mobileHIVE - Final Project Report

# StreetDroids - A Mobile Context-Aware Game for Android

University of Applied Sciences Bremen

University of Applied Sciences Bremerhaven

16th of April, 2010

# Contents

# 1. Executive Summary[1]

[1] Isabella Lomanto, Jana Wedekind

*“In a gathering and hunting society children play with arrows, in an information society children play with information.”* (Levy, 1997)

Originally the master project was designed to be two separate ones: *Game-based learning in Museums* from the University of Applied Sciences Bremerhaven and *Contacts in Context* from the University of Applied Sciences Bremen. Putting them together in the *mobileHIVE* project confronted both groups with some great advantages as well as some difficulties. In the beginning this fusion meant for the whole group to find a new combined project that fit to most of the criteria of the two old ones. After long discussions and a lot of brainstorming a new idea evolved, which took the game out of the museum, and placed it in the city of Bremen. The new combined *Context-aware Mobile Learning Game* was born. Though the general description for the *mobileHIVE* project was found, now an idea for the actual game was needed. Starting off with several objectives, soon the vision of a treasure hunt formed and became the starting point for the game development.

Several games on the market were studied in order to see what other institutions or companies were working on at the moment, and to find out where our project could fit in. One conclusion of this preliminary research was that learning games are sometimes seen as non appealing and behind the times by students. Technology evolved and in the last decade the usage of mobile phones increased immensely. New technologies therefore allow new ways of user interaction. Especially the application of touch interfaces within affordable devices gives developers new possibilities. By using new technologies and devices (like the *Android G1*[2]) a game can be created, that is more appealing to new generation of users. These new technologies support approaches such as context- and location-awareness, which can be used to make games that are more connected to the real world and to the community around the user. This can be seen for example in *MobiMissions*[3] from the Futurelab[4], a location-based, social, mobile phone experience.

The main question deriving from the research was how to create a context-aware mobile learning game where the player can experience a different way of learning through interaction with the environment and other players. The decision was to create *StreetDroids,* an open game where the user community can contribute to its expansion and moreover the game mechanics allow the implementation of different games scenarios (e.g. a game in the city of Bremen) as so called maps are used to introduce the player to a certain story. In a first step an exemplary map that uses the history of the old city of Bremen was implemented. The game took place in the real world, which is called pervasive gaming or real world mission games. Jane McGonigal defines the term pervasive gaming as follows: *“Pervasive gaming is an experimental genre in which at least some of the gameplay transpires in real world environments with the aid of mobile and ubiquitous computing technologies.”* (McGonigal, 2007) As an inspiration for this genre

[2]The G1 was the chosen Android mobile device for the *StreetDroids* Project. Further information about this device is available in: `http://www.androidg1.org/`

[3]`http://futurelab.org.uk/projects/mobimissions`

[4]`http://futurelab.org.uk/`

of games the artistic group *Blast Theory*[5] has to be mentioned. The game *GPS Missions*[6] is another example showing how up-to-date the idea of using mobile devices within real surroundings is.

*StreetDroids* is a context-aware mobile game where players are encouraged to learn by exploring and interacting with the city. It was developed over a year by the sixteen members of the *mobileHIVE* project and the two project supervisors Prof. Dr. Ulrike Erb (HS-Bremerhaven) and Prof. Dr. Thorsten Teschke (HS-Bremen). This report describes the work that led to the development of the *StreetDroids* game. The content of the report is divided in three parts: concept of the game, implementation, and results.

Chapter 2 explains in detail the concept of the game starting from the original idea. This chapter sets out the theoretical foundation that supports our research, and the conceptual structure used in the game design. It includes an overview of the game mechanics and a thorough description of how the collaboration and learning aspects were applied in *StreetDroids.* In the end, the topic of content re-usability is addressed in section 2.5. Here the content chosen for the prototype is presented, focusing in the reasons why this particular content was used and how different contents can be applied to *StreetDroids* by using the web platform.

A third chapter covers the actual implementation of the project. This chapter deals with the technical and design decisions that gave shape to *StreetDroids.* It begins by presenting the basic technologies employed in the development of the game and its online community. Then it describes in detailed the system architecture and the concrete implementation of both the mobile and web platform, including the problems that occurred and the solutions applied. Furthermore, this chapter offers an overview of the final stage of development achieved in the final prototype.

Finally, in Chapter 4 the main results of the project are described and reviewed. Here the prototyping process is explained, including a reference to the demo videos of the functionalities of the final prototypes and their applications. Section 4.2 gives a complete account of the evaluation process and an extensive analysis of the play-testing results. Additionally, this chapter presents the future work that can be done to expand or continue the project research.

[5]`http://www.blasttheory.co.uk/bt/type_games.html`
[6]`http://gpsmission.com/`

# 2. The concept

## 2.1. StreetDroids Concept and Game Mechanics[1]

[1]Cristina Botta, Isabella Lomanto, Joatan Preis Dutra

### 2.1.1. *StreetDroids* Original Idea

The original idea was to build an outdoor game based on the concept of *treasure hunt*, where the player must find and collect items through real scenarios – such as the old city of Bremen, for instance. The game serves as background to achieve the some learning goals related to history; the original goal was to enable students to learn about history through personal experience at the original historical places, supported by mobile devices, making the city itself as a "live museum". This could be advantageous feature for the learning process, adding extra external source beyond books, in reason of a direct contact with the subject matter in a playful approach.

The original concept for the game started with the idea of offering a different and attractive learning material for the students, combined with the concept in using the city as an open playground. Teaching history as a subject could be improved if it is possible to use real world experiences, which would make a different approach on traditional learning methods. With an electronic gadget, here represented by a mobile game, the kids would be more motivated to explore historical places while learning about their importance.

A guided outdoor learning possibility could be expected to motivate especially the young students to take initiative and improve the exchange of information between them. A treasure hunt game running on the cell phones of the students could offer a fun way to deal with historical contents about the environment they are placed. Another advantage of this game concept lies in the fact that a treasure hunt could be composed by any author (i.e. teachers) with enough knowledge of the location. The work could be done from anywhere, through an online platform, integrating several kinds of media (audio, images, video and text). It means, for example, a scheduled travel to a near city could be changed into a treasure hunt adventure: the teacher just has to upload the map into the cooperative system and add the puzzles to provide the learning content for the planned route.

With its potential for enriching traditional classes, the game could be part of local historical lessons for a school. For the sample adventure, the old history of Bremen will be uncovered through the game route tasks. The students can better integrate the facts during the game in their already existing conceptions, when they have some previous knowledge about the topic. A simple touristic guide could be a derivation of the original idea, but it will not be attractive enough for the educational purpose.

The history of Bremen covers important aspects that are crucial for understanding the impact and influence about the past facts that shaped the present of the city. For instance, Bremen was part of the *Hanseatic League* (Hanse, 2009), and was called "Freie Hansestadt Bremen" (BremenOnline, 2009), there is a Roland and Stadtmusikanten statues around of the City Hall, leading to questions as: what are the meanings behind them or why are these momuments important? Here, the learning exploration possibilities are wide open, that is why the first target for the game was the old city of Bremen and its historical aspects.

The game original idea supports constructivist learning, as "*constructivism, like objectivism, holds that there is a real world that we experience. However, the argument is* *that meaning is imposed on the world by us, rather than existing in the world independently of us. There are many ways to structure the world, and there are many meanings or perspectives for any event or concept*" (Duffy and Jonassen, 1992).

It could be expected that users will be able to enhance their learning while playing adventures, because learning subjects come into tangible real-world reach and users must engage personally and apply multi-strategy problem-solving in order to complete the adventures. More than just a game, it can be a powerful – easy to use and adaptable – tool to the organization of creative and fun activities, associating the playful aspects to the educational objectives.

### 2.1.2. *StreetDroids* Aims and Objectives

The main objective of the mobileHIVE project was to create a context-aware mobile game in which players gain knowledge by actively interacting with the environment and other players. *StreetDroids* was developed in response to this objective, following the hypothesis that such a game could bring about a more immersive and dynamic game experience. Hence, *StreetDroids* should allow the players to connect the game world to the real world, so that they could join digital content with real life objects.

The specific aims in the mobileHIVE project were:

- to understand how mobile games can enhance and motivate learning processes through the exploration of a given environment.
- to create a mixed-reality environment, where players can collaborate inside and outside the game; inside by context-aware interaction and outside through an online user community.
- to develop a context-aware application for the Android platform that allows for content reusability.

### 2.1.3. Description of the Game Mechanics

A story map is what glues the theme of the game together, in the example developed by our team it is the history of Bremen. A story map can contain one or more missions related to the same theme. These missions are made of two or more puzzles, which are connected in a predefined circuit. When the player begins she can not see the location of these puzzles. Since the circuit can be started from any of the puzzles, the one which is closer to the player's location when he enters the game will be revealed as a starting point. A puzzle is an activity that has to be performed at a distinct place in order to gain information and items.

Non-playable characters (NPC) are used to guide the player through a mission. They are the ones to give the player hints on how to solve the puzzles, and information on how to find the next puzzle in the circuit. In the case of the test scenario that could mean that the characters giving the player information are directly related to the history of Bremen, and to the current location they are talking about. This helps to keep the information in context and make the game experience more immersive for the player. Instead of having the device simply displaying some textual information, the NPC is part of the story and can react to the player's decisions.

Puzzles have the objective of creating a bridge between player and environment. The information that is given in the puzzles is only complete when the player examines the place/building that is related to it. Thus, puzzles can only be solved with information from both worlds, the real and the digital.

After successfully solving a puzzle the player will receive an item from the NPC. These items are stored in the player's inventory and have some information attached to them. The information attached to the items is consisting of short texts that are supposed to help the player have a better understanding of the story.

All missions have to be played to finish the whole story, but with each mission a part of the story will be completed, missions can therefore be played independently. An essential part of the game concept is the possibility to contribute to its expansion through the online community. This means that players are able to create the following parts of the game: stories/missions, puzzles, characters, and items. All created parts should be re-usable and editable by anyone in the community.

## 2.2. Conceptual Structure[2]

[2]Cristina Botta, Isabella Lomanto

Following the objective to create a context-aware mobile learning game where the player can experience a different way of learning through interaction with the environment and other players, we identified the main design requirements that define the game mechanics. Besides being context-aware our game should be reusable in different scenarios or contexts, allow and encourage collaborative play, support and enhance mobile learning, and be engaging and fun at the same time. Including all these requirements in a single application presented a major challenge. As a design solution we came up with the main features that differentiate our game and respond to our preconditions. According to these features our game is defined as:

- Spatially expanded
- Location based
- Spatial narrative
- Context aware
- Mixed reality mediation
- Collaboration
- Open play

In the following sections the main features of our game will be explained in detail.

### 2.2.1. Spatially expanded

Since our goal was to design a context-aware game where the player can interact both real and virtually with the environment we decided to incorporate in the game mechanics some of the characteristics that identify pervasive games. While there is not one precise or completely accepted definition of this game genre, pervasive games can be distinguished from traditional game genres because they blur the boundaries between game fiction and reality by using the real world as a gameplay area (Montola, 2005), (McGonigal, 2007). Montola's approach to pervasive games puts emphasis on how this type of games disrupts and expands the magic circle (Salen and Zimmermann, 2003) that defines traditional game experiences. He argues that this magic circle can be expanded in one or more of three directions: socially, spatially or temporally. He explains that these transgressions produce a sense of ambiguity which would in extreme cases make the game appear as if it was not a game (Montola, 2005). McGonigal has also identified these extreme cases of pervasive games as alternate reality or immersive games, which she argues are based on a *This is Not a Game* rhetoric. However, in the design of *StreetDroids* the focus was not in this type of pervasive games. The objective was not to completely blur the magic circle but to examine and experiment with the design possibilities that emerge by altering some aspects of it. In particular, we focused on the potential of the spatial expansion offered by pervasive play. Montola argues that what he calls spatial expansion in games brings the opportunity for people to reclaim public space, to explore it in different ways and to rediscover it (Montola, 2005).

Nevertheless taking the gameplay to the public space or allowing the player to move into different scenarios outside the virtual game world is not enough for a game to be considered spatially expanded. Montola affirms that mobile games are not spatially expanded in nature, it is the relation they establish with the physical world and the meaning of this relation within the game what allows spatial expansion in mobile games. *"The spatial expansion only applies to games that are affected by the player's spatial context, usually in relation to physical places or to other players."* (Montola, 2005).

In order to be able to enhance player experience and motivate learning processes, exploration as a mode of interaction is at the core of the game mechanics of *Street-Droids*. To design this meaningful exploration it was very useful to consider the distinction between space and place. Different studies such as (Davies, 2007), (Dourish, 2006), (Reid, 2008) have shown that the notions of space and place can be used to create player immersion in pervasive and location-aware games. Of special significance are the arguments presented by Paul Dourish. For him space and place are both products of social practice. However, he finds that whereas space describes geometrical arrangements that might structure, constrain, and enable certain forms of movement and interaction, place denotes the ways in which settings acquire recognizable and persistent social meaning in the course of interaction (Dourish, 2006). This distinction is fundamental for designing an active and more immersed relation with the environment in games. In particular for pervasive or location-aware games it means that the players will be moving in both space and place; in space by navigating through a geographical map, and in place by exploring and emotionally interacting with the physical environment.

### 2.2.2. Location based

The design of our game as a location based game responds to the idea that spaces and places store and communicate narrative, aesthetic and psychological information (Davies, 2007). We wanted to use the interaction with space as a medium to deliver learning content and to create greater immersion in the game. For this reason we set the city, in particular the Old Town of Bremen, as the game world where the actions will take place. We encourage the players to explore the environment by interacting with real world people, objects and places as well as with virtual ones. As Jane McGonigal explains that the goal of pervasive game design is to take advantage of the possibilities offered by ubiquitous computing technologies in order to allow the players to be immersed in reality itself, rather than in a digitally rendered virtual reality (McGonigal, 2007). Following these ideas we designed a map based interaction with the space in which the players can choose either an open map where they move freely and find the puzzles of the game as they go, or a story map that is based on a story that will indirectly take them to certain related puzzles by discovering and following hints embedded in meaningful places. In either case, the open or the story map, players have the freedom to start at any point they want. Even in the story map we wanted to give the feeling of free exploration by having the route of the mission or connection between the puzzles hidden.

Some of the examples of good and interesting pervasive, location aware or real-world mission-based games that we considered for designing our game are: *Uncle Roy All Around You*[3], *MobiMissions*[4] *Interference*.[5] What these games have in common to ours is that they set the game in the urban environment by placing the game activities and missions into the real world. These three games also incorporate narrative elements and open channels of player-to-player communication. As well as these games we looked at projects that involved user content creation of missions, games or activities embedded in the real world, such as *GPSMissions*[6] and *Mscape*[7]. These are systems that allow players to design their own context or location aware experiences supported by a community that share and exchange their creations.

We believe that one of the successful mechanisms employed in these games is giving the player the ability to place, collect or interact with virtual objects embedded in the real world as part of the game play. This enriches the game experience and allows for emergent behaviors (Reid, 2008). For this reason we have included these type of interactions as part of the game mechanics. However, this type of interaction with the physical world is still limited in many of these games because it reduces exploration to a simple mechanic of going to a place and triggering a reaction in the game. Our game design aimed to support a more complex and meaningful exploration, which would allow the player to relate emotionally with the environment. Our goal when defining our game mechanics was to enable the player to experience the game environment as both space and place.

### 2.2.3. Spatial narrative

One of the first choices that we made when designing the mechanics for our game was to include narrative elements and role-play interactions. The idea was to have a story drive the game play and make the players take the roles of different characters and relive the story through the characters' partial perspectives. However, if this was meant to be the main structure of our game, we had to simplify the story and make the roles of the characters more general in order to allow the reusability and the replayability of our game. We decided to keep the idea of having a story drive the flow of the game play and the different characters perspectives but we chose a specific form of narrative, the environmental storytelling.

According to Henry Jenkins environmental storytelling "*creates the preconditions for an immersive narrative experience in at least one of four ways: spatial stories can evoke pre-existing narrative associations; they can provide a staging ground where*

[3]`http://www.blasttheory.co.uk/bt/work_uncleroy.html`
[4]`http://www.futurelab.org.uk/projects/mobimissionsand`
[5]`http://www.pervasive-gaming.org/iperg_games13.php.`
[6]`http://gpsmission.com/`
[7]`http://www.mscapers.com/`

*narrative events are enacted; they may embed narrative information within their mise-en-scene; or they provide resources for emergent narratives*" (Jenkins, 2004). This type of narrative worked quite well with our game idea because we needed the environment to be the main character of our story and to let the events be told in great amount by the interaction with places and objects. For our game "spatial stories" have the benefit that they are episodic, privilege spatial exploration over plot development, have broadly defined goals and are driven by the character's movement across the map (Jenkins, 2004).

We applied this idea of spatial narrative into our game by creating one scenario or story map where a main but general story sets the atmosphere and the principal conflict by describing the ultimate objective of the game. The players choose to follow the particular view of one or more of the characters in the story to reveal the solution to conflict of the story. However, the role or mission of each character is compelling in its own terms and could be played independently and still be meaningful and enjoyable. We decided to have these two levels of the story to allow a more open game play, that supported single or multi-player modes and that gave the player the opportunity to choose which, when and how many characters he or she would play.

### 2.2.4. Context awareness

The concept of *context* have been long discussed in the research on human computer interaction. Perhaps one of the definitions most widely accepted is the one by Dey et al. 2001 *"any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves"*.(Dey et al., 2001) However, Terry Winograd (Winograd, 2001) makes an important adjustment to Dey et al. definition. This author proposes a more specific definition that focuses on the communicational aspect. He argues that context is an operational term, and that some information becomes context because of the way it is interpreted and not because of its inherent qualities.

Our idea of a context aware game falls into Dey and Winograd perspective, because we aim for a form of interaction in which the responses of our system are triggered by the contextual information that is relevant in each moment of the game. As the player moves and interacts with the environment the context of the game play is constantly changing and a context aware responsive system is needed to support this form of interaction. The context in our game does not only include the location, but a broader spatial relation to places, objects or other players, as well as the history of the player (where he or she has been and where will go) in the game.

From the game experience point of view the ability of the system to respond automatically to the changes in the players context produces a sense of coincidence which is felt as a magical unexpected event by the player. Coincidence is defined by Josephine Reid as *"the noteworthy alignment of two or more events or circumstances without obvious causal connection"* (Reid, 2008). She also argues that having an understanding and knowledge of the environment can maximize the chance and effectiveness of these coincidental magic moments, which can be also artificially created inside the game.

Finally we believe that context awareness helps the player feel free to make meaningful choices that influence the game outcome. However, the player does not have to literally tell the system everyone of his or her choices. On the contrary, the idea is that his or her interaction with the physical space will be *naturally* translated into the virtual environment of the game and vice versa. As Katie Salen and Eric Zimmerman argue the ability of the player to freely explore the environment and the interaction with the context that surrounds this exploration produce emergence and leads to unpredictable narrative experiences (Salen and Zimmermann, 2003).

### 2.2.5. Mixed Reality

*Mixed reality*, or *augmented reality*, explores the application of computer-generated images in live-video streams as a way to expand the real-world. Milgram and Kisinho (Milgram and Kishino, 1994) established a *Mixed Reality* dimension that attempts to define a continuous scale along which environments that mix the real and virtual can be defined. At one end of the spectrum are totally real environments with nothing virtual in them at all. At the other end of the spectrum are virtual environments without any real components. In the middle are *augmented reality* and *augmented virtuality*. *Augmented reality* would be according to this definition, a combination of the real and the virtual, which contains more real than virtual. This could be an environment in which virtual elements, like people or objects, are added to a real location. The other side of *augmented reality* is *augmented virtuality*, and this may be thought of as adding elements of reality to a virtual environment. Here one might be projecting a real person or an object into an otherwise virtual environment.

The above definition was created with immersive technologies in mind, like for example having the player wear a helmet or goggle and seeing real-time virtual information overlaid on a real environment. Klopfer (Klopfer, 2008), suggests a more suitable spectrum for games that are still *augmented reality*, but use commercial off the shelf technologies, which is the case of *StreetDroids*. He has used a spectrum that defines the weight of the augmentation along a continuum from light to heavy. The weight refers to how much virtual information is provided to the player. A lightly *augmented reality* has the player using a lot of the physical reality and accessing virtual information quite rarely. Players may look at the virtual information every few minutes or even hours. A heavily *augmented environment* relies on frequent access to virtual information. That information may be accessed on the order of seconds or even continuously.

With *StreetDroids,* the objective is to use *mixed reality* to emphasize experiences where the environment matters in a non-trivial way, and does not simply act as a background for a game with overlaid media. From the artifacts found in a real city to the people sharing experiences and their creations with the community, the place and people are what gives the experience its meaning. *Augmented reality* games combine different forms of media and real-world presence to create an experience that actively blurs the border between fiction and reality. It encourages the participants to engage with reality, not escape from it. As in pervasive gaming it blurs the boundaries between what is game and what is real, the “magic circle” (Salen and Zimmermann, 2003).

Games and any kind of participatory simulation propitiate a high level of personal engagement. By using *mixed reality*, and pervasive gaming principles, we want players to be more aware of the environment around them, what happens (or has happened) there, and the people who inhabit it. *Augmented reality* is also a great way to simulate a real life experience. Being in the field enables the player to get a much better sense of the terrain that he or she has to work with, it allows for a more authentic feel, and also may change some decisions the player makes during the game. The player can see who are the people using that space and how they interact with it, if it is noisy or has a lot of traffic, and therefore be more informed about the real consequences an action has towards this place.

#### 2.2.5.1. What we wanted

At first the objective was to make moderate use of augmented reality (AR) by overlaying images and objects on the camera view of the mobile device, and have players pick up and use virtual objects that would “be” laying around the city like any real object. We also wanted to use AR in an interesting and meaningful way and not just for the “novelty factor”. The use of AR would have to make a real difference in how the player interacts with the world around him to be in the game.

Some interesting ideas arose from the brainstorming meetings:

- Have a wall in the city center where people could leave virtual/AR notes, clues and pictures for other players.
- Mark items for other players (e.g. here is a door that can be opened with the key from the other character).
- Players can leave items behind for other players (e.g. a key that only one specific character can receive, but only another one can actually use).
- An overlay showing how a building looked in the past.

One requirement for AR to be used was that it had to run directly on the mobile device, without the help of any other device. This decision was taken for two main reasons. The first being accessibility, so more people could play the game without having to own a lot of electronic equipment. Also one of our objectives was to create a community and expand the game, but if players have to own a lot of expensive equipment this would become almost impossible. The second reason is portability, to give the player freedom to come and go easily, and to be able to just pick up and play anytime, anywhere.

Another requirement was that it had to work outdoors, since the whole idea of the game is based on space and environment. It also would have to work without fiduciary markers[8], since it would be impossible for the team, and later for the community to go around the city sticking markers to buildings.

#### 2.2.5.2. What did not work

When the project started, the objective was to implement the team's ideas with off the shelf technology and the available AR toolkits and SDKs. After many members of the team researched on the subject for quite a while it was concluded that using this level of AR would be a huge project on its own. All prototypes and games found ran not from the mobile device used for playing, but from a laptop being carried in a backpack, or from servers. Some of them also used goggles[9].

#### 2.2.5.3. What is still there

Using Klopfer's definition (Klopfer, 2008) *StreetDroids* is on the lighter spectrum of *augmented reality*. The augmentation comes from the use of location awareness, virtual NPCs that the player can only interact with when he or she is at the right place, information and puzzles connected to the players current position, the use of a virtual compass and map, and awareness of other people playing in the same area, which would trigger special features.

*StreetDroids* makes the location in which it is being played a key factor to the gameplay. Players move in real space and get the information from the device when they reach certain places or trigger points. Without a real environment there is no game, since players are required to explore places in a real city to find the answers to the puzzles. The information is given to players in the device display by virtual NPCs, but still the most important part of the game is to explore and observe the real environment. Some players might even consider the possibility of interacting with the locals by asking questions that might help to solve a puzzle, consequently not having to pay for a hint from the NPC. This is also another way of stimulating players to talk and interact with the people around them.

### 2.2.6. Collaboration

There are different ways to collaborate in *StreetDroids*. One is directly while playing the game, by participating in puzzles that can only be solved by two or more people. The other way of collaborating is by creating content, expanding the game, and being an active member of the online community.

#### 2.2.6.1. Player-player interaction

Conversation theory (Sharples et al., 2005) states that the process of sharing information, of having a continuous communication, helps the learner to achieve a better

[8]`http://en.wikipedia.org/wiki/Fiduciary_marker`
[9]`http://www.ipcity.eu/?page_id=10`

understanding of the learning objectives.

In *StreetDroids* this sharing happens either through interaction with the online community or through player-player interaction during the playing of the game. One planned feature was for players to be notified if anyone nearby is also playing the game giving them the option to communicate through chat, and the possibility to help each other or play "together" even if using separate devices. Another feature that was planned, but not implemented, was to have multiplayer puzzles that would be activated when two or more players are in the same area. The player would have the opportunity to agree to play together, or to not take part in the player-player interaction at all. If a player agrees to participate in a collaborative puzzle she would get a reward, e.g. receive additional coins or an extra item. These two features can also help to build a sense of community through the sharing of experiences and information. This could then encourage people to meet and play together or start being active on the online community.

The game mechanics are also open enough to allow different ways of playing the game and different kinds of collaboration. If, for example, a map for a school class is created, the class can be divided into teams that play against each other. Missions could be done in a way to complement each other, and within a team the players would have to exchange their knowledge in order to achieve their common goal, to win the game.

The game is not dependent on in-game collaboration to be played though, and it was also avoided to have a game that is only playable with a predefined number of people or only at a certain time. By doing so the game lost some collaboration aspects, but gained a lot of freedom. This subject is developed in more detail in section 2.2.7, "Open-play".

#### 2.2.6.2. Community

In finding each other, players discover what Bernie DeKoven would call their *"play community"* (DeKoven, 2002). DeKoven defines this community as a group of people who want to play together, and may change games, and might even recreate game rules with the objective of creating a balanced playing field for its members. The play community has a special tie, one that is different from what scholars of computer-mediated communications refer to as *"communities of interest"* or *"communities of practice"*. Play communities form around shared play styles and preferences.

> *"Different communities of play have different characteristics that arise out of the combined play styles of the individuals within them, each of whom is, in turn, transformed by the group play style. These play styles are also both influenced and transformed by the spaces they are enacted in."*(DeKoven, 2002)

Another subject that is deeply connected with communities is fan culture. Henry Jenkins has documented the fan culture of the so called Trekkies (Jenkins, 1992), fans of the television show Star Trek, who have not only taken up, but also extended the mythology of the classic science fiction world by creating, for example, alien dictionaries and fan fiction. Trekkies are considered the pinnacle of nerd-dom in the US, and they are just one example among many (X-Files, Lord of the Rings, Final Fantasy, are others). All these fan communities foreshadow the growing participatory culture of play, one in which players themselves take the central stage in the playground.

This player-centric approach views the game in the context of the agency it yields to players, the experience they have while playing, and the social constructions that take place in the context of play communities. *"These communities set people in motion, function as a cultural attractor and activator, and also create a common ground between diverse communities"* (Jenkins, 2006). Players and fans get more out of the experience if they share it with others.

> *"When motivated by the group, players find in themselves new talents, abilities and skills, which are further enhanced through a process of group feedback. Furthermore, based on the type of games players gravitate towards, we can, to a certain extent, anticipate overall patterns of emergence based on players' play styles, predilections, and resident skills."(Pearce, 2009)*

The *StreetDroids* community works as hub, a way of bringing very different people together. What unites all these different people from all over the world is playing *StreetDroids*. But from there people can find their own niches, being it the real spaces they visit or live in, common interests, or even competition. The community serves also as a central for the player, where they have a profile, can see what they have already accomplished in the game and the information and items they have already collected in their inventory.

The community gives many possibilities for people to find what they like, and if they cannot find it they can create it, as long as it has some relation to a physical space. These shared interests could be since living in, or simply liking, the same city to only making puzzles about cheese, or only in places a certain TV show was filmed. People are more encouraged to try something if they have someone to share it with and get feedback from.

Competition can happen while playing the game, for example, who is faster, but in the community there are many other possibilities. Instead of using a point system within the game, the player can achieve different levels of experience, which will give him status in the community. The level of experience increases not only by playing story maps or puzzles, but also by collecting objects and information within the game, by taking part into collaborative puzzles, and by contributing actively in the community.

Contributing by, for example, helping other players or creating content to expand the game, can creates its own competition. Who makes the best puzzles, who has the most creative stories or characters, and even competition between cities, like which one has the most puzzles, or the most interesting ones, or the most active local community. The possibilities are endless, and it almost impossible to predict what kind of community *StreetDroids* would be, which kind of people would be the most active and influential.

#### 2.2.6.3. User generated content

Pierre Lévy suggests that the *“distinction between authors and readers, producers and spectators, creators and interpreters will blend”* to form a *circuit* of expression, with each participant working to *“sustain the activity”* of the others. The artwork will be what he calls a *cultural attractor*, drawing together and creating common ground between diverse communities (Jenkins, 2006).

Online communities and games provide a networked media environment in which players have the opportunity to modify, rebuild, and adapt a game’s message at the levels of narrative, gameplay and/or cultural space. Players immerse in the challenge of creating both permanent and evanescent elements in games and game systems. Sometimes these elements are not explicitly part of initial game design, and often subvert it. While it is only a relative minority of players that participate in game modification, their contribution to the overall game community makes sure that a continuous flow of new game modes, contexts, and content expands and contributes to keep the game alive.

The online environment also encourages and quickens distribution, as it allows for the propagation of instructions and advice, and lets players form spontaneous working groups. Even single player games, such as Maxis’ The Sims[10], have the opportunity to become an actual multiplayer game through the communities that are born around them.

Keeping *StreetDroids* alive, with its content up to date, and independent of its developers was one of the motivations to create a community, since after the project time was over there would be no one to take care of it or create more content for it. Unless the developers would be active in the community themselves, of course. At this point, as Pierre Lévy suggested, the distinction between developers and players would disappear completely.

*StreetDroids* also falls in the category of a single player game that could be turned into a multiplayer experience through its community. Either by people working together to create a *Story Map*, or by trying to create an actual multiplayer map, which was not created by the developers, but could be done by a creative community. As stated in the second paragraph, it is not unusual for players to alter the way a game is supposed to work. It would have been definitely interesting to see what kinds of things people would try to do with *StreetDroids*.

According to researchers Salen and Zimmerman (Salen and Zimmermann, 2003) player-produced modifications can occur in one of two ways: production occurring within the magic circle moving outward (inside > out), or production taking place within culture that moves inward to affect the game (outside > in).

An example of the former would be machinima, and the huge community that exist around this culture. There are even series created from games like Halo, as for example *Red vs Blue*[11], a comic science fiction series.

[10]`thesims.ea.com/`

[11]`http://halomachinima.wikia.com/wiki/Red_vs_Blue`

The (outside > in) aspect of collaboration is already becoming mainstream even in commercial games, and as an official feature instead of hacking, which was the norm until recently. Some examples of this are creating tracks on a racing game, like in *Excite Bike*[12], or music that can later be used in the game, like in *Guitar Hero*[13], or even the whole level and characters, like in *Little Big Planet*[14]. These all can be later shared in online communities like *XBox Live Arcade*[15] or the *PlayStation Network*[16], for example.

We wanted to create the (outside > in) aspect of adding and/or modifying the game, by allowing players to create their own experiences and contribute freely to expand the game world. At first the idea was that players would, through an editor in the online community, be able to create story maps about any subject that interested them. Thus if they wanted to create a food, music, or an architecture story map it would all be possible.

Later we realized that this could greatly reduce the number of people creating content. Very few people would have the time and the will to create characters, puzzles, a whole story, and tie it all together. Therefore it was decided that the whole system should have a lower entry barrier, be more reusable, and faster for the user to be able to create and contribute with something. The objective was to keep the creation of content as accessible, as fast and as easy as possible, so more people would want to use it. By having a lower entry barrier and giving the creators as much freedom as possible we also hope to create as diverse a community as possible.

The editors now would give community members the possibility to contribute by creating or modifying puzzles, characters, items, and missions/stories. All created parts being re-usable by everyone and independent of each other. This means members can use existing puzzles to build their own story maps, as well as building a story map from scratch. It would now also possible to only create a single puzzle or only a character, that can then be re-used by other members if the so desire. This way a person in the community who knows a lot about the history of a certain place, but just wants create puzzles, because he is not, for example, interested in storytelling, could do so. And lets say another community member loves to write, but has no patience to think about puzzles, she could simply integrate already created puzzles into her *Story Map*.

### 2.2.7. Open play

As mentioned earlier, we wanted to avoid a game that is only playable with a predefined number of players or only at a certain time. Our objective was for players to be able to play at any time they want, anywhere there are available puzzles, thus creating a more spontaneous experience. Another objective was to allow the game to be continuously played, and not be tied to sessions. One example of how this could cause the player to give up on playing the game is the possibility of people not finding anyone to play

[12]`http://en.wikipedia.org/wiki/Excitebike#Modes`
[13]`http://en.wikipedia.org/wiki/Guitar_Hero_World_Tour#Custom_songs`
[14]`http://en.wikipedia.org/wiki/LittleBigPlanet#Content_creation`
[15]`http://en.wikipedia.org/wiki/Xbox_Live_Arcade`
[16]`http://en.wikipedia.org/wiki/PlayStation_Network`

with. In this case the player would have to wait, or set an appointment with friends or people in the community to be able to play. Also being only able to play when the game designers hold sessions would force people to play only at very specific days and times, and maybe the game would be actually played only once or twice. These were disadvantages noticed in most mobile learning games, like for example *Environmental Detectives*[17] *Mobile Game Quest*[18], *Frequency1550*[19], and *Anywhere Somewhere Everywhere*[20].

The game was designed to facilitate entry of new players, but also to be complex and diverse enough to interest old players into coming back. The main way used to make starting to play as simple and as fast as possible is through what we call *Open Map*. The *Open Map* would display all puzzles ever created for the location the player resides at the moment. Players should be able to filter this map by tags of interest, for example display only puzzles relating to history. In the *Open Map* all puzzles are visible to the player and are not connected in a predefined route. The player can choose freely how many puzzles and in which order he wants to play. This makes for a more casual game experience, but also allows the player to experience the game immediately, and have a short or long game depending on how much available time she has.

The game experience can also be longer and complex through the *Story Maps*. The *Story Maps* have, in contrast to the *Open Map*s, connected puzzles with a predefined, closed circuit. Although the entry-point into the story is depending on the position of the player in the beginning, the amount of puzzles in a certain mission is always the same and they are played in a circuit. The puzzles are connected by a story that is revealed while completing the puzzles on that map. Players will navigate through the *Story Maps* by following the clues that will be given by a NPC. A *Story Maps* can contain several *Missions*, each with a different character. All characters have to be played to finish the whole story, but with each mission a part of the story will be completed, missions can therefore be played independently and can be stopped and restarted at any time. The *Story Maps* create an experience that connects the player to the location, and in the case of the map created as an example by our team, to Bremen's history. *Story Maps* require more time and dedications from players, but also compensates them with a more meaningful and complete experience.

[17] `http://education.mit.edu/ar/ed.html`
[18] `http://www.waag.org/project/gamequest`
[19] `http://www.waag.org/project/frequentie`
[20] `http://www.mrl.nott.ac.uk/~hms/AnywhereSomewhereEverywhere.html`

## 2.3. Collaboration Concept[21]

[21] Nesrin Abdelrazik, Alexander Tyapkov

Besides single player mode a concept for multiplayer mode was developed for the mobile game *Streetdroids*. In this part of the game the group focused on creating a collaborative gaming experience where content and communication are blended in order to support the social interaction between the players. The following chapter reviews the idea of the collaborative puzzle and presents an example of how the puzzle could look like, but first the current state of collaborative games will be presented.

### 2.3.1. Current state of collaborative games

In recent years the number of mobile devices, such as mobile phones and PDA's increases dramatically. The manufactures investigating new markets are proposing solutions with wider functionality. Indisputably the physical functionality of the mobile devices should be supported by a variety of software allowing users to perform almost all the necessary operations. One of the most profitable types of software can be considered to be games. To help developers profit from their applications the majority of mobile platform's owners provide access to the Internet markets where programmers can sell their works. From the rapid development of the technology – in terms of the decrease of the mobile devices' size, deployment of diverse sensors and methods of communication between devices as well as a simplification of game marketing mechanisms – one could expect an increase in the number of games in which users should interact with it others. However, the number of collaborative games is negligibly small. Even more if one consider educational collaborative mobile games, then the number of such applications tends to zero (Zagal et al., 2006). Nevertheless games are appreciated to be important and enjoyable means of education. Playing game people find more about their skills and limitations in a stress-free environment.

The development of collaborative learning games not only needs versatile skills in programming and design but also appropriate knowledge in educational questions and psychology. According to the research by José Luis González Sánchez et al. (2009) there are three ways of interaction during the learning process: competitive, individualistic and cooperative. The majority of current methods and games are based on the competitive approach, which encourages players to become better by comparing themselves to others. However, the research describing student's interaction states that: *"the vast majority of the research comparing student-student interaction patterns indicates that they learn more effectively when they work cooperatively*" (González Sánchez et al., 2009). The same approach can be applied to the learning games and we can state that players learn more efficiently when they are playing in a group. The research mentioned above provides the guideline which can be used in our puzzle to make the game collaborative and learning efficient. This guideline is based on the following definitions which were later used in the developed collaborative puzzle: *positive interdependence, individual accountability, face-to-face interaction, social skill,* and *self-analysis of the group.*

Based on the proposed definitions and the guideline to create educational video games it was decided to mark out the most crucial rules and make up a document. The aim of this document was to develop the basic framework fitted with examples which should later help in development of an educational collaborative puzzle. According to the definitions described above it was proposed following:

**Positive interdependence**

Players should be aware that they are a team, so group success or failure will represent individual success or failure. In order to achieve that it was proposed to organize the players into the teams with common attributes to create a group identity. Moreover after solving the puzzle players should receive a reward which should be shared with other players. In some cases the creation of the team, either virtually or physically, is not enough therefore the variety of activities in which players can contribute something to the group should be proposed. Eventually there should be group and not individual evaluation of challenge after solving the puzzle and the possibility to compete against other groups should be offered as well.

**Personal accountability**

Each player should share his or her knowledge with all the participants and learn from his or her partner's contributions. That can be achieved by representing the contribution of each player during the game play or by awarding points to the team when individual activities are great. Also situations in which one player contributes more than other should be considered. In this case a balancing mechanism which will propose more difficult surprising tasks to the leader should be realized.

**Face-to-face interaction**

The aim of face-to-face interaction is to establish social relations between group members. This can be achieved when players discuss possible solutions and points of view, helping other who are finding task or puzzle difficult. Possible solutions which can be used in developing puzzle can be including of challenges or activities in where all group members must respond in the same way, forcing team members to decide which of them solves the challenge. Such implementations in which each player contributes or solves the part of the puzzle or where the whole team is competing against another team are also possible.

**Social skills**

While solving puzzles players should organize themselves, make decisions and show leadership and conciliation abilities. In order to achieve it the developed puzzle should allow establishing a leader role which the group members play in turns. The leader can manage the work by assigning members to a task, giving advice, helping the partners in difficulties. The logic of the puzzle can give feedback to other group members to establish discussions and social relations or show the problem on a device so that the device owner will not be allowed to solve it but must explain the problem to the rest of the team members.

**Self-analysis of the group**

The group of players, organized virtually or physically, must self-analyze itself to discover whether the found solution is effective and the targets of the puzzle are being achieved. This can be done if players are able to see the statistics and analyze which members succeed in the puzzle and which of them need help from the team. The competition against other groups can also show not only the success of one team member, but the progress of the whole team. After the document which gathers key requirements was done the project team prioritized them and began to compose the mechanics of the collaborative puzzle; keeping in mind the existing game structure and design as well as time limitation.

### 2.3.2. Collaborative puzzle

#### 2.3.2.1. The general idea of the collaborative puzzle

The idea in this puzzle is to create a puzzle that supports communication and collaboration among the players. A number of players meet face to face in a specific meeting point. Here they receive a puzzle which they have to solve together. The team is not being told exactly how the puzzle should be solved. They are given a list with different smaller tasks and they have to discuss which tasks on the list they think should be done to solve the puzzle and which one should not be done.

All the tasks on the task list ask the player to go to a specific place and get something and come back to the meeting point. After the players have done the tasks, they will see if the tasks that they chose to do have really solved the puzzle or not. If they have chosen the right tasks and solved the puzzle they all win a special item, if not they all lose. After the collaborative puzzle, players can continue their own mission.

#### 2.3.2.2. Puzzle mechanic

The puzzle mechanics was made up as a separate document describing the flow of the puzzle from player's point of view. Each step of the algorithms was supplied with priority showing how important is the realization of this step; involved parts, describing whether client logic, server logic or client-server communication are involved into the step. Additionally each step has attached screen from the developed prototype what simplify the understanding of the puzzle.

As the general flow of the game in which the player solves the puzzles was already described above, here the puzzle will be described more detailed. To understand the mechanics of collaborative puzzle written below it is obligatory to read the previous sections explaining the general idea and structure of the game. Also the full description of the puzzle can be found in puzzle mechanics document and collaborative prototype.

The player going from one puzzle location to another will be interrupted by a notification if he is within a certain distant to the collaborative puzzle. A Non Playable Character (NPC) pops up on the player's screen and proposes him to participate in the collaborative puzzle. It is up to the user to accept the invitation or to continue playing the started mission. If the user is interested in the collaborative puzzle, where he should interact with other players, he will receive all the necessary information about the place to go and meet with a future team. After arriving to the place in appropriate time the player joins the puzzle and meets the other participants, which helps to organize the team and break the ice between participants. After all team members are at the start location the non-playable character will confront them with a problem which they have to solve. To solve the problem they have to choose between smaller tasks to do on a task list. They can discuss and decide who will take which task and begin solving the puzzle. Generally for every task each player should walk from one physical location to another and bring one virtual item to solve the problem. The virtual items as well as locations differ depending on the puzzle's content. The aim of the players is to choose which items are the most necessary in the current situation, bring them to the start location of the puzzle and solve the problem using them. If the combination of the items which were brought to the start point is correct all the players get a special item as a reward, if not they all lose. Afterwards the collaborative puzzle should be considered to be finished and players can continue playing their own missions.

Later a developed design example will be shown to illustrate the puzzle mechanics. While developing the puzzle mechanics the team members succeed in deployment of most of the crucial elements for making the game collaborative, which were described in the beginning of the paper. Nevertheless the implementation of some of the elements is still under the question due to some of the results of the user evaluation.

#### 2.3.2.3. Puzzle technical details

The implementation of the collaborative puzzle has begun from the diagram describing the communication process between the client and server, see Figure 2.1. According to the architecture used in the project the server cannot request the information from the client. Therefore the client should continuously send the information about its location to the server. It needs additional programming and reorganization of the structure on the mobile part while existing puzzles don not need continuous data transfer. Also comparing to the developed puzzles it was proposed to split the algorithm of the communication into three steps. The first step is responsible for defining if the player is situated in the area of the collaborative puzzle and can accept the notification; the second requests the collaborative puzzle data; and eventually on the third step it should be checked whether puzzle is solved or not. Previously after receiving the puzzle data the mobile client could reason independently about the rightness of the solution, but after involving more players this function should be transferred on the server side. Future work can include the realization of the proposed algorithms as well as reorganizing the developed puzzle structure used on the mobile side. The server side also needs laborious work especially while creating the logic of the puzzle which involves multimode communication.

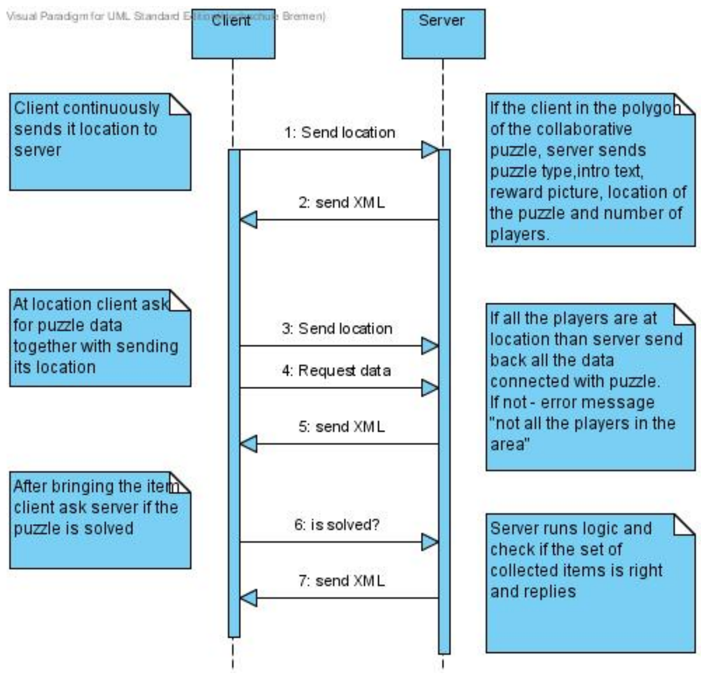

Visual Paradigm for UML Standard Edition(Hochschule Bremen)
Client
Server
Client continuously sends it location to server
1: Send location
2: send XML
If the client in the polygon of the collaborative puzzle, server sends puzzle type,intro text, reward picture, location of the puzzle and number of players.
At location client ask for puzzle data together with sending its location
3: Send location
4: Request data
5: send XML
If all the players are at location than server send back all the data connected with puzzle. If not - error message "not all the players in the area"
After bringing the item client ask server if the puzzle is solved
6: is solved?
7: send XML
Server runs logic and check if the set of collected items is right and replies

Figure 2.1.: Diagram showing data exchange between client and server sides.

#### 2.3.2.4. Design

The collaborative puzzle is a part of the whole game so there was no need to design everything new. To keep the same style and design like the single-mode puzzles many elements from the single mode where used in the collaborative puzzle. Of course for the content of the first example “Roland on fire” many new elements needed to be designed, but for the puzzle in general only two new elements had to be designed. The first one is a person symbol that illustrates how many players have arrived to the meeting point and how many are still missing.

On the screen (Figure 2.2) the player can see the number of players needed for the puzzle though the number of persons on the screen. The green person show players who have arrived to the meeting point and the grey person shows the players that are missing.

The second element that had to be designed was the task list (Figure 2.3). To make it easy for the player to understand the tasks the task list was made very simple with sentences one after the other. In the end of each sentence there is a little checkbox for the players to choose a task.

### 2.3.3. Example: Roland on Fire

In this chapter an example for how the collaborative puzzle could look like will be presented. The content of this collaborative puzzle is connected to the mission Old City of

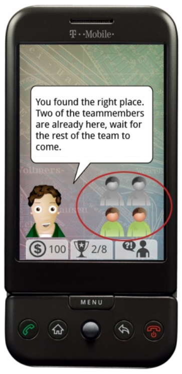

You found the right place. Two of the teammembers are already here, wait for the rest of the team to come.

Figure 2.2.: Person symbol

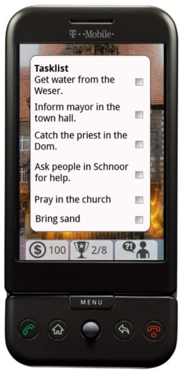

Tasklist
Get water from the Weser.
Inform mayor in the town hall.
Catch the priest in the Dom.
Ask people in Schnoor for help.
Pray in the church
Bring sand

Figure 2.3.: Task list

Bremen. It makes the player familiar with a historical event in the 1400 century where the wood Roland statue was burned.

The example here is only a rough explanation, because a detailed explanation with all screens would be too long. In order to get the notification about the game, the player should be on the way from the previous finished puzzle to the next one. If the player is within 300 meters from the collaborative puzzle she will receive this notification telling her about the collaborative puzzle (Figure 2.4).

Player can either accept it or refuse it. If the player refuses to play the puzzle he simply continues his own mission, but if the player accepts he will be asked to go to a meeting point. In this example the player has to go to the Roland statue on the market square in the city to meet the rest of his team.

When the player has arrived to the Roland Statue she can see how many players are there and how many still need to come. This information helps her find her team members. In this example four players are needed for the puzzle, two of them have arrived and two still need to come (Figure 2.5).

After all players have arrived the puzzle begins. In the beginning of the puzzle the players see a screen with the Roland statue burning. Then the NPC pops up on the

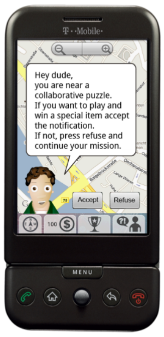

Hey dude,
you are near a
collaborative puzzle.
If you want to play and
win a special item accept
the notification.
If not, press refuse and
continue your mission.

Figure 2.4.: Notification

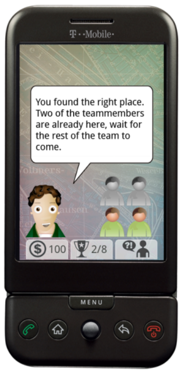

You found the right place.
Two of the teammembers
are already here, wait for
the rest of the team to
come.

Figure 2.5.: Meeting point found.

screen and tells the back-story of this event and explains the problem that the team has to solve. Then the players must put the fire out and save the Roland (Figures 2.6 and 2.7).

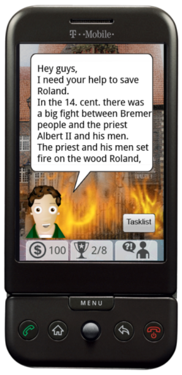

Hey guys,
I need your help to save
Roland.
In the 14. cent. there was
a big fight between Bremer
people and the priest
Albert II and his men.
The priest and his men set
fire on the wood Roland,

Figure 2.6.: Puzzle explanation 1

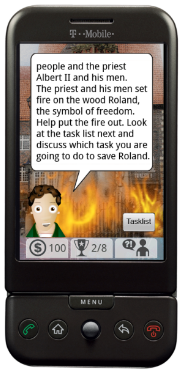
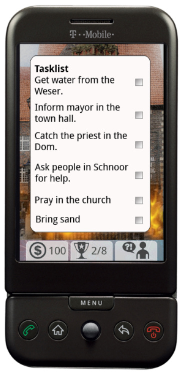

people and the priest Albert II and his men. The priest and his men set fire on the wood Roland, the symbol of freedom. Help put the fire out. Look at the task list next and discuss which task you are going to do to save Roland.

Figure 2.7.: Puzzle explanation 2

After this explanation a list of tasks which the players can do to put the fire out will be showed. Each player selects the task that she wants to do by choosing the checkbox next to the task (Figure 2.8).


**Tasklist**

- Get water from the Weser. ☐
- Inform mayor in the town hall. ☐
- Catch the priest in the Dom. ☐
- Ask people in Schnoor for help. ☐
- Pray in the church ☐
- Bring sand ☐

Figure 2.8.: Task list

The NPC explains the player where to go and how to get the item. When the player arrives to the right place she receives an item. In this example the player chooses to get a buck of water from the Schlachte to put the fire out (Figure 2.9). It is then possible to get information about the Schlachte or to skip this information and go back to the meeting point (Figure 2.10).

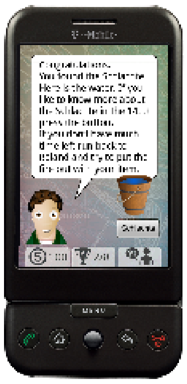

Congratulations.
You found the Schlachte.
Here is the water. If you like to know more about the Schlachte in the 14th c. press the button.
If you don't have much time left run back to Roland and try to put the fire out with your item.

Figure 2.9.: Item found

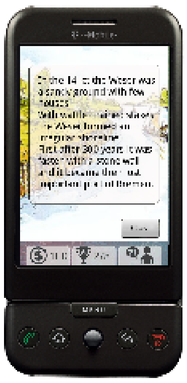

In the 14th c. the Weser was a sandy ground with few houses.
With [illegible] the Weser formed an irregular shoreline.
But after 300 years it was faster with a stone wall and it became the most important part of Bremen.

Figure 2.10.: Information about Schlachte

When the player comes back to the meeting point she can use the bucket of water to put the fire out (Figure 2.11).

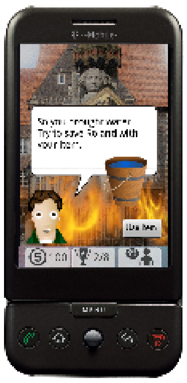
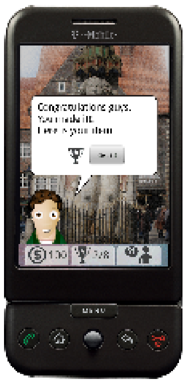

So you brought water.
Try to save Roland with your item.

Figure 2.11.: Use item

If the item was useful the fire becomes smaller, if not, it does not change. In this example the bucket of water helped putting the fire out. The player can then choose another task from the list or wait for the rest of the team to come.

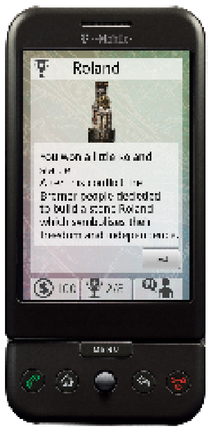
After all the players are back with the items they will see if the items all together put the fire out and saved Roland or not. In this example the items put the fire out and Roland was saved (Figure 2.12).



Congratulations guys.
You made it.
Here is your item

Figure 2.12.: Won the puzzle.

The team members all win a little Roland statue followed with some information about it (Figure 2.13).



Roland

You won a little Roland statue.
After this conflict the Bremer people decided to build a stone Roland which symbolizes their freedom and independence.

Figure 2.13.: Reward.

The collaborative puzzle ends here. After that the players can continue their own mission and play single-mode puzzles.

### 2.3.4. Final Considerations

The development of the collaborative puzzle includes interdisciplinary fields of study such as psychology and pedagogy together with programming and design. In the preparatory steps towards the implementation of such puzzle the appropriate research in the field of collaborative games creation was done. It allowed understanding the most important elements which make any game collaborative and use them later during the design phase. During the design phase the necessary algorithm was formed explaining step-by-step user's actions and the logic of the puzzle which should be programmed on the server side. Additionally the clickable prototype was drawn using available free web tools. The developed prototype offers a common ground both for designers and programmers and simplifies further work. In spite of the design phase being finished the technical implementation has not yet been started. Proposed way of server-client communication and the logic algorithm should simplify programmer's work but does not exclude possible difficulties. These difficulties can influence some steps of the algorithm but nevertheless will not change it drastically. The general idea of a collaborative puzzle was proposed to the testers during the evaluation phase and, although the idea of meeting face-to-face with other players generated some concerns, it was overall well appreciated which proves its potential.

## 2.4. Learning Orientation [22]

[22]Nesrin Abdelrazik, Jana Wedekind

> "*Mobile applications can vary continuously because of changing circumstances and different user needs (Tarasewich, 2003). The object of mobile learning design is inseparable from the design, the context or activity of use.*" (Uden, 2007)

One of our research questions when defining the concept of our project was: How can learning be integrated in a meaningful way and in such a manner that it will be unconsciously for the player? The most prominent theory that we used to approach this problem was the Activity Theory, which will be described in detail later. Further, the learning objectives will be named as well as the process of production will be highlighted. The structure is:

- Learning in Games
- Five levels of learning
  - Learning theories
  - Why do we need learning objectives?
- Learning Objectives
  - General learning objectives
  - Concrete learning objectives

### 2.4.1. Learning in games

In order to define the learning objectives, it was very helpful to research on how players learn in games. This helped us defining learning objectives that are interesting for the player and in the same time effective in the learning process. The main factor for effective learning is the motivation. That is why our first question was: "What motivates young people to play computer games?" When they were asked this question these were the qualities that were mentioned in interviews: "nice presentation", "good storytelling", "things happen", "I like competition", "I like action", "I can test my skills", "I can be somebody else" (Pivec et al., 2004).

From these statements we could conclude that the ability of games to engage and motivate player is directly related to the design and the quality of the content of the game. The challenge in our game was to create a high quality game that in the same time integrates learning. According to Prensky (Prensky and Thiagarajan, 2007) learning occurs over five levels in video and computer games.

#### 2.4.1.1. Five levels of learning

**Level 1: Learning How**

The basic thing that a player learns from a game is how to do something. The skills that the player learns can be defined from the beginning in the learning objectives, but sometimes they may extend far beyond the context of the original application. For example in our game the player can use a compass to find a specific place. In the original context it is only a tool, but in practice the player will learn how to use it.

**Level 2: Learning What**

In the second level the player learns what he or she is going to do. The player here learns the rules of the game needed to play successfully. The nature of computer games provides for a repeated “trial and error”, because it is faster even by complex rules. The rules in our game are designed to be easily understood by the player. This allows the player to get started with the game fast. In this level the learning outcome in our game will not be very high.

**Level 3: Learning Why**

Level 3 deals with the particular strategies and tactics that the player develops while playing. In multiplayer games one of the strategies is to understand, negotiate and deal with other players. Since our game is supposed to be collaborative, it will contain puzzles where a player can collaborate with another player if they are in the same area in order to achieve better results. This is one of the strategies that the player can use in our game to complete the game successfully.

**Level 4: Learning Where**

In this level the player learns about the world in which the game is set. Mostly it reflects the values and culture of its developer. In our game this level of learning is very apparent, because the maps that are developed in the game reflect the world of its creator. The first map that the team created is about the history of the city Bremen. This map reflects the world of the team. Since most of them are exchange students and new in Bremen they are interested in the history of Bremen.

**Level 5: Learning When/Whether In the last level of learning**

Prensky considers how games influence the player to make value-based decisions of *right and wrong* in the real-world environment. At the current status of our game this sort of learning is not very presented, because the content of our map only gives facts that do not present any specific views that could influence the player’s decisions. But maybe some of the maps that will be developed by other players in the future may have more of this sort of learning.

### 2.4.2. Learning theories

When developing a learning game it is important to follow a learning theory that provides you with conceptual framework. The three main learning theories are behaviorism, cognitivism and constructivism. The theory of Behaviorism focuses on objectively observable behaviors (Phillips and Soltis). It defines learning as nothing more than the gaining of new behavior. Cognitivism on the other hand looks beyond behavior to explain brain-based learning. "*The cognitivist paradigm essentially argues that the 'black box' of the mind should be opened and understood. The learner is viewed as an information processor (like a computer).*" (LearningTheoriesKnowledgebase, 2010) Last we have constructivism who states that humans learn and construct knowledge out of their own experiences. "*The learner is an information constructor. People actively construct or create their own subjective representations of objective reality.*" (LearningTheories-Knowledgebase, 2010) So the learning here is a process where the learner actively creates new ideas. In our game *StreetDroids* we chose to follow the theory of constructivism. The game encourages the player to explore different places by physically walking around and solving puzzles on the mobile device. Players will get some information about a place which she has to find. This is the first step where she learns something about the place. After finding the place, she has to solve a puzzle that also contains some interesting facts about the place. To support the player by solving the puzzles the game offers several hints. According to constructivism learners are motivated to learn when they experience successful completion of challenging tasks. That is what these hints are for. They help the player to successfully complete the challenging puzzle and motivate him or her to continue the game.

The theory of constructivism also calls for collaboration among learners to support the learning process. In *StreetDroids* we developed some puzzles that encourage the player to collaborate with other players. By solving these puzzles, the player will get some special rewards.

### 2.4.3. Why do we need learning objectives?

A successful game needs clear goals for the creator as well as for the player. For that we had to define the learning objectives of our game from the beginning of our concept design. "*Learning objectives are statements that describe significant and essential learning that learners have achieved.*" (TeachingSupportServices, 2003). In other words, learning objectives describe what the player will know and what he or she will be able to do after playing our game. They basically answer to two questions: Why should the player play our game? and What can the player learn from our game?

"*Learning objectives refer to observable and measurable: knowledge, skills, attitudes.*" (TeachingSupportServices, 2003). In the first scenario of *StreetDroids*, the Old City of Bremen, the knowledge that the player will learn are the historical facts that will be integrated in the game. The skills that will be achieved are for example: learning how to play with a mobile device and how to use a compass. Through the collaboration with other players the player may acquire new social attitudes.

When writing the learning objectives we tried to use the acronym SMART (Teaching-SupportServices, 2003) which stands for Specific, Measurable, Achievable, Relevant, Time-based. Of course it was not always possible to define the learning objective so specific and exact like the SMART suggest, but it was good to keep these words in mind while were writing learning objectives.

### 2.4.4. Learning Objectives

#### 2.4.4.1. General learning objectives

*StreetDroids* is designed to be a context-aware mobile learning game, which means that by playing the game learning objectives can be achieved. The modular setup of the core mechanics will allow the users to develop their own game ideas and make them available to other players as missions. That also means that it can not be foreseen if everyone will try to include learning objectives within a mission. It is possible though, as will be demonstrated with the map for the city of Bremen, to include learning objectives in a mission (see also 2.4.4.2 Concrete Learning Objectives). Offering example missions right from the beginning will help the players to understand what is possible with the given concept.

Mobile Learning (m-learning) (Sharples, 2007) is part of e-learning and therefore we can fall back on some of the learning paradigms presented in e-learning. Nevertheless there are some features specific to m-learning, such as the mobility aspect offered by mobile devices. Learning with mobile devices has a strong focus on the learner (human centered) and gives them the possibility to learn within a given environment (situated learning).

In the following the general learning objectives that apply to the whole game mechanics and not only to explicit missions are presented. These objectives are defined as follows[23]:

1. *Players can locate visited places on a map and connect them to the real world. Further, they can use a compass for navigation*.
   The game allows the player to learn more about navigation, as he has to find the location for the puzzles. He can use a compass in order to facilitate the search, but therefore he needs to understand the concept of a compass.

2. *Players are enhancing their soft skills.**
   Soft skills are trained by demanding skills from the player as problem solving, communication, working in groups, and so on. This is important as nowadays these skills are gaining more and more value as job entry requirements.

3. *Players are improving their communication skills*.*
   It is important to learn to communicate and work with others in today's society as never before. All is about teamwork and how one can interact in these situations. The game is trying to help the player to learn to communicate with other people, even if she does not know them well, for example by creating the community, where the players can exchange information. Conversation(al) theory [Sharples, 2005] describes that the process of sharing information, of having a continuous communication, helps the learner to achieve a better understanding of the learning objectives.

[23]The * marked bullets are adapted from (Klopfer, 2008).

4. *The game allows players to collaborate with each other.**
   The game mechanics are open enough to allow different ways of playing the game. If, for example, a map for a school class is created, the class can be divided into teams, that play against each other. Within a team, the players can help and support each other in the best possible way. They can exchange their knowledge in order to achieve their common goal, to win the game, as fast as possible.

5. *The game is context-aware, which allows the player to implement scenarios for different locations and context.*
   *"Many scholars agree that learning is most effective when it takes place as a collaborative rather than an isolated activity and when it takes place in a context relevant to the learner."* (Kofod-Petersen and Petersen, 2008)
   Designed as a community game that can be expanded by the players themselves, the location of the game is not fixed by the game mechanics. This change of location allows the players to implement the game in different context that are meaningful for their story or puzzles. The learning objectives therefore can be combined with relevant location and context, as can be seen with the exemplary map for the history of Bremen. The learning actually takes place at the location that is asked for. If for instance the player is supposed to learn that the town hall of Bremen is on the list of *UNESCO World Heritage Sites* he stands in front of the town hall and can examine it. Thereby he has the chance not only to solve the puzzle, but to experience the building on its own. Different to a picture in a book the learner can form his own view by discovering the building.

6. *The game encourages players to create their own maps and characters (user-generated content) / The game encourages players to be an active part of the community.*
   Players receive the possibility to become an active part of the community by creating their own maps and characters, to enforce the identification with the game, respectively the community. Creativity is encouraged thereby, as well as the contention with other players.

7. *The game improves the player's ability to solve problems.**
   By offering different puzzle types to the players, their ability to solve a problem is improved. Further, a puzzle type can be applied to different context, as mentioned above, therefore the player is encouraged to apply her knowledge of solving a type of puzzle to a different scenario.

8. *Players are qualified to work in groups*.*
   Players are encouraged to communicate with each other within the game. Although not being a requirement to the game itself, this allows players to share their knowledge and helps them solving problems.

9. *Players are applying inductive reasoning*.*
   The players develop skills to solve complex problems in a non-linear way and also

some of them require a deep understanding of systems and processes.

10. *Players are applying* s*ustained reasoning.**
    This objective means, that the game will support to solve problems that are ongoing and use a variety of resources to do so. The player can for instance try to solve a puzzle by talking to other people, by researching in the Internet, or maybe fall back on knowledge already gained. Further, it will be possible to save the game in between and go back to an already started mission.

11. *Players are prepared to organize and navigate information structures and evaluate information.**
    As mentioned in item 10 already, the learners must collect and evaluate information from a variety of sources. This helps them to deepen their understanding of the task and to support sustainable learning.

#### 2.4.4.2. Concrete learning objectives

The concrete learning objectives apply to specific puzzles in the game. They were implemented exemplary for the map of the city of Bremen. The learning objectives are based mainly on the activity theory, that will be explained with the help of examples from the game.

*"Activity Theory provides a theoretical language for looking at how an educational game or resource mediates players' understandings of other phenomena while acknowledging the social and cultural contexts in which game play is situated."* (Squire, 2006) A theory which has a strong focus on the context the learning occurs in, as well as on communication, is activity theory. Originating in the cultural and historical psychology of Vygotsky and Leont'ev, it "*includes collective activity, community, rules and division of labour that denote the situated social context within which collective activities are carried out*." (Uden, 2007) The five basic principles of activity theory are:

1. Hierarchical structure of activities
   "*As in situated learning or a constructivist environment, cognition is defined and shaped by its relation to a given context. This means that we must not only learn in context, but also by context.*" (Snow, 1994) Also Jonassen describes that "*In activity theory, activity is a precursor to learning. Knowing can only be interpreted in the context of doing.*" (Jonassen and Rohrer-Murphy, 1999) Therefore all learning occurs while actually playing the game. Only with the context of the game the puzzles make sense and are solvable. Given the example of the map for the Old City of Bremen this means that a puzzle needs to be solved in a specific location (context) in order to fulfill the learning objective. This could be for example a puzzle located at the St. Petri's Dom in Bremen, where the objective is to explore the history of the cathedral by comparing the first cathedral with the current one.
   Activities are "*(...) consisting of a subject and an object, which are mediated by a tool. A subject can be an individual or a group, engaged in the activity. An activity*

*is undertaken by a subject using tools to achieve an object (objective), thus transforming it into an outcome.*" (Kuutti, 1996) The division of an activity can be seen in figure 2.14, which is displayed as an Engström's activity diagram (Uden, 2007).

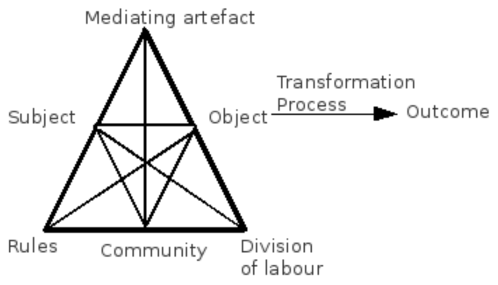


Figure 2.14.: Structure of an activity

Adapting the shown structure of figure 2.14 in the Engström's diagram for the game would have the result seen in figure 2.15.

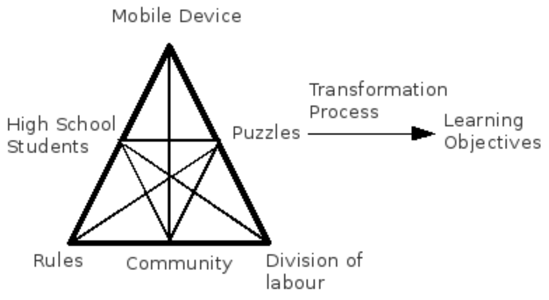


Figure 2.15.: Game structure in an Engström activity diagram.

Every mission in *StreetDroids* consists of several puzzles that need to be solved in order to complete a whole game. In each puzzle the players needs to undertake, they have to become active. In the game, the mobile device occupies the role of the mediator respectively the tool, that supports the players. The subject consists of one or more players, which will use the tool in order to achieve the learning objectives.

2. Object-orientedness
"*This principle specifies the activity theory approach to the environment with which human beings are interacting. Unlike Piaget and Gibson, activity theorists consider social and cultural properties of the environment to be as objective as physical, chemical, or biological ones. These properties exist regardless of our feelings about them.*

*So human beings live in an environment that is meaningful in itself. This environment consists of entities that combine all kinds of objective features, including the culturally determined ones, which, in turn, determine the way people act on these entities.*" (Kaptelinin, 1995)

3. Internalization/externalization
   "*Activities can be either internal or external but they need to be analyzed together for a proper understanding to be achieved. Internalization relates to the human being's ability to imagine, consider alternative approaches to a problem, perform mental simulations. Externalization transforms an internalized action into an external one.*" (Hardy, n.d.)
   Taking as an example the puzzle about the 15 windmills surrounding Old Bremen. The learning objective is to understand that surrounding the city with windmills used to be a tradition in the area. After solving the puzzle, an old map of Bremen is overlaid by the new one to display the changes in size and shape that Bremen has gone through. In order to solve the puzzle the player has to play through several scenarios (internally), before he decides which placement of the mills will be correct. By placing the windmills on the display, the internal chosen scenario gets externalized.

4. Mediation
   "*Human activity is mediated by artefacts – tools both internal and external. These tools may be signs, language, instruments or machines.*" (Hardy, n.d.)
   As mentioned before, the mobile device occupies the role of the mediator, that supports the player in achieving her goals and therefore reaching the learning objectives. It provides the necessary information to solve the puzzles and play the game. Further, in certain situations the player is able to contact other nearby players with the help of the mobile device.

5. Development
   "*According to activity theory, to understand a phenomenon means to know how it developed into its existing form. The principle of development gives an opportunity to conduct thorough, scientific analysis of complex phenomena while avoiding mechanistic oversimplifications.*" (Kaptelinin, 1995)

As can be seen, the activity theory is very appropriate for the implementation of the game's learning objectives. Its focus on the learner being an active part fits better than other traditional learning theories that are more directed towards computer-mediated learning. Further, it takes into account the changing context of the game, which is one of the main requirements.

## 2.5. Content Choice and Use[24]

[24] Dema El-Masri, Catalina Payán

The applied content to *StreetDroids* is, as explained later in detail: the history of old Bremen. Yet other contents can be applied ranging from sports to shopping centers and shops, based on the location, environment and the availability of potential content that can be used. A sports-related game could be for example locating football stadiums, tennis courts, sport museums, sports bars, or other possibilities and playing different challenges in these locations. Each challenge leading to another until the mission is solved. The main idea of the game is to solve a number of puzzles in one mission at least, each puzzle then leading the player with location-related clues to the next location and so on. This customizable content creation function is facilitated by the web end, explained in more detail in section 2.2.6.2of the report. The web platform of *StreetDroids* presents what we like to call as a "toolbox" for the creation of further content. There are editors for the puzzles, the characters, and the missions. Each one of these editors allows players to upload any content they want, thus creating customized puzzles, characters, and missions, which includes customizing everything related to the puzzle of choice as items to be collected after succeeding in the puzzle. This aspect of creating the players' own content in the game is also recognized by game developers. Cary Rosenzweig, president and CEO of IMVU in (Nutt and Kumar, 2008) states, *"Of most people who create items, most of them create one or two items. People do what they do mostly for affirmation; because being a creator is cool and they like the status – it's their choice if they want to turn the credits to cash."* He also goes on to say that *"Amateurs are sometimes better than professionals because they're closer to the truth, to the social reality"*. For such reasons, it is an advantage to have users participate in the content creation of games. It enriches the game experience by offering the player with many choices of content, and also offering an opportunity for the player to make his own content.

### 2.5.1. History as Content

The choice of content was not a simple one. The game is a location-aware game, and therefore can be played almost anywhere and use content available from the surrounding environment. Despite the vast amount of potential contents that can be applied to the game, the team decided to apply content from our nearby surrounding environment, that being the Old City of Bremen. The team resides in this city which is a rich city in its history and importance in Germany in general, and in North Germany specifically. Development of games also reflects interest in the surrounding culture, and according to the fourth level of learning by (Prensky and Thiagarajan, 2007), games mostly reflect the values and culture of its developer. Research was conducted for that reason on the numerous historical buildings, statues and objects in the city to which the game's concept could be applied, and could be puzzles created. A content team was formed and designated to do research about the city's history and apply this content into the game and the mobile's context.

Though it is not a traditional learning game, learning aspects can be applied to it. This is something we sought to achieve when applying the content of Bremen to the game. The content presented us with a good opportunity for mixing game play and entertainment, together they can be used as a trigger for history education. Learning history through a game makes it a lot less likely to be forgotten than from a book (Blecic et al., 2002). Presenting information in such a way makes learning a more engaging process; also the mobile device in use is a device most people are accustomed to and does not require special skills to operate, especially amongst young adults.

To understand history in a better manner the learner usually has to imagine how the world was hundreds of years ago. This process is not easy just by being based on facts and dates from books, as it is in most school systems -where the past is seen through the lens of the present. Introducing a tool as the mobile device which allows for a more personal way of approaching knowledge, as well as animations, Role Playing Games (RPG), illustrations, audio and video related information, the process of learning will become easier.

Blecic et al.(Blecic et al., 2002)stated the main advantages of teaching/learning history through play, these advantages can also be found in *StreetDroids*, and are explained in the following. Play is known to be fun, and what is learnt through fun activities is less likely to be forgotten than what is learnt through traditional means. This is an aspect we incorporated in the game and content design, through designing the content in a more-approaching and engaging manner. An example of that could be challenge; many players see more fun in a game when it challenges their skills and information -being a learning game, or having learning aspects applied to it- and that plays a positive role in building up elements of fun, enjoyment and challenge. Play usually requires different skills to be deployed. In the case at hand such skills include coordination, precision, historical knowledge, and the use of the technology. The project started with an aim to have a collaboration part, where players can compete or collaborate, for time and technological constraints this aim has not been yet fulfilled, but the concept of it was developed. Nevertheless if developed in the future, collaboration and other skills as conflict management will be encouraged in game play.

According to our research, only few examples exist of implementing history education via mobile devices in a game-based learning and context-aware manner. The project *Frequentie 1550* (Akkerman et al., 2008) developed by the Waag society is an example. The game's purpose is to teach secondary-school students about the history of Amsterdam by walking through it, completing game assignments and communicating with other team players.

Another project is *Una giornata di Gaio ad Egnathia* (Gaius' Day in Egnathia). The paper version of this game showed that playing historical roles motivates students. This game is structured as a scavenger hunt game, with the players solving the case, exploring the area and discovering the hidden secrets in Egnathia, an ancient city in Apulia the southern region of Italy which dates back around 1000 BC (Ardito et al., 2007).

Both projects have been tested with many school students and produced positive results among students and teachers. The game developed in the MobileHive project is very similar to the examples above, because it uses history as content, mobile devices as tools, location-awareness, it applies learning aspects, and players play historical roles in it.

Bremen's history goes back approximately 900 BC, which covers a lot of historical events. In our history research we discovered that some of the interesting places have gone through many different incidents and its attributes have been changed several times in history. For example the church St. Peters Dom, goes back to the year 810 (Dappen, 2008), and has been rebuilt several times changing its form and making it much different than it was in the past. With such an example the goal would be to clarify these changes and make them evident to the player in an engaging manner.

In order to cover the main incidents in the city's history and maximizing the potential learning outcome, the narrative concept of time traveling was developed. We divided the history in four periods, (9-12 centuries; 13-16 centuries; 17-19 centuries; and 19-20 centuries) and for each period of time we developed a route that contains 5 to 8 historical places for the player to visit. Each route will have a guiding character that helps the player though the puzzles and the travel. The player can choose from these four characters, each one has been a famous figure from the past in Bremen and has played an important role in shaping its history. The characters combined cover the whole history of the city, and each character takes the player through a route of historical places where the user will receive challenges to solve puzzles in different locations related to the character's time era. Towards the second semester of the project, the content's structure was changed. Instead of having four time eras with four characters, eras were combined resulting in two missions. The first two eras were combined to become one mission, as the last two became another mission as well. This was done for different reasons, such as having faced a number of technological difficulties and time constraints during that period. Making these changes allowed the team to focus on solving these technological problems and delivering on time, rather than having an excess amount of content not applied, therefore the content had to decrease in size.

### 2.5.2. Applying content to the puzzles

Mobile social games use the interactions made between players and their relationship with the physical world to provide an entertaining experience (Casey et al., 2007), and at the same time they reveal how users socially interact when they use the devices. *StreetDroids* is a task-based play game, in which the players can create their own content in a non-linear manner, developing a whole new experience, with players exploring and making use of their environment and its attributes. This content is used to promote what is called a mobile "information encounters". A distribution of interactive multimedia digital content designed for social networking and community content sharing, as well as for entertainment and commerce (Churchill et al., 2004).

As it is explained above *StreetDroids* is a mobile game where the users can create and apply whatever content they want in order to explore or interact with their surroundings, because the game can be customized the user can interact with almost every feature the game can offer, including maps, locations, missions, puzzles hints, rewards and characters.

However there are some restrictions and guides in order to apply the content, because not every content its 100% suitable to every feature of the game.

**Location:**

Because it is a GPS based game, *StreetDroids* can only be played outdoors, therefore it is recommended to play in an open space. In addition the battery life is another component the user has to keep in mind; as a result it is better to have the locations for the puzzles near but not next to each other because the GPS is not precise enough and the application can mix the locations and display a specific puzzle in a wrong location.

**Location hints:**

The location hints are the ones leading the player to the next puzzle in a new location. It is similar to a scavenger hunt; after the player finishes a puzzle he must go in the direction of a given hint which will lead him to the next task. These hints require the player to pay attention to his surroundings. As these hints are usually related to buildings, streets, and sights it is recommended for the content creator to know the place where the puzzle is going to be placed because he will have to explain in a narrative way where to find the next location.

**Puzzles:**

Puzzles are the challenges the player will go through once he arrives to a specific location of the map. The puzzles developed for the *StreetDroids* game can be reused by applying different content every time. Despite the fact that the puzzles are made for the android mobile device they are quite diverse, making use of the many capabilities of the device and implementing many puzzle concepts most potential users are familiar with. Nevertheless the puzzles have to cope with the android properties, screen size, interaction type, and network connection.

The challenge of developing puzzle content is to combine entertaining challenging puzzles with interesting facts, and at the same time consider the capabilities of the technology in this case the android mobile device. At the moment the *StreetDroids* game has three puzzles available:

- Hotspot puzzle:
  In this puzzle the player is presented with a location task and by touching the screen she has to find in an specific object in the image. It is recommended on this puzzle to look for tasks or images that have many possibilities for the user to click on, it is important for the image to be as wide as possible that means general view without close ups, etc, in order to make finding the correct object more challenging. However it is also advisable to pick objects as big as the finger print because of the small screen of the device the small objects can be hard to touch and find.

- Image recognition puzzle:
  In this puzzle the player is presented with a given picture with a blank space on it, and he has to fill in the missing piece by taking a picture with the mobile camera. In this puzzle the given picture for the task has to be taken by the creator of the puzzle with the android camera, in order for the player to take the exact same

image, it is also important to pick an image not far away from the player because of the zoom restrictions. The selected image for the content of the puzzle must be easy to crop and it is advisable to only take a detail of the entire object, in that way the puzzle turns into a more challenging experience for the player.

- Drag and drop puzzle:
  For this puzzle the player has to recreate a picture with some given parts. This is a variation of a jigsaw puzzle, the player will find the pieces of the image in the top of the screen and she has to drag and drop them in the correct location. For this puzzle it is recommended to work with an image that can be split in many parts, therefore it is better to work with detailed images like a façade where all the part are looking similar and it is difficult to tell the difference between them, as a result the player has to reconstruct the image according to what she is looking at nearby her location.

**Puzzle hints:**

Sometimes the players cannot solve the puzzle, for this reason, and to ensure that the player can find the next location, hints are given when the player does not succeed for the first, second, and third attempt to finish the challenge. These clues do not only help the player finishing the puzzle but also draw the player's attention to the many details surrounding the challenge. For the puzzle hints it is recommended to be as specific as possible without giving the answer, otherwise the player will just look for the hints as solutions without paying attention to the challenge.

**Rewards (items):**

At the end of every puzzle the player will be presented with an item as a reward, that confirms of her success in the given task, this reward can be something associated with the specific puzzle or with the entire mission.

## 2.6. Documentation [25]

[25] Cristina Botta, Jana Wedekind

## 2.6.1. Documentation

Every project needs documentation. Usually it is not the most beloved part of a project, nevertheless it is necessary and helpful. For the development of our game three things were found crucial: To model a vision of what we want to achieve with the master project, to define the game mechanics, which are the core of each game, and to have a way of organizing features and priorities. Therefore three documents were prepared in order to support all project members in internalizing the goals of the project and achieve a common understanding of the game.

### 2.6.1.1. Vision Document

For the project a vision document was needed, in order to define the most important topics and the vocabulary needed in the development of the project. The most common vision documents originate from the Rational Unified Process (RUP)[26]. The RUP is an iterative software development process developed by IBM. It is more like a framework that can be adapted to the different requirements in software projects. It provides process models, role descriptions, document templates, and software (e.g. Rational ClearQuest, Rational Rose, and more) for software development. The vision document in the RUP is a general vision of the core project's requirements, key features, and main constraints. It is used to achieve a common understanding of project goals between all project members (project managers, customer, developers, and designers). In the RUP documentation the vision is described as follows: "*Defines the stakeholders*[27] *view of the product to be developed, specified in terms of the stakeholders key needs and features. Containing an outline of the envisioned core requirements, it provides the contractual basis for the more detailed technical requirements.*" (RUP, 2003b) Usually the vision is directed also to the customer/stakeholder of the project. As we did not have them, we focused on the description of our project in order to develop a common understanding of what we wanted to achieve in the project and how we wanted to do it. It was used as the main reference for all team members in order to reflect on our goals for the end of the project. The document helped to build a common language between team members to refer to the game concepts. Further, the vision helped us to present our idea to possible customers or external persons that were not involved in the development process. Using the vision, we described the state-of-the-art and then proposed how we wanted to differentiate ourselves from it: *what were the innovative parts in our game?* Moreover, a formal description of the technical requirements was included.

[26]`http://www-01.ibm.com/software/awdtools/rup/`

[27]*"The Stakeholder role is responsible for representing an interest group whose needs must be satisfied by the project. The role may be played by anyone who is (or potentially will be) materially affected by the outcome of the project."* (RUP, 2003a)

#### 2.6.1.2. Game Mechanics Document

"*A game mechanic is simply any part of the rule system of a game that covers one, and only one, possible kind of interaction that takes place during the game, be it general or specific. A game may consist of several mechanics, and a mechanic may be a part of many games.*" (Lundgren and Björk, 2004)

The purpose of the game mechanics document was to define the game mechanics and make them accessible to the whole project group. Within the document it was described how the game will work and the approach taken to achieve meaningful gameplay. By writing down the rules of the game misunderstandings and contradictions can be discovered and solved. The document itself was also subject of an iterative process, as rules were discussed, changed and if necessary dismissed again throughout the whole conception phase.

#### 2.6.1.3. Product Backlog

To help organize the tasks and priorities in the project a document called Product Backlog was used. This document is originally part of Scrum[28], which is an iterative, incremental framework for agile software development (Wikipedia, 2010a).

The Product Backlog is the master list of all functionality desired in the product, so it is a document for the entire project. It contains broad descriptions of the required and wished features divided by priority (Very High, High, etc.), rough estimates of development effort, and value for the project. This document is open and editable, because as more is learned about the product and its customers, updates and changes will be made to it during the time of the project.

The Product Owner prioritizes the tasks in the Product Backlog and describes the top items to the team. The team can then determine together which items can be completed until the end of the sprint. These estimates help the Product Owner to suggest a timeline and, to a limited extent, the priorities. Conceptually, the team should start at the top of the prioritized Product Backlog list and draw a line after the lowest of the high priority items they feel they can complete. In practice it is not unusual to see a team select, for example, the top five items and then two items from lower on the list but that are associated with the initial five. Or if the *add user profiles* and *add tagging support* features have the same business value, the one with the smallest development effort will probably have higher priority (MountainGoatSoftware).

[28]Scrum will not be discussed on this report as it was not used in the project.

# 3. The implementation

## 3.1. Basics

### 3.1.1. Mobile Platform Selection[1]

[1] Alexander Tyapkov

The mobile platform plays a significant role in our project, and it influences the further steps connected to the choice of architecture and programming methods, as well as work division between the members of the team. According to the concept puzzles are the core of the game, and can be filled with any appropriate content and be played at any location. It was clear that the user should have a mobile device. Nevertheless no additional information was provided, making the choice of the mobile platform difficult. During the initial phase of the project there were around five platforms present on the market, each with differences in availability and functionality. Among them are: Android Cupcake developed by Open Handset Alliance, Blackberry OS 4.7 by Blackberry, iPhone OS 3.0 by Apple, S60 5th edition by Nokia, Windows Mobile 6.5 by Windows and Palm WebOS proposed by Palm. The goal was to analyze the functionality, availability, and price of the existing platforms, as well as current trends showing the direction of the platform's market development, and eventually choose the mobile platform that would best fit the the project's needs.

#### 3.1.1.1. Criteria definition

Before selecting the platform a requirements analysis was performed. This helped to define the criteria and understand what features of the mobile platforms were the most important for the realization of the project. These criteria were based on the requirements collected from the project's vision document (section 2.6.1.1).

**Input methods**

In order to implement a context based game a variety of input methods should exist. By context, we can broadly understand all the information, which can be important during the game play and which can influence it. The logic of the game should process the information from the user's input or from sensors and check the ways it influences the application. Almost all modern devices and mobile platforms support a variety of sensors such as, for instance, GPS receiver, accelerometer and compass, which can be used to gather data. Indisputably that location of the player is the most widely used type of context information and has been implemented in thousands of applications. We also considered the location of the user as a crucial feature which can help to create an innovative game. Nevertheless we do not exclude other types of sensors such as the accelerometer, a compass or even a thermometer, which could become the key feature in a puzzle.

**Machine-user interaction**

In the vision document and project's description the communication between players was defined as a major component of the game. Players can interact in the real world by speaking and exchanging knowledge, or can communicate through the game using the proposed collaborative features. The third possibility to consider is when one device is used by two or more players to solve puzzles or to complete the mission. In this case players can interact either using the mobile device or by direct communication.

All the scenarios of user-machine interaction or even user-user interaction including a mobile device should be considered in order to make an attractive game. For these reasons appropriate hardware and software is needed to make the interaction clear for all the players. Nevertheless it should be noticed that the problem of machine-user interaction can be solved not only by deployment of innovative methods proposed by manufactures but also by joint laborious work of programmers and designers.

**Team knowledge level**

The *StreetDroids* project is educational and has limited human resources. All the participating students have versatile academic and programming backgrounds. Interviewing among the programmers showed that the majority has experience in the Java language. This defined the preferable programming language for the mobile part of the project, but it did not exclude other programming languages, as project members could still learn the necessary skills. It just shows that the mobile platform should be chosen properly and according to the developers' knowledge level, which in turn can help to save time for development.

**Community and documentation**

The existence of a community around the platform project signs that mobile platform is needed and therefore it will be easier to receive support and clarify the problematic questions. Besides, programmers usually share their code snippets and inform the community about found bugs. This information will help to concentrate on innovative ideas rather than on implementation and debugging. After publishing the application and receiving the player's feedback possible limitations and drawbacks of the game can be found. In general the presence of a well structured platform documentation and programmer's guide is an obligatory condition for the project.

**Platform's future**

The mobile platform market is new, and no one can predict what will happen with a new platform within a year. It could disappear or the producer could find that the platform is not profitable and stop supporting it. These variants could lead to the project's cancellation, or it would force the project to migrate to another platform, and consequently the team would have to rewrite the code. It means that attention should be paid not only to the technical part, but also to questions of the market investigation. The comparison of platforms should reveal current market leaders and make predictions about their future.

**Market possibilities**

Nowadays almost all mobile platform manufactures allow programmers to publish their applications and sell them via store markets. Usually the minor part of the earnings goes to the manufacturer as a payment for using the market. Nevertheless, the conditions of entering the market vary and should be compared. The comparison defines which companies have application stores and which not, and what the conditions and possible limitations are.

**Attractiveness for the target group**
In the previous sections of the report it was highlighted that out primary target group are students from ages 14 to 20. As a mobile group, students have more time and freedom to walk unaccompanied around the city and to interact with the community. In our work it is assumed that young people are more opened to innovation, thus the new platform with a variety of interaction methods can definitively become one of the attractive features of the game.

Based on the criteria described above we have begun to compare and analyze existing mobile platforms.

#### 3.1.1.2. Platform choice

The most important criteria which were marked out from the general list were:

- an easy development start
- the variety of input methods
- the presence of documentation
- the price of the devices

Easy development start means the correspondence of the platform language with the previous experience of the team. The variety of input methods is what allows the creation of a rich context application and the receiving of necessary educational knowledge. The presence of a well-structured documentation and a reasonable price of the devices are also taken into the consideration. These parameters allowed the start of an investigation of the mobile platform market and comparison of the existing platforms. The comparison was done according to basic functionality, user interface, core functionality and market information. All the compared data was organized in tables, allowing for an easier overview.

Examining the defined criteria it was concluded that only Android Cupcake and iPhone OS provide a wide range of input methods necessary for the forthcoming project (Topolsky, 2009). The number of these methods varies depending on the device, but almost all of them are equipped with accelerometer, GPS receiver, compass, and camera, what allows the implementation of rich context-based applications.

When speaking about machine-user interaction it is necessary to point out Apple as a leader. They succeeded in the development of easy-to-use, friendly user interfaces. Other companies also continued developing their ideas and innovations and improving the usability (RubiconConsulting, 2008). Best practices proposed by manufactures should be considered and used in the project, but that does not mean that the iPhone should have been chosen as the project's platform.

Programming languages vary from platform to platform. Referring to the platforms comparison (Topolsky, 2009) only Android Cupcake and Blackberry allow to program in Java. The iPhone requires the use of MacOS in order to program in Objective C. Moreover, ObjectiveC would need additional time to be learned by all members of the team. From this point of view the Android platform is more advantageous.

Due to many factors market analysts do not predict the failure of some of the considered platforms in the future. Each company positions their platform differently and have their own target groups (SkyhookWireless, 2009). By now both, Android and iPhone, are counted as platforms for youth. Nevertheless, both of them are considered expensive for our target group, a condition that can change after new devices are released on the market.

At the time the comparison was made all manufactures provided clear documentation, excluding Palm. Also, the community gathered around the platforms differed. The attention of the majority of the audience was drawn to the iPhone and Android platforms, what means also that these platforms have the biggest audience of programmers.

Additionally it is necessary to notice that our project is educational and with no doubt the popularity of the platform, as well as the market possibilities, are not so important as for instance the available input methods and programming language. From this point of view Android Cupcake fits better to our project.

#### 3.1.1.3. Conclusion

After the market research was finished it was decided to use Android as the project's platform. This year of platform usage concludes the rightness of the choice. Nowadays the market presents a variety of mobile devices supporting Android, what shows the popularity of the platform among users and interest from the manufactures. The platform is relatively cheap and the appearance of new devices decreased the price. The Android market has already more than 100.000 applications and best conditions to enter the market. Of no small importance is also the fact that our programmers gained the rich experience of using the features of the Android SDK and the creation of programs and games for Android.

### 3.1.2. Web Platform Selection[2]

[2]Till Hennig

The implementation of the web-platform (web-frontend and webservice) needed to be accomplished in approximately six months. Even though the project spans two semesters (equaling less than 10 months of effective working time), the entire time was not available to the developers since conceptual aspects needed to be discussed first.

For a web-platform of this dimension this is a tight time-plan forcing the students to use technologies that allow rapid development techniques. An additional challenge is the fact, that the concept is rather open and flexible in terms of its definition of scope. The flexibility for letting users contribute multi-faceted content results in a high level of abstraction for the developers.

These requirements make it nearly impossible to recycle existing web-applications which have been constructed for building web-communities; the effort for customization would have caused the students to spend more time trying to understand and change the code of the original developers instead of coming up with their own solutions and thus minimizing the academic aspects of the project.

The alternative are so-called frameworks, which the developer Jacob Kaplan-Moss characterizes as helpers that "operate at a high conceptual level"(Kaplan-Moss, 2009) allowing developers to focus more on the conceptual fundamentals of what is being built rather than on trivial implementation details. He goes on by saying that frameworks provide "much larger building blocks", which he illustrates by comparing the development of a web-application with the building of a house. Building it from scratch with raw materials, gives you all the flexibility you could possibly want, but the construction requires an enormous amount of planing and time. According to Kaplan-Moss, frameworks are like "factory-build homes", which offer a variety of choices for all kinds of components. The architect is more restricted because he is forced to choose from a predefined set of rooms, however most aspects can be customized and the time required for construction is a fracture of that for building a house from scratch. Continuing the analogy of Kaplan-Moss, using an existing application would be like moving into a Hotel room: Everything is furnished and there is no room for expansion, except renting another room. This would equal to adding a new, independent application for added functionality with a integration that would leave a lot to be desired. Kaplan-Moss goes on by saying, that "good frameworks encourage rapid development (...)" and that "It's no coincidence that the Age of the Framework is also the Age of Agile. Agile, XP, Scrum, etc. frameworks are at their best when used in a rapid-iteration style." Rapidly prototyping is possible, because most of the trivial and repetitive tasks are taken care of by the framework. The developer can go straight to implementing the core of the application. Components are are functional at all times and more functionality can be added iteratively as needed. By not frustrating the developer, the author says that, albeit sounding silly, frameworks actually make "(...) development fun", which is important because "fun motivates, leads to experimentation, and hence to innovation."

Another important aspect is the cost and license, and the philosophy behind the software that is used for our project. It is self-explanatory, that a student project does not have the necessary monetary funds to purchase commercial software. Commercial software is often optimized for scalability as well as redundancy. Large portions of the initial purchase costs can also be attributed to support services. These are features that are not a necessity for the project at hand, in fact they often have a negative effect as they create an ecosystem around these software platforms, where related services or products are also commercial. The main problematic lies in the architecture of commercial products.

Most of these applications are build upon proprietary technology that deliberately restricts interoperability to lock the customer in, however, it is our intention to experiment and rely on external APIs for certain information and operations, such as displaying geospatial information. Moreover, commercial products tend not to publish their source code, meaning they distribute compiled versions which cannot be altered. The academic context of this project demands that critical questions are asked: How is the solution implemented? How could it be improved? Without having access to the source code, these questions cannot be answered. Open-source projects follow a different maxim: All of the source code is published publicly. Most open-source licenses allow the code to be used free of charge for any purpose, even commercially. Support can be purchased if needed. The fact that the source code is freely available attracts developers to improve upon it which results in quicker development cycles and design decisions which can be followed and influenced using the also publicly held discussion.

As previously stated, there are several requirements that must be met in order to be suitable for the project's concept and time-frame. For most major programming languages, more than one framework exists that could be considered for the project - a decision was made in favor of Python, in conjunction with Django, for the following reasons:

#### 3.1.2.1. Rapid programming language

Not only does the chosen framework support the process of rapid development, the programming language itself is construed as being as flexible as possible and easy to develop for. Python is considered to implement "intuitive object orientation" meaning it offers very high level dynamic data types and full modularity using duck-typing instead of strict-typing. In comparison to the improvised object-orientation of PHP and the strict object-orientation of Java, developing in Python results in considerable time advantages. The clear and readable syntax, which bears resemblance to pseudo code, brings a shallow learning curve about allowing students with a background in any programming language to read and write Python code in a short amount of time. In accordance with the requirements set forth, Python is published under an open source license that makes it freely usable and distributable. Powering, and receiving support by, large corporations such as Google ensure the constant improvement and longevity of the platform.

#### 3.1.2.2. "Batteries included"

Python attributes itself as "batteries included"[3] to describe the extensive standard library, which includes libraries ranging from a webserver to a GUI toolkit. For almost everything else, third party components are available as open-source, making Python suitable for all problem domains of the project. Many other languages can easily be integrated by using language wrappers. This means that one programming language is sufficient - students can operate at a high conceptual level, using advanced language constructs instead of learning the basics of many different languages. The result is code of high quality and a deeper understanding into advanced programing concepts.

#### 3.1.2.3. Availability of Django and its extension GeoDjango

Django is a Python-based Web Frameworks that fulfills all of our requirements. It complements Python in its rapid development approach, offers sophisticated buildings blocks (such as a object-relational mapper, internationalization, a template system, etc) that make getting started easy, it is open-source and its high level of abstraction, which is often referred to as "automagic", surprises the developer because of its simplicity and thus ensures he is having fun. As outlined before, the web-platform enables the user to contribute spatially enabled data. A specialized branch of Django that adds support of geometry fields and extends the ORM to allow spatial queries is available under the name GeoDjango. Geospatial information can be seamlessly integrated into Django applications for a web-enabled GIS with the comfort known from Django.

[3] `http://python.org/about/`

### 3.1.3. Android in a Nutshell[4]

[4]Vahe Markarian

The Android project started with a team, who went to work at Google on July 2005. The idea was to develop the first open source mobile device operating system which would allow handset makers and carriers to provide a flexible and upgradable system. While the team was working on the project, Google started informing several leader companies in search for cooperation. Soon after the Open Handset Alliance (OHA) was formed.

Currently OHA reaches to 65 mobile and technology leader companies (OpenHandsetAlliance, 2010a), among its members are Google, HTC, LG Electronics, Motorola, Samsung, Sony Ericsson, Toshiba, Asus, Intel, NVIDIA, eBay and other handset manufacturers, mobile operators, semiconductor and software companies. The cooperation between these companies led to the creation of their first project, the Android, a Linux-based open source mobile device OS which was unveiled on November 2007 (Google, 2010e).

OHA members are strongly devoted to openness, as they claim “Increased openness will enable everyone in our industry to innovate more rapidly and respond better to consumers’ demands” (OpenHandsetAlliance, 2010b). The Android platform was built to be truly open, which would allow developers to create applications to replace the phone’s core functionalities such as replacing the default web browser, image viewer, or messaging application. As of 21 October 2008, Android is available as open source under the Apache License, which allows the vendors to freely add proprietary extensions without submitting to the open source community. Android is based on the Linux kernel, hence these components are licensed under the GNU General Public License or GNU Lesser General Public License, the LGPL. The Android platform is also using several open source libraries under various licenses, such as LGPL, BSD, MIT, etc.

Android Applications area equal, which means any application that comes with the phone is no different than those written by any developer. It comes with a set of built in applications, like desktop display, telephony, web browser, email, media player, and other applications. Minimum hardware requirements are 128MB of RAM and 256MB of Flash Memory. The platform supports large screen resolution and keyboard interface. It also supports video, camera, touch screen, GPS, accelerometers, and 3D graphics.

To develop Android applications one must download and install the Android SDK[5], which includes device emulator, tools for debugging, memory and performance profiling, and a plug-in for Eclipse IDE[6], which is the supported development environment. The installation of the Eclipse plug-in ADT (Android Development Tools) is required to add integrated support for Android projects and tools.

#### 3.1.3.1. Android Anatomy

Android Anatomy consists of several layers that structure the Android operating system. These layers include the Linux kernel, native libraries, Android runtime, and application

[5]Software Development Environment for developing Android applications. It can be downloaded at: `http://developer.android.com/sdk/index.html`

[6]An Integrated Development Environment for developing software. It can be downloaded at: `http://www.eclipse.org/downloads/`

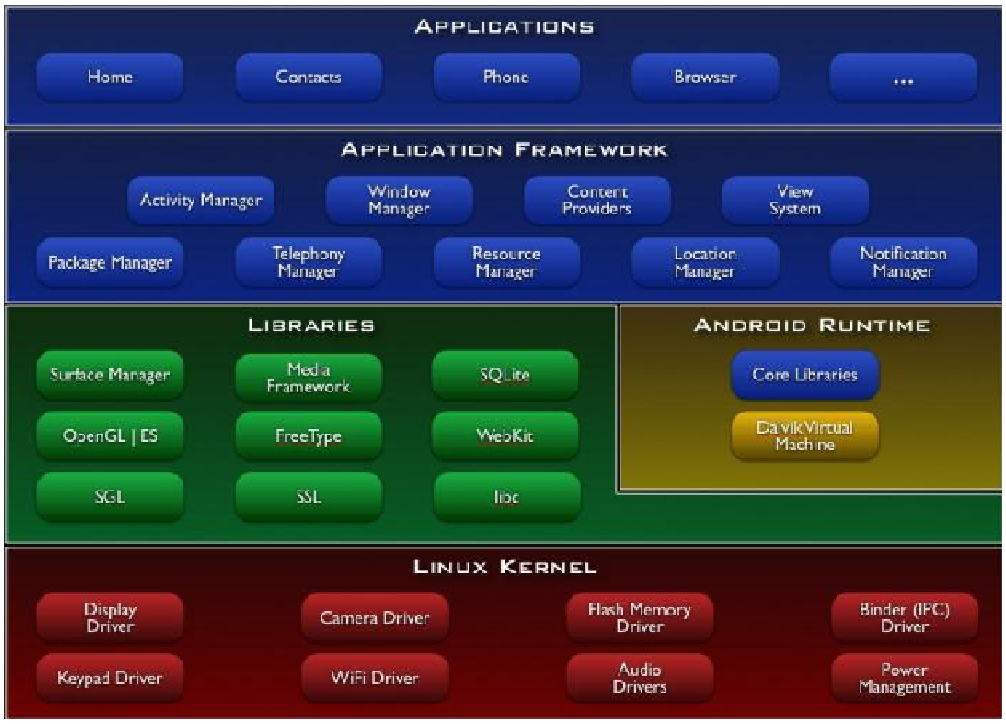


Figure 3.1.: The system architecture of the Android platform.

framework.

**Linux Kernel**

The Android architecture relies on the Linux kernel version 2.6 , which corresponds to the red layer illustrated in figure 3.1. The kernel manages the core system services such as security, memory management, process management, network stack, and driver model. Therefore, the kernel takes care of hardware drivers to run on the mobile device. Android is designed to run on virtually any ARM-based Linux kernel environment, and it provides support for Qualcomm MSM 7K chipset family. As new versions of the Android platform appear, support for other major chipsets will be added.

**Native Libraries**

A set of open source libraries, the green layer in figure 3.1, are implemented in the Android platform to be used by various components of the system. Developers may find these libraries appealing to create innovative applications for Android. These libraries are mostly written in C/C++ programming language. For example, the 3D capabilities of Android are due to the OpenGL ES 1.0 API, which supports hardware 3D acceleration and highly optimized 3D software rasterizing. Another example is the SQLite library which is a powerful and lightweight relational database engine available to all applications. It is quite effective in embedded systems and yet provides most features found in the SQL-92 (SQL, 2010) standard.

**Android Runtime**

Android runtime, the yellow layer in figure 3.1, consists of core libraries and Dalvik VM[7]. Core libraries, written in Java language, consist of a set of Java SDK core libraries intended for the use of embedded systems and other libraries that facilitate mobile device functionalities. Whereas Davlik VM is specifically designed for embedded systems, to transform the Java format into Dalvik format. A special tool named “dx” is integrated in the Android platform, it converts generated byte code from .jar to .dex file format, which enables the byte code to run more efficiently on the small processor. For that reason every Android application runs on its own instance of Dalvik VM, yet enabling multiple instances of Dalvik VM running on a single device simultaneously.

**Application Framework**

The Application framework is composed of APIs written in the Java language. It is a toolkit that simplifies the use of components within the applications. Therefore, applications may use other application’s capabilities under the security restrictions enforced by the framework. For instance, contacts in the address book can be directly marked on a map relying on their address.

**View System:** UI components such as lists, grids, text boxes, buttons, etc.

**Activity Manager:** manages lifecycle of an application and provides a common navigation between applications.

**Resource Manager:** allows applications to access non-code resources like strings, graphics, and external files.

**Content Provider:** enables applications to access shared data from other applications such as contacts, messages, etc.

**Notification Manager:** is used to show customized alerts in the status bar.

**Package Manager:** stores a list of installed applications in the system.

**Telephony Manager:** includes necessary API for telephony related functions.

Above the application framework are all the applications that are either shipped with Android or created by any developer. Due to the equal nature of applications in Android, end users can optimize their mobile device by installing customized applications which can completely adapt the system to their needs.

[7]An extremely low-memory based Virtual Machine designed for the Android platform to work well during low power, CPU, memory, and data storage situations.

### Activities

Android applications are divided into four building blocks: Activity, Broadcast Intent Receiver, Service, and Content Provider. While developing an application, one must define which of the mentioned components should be used in an application by listing in a file called AndroidManifest.xml. Accordingly, every Android application must include this XML file in its root directory. The manifest file holds the capabilities and requirements of the declared components before running any application code.

### Activity

An activity is a user interface component, which displays a single screen in the application. For instance, an activity will be listing all the contacts in the address book, and another activity can be displaying only one contact with all its information. Each screen is implemented as an activity, and moving to another screen is realized by starting a new activity. While a new activity is started, the previous one is paused and queued in the history stack. Users may control the activities by going back and forth among the paused activities, or even remove an activity when it is no longer in favor of the user.

Android manages screen to screen navigation by a special class called Intent. Intent has two important properties, action (MAIN, VIEW, PICK, and EDIT) and the data (URI – Unified Resource Identifier) to act upon. For instance, viewing a picture in the image gallery is done by creating Intent with the VIEW action and the data set to the URI representing the picture.

### Broadcast Intent Receiver

It is used to respond to any external event. For example, an alarm notification must be generated while receiving an SMS. Broadcast Intent Receivers are defined in AndroidManifest.xml or in the code itself.

### Service

It is a task, running in the background. In other words, one can open an application, and keep the service running while browsing other applications. For instance, one can open a media player and listen to music and at the same time open a web browser without interrupting the music from playing.

### Content Provider

It is a class, implemented to share data with other applications in a standard set of methods. Hence applications can make use of other applications by obtaining their data and manipulating it in another fashion.

### 3.1.4. Django in a Nutshell[8]

[8] Pia Storck

The following section provides an overview of Django, a framework that was used within the project for the web platform. After introducing its core features, special attention will be given to the model-view-controller design pattern and to the different components of that programming tool.

#### 3.1.4.1. Introduction to Django

Django is a web framework written in Python[9] that "encourages rapid development and clean, pragmatic design" (Django Project n.d.) by simplifying common web development tasks. It is released under the BSD License and considered to be an effective tool for creating database-driven web applications which are easy to maintain. By providing a programming infrastructure, it is easier to build better applications with clear code in less time. "High-level abstractions of common Web-development patterns, shortcuts for frequent programming tasks, and clear conventions for how to solve problems" (Holovaty and Kaplan-Moss, 2009a pg.3) contribute to Django's goal of facilitating the creation of complex websites.

In addition, the framework relies on the reusability and "pluggability" of components as well as on the so called DRY[10] principle. This rule states that "every piece of knowledge must have a single, unambiguous, authoritative representation within a system" (Hunt and Thomas, 2000) to preserve flexible and maintainable code.

Besides, the web framework offers sophisticated building blocks such as an object-relational mapper, a template system and other useful components which are cleanly separated from each other. Thus, just like Python, Django follows the "batteries included" philosophy which implies that a large standard library with tools for common tasks is available. Originally developed to manage news-oriented sites, Django is now used by web developers around the world[11].

#### 3.1.4.2. Model-View-Controller Pattern

Like other frameworks Django follows an architectural pattern called model-view-controller, or MVC for short. The pattern divides the application logic from the input and screen presentation. While the model represents the data and the business rules upon which the application operates, the view renders the content and specifies its presentation. The controller, on the other hand, "defines the way the user interface reacts to user input" (Gamma et al., 1995 pg.4). By decoupling these three kinds of objects, flexibility and the reuse of existing code is facilitated.

According to the Django Project this design pattern is interpreted slightly differently. In Django a view is only a callback function for describing which data is presented. How it is displayed is specified by the template which is loaded by the view that renders it with the retrieved data. As a result, the content is separated from the presentation. A

[9]Python is also used for settings, files and data models (Django Project n.d.).
[10]DRY stands for "Don't Repeat Yourself" (Hunt and Thomas, 2000).
[11]For example, the Washington Post `www.washingtonpost.com` uses Django for parts of their website.

simple database-driven application will at least be split over three Python files (models.py, views.py and urls.py[12]) and an HTML template. Complex projects, of course, will consist of a lot more files. The idea behind this approach is the separation of concerns because when the components of a Django-powered web application are loosely coupled, they "can be changed independently without affecting the other pieces" (Holovaty and Kaplan-Moss, 2009b pg.6).

#### 3.1.4.3. Components of the Django Framework

As mentioned previously, Django sets high values on reusable applications which allow the seamless integration of additional functionality so that the entire project remains easy to manage. The core framework consists of several components such as an object-relational mapper in which the database layout is described in Python code. Furthermore, Django comes with an URL dispatcher that supports clean and elegant URL schemes. The view system is responsible for processing requests while the template system separates the design not only from Python code but also from the content. Other components are a cache system and a lightweight, standalone web server for development and testing. Also commonly used is the forms API, a library for handling forms. In addition, the core provides a dynamic admin interface, built-in tools for generating non-HTML content and the functionality for internationalization.

Besides, several bundled applications exist to extend the core functionalities of the main Django distribution. These extensions include an authentication and commenting system as well as tools for generating web feeds and a branch for creating GIS applications[13]. In conclusion, it can be said that Django is a very flexible and easily extensible Python-based web framework which saves time and supports developers in building high-quality web applications.

[12]The urls.py file represents the controller that assigns a given URL pattern to a view.
[13]`http://geodjango.org`

## 3.2. System architecture

### 3.2.1. Overview[14]

[14]Yarik Sheptykin

Under the system which is being referred to in this section it is understood a complex software solution for the requirements set in the conceptual design document of the *StreetDroids* project. The software solution is complex because it is composed of many conceptual parts which provide different functions and use various technologies. The architecture of this system was introduced based on the initial conceptual requirements of the game mechanics document discussed in section 2.6.1.2. Along with the concept of the game itself, its mechanics, rules, players and environment, the game design document establishes a set of requirements which the software solution must meet to implement the designed game. Among the many requirements the ones which influenced the system architecture the most are:

- *StreetDroids* is a mobile game
- players must have the possibility to contribute with their own content to the game
- players must have a community where they can share their play results and compete against each other

Though the game is defined as a mobile game the other requirements make the system too complex to build it on a mobile platform only. The fact that the players must have a community where they can communicate and share content, forces the system to be always available and act as a common server for many peers. Therefore this part of the system logic cannot be put on a mobile device, which is always in movement and does not have a stable connectivity to a common communication media. Another obstacle in implementing the whole system on a mobile platform is a limitation in computational and data storage resources normally fixed by a device vendor. These arguments led to the decision to split the solid system into a set of interconnected pieces.

It was decided to break the system into two main parts: a server and a client. The server should be always available to a wide range of users and therefore should have a static well known address in a common communication environment. The clients should be as light as possible and provide the basic interaction functions to the end player. The client-server system architecture is a widely implemented model used for many software business solutions (Renzel and Keller, 1997). As the most suitable communication environment it was chosen to use Internet, which nowadays is a common communication media for many technologies, and is able to bring together mobile and PC users, which is the most important for the project.

In a situation where the system architecture is being split it is very important to find a good balance in placing the system logic on its parts. Tuning such a balance many aspects of a system performance should be taken into consideration. First of all, the limitations that a mobile platform puts on the application must be considered. Eventhough mobile platforms constantly grow in terms of computational rates, data exchange speeds, and data storage sizes, they still put noticeable restrictions on user applications. Therefore it seems obvious to put most of the system's components on the server, and leave the client for the interaction with the player only. Nevertheless this radical approach would require a lot of communication between the client and the

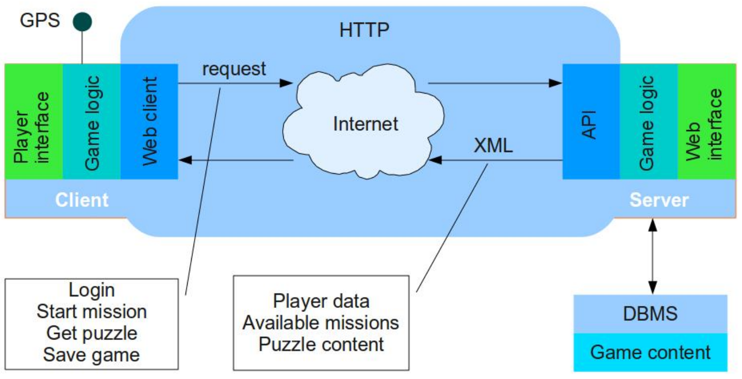


Figure 3.2.: Architecture that has been adopted in the StreetDroids project for a software system implementation.

server, which in turn decreases the performance, increases the costs, and therefore debilitates the user experience.

All these arguments were considered while developing the system architecture which resulted in the most suitable solution for the project requirements. Most of the logic related to the game itself was implemented on the mobile side of the system. This logic includes the navigation, the puzzle launch control and the interaction with the player inside the game. There are also some parts of the game code that were put on the server side. The logic that defines the sequence of puzzles, puzzle locations and puzzle contents is placed on the server.

This kind of separation helped the application to reach the highest productivity, and also gave much flexibility for the game's customization through user generated content. Nonetheless the separation of the game logic brings dependencies into the system components and prevents the logic from localizing itself in one certain place, which makes the adaptation to any possible conceptual changes harder.

Considering all advantages and drawbacks discussed above a sketch of the system architecture, shown in figure 3.2, has been suggested.

Figure 3.2 shows the architecture that has been adopted in the *StreetDroids* project for a software system implementation. This architecture is based on the client server architecture paradigm and composed of one *StreetDroids* game server, many *StreetDroids* game clients and one global *StreetDroids* game data storage.

The game server is the part of the system architecture that offers access to the game content which is stored in a database managed by a database management system. It provides two interfaces to the game data. One is used by game clients for downloading puzzles and puzzle content. Another interface allows the game users to view, edit and create the content for the game. Both interfaces additionally allow the players to login and control their accounts, it also allows them to play and to contribute to the game's content. The interface used by the game clients is called API. Along with the data access this interface defines a set of functions for a game flow control. The other interface is built to give the web users the ability to share the results of their play, control their accounts and to create new content for the game, such as custom missions and puzzles.

The game client is a piece of software running on a mobile GPS-enabled platform. It is composed of three main parts: the player interface, the game logic and a web client. Through the player interface a player interacts with the game. In response to the player's action the interface generates game events, which are processed by the game logic. The game logic controls the game flow based on the programmed reaction on the game events coming normally from a player or from a GPS module. Some events are also generated by the web client, which is used for exchanging the game data between the client and the server. The web client gets access to the game data through the API mentioned above.

The communication medium used by the web client to call API functions is the Internet. Every request to the API server is done via HTTP, as it was decided to use an HTTP server for both, the web and the API frontends. Every data being transferred between a client and a server is wrapped into an XML document and sent as an HTTP content along with an HTTP request/response. All API calls are made based on the "pull" method, which means that the data is delivered to the client only after the client has requested it, therefore the API server never starts a communication session. The game server API and the communication rules are further discussed in the following sections.

The architecture sketch presented in this section has been suggested and accepted in the beginning of the project development. Though it does not define precisely an architecture for the software solution for the initial project requirements, it outlines the ground for such a solution. In fact, this architecture served as a basis for a software architecture developed to program an application logic, which is split into a client and a server architectures respectively. Both this architectures are explained in details in sections 3.2.2 to 3.2.4.

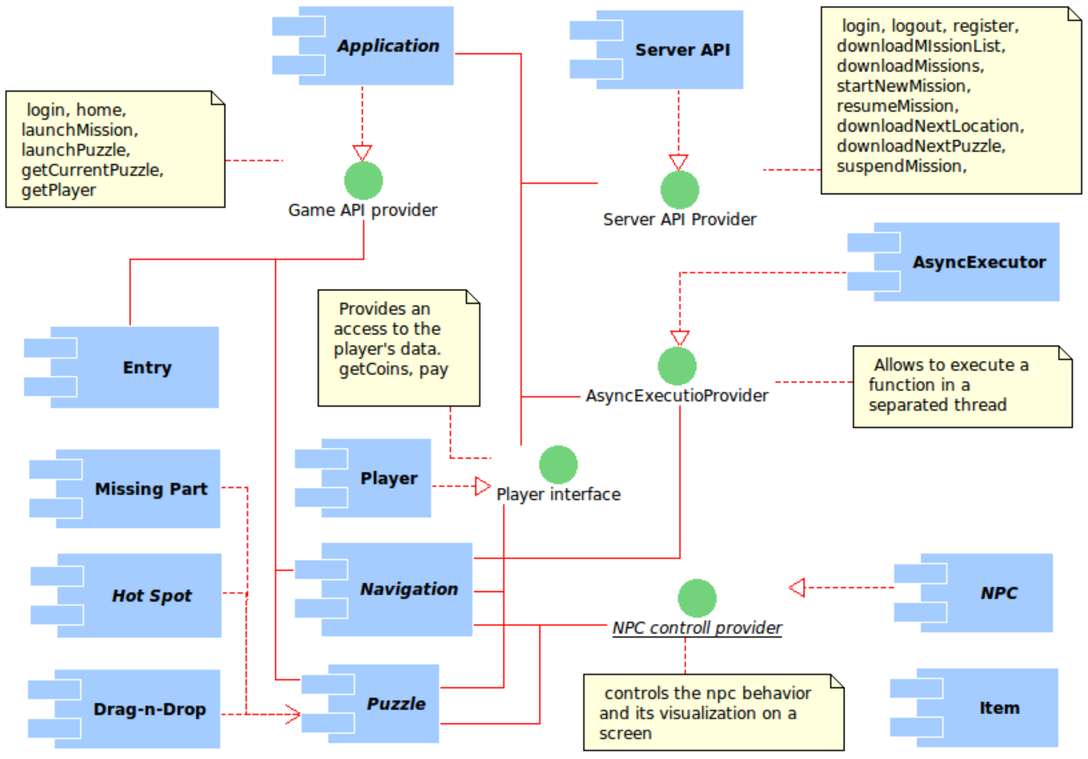
### 3.2.2. Client Architecture[15]

[15]Vahe Markarian, Yarik Sheptykin

The client architecture in the scope of the *StreetDroids* project is a software design document that defines application components and their relations, which were used to program the client part of the *StreetDroids* game on a mobile platform. The architecture has its roots in the system architecture discussed in the previous section. It takes the client part, defined in the overall system design, and specifies down an architecture for each of its particular components. Therefore the architecture discussed in this section is a particular branch of the entire software system, and consequently its concept is influenced by the interfaces of the other parts. Though the overall system architecture has defined a clear interaction between its parts in practice it is very hard to follow these guidelines. During the system implementation specific peculiarities of every platform constantly introduce slight changes to the initial interaction model. Consequently the architecture of each system part is always being influenced by the architectures of other systems, and also constantly evolves along the application development. These facts have been considered while developing a flexible and a successful client architecture, which is a result of an evolution driven by a constant research, platform evolution, and the client application development itself.

The client application development in the *StreetDroids* project consists of two phases. The first phase is the phase of the game prototype, which was conceived to test the platform's features and to prove the implementability of the game concept key features. The game prototype was build under a special client architecture, which was a simplified version of the architecture used for the final application. Although the prototype architecture had been simplified it was still very complex. It was composed of sixteen classes, and used a game loop very similar the one used in the final game implementation. The prototype successfully tested the core game mechanics, and much of the prototype architecture was later adopted for the final client architecture.

The second phase in the application development started with an attempt to evolve the prototype into the final software solution. At this point the architecture had been through many changes. Many new components were added to accomplish the requirements set in the game mechanics document. The old components and their relations were altered as well, because a constant research in the mobile platform architecture brought many new solutions to the old problems. Most of the changes were introduced because of specific feature adaptations of the chosen platform, which led to a considerable code refactoring. This happened because the initial architecture was too general and did not make use of the platform's peculiarities and specific mobile requirements the platform had been designed with. It is not the fault of the preceding architecture, but rather a necessary step for its evolution, since the concept used for the previous architecture was kept, eventhough the implementation at some points was changed. According to the concept of the prototype architecture there were several generalized blocks the software was designed around. These parts are the game logic, the GPS processor, the HTTP client, and the player interface. These blocks remain in the current software solution, but they are implemented in a slightly different way. For example, the game logic in the final implementation still launches a puzzle when the player reaches the right location, but instead of running the puzzle in the current activity it starts a new one on top. There were many other similar changes, which were made first of all to



Application
Server API
login, logout, register, downloadMIssionList, downloadMissions, startNewMission, resumeMission, downloadNextLocation, downloadNextPuzzle, suspendMission,
login, home, launchMission, launchPuzzle, getCurrentPuzzle, getPlayer
Game API provider
Server API Provider
AsyncExecutor
Provides an access to the player's data. getCoins, pay
Entry
AsyncExecutioProvider
Allows to execute a function in a separated thread
Missing Part
Player
Player interface
Hot Spot
Navigation
NPC
NPC controll provider
Drag-n-Drop
Puzzle
controls the npc behavior and its visualization on a screen
Item

Figure 3.3.: The diagram is composed of generalized software modules – the components

provide a better experience to the player.

One of the most important parts of the architecture adaptation is its visualization method. Since the architecture is mostly implemented by many different people in a long period of time it is important that all have a common vision and understanding of the existing architecture. During the architecture development in the *StreetDroids* project the unified modeling language (UML)[16] was used. As mentioned before, the prototype architecture was already very complex, but additional changes done in the second development phase made it even more complex. The prototype architecture could be clearly visualized as a UML diagram of application classes. Unfortunately the complexity of the final architecture does not allow it to be presented in a clear UML class diagram. Because of its big size this type of representation will reduce the readability, and can increase confusion among its readers, which is very undesirable. Therefore it was decided to generalize application classes into blocks, and present the client architecture as a diagram of these blocks with the maximal possible scale of precision.

[16]The Unified Modeling Language `http://www.uml.org/`

The diagram presented in figure 3.3 is composed of generalized software modules called components. Each component depicted as a blue rectangle is actually composed of many classes grouped by functionality. The diagram shows that some of the components provide functions to others components through interfaces. Each interface, shown as a green bubble, defines a set of methods which the components can call. For every interface there has to be a component that implements the functionality defined in it. There are three main functionality providers: the application, the server API and the asynchronous executor components. Many of the functions used by other components are implemented in those modules. As it shows from the component name the server API provider component implements methods for accessing the game server API. This component is vital for the game, because as it was discussed in the previous section a part of the game logic is implemented on the server. The functionality provided by this component is consumed mostly by the application component, which is the main component of the system. This component groups source codes, which control an application flow. It also provides access to the global application data, and its services are consumed by other core components such as the puzzle and the navigation.

The navigation component except application functionality consumes also a functionality provided by the asynchronous executor component. This kind of functionality is not vital for the system execution, but it provides functions that allow a piece of code to be executed in a separate thread. The methods of this block are used by the application and navigation blocks code to execute long term, high CPU or IO consuming operations. For example, the navigation module asynchronously downloads the next puzzle while the player is walking around the city, and when the player finally reaches a puzzle location the puzzle instantly pops up, because its content had already been loaded onto the device. Additionally there is the player component, which also provides a set of functions required for the game implementation. Those methods are used for user data manipulations. In the current application the player interface is mostly used by the puzzle component, where there is a need to access player's coins. Similarly to the player component the NPC component also provides a small interface for manipulating the NPC behavior.

Of course, on the lower layer - on the layer of higher details - there are more interfaces that define the actual communication between the system elements, but they have not been included into the component diagram because of readability issues.

Components which do not provide any functionality normally consume it from the others. The two "biggest" such consumers are navigation and puzzle components. These two components include GUI parts, therefore they redirect some of the consumed functionality to the user with new functions added to it. The navigation component provides the user interaction while the player is walking around the place looking for a puzzle location. This module is responsible for giving play hints, displaying the compass, monitoring the user position, and launching puzzles in time. The puzzle component includes the logic that controls the interaction during the puzzle play. This component is composed of several puzzle type blocks; each block includes specifics to a puzzle type classes.

Another component that has an GUI is the entry component. It contains classes that handle interaction with the user before the actual game starts. It includes start menu and login screen handlers. There is also a component which neither consume nor provides any functionality. Instead, the elements included in the item component are used by other modules. For example, the item consists of a set of classes which model items of the game instances of which are used by the other components. The architecture presented in this section has successfully been used in the *StreetDroids* project for the client side application development. A components diagram shown in figure3.3 gives a general overview of this architecture, which is too complex to be visualized in this paper with a higher level of details, but it definitely deserve some attention because of the many interesting solutions it gives. The architecture which is currently in use has evolved during the project's development, and is possible to continue this evolution if further development happens.

#### 3.2.2.1. Activities

One of the platform's peculiarities, which has been mentioned in the previous section, is an activities concept of the Android mobile platform application. In the official Android documentation an activity is defined as "a single, focused thing that the user can do" (Google, 2010a). An activity can be compared to a single application window, neither complex nor too simple. Having more than one layout in a single activity is possible. Every Android application, except from a service, has at least one activity, which is at the same time a main entry point into the application. According to the Android software system architecture activities are in a way separated parts of an application. In case there is more then one activity, they can share the same data storage and work on common objects, but their execution is separated in time, and is controlled by the user's focus. The concept of activities is designed in a way that each activity of an application could be used by another application, which promotes code reusability, but it also brings some requirements to Android application developers. These platform features have been considered while implementing the migration from the prototype architecture to the final application architecture. The architecture was changed in a way that allowed the separation of the application code into the corresponding activities. There are three layers in the user interaction sequence, which created a basis for breaking the application execution into activities. These three layers are, an entering layer, a home layer, and a game layer. The entering layer groups such actions as starting the application, logging in or creating a new account, and adjusting the preferences and language selection. The home layer includes the functionality that the user receives after a successful login; these actions are editing the personal data, browsing and resuming paused missions and starting new missions. The last interaction layer groups actions that happen in the game itself, like navigation, puzzle launch, NPC interaction, etc.

Though this kind of division is intuitively clear and is easily derived form a simple application flow, such as login, choose mission, or play the game, it cannot be directly mapped to the activities architecture because each of these parts has much more interaction components than one mobile screen can fit. Therefore it was decided to do a further breakdown of the application interaction phases into simpler blocks. To gain the best profit from splitting an application into activities it is important to keep balance between overwhelming the application with too many activities and creating few but “heavily” programmed ones. Both of the extremes affect the performance and the memory consumption. To prevent this from happening the interaction load has been considered for the activities architecture design. Interaction phases which require a lot of code and resources have been broken into separate activities. A good example is the game play phase, which has been divided into the navigation and the puzzle play activities. However, the home user phase has been left as a single activity with several layouts interchanged, because it does not have much interaction on all its screens. Nevertheless if the further development of the user interface adds more complexity to the home activity it can be sub split to keep the interaction balance aligned between application activities.

Another specific feature of the Android mobile platform is the ability of one application to use activities shared by the other application. This feature is possible because each activity in an Android application is a sort of isolated application module (Rogers et al., 2009). Each activity can, for example, launch a new activity, but cannot close it without a direct system activity stack manipulation. An advantage of such a feature is that there is no need for a custom activity design if an activity with a close functionality is shared by another application or the Android system itself. It is also considered good practice to share activities of a custom application so other developers can access and reuse them. Despite many positive sides this feature also has drawbacks. First of all, “borrowing” an activity from another application introduces undesirable dependencies. Sharing an activity might also lead to a very general activity design, which affects the performance of the host application. It also affects the way the application data is shared among activities. If activities are designed for “export” they should be isolated as much as possible from the other application parts, and operate on the data bundle given with their call.

Unfortunately all activities in the *StreetDroids* project are tightly connected with the global application data. Therefore it has been decided to ignore the discussed feature, and neither use foreign activities nor export inner ones. Based on discussed above decisions the activities architecture, shown in figure 3.4, has been developed. The architecture is presented as an activity tree which gives it a clear visualization. The big green boxes correspond to the activities. Each activity has at least one screen the user can interact with. If an activity has more then one screen it can switch between they are a shown in the smaller blue rectangles. The start menu and login activities belong to the entering layer and provide the functionality that allows to manipulate the global game preferences, receive help, login to a game, or register in the system if an account has not been created yet.

The next interaction layer is presented by a single activity which only groups functions related to the user’s home page. The last layer on the diagram is the game play layer that is implemented with two activities: a navigation and a puzzle activity. The navigation activity includes all functions needed for the user to navigate around. Though it has only one screen it is very rich with interaction elements, and it has much program-

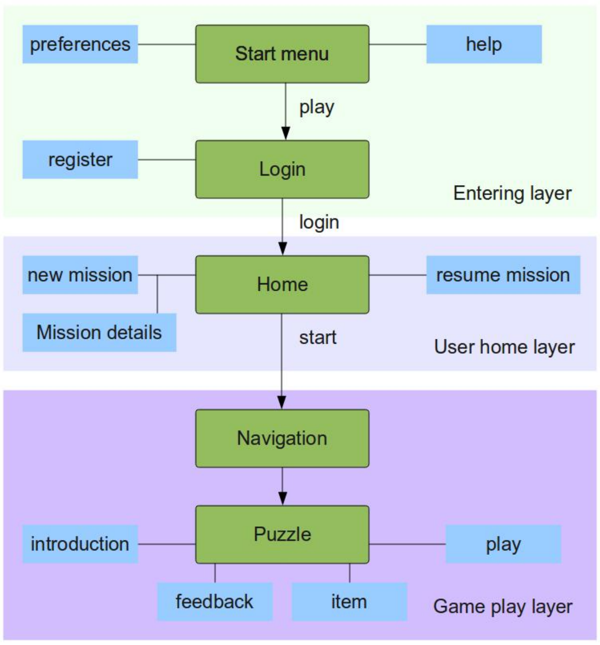


Figure 3.4.: The structure shown presents an activity composition of the *StreetDroids* project.

ming code behind it. When the player reaches a puzzle location the navigation activity launches the puzzle activity, where the user is given all required functionality to play each specific puzzle.

This activity structure accurately outlines the activity structure used in the final application design concept, eventhough it lacks some screens defined in the game mechanics document. This happened because during the game's concept implementation all features have been prioritized, and only the features with high priority made their way to the final application. This nevertheless does not mean that features with lower priority will never be included in future releases.

It is worth to be noticed that some activities have many screens, while others have one screen only. In the passages above the importance of balance has been mentioned, and it might seem that this balance is not well tuned at first glance. This is both, right and wrong, because of the following reasons. In one hand, the activities architecture which is currently used in the application really lacks some balance if, for example, the login and the home activities are compared. The login activity has two screens only and the programming logic of those screens is not very complex. On the other hand, the home activity has twice as much screens and the logic behind them is considerably more complex. Nevertheless, while developing this activities model a possibility for further development has always been considered. The point is that some features from the login activity were given a lower priority compared to the ones the home screen had. Therefore those features were not implemented yet, but if they are included in future releases there is already a place reserved for them. It is expected that this tiny design peculiarity helps to make further development easier.

#### 3.2.2.2. Package Structure

The game itself is divided into several packages, which are the building blocks of the client architecture. The packages contain all the necessary classes in an organized fashion. The arrangements of the classes within their packages are done based on the usability of the classes with relation to other classes. The packages are as follows:

**API:** An interface that allows communication between the server and the client via XML documents. It allows the game client to submit and receive data about the puzzles, missions, geo locations, sessions, etc.

**Application:** The main application of the game that holds all the activities for the game play. It is also the global section between all the activities to share and access data.

**Async:** Performs asynchronous API operations to improve game functionality. It runs tasks in separate threads, thus allowing multiple operations to be executed simultaneously. For instance, while downloading puzzle location, the puzzle itself could also be downloaded at the same time to allow quick game responses.

**Compass:** Custom compass functionality integrated in the game.

**Hint:** Functionality of game hints that consists of either geo location hint or puzzle hint.

**Httpclient:** Handles the communication between the game client and the game server.

**Item:** Functionality of game items, which are the reward when a puzzle is successfully finished.

**Location:** Responsible for user and puzzle locations. User location consists of single geo location, while puzzles have a single geo location as center and a set of geo locations around the center that form an area.

**Log:** Specifically designed for evaluation purposes to track down the usability of the game.

**Login:** Handles the interaction that allows the user to login, create new account, or play as a guest.

**Map:** Provides functionality to display Google Maps, as well as to overlay graphic such as user and puzzle location markers.

**Mission:** Game mission functionality that allows players to start, store, and skip while playing.

**Motion:** Listens for device's accelerometer and GPS location motion changes. Triggers events when the player moves, reaches a puzzle or hint locations, or moves away from the puzzle location.

**Navigation:** Handles displaying of the map and launching of the puzzles based on the motion events.

**NPC:** A Non Playable Character interface, which is used to interact with the player throughout the game.

**Puzzle:** Consists of generic and specific puzzle classes. Specific puzzles are singular activities which have different puzzle logic, while the generic puzzle provides a common structure for specific puzzles.

**Startscreen:** The starting point of the game. It allows players to select different activities of the game.

**User:** Handles player functionality.

### 3.2.3. Server Architecture[17]

[17]Till Hennig, Alexander Tyapkov

#### 3.2.3.1. Deployment and Dependencies

**Serverside Dependencies**

The following software is used at various points throughout the core of the project:

**lxml** (2.2.6) - "lxml is the most feature-rich and easy-to-use library for working with XML and HTML in the Python language." Used for parsing XML documents[18]

**PIL** (1.1.7) - "The Python Imaging Library (PIL) adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities." - Required for image processing, see sorl-thumbnail[19]

**django-tagging** (0.3) - "A generic tagging application for Django projects, which allows association of a number of tags with any Model instance and makes retrieval of tags simple." Used for categorizing puzzles[20]

**sorl-thumbnail** (3.2.5) - "Our goal is to make the best thumbnailing application for Django, balancing simplicity and extensibility." Used for creating resized variants of image media automatically[21]

**Clientside Dependencies**

**Mootools** (1.2) - Javascript framework for user interface interaction[22]

Dependent on mootools are the following libraries:

**Lasso.Crop** - Used for "Photoshop"-like image cropping in various editors[23]

**GearsUploader** - Enables uploading multiple files at once[24]

**MavDialog** - Modal-window user interface[25]

**Deployment**

Django projects can be deployed to a number of different webservers, such as Apache using either mod_wsgi or mod_python, or any webserver that support the FastCGI, SCGI, or AJP protocols. Please refer to the documentation for the advantages and disadvantages in each of the solutions[26].

[18] http://codespeak.net/lxml/
[19] http://www.pythonware.com/products/pil/
[20] http://code.google.com/p/django-tagging/
[21] http://code.google.com/p/sorl-thumbnail/
[22] http://mootools.net/
[23] http://www.nwhite.net/?p=328#more-328
[24] http://bitbucket.org/kmike/gearsuploader/
[25] http://maveno.us/library/public/mavdialog/
[26] http://docs.djangoproject.com/en/dev/howto/deployment/

In the case of *StreetDroids*, the project is deployed to a server running Debian 4.06 and *Cherokee Web Server*[27] "a very fast, flexible and easy to configure Web Server". Cherokee is very lightweight in terms of memory footprint and runs sufficiently efficient even on systems with limited hardware resources. The web-application runs on Python 2.5 and Django with the GeoDjango extension in the latest stable release (1.1.1 at the point of writing). The environment ("webapp") the application is executed in is isolated by virtualenv which manages dependencies and versions of Python packages.

To change the host system, the administrator needs to adapt the system specific settings (such as file paths or database configurations) in the `streetdroids.settings` module.

#### 3.2.3.2. Overview

The implementation follows the structure of a Django project. Functionality belonging together is encapsulated into a unit called an "application". Applications are implemented according to the recommendations by James Bennett for creating "reusable apps" (Bennett, 2008), of which the highest principle is "Do one thing, and do it well" ((Bennett, 2008) Pg. 4). Reusable, or "pluggable", apps allow the seamless integration of additional functionality without the necessity for altering existing code, which makes the project and its individual apps flexible and easy to manage.

By default, an application consists of:

- a `models` module, which realizes the persistently stored concept of the application
- a `views` module, which declares functions that interacts with the model instances
- a `urls` module, which links URLs to a view functions

Applications may be extended by adding attentional modules (as detailed in section 3.5.2). New applications can be created by using the `manage.py` command `startapp`.

The common functionality of the *StreetDroids* project is separated into the applications `core` and `supplements.core` contains all elements (models, views, webservice) that are required and relevant for the game-logic. The application `supplements` contains elements not specific to any puzzle-type and not required for the game logic. These elements range from non-playable characters (and the logic for creating them) to images (and the logic for storage, resizing, etc).

In the following section the main concepts and their functionality will be presented.

#### 3.2.3.3. Core

The following concepts are implemented as models in the module `streetdroids.core.models`. These models are persistently stored in the database.

[27] `http://www.cherokee-project.com/`

- `Mission`: A mission is a collection of puzzles.
- `MissionSession`: A mission-session is the link between a mission and a user. When a user selects a mission for playing, a `MissionSession` instance will be created to keep track of the user's progress (also see `PuzzleStatus`).
- `StarredMission`: A starred missions is a mission a user has bookmarked, i.e. he is interested in playing but not in this instant.
- `Puzzle`: `Puzzle` is an abstract base class. All puzzle-type implementations need to inherit from this class. It implements the common attributes and access methods for all concrete `Puzzle` implementations
- `PuzzleStatus`: The `PuzzleStatus` class is used to track the status (if it has been solved, failed, skipped) of a `Puzzle` within a `MissionSession`.
- `Hint`, `LocationHint`, `PlayHint`: The `Hint` models have the purpose of storing information that is specific to a `Puzzle`, and aid the user in accomplishing the current task, such as finding the puzzle, solving it etc. Each `Puzzle` can have multiple hints of each type.
- `Texts`: Contains all customizable texts related to a puzzle, such as introduction, success and failure etc. These attributes have been separated from the `Puzzle` model to allow language localization of puzzles.
- `UserProfile`: Because the `User` model is mandated by the framework (contains a fixed set of attributes, such as username, email, name, etc), an additional model, here `UserProfile`, is used to add project specific attributes to a `User` instance (such as last know position)

#### 3.2.3.4. Supplements

The following concepts are implemented as models in the module `streetdroids.supplements.models`. These models are persistently stored in the database.

- `Item`: Model for the reward the user receives after successfully solving a puzzle
- `Npc`: The implementation of the guide in form of a character. Contains references to the individual elements (See `NpcElement` below) and their offsets (relative position to each other) as well as a rendered image of the NPC.
- `NpcElement`: The individual graphical element that a NPC consists of.
- `ImageBase`: An abstract base class for all following `Image` models. Contains common attributes required for all images (such as the height, width and path to the image)
- `Image`: The model for storing the original image as uploaded by the users.

- `DerivedImage`: An abstract class for all images that have been modified. This model inherits from the `Image` model and adds additional information about the performed image transformation and a reference to the `Image-instance` it was derived from.
- `TempImage`: Inherited from `DerivedImage`. `TempImage` instances are temporary and have a limited lifespan. They may be disposed of after their expiration date has passed. `TempImage`-instances are created during the puzzle's creation process.
- `ShoppedImage`: Inherited from `DerivedImage`. `ShoppedImage`-instances are clones of `TempImage` instances that have associated with a `Puzzle`-instance. `ShoppedImage` instances do not expire.

#### 3.2.3.5. Anatomy of a Puzzle Application

In the context of the project, every puzzle implementation, or puzzle type, is implemented as an application. These apps follow the naming convention of `X_puzzle`, with X being the name of the puzzle, i.e. `dragdrop_puzzle`.

Every Puzzle application must realize certain requirements in order to seamlessly integrate into the core of the *StreetDroids* project. These requirements are:

- a `models` module with exactly one model which inherits from `core.models.Puzzle`. This is the central concept of the puzzle application.
- an `editor` module which provides
  - the variable `EDITOR_FORMS`: a list of `Form` instances which are included in the Puzzle creation process
  - the function `create_xml_from_input()`: required to return a valid XML fragment of the data collected from `EDITOR_FORMS`
- a `webservice` module which provides
  - the function `new()` which creates a `Puzzle` instance (and all supporting elements from an XML document)

Furthermore, most default behavior (such as which template is rendered by what view) can be customized by overwriting the default implementation.

Please refer to the sections 3.2.4 "Client-server communication and server API" and 3.5.2 "Technical details and structure of editors" for a detailed explanations of the members above.

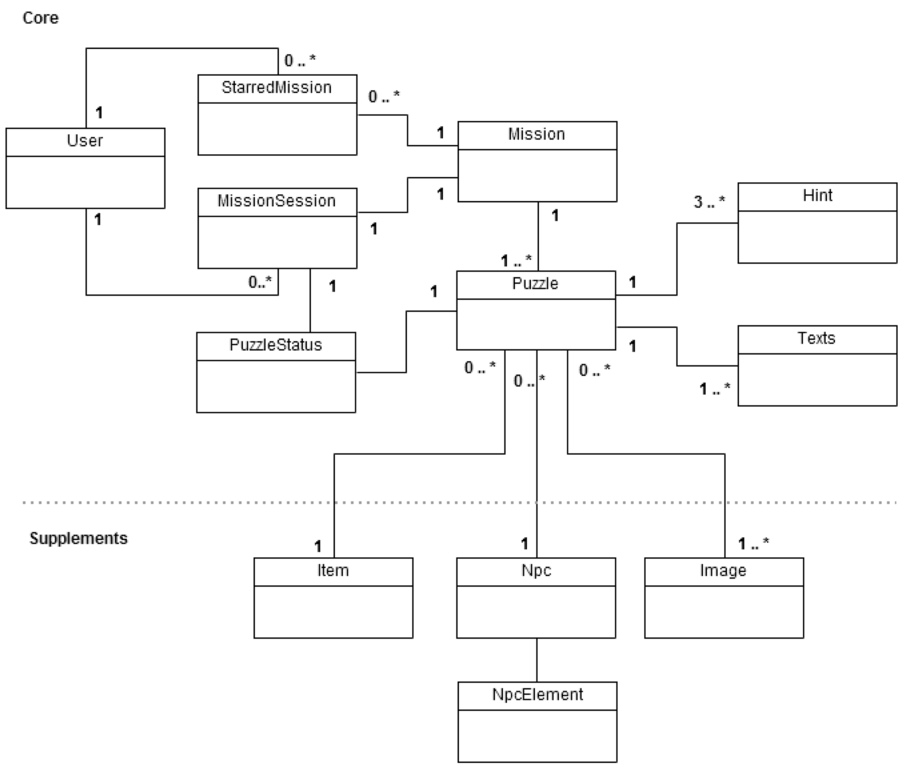


Figure 3.5.: The diagram highlights the relationships between the classes of `core` and `supplements` and has been simplified in that regard (for example all `Image` models are represented by one class)

### 3.2.4. Client-Server Communication and Server API[28]

[28] Till Hennig, Yarik Sheptykin

The term "webservice" consists of the two terms "web" and "service". A "service", in the context of computer science, is an offer of some kind, that can be consumed by a client. Usually data or an interface to other functionality is offered. The prefix "web" refers to the way the service may be accessed, meaning it is accessible using the Internet.

The fact, that the service and the client are meant to be on two independent systems that are only linked by a network connection, requires a concrete specification of the methods and the formats of that data that can be exchanged between the actors.

Moreover two independent systems are designed and supported by different developer teams, therefore a concrete methods specification is needed also for a common system understanding among developers.

Webservices offer the following advantages:

**Interoperability:** The system architecture is adaptable and does not need to be homogeneous, because specifications for the communication are transparent and platform independent, systems based on different operating systems and programming languages are able to communicate with each other via a commonly agreed protocol. Either side of the the communication can be exchanged seamlessly as long as the the protocol is followed.

**Distributed Computing:** The interoperability between system enables distributed computing, i.e. the delegation of tasks to other computers, so called "nodes". Using external nodes is useful in those cases, where the requesting node is lacking hardware resources (processing capabilities, storage space, etc), such as mobile devices. The client initiates the processing requests and receives the result, but is not involved in the calculation that happens, this task has been distributed. In the context of *StreetDroids*, this principle has been applied in the "Missing Piece"-Puzzle: Instead of executing the complex image analysis on the mobile client itself, it queries the webservice, which requires only a fraction of the time for the analysis, and then returns the value.

##### 3.2.4.1. The function of the webservice in the context of *StreetDroids*

As previously described, webservices enable the interoperability between systems. In *StreetDroids* this has been utilized in the following ways:

**Communication between mobile-frontend and server**

The primary function of the webservice is to respond to requests from the mobile client.

The exchange of data between the mobile-frontend and the webservice is essential for keeping the application and its content up-to-date as well as enabling context-awareness. The mobile device reports its current status (location, score, ...) to the centralized server which manages it and makes it available to the other clients.

The specification of requestable resources is discussed in detail in a later section; anything from text to binary files such as images and videos can be transferred. The information flow can either be from the client to the server or viceversa, however the communication initiator will always be the client (information pull).

The information served by the webservice is data whose inclusion in the install package is not reasonable. The reason is either that the data is generated on-demand and personalized for a specific user, for example game play data containing data specific to the user's location, or that the data file is too large in terms of file size (media files). In accordance to the game concept the content is one of the customizable parts of the game, therefore it cannot be included into a client installation package.

**Communication between web-frontend and server**

A webservice is also used during the Puzzle creation process. Data in a standardized format that is sent to the webservice is transformed into a puzzle. Puzzles can be created by a multitude of frontends - be it the implemented web-frontend or a standalone desktop application.

#### 3.2.4.2. Considerations

It was mentioned above that the PULL communication model is used for accessing the game data. This decision is considering the high mobility of game clients. With a constant location change their connectivity, and therefore a network address, changes along. Thus the server is not able to initialize a connection to a mobile client. Nevertheless, despite moving mobile clients can always reach the server when connected to the Internet, because the server has a static network address assigned to it. This discussion contributed to a high level communication protocol selection. It has been decided to use HTTP, the fundamental web protocol (Fielding et al., 1999), to exchange application data.

The communication between a client and a server system part is conducted by a specially designed protocol. The protocol was built on top of HTTP and uses its requests and result codes for exchanging documents. However, instead of HTML content sent traditionally over HTTP (Fielding et al., 1999) the designed protocol requires a payload to be formatted with XML. The XML allows to exchange data of any type, even a binary code (Bray and Paoli, 2008). It is also easy to parse and there are many libraries available which provide an XML parsing functionality[29]. Both features have high importance for the *StreetDroids* project. Support for any data type is necessary for the game content to download. For example, a puzzle content along with a simple text can include pictures and other media resources. The availability of the lightweight XML parsers is important for mobile clients which might be limited in their computational resources.

The communication design protocol introduced above defines a set of services or API a game server provides to its clients. It also establishes rules for accessing the protocol

[29]a list of parsing tools `http://www.xml.com/pub/rg/XML_Parsers`

services. The API and the accessing rules are discussed later in this section in more details.

#### 3.2.4.3. The RESTful *StreetDroids* API

REST stands for Representational State Transfer and is an architectural style - or "pattern" - for implementing web services. REST is not a standard (there is no REST specification) but a RESTful webservice often relies on existing standards of the web, such as HTML, XML, etc.

"The problem is, most of today's “web services” have nothing to do with the Web. In opposition to the Web's simplicity, they espouse a heavyweight architecture for distributed object access [...]. Today's “web service” architectures reinvent or ignore every feature that makes the Web successful.” (Richardson and Ruby, 2007) They introduce “... complexity [...], (which is) impossible to debug, and won't work unless your clients have the exact same setup as you”, for example by reducing HTTP "to a transport protocol for an enormous XML payload that explains what's “really” going on." This is greatly counterproductive in regards to the interoperability between heterogeneous systems.

Instead, RESTful web services aim to be "[...] amazingly, a simple, open (for now), almost universal platform for networked applications." by following recommendations described in the following chapter:

**Characteristics of RESTful Web Services**

REST web services can be characterized as:”

- Client-Server: a pull-based interaction style: consuming components pull representations.
- Uniform interface: all resources are accessed with a generic interface (e.g., HTTP GET, POST, PUT, DELETE).
- Stateless: each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.
- Cache: to improve network efficiency responses must be capable of being labeled as cacheable or non-cacheable.
- Named resources - the system is comprised of resources which are named using a URL. Interconnected resource representations - the representations of the resources are interconnected using URLs, thereby enabling a client to progress from one state to another.
- Layered components - intermediaries, such as proxy servers, cache servers, gateways, etc, can be inserted between clients and resources to support performance, security, etc.” (Costello)

While most of the characteristics have been taken into account while implementing the *StreetDroids* web service, one aspect that has been neglected by design is the fact that requests are stateless. The reason being, that the existing authentication mechanism of the framework that the web service is realized with relies on sessions for keeping track of the state of a user. In order to reuse as many of the existing framework functionality for handling users the decision was made to disregard that principle instead of implementing a solution that would fulfill that characteristic just for the sake of being conform.

**XML Format**

Above was discussed that the communication protocol uses XML to format the data being exchanged between remote system parts. The developed protocol defines also the format of the XML documents for each particular case of data exchange. The design of the *StreetDroids* XML format generally follows the guidelines of Google's XML Style guide. (Google, 2008)

The guidelines state that an "Attempt to reuse existing XML formats whenever possible" should be made. Reusing a standard was not possible due to the lack of standards for the specific purpose of *StreetDroids*: The requirement for highly customized data structures which are optimized for consumption by mobile devices, i.e. minimize the data transferred to a reasonable and relevant minimum.

However, some elements from standards were included and made "sensible use of the prescribed elements and attributes", as per recommendation 2.1. Geospatial information is represented in the GeoRSS-Simple format, which "is designed to be maximally concise, in both representation and conception." "Each of the four GeoRSS objects require only a single tag."[30] (Point, Polygon, Box, Line, Circle) which makes the format human-readable and allows the reuse of existing parsing libraries.

A recommendation for designing XML documents is to "use common sense and be consistent. Design for extensibility. [...]." This leads up to the question whether to use attributes or elements as the primary carrier for information. According to the author of Google's XML Style guide, "Attributes are more restrictive than elements, and all designs have some elements, so an all-element design is simplest - which is not the same as best." The self-conception of *StreetDroids* is to be a framework, which allows end-users to create content, and developers to extend the platform by adding new puzzle types. With respect to this fact a flexible, extensible, as well as easy to understand format was needed, which resulted in a decision in favor of a mainly-element based design.

**API**

In the context of the *StreetDroids* project the application programming interface (or API) is an interface to services provided by the central game processing unit. According to the software system architecture the API, or "game server", is the system component

[30] `http://www.georss.org/simple`

that provides services consumed by the game clients. Based on the game design and defined by the designed communication protocol discussed previously the API provides a list of resources detailed in table 3.1.

The calling sequence of API function is of importance. For example, a new puzzle location cannot be retrieved unless a mission has been started or resumed. Some resources are also privilege protected. For example, some of API functions are only accessible to authenticated users. These resources are indicated by the "Protected" column in the table above. A more thorough documentation for the publicly available API is published at the *StreetDroids* wiki.

#### 3.2.4.4. Server implementation

The core of the implementation for the webservice is located in the module `core.webservice`.

The XML-generation approach follows Django's "MTV"[31] - Pattern (Model, Template, View) - however, instead of rendering HTML templates that are destined to be viewed by the user, the webserver outputs XML from XML templates (located in in the "xml" subdirectory of the templates directory).

An alternative would be generating XML documents purely programmatically using a XML library. This would have the advantage of consistency checks during all steps of the process, but the implementation would be far more abstract and less visual than using templates. Due to the lack of programming knowledge this approach was not feasible.

Each of the accessible URIs (detailed before) is implemented by a method - the method must accept an request-instance as the first argument.

Some API URIs are accessible only when certain criteria are met: A requirement can be the HTTP-method used for the request, the payload of the HTTP-request or the existence of a valid user credentials. This functionality is implemented in decorators (in the module `core.decorators`) - the following are available:

- `permitted_methods(methods)` - takes a list of accepted HTTP-methods as argument - restricts access to certain HTTP-methods
- `validate(dtd_file)` - takes the path to a dtd-file as argument - validates the payload using the referenced dtd-file
- `exchange_plug()` - pass-through for currently active puzzle
- `login_required_404()` - checks whether the request has been made by an authenticated user
- `mission_session_required()` - makes sure that a mission session has been activated

[31]`http://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-\ \mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-\ \template-how-come-you-don-t-use-the-standard-names`

| URI | Description | Protected | Method | Response |
|---|---|---|---|---|
| /resources/ | Lists all available api resources | No | GET | 200: ReturnsXML<br>405: Method not allowed |
| /register/ | Used to register a new user | No | POST | 200: User was created successfully<br>400: Request not in expected format<br>405: Method not allowed<br>500: Username is already taken |
| /login/ | Used to log an existing user in | No | POST | 200: User was logged in successfully<br>400: Request not in expected format<br>405: Method not allowed<br>500: User does not exist or incorrect Password |
| /logout/ | Used to log an existing user out | Yes | GET | 200: User was logged out successfully<br>404: User is not logged in<br>405: Method not allowed<br>500: Attempting to logout without being logged in |
| /ping/ | Used to send a location update | Yes | POST | 200: Location saved successfull<br>400: Request not in expected format<br>404: User is not logged in<br>500: No active mission-session |
| /missions/ | Returns a list of all available missions | Yes | GET | 200: Returns XML<br>405: Method not allowed<br>500: |
| /sessions/ | Returns a list of all unfinished<br>mission-sessions for the logged in user | Yes | GET | 200: Returns XML<br>404: User is not logged in<br>405: Method not allowed<br>500: |
| /session/new/ | Creates a new mission-session<br>for the specified mission | Yes | POST | 200: Returns XML of first puzzle<br>400: Request not in expected format<br>404: User is not logged in<br>405: Method not allowed<br>500: No such mission |
| /session/resume/ | Resumes the specified unfinished<br>mission-session | Yes | POST | 200: Returns XML of first unfinished puzzle<br>400: Request not in expected format<br>404: User is not logged in<br>405: Method not allowed<br>500: No such mission session |
| /session/suspend/ | Suspends the specified unfinished<br>mission-session | Yes | GET | 200: Mission was suspended<br>404: User is not logged in<br>405: Method not allowed<br>500: No open mission session |
| /session/puzzle/location/ | Returns the location stub for<br>the currently active puzzle | Yes | GET | 200: Returns location XML of puzzle<br>404: User is not logged in<br>405: Method not allowed<br>500: No open mission session |
| /session/puzzle/data/ | Returns the complete puzzle data<br>for the currently active puzzle | Yes | GET | 200: Returns data XML of puzzle<br>404: User is not logged in<br>405: Method not allowed<br>500: No open mission session |
| /session/puzzle/skip/ | Skips the currently active puzzle | Yes | GET | 200: Returns location XML of next puzzle<br>404: User is not logged in<br>405: Method not allowed<br>500: No open mission session |
| /session/puzzle/solved/ | Marks the currently active<br>puzzle as completed | Yes | GET | 200: Returns location XML of next puzzle<br>404: User is not logged in<br>405: Method not allowed<br>500: No open mission session |

Table 3.1.: List of resources provide by the API.

**Special case: /puzzle/create/**

Most of the aforementioned API methods do not have any noteworthy implementation details, as they simple execute CRUD (create, read, update, delete) operations. The method `create_puzzle` which is accessible under the URI `/puzzle/create/`, is more complex and will be discussed in more detail. It is responsible for accepting data which will be turned into a Puzzle; or more precisely, it must unserialize XML data sent over an HTTP POST request in order to instantiate a concrete Puzzle subclass and all its supporting entities, and save them persistently.

All `Puzzle` classes share a common set of attributes - those inherited from the `Puzzle` class. However, in most cases the `Puzzle` subclasses will add its own attributes. The webservice is only capable of handling the common information, as the processing of the puzzle specific data needs to be implemented by the Puzzle application. The result is a webservice that is both generic and modular because the implementation details are consistently located within the puzzle application, and therefore satisfies the principles of being "pluggable".

When the `create_puzzle` method is called the XML document will be transformed into an object by `lmxl.objectify()`, which allows access to the XML document as if it were an ordinary object. The common bits of the XML document (such as the name, location, language, hints, etc), are casted from strings to their corresponding Python data types (with the help from the utility methods in `streetdroids.core.parse_utils`), and pre-populated into a `Puzzle` instance.

Because the XML document contains the type of puzzle that has been created, it is possible to call the function for handling the puzzle type specific data. The implementation of that functionality needs to reside in the module `webservice` of the puzzle package in the function `new()`. It receives the `Puzzle` instance as well as the objectified XML document as arguments and is responsible for casting the Puzzle instance to a concrete Puzzle implementation, and saving it as well as creating instances of all supporting models.

### 3.2.4.5. Client implementation

To allow the client to access the game content and the game flow control a game API client is required. The API client is an independent part of the client software that is able to communicate local application calls to the remote server using the specially designed protocol. The API client is implemented as an application library. It provides a local interface to the remote server functionality. Since the client software is being written in Java programing language the API client is also programmed in Java. The history of the API client development starts with the first game prototype implementation. Since then it has dramatically evolved. In the final application release the API client is expected to become a fast, reliable and transparent interface between the client game implementation and the game server. Along with a transparent API functions implementation the API client tends to provide tools for a comfortable work on transmitted data. An API client package includes classes that help to convert formatted XML data into game objects. It also includes functionality for the base64 encoding and XML parsing.

The server API in the client application is defined as a Java interface of server functions. All the local API calls are done using that interface, which makes the client architecture design very flexible - an implementor of the API interface can be easily replaced even at runtime. It allows the client to stay compatible with several versions of the communication protocol at the same time, if there is ever a need for it. This feature promotes an easier further enhancement of the protocol.

Implementing the API interface the API client needs a way to access networking to communicate with the game server. The latest client implementation uses an Apache HTTP client package for this reason. The Apache HTTP client has been chosen because of its high quality implementation of the latest HTTP features (ApacheSoftware-Foundation, 2008). Moreover, it is included as a default HTTP client into the Android mobile framework[32], which the current client application is being developed for. The Apache HTTP client takes care of an entire HTTP communication part, and therefore allows the API client to concentrate on the communication protocol implementation. Among many other features the Apache client supports encryption[33], which can become important if it will ever be decided to protect the API from public access.

The latest API client source consists of one Java class that implements API functionality defined in the API interface and several helper classes. The helper classes include API exceptions family and tool classes for retrieving data from exchanged XML documents. For paring XML content the tool classes use an official Java implementation for the DOM parser standard provided by the Android mobile framework[34]. The DOM parser has been chosen because of its ability to convert an XML document into a document object model[35]. This gives an easy access to any document node at any time, and simplifies navigation and search through a node structure. There are tough arguments against DOM application in the mobile client code. The DOM parser can allocate a considerable amount of memory to build a document object tree. Unfortunately memory on a mobile client is one of the highly limited resources in the Android mobile framework[36]. Considering this fact the DOM parser can be replaced by a less resource consuming parser. Nevertheless the DOM parser will remain for a while because it is the easiest tool to work with XML documents among others included. The client software developers team consists of students with different programing experience therefore the ease in this particular case is a very influencing argument.

Though the API client included in the latest game client implementation has being actively developed it still lacks a support for some API functions of a low implementation priority. These functions are subject to development and are expected to appear in the

[32]"org.apache.http - The core interfaces and classes of the HTTP components" `http://developer.android.com/reference/packages.html`

[33]"Supports encryption with HTTPS (HTTP over SSL) protocol" `http://hc.apache.org/httpclient-3.x/features.html`

[34]Provides the official W3C Java bindings for the Document Object Model, level 2 core. `http://developer.android.com/reference/packages.html`

[35]The parser converts XML into the DOM `http://www.w3schools.com/dom/dom_parser.asp`

[36]"Don't allocate memory if you can avoid it" `http://developer.android.com/guide/practices/design/performance.html`

future library releases.

## 3.3. Puzzle implementation

### 3.3.1. General Aspects[37]

[37]Felix Oey, Hima Bindu Vudathu

Several ideas have been considered for developing different kinds of puzzles. After a thorough brainstorming three different puzzles were decided to be implemented: Hotspot puzzle, Drag and Drop puzzle, and Missingpart puzzle. In the following sections the generic aspects of the puzzles (what they are, where they are located, and how they are implemented), will be described. Initially puzzles were implemented as different individual puzzles. Later as the project group came up with the idea that more puzzles can be developed by an online community, it was decided that there should be a general way of implementing any puzzle with adaptations made to it. Therefore, a generic puzzle class, which has different methods that can be implemented by the specific puzzle classes as per the specific puzzle, was developed.

#### 3.3.1.1. GenericPuzzle

Each puzzle inherits from a generic class called `PuzzleActivity`. There is also another generic class called the Npc, which will be described later. The `PuzzleActivity` class is an abstract class which extends from the `Activity` class of the android.app package. This `PuzzleActivity` class also implements an interface called `PuzzleActivityInterface`. The interface has different methods which are implemented by the abstract `PuzzleActivity` class. First, the `PuzzleActivityInterface` is described. Later, the `PuzzleActivity` class will be described and the other generic classes like `Npc` and `Puzzle` will be described.

#### 3.3.1.2. PuzzleActivityInterface

The interface has several common methods which are implemented by the generic `PuzzleActivity` class. These methods can be overridden by the specific puzzle class to alter the behaviour as it suits to the specific puzzle. The method's signatures in the interface are shown below. The functionality of the methods is described in the `PuzzleActivity` class in the next subsection.

```
public void showItem(Item I)
public void showPuzzleItem()
public void showNPC()
public void hideNPC()
public void sayFeedback(String text)
int getCoinsLeft()
void onPlayPuzzle()
void onPuzzleSkipped()
Puzzle getPuzzle()
public void onWrongTry(String wrongTryText)
public void onWrongTry(String wrongTrySpeech)
public void wrongTry()
public void onRightTry(String rightTryMessage)
public void rightTry(String rightTryMessage)
public void rightTry()
public void onHintRequested(int coins)
public int giveHint(PuzzleHint hint) throws
    PoorPlayerException, NoMoreHintsExceptions
public PuzzleHint giveHint() throws PoorPlayerException,
    NoMoreHintsExceptions
public void solved(String contgratulationsText)
public void onGameOver()
public void givePlayInstructions()
```

#### 3.3.1.3. PuzzleActivity

This is the generic class which implements the `PuzzleActivityInterface`. It groups all the functions and methods that are common to all puzzles. These are described below:

- having the start screen
- having the game screen
- having the item screen
- handling of NPC events and buttons
- having a bottom menu on the game screen
- showing the alert dialog when the puzzle is about to be skipped

The methods in the interface are described below:

**`public void showItem(Item i):`**
This method displays the item which the player might receive after successfully solving the puzzle. Each puzzle has a different item, which is received on successfully solving the puzzle**.**

**`public void showPuzzleItem():`**
This method displays the screen with the item received after solving the puzzle.

**`public void showNPC():`**
This method pops up the NPC of the specific puzzle to guide the player in solving the puzzle.

**`public void hideNPC():`**
This method hides the NPC after the information of the puzzle is shown on the screen.

**`public void sayFeedback(String text):`**
This method gives feedback as the puzzle is played. It provides information that the player has solved the puzzle successfully if the puzzle is solved. If the puzzle is not solved, this method provides feedback of the same.

**`int getCoinsLeft():`**
This method is called when the player buys hints in trying to solve the puzzle. The method returns the number of coins the player currently has.

**`void onPlayPuzzle():`**
When the player goes from the introduction screen of the puzzle explanation to the exact puzzle screen, then this method is called.

**`void nonPuzzleSkipped():`**
When the player does not want to play the puzzle at all, or he gives up after a certain number of tries, this method is called. The player has to click YES when the puzzle skip dialog screen is displayed.

**`Puzzle getPuzzle():`**
This function gives the developer to get a pointer to the data that has been downloaded for the current puzzle.

**`public void onWrongTry(String wrongTryText):`**
This method is called by the generic puzzle activity class when the player tries to solve the puzzle but fails. First of all, the implementation of this method in the generic puzzle activity class decreases the number of available tries. If the player still has free tries the NPC pops up and tells the player the text given as a parameter. In case the player is out of free tries, the application checks if the player is still able to proceed (if he has enough coins and hints). The player is shown a NPC that gives him the possibility to buy a hint. If the player can, the method locks the puzzle which forces the player to buy a hint. Otherwise the game is considered as over and the `onGameOver()` method is called after the player hides the NPC.

**`public void onWrongTry(String wrongTrySpeech):`**
This method redirects to the `onWrongTry` method. The method is invoked with a given text parameter.

**`public void wrongTry():`**
This method is the same as the one described above with just one difference. Here the text displayed by the NPC is the default text which tells the player the number of free tries that are still left.

**`public void onRightTry(String rightTryMessage):`**
This method is called every time the player makes a correct try in the game. The implementation of this method in the generic puzzle activity class displays the NPC, who gives a feedback to the player about the correct action.

**`public void rightTry(String rightTryMessage):`**
This method simply redirects the call to the `onRightTry` method and sets the message the NPC is going to tell the player.

**`public void rightTry():`**
This method calls the `rightTry` method with the generic positive feedback message.

**`public void onHintRequested(int coins):`**
This method is called when the player clicks the hint button on the NPC menu to get a hint for solving the puzzle. The implementation of this method in the generic puzzle activity class gives the player a random hint if the player has enough money to buy one.

**`public int giveHint(PuzzleHint hint) throws PoorPlayerException, NoMoreHintsExceptions:`** This method displays the NPC showing a hint in the textbox. The method might throw an exception when the player does not have enough coins to buy a hint. This method deletes the given hint from the puzzle hint array because the same hint can't be used twice in the puzzle. It might also throw an exception when there are no more hints available.

**`public PuzzleHint giveHint() throws PoorPlayerException, NoMore HintsExceptions:`** This method gets the first hint from the puzzle hint array and shows the hint to the player. The hints are filtered by the price, which means that only the hint for which the player has money for is being chosen and displayed.

**`public void solved(String contgratulationsText):`**
This method is called when the puzzle is successfully solved, and a screen with the NPC congratulating the player is displayed. When the NPC is hidden the puzzle intro screen is displayed and the next puzzle instruction screen is shown to the player.

**`public void onGameOver():`**
This method is displayed when the game is over. The game will be over when there are no more hints available, and the player cannot buy any more hints and has no way to solve the puzzle. This method is not called when the puzzle is skipped.

**`public void givePlayInstructions():`**
This method displays the NPC with a little bubble box that contains the instructions of how to play the game.

#### 3.3.1.4. Puzzle class

There is also another class called the `puzzle` class. This class is the base class for all the kinds of puzzles present in *StreetDroids*. The class has a constructor and methods which are used for downloading the puzzles and parsing the data. The functionality of the constructor and methods are described below:

**`public Puzzle(XmlDocument xml, GameHttpClient httpClient) throws RequestFailedException:`** This is a constructor which creates an instance of the puzzle from an XML document received from the server. Most of the binary data used in the game is not encapsulated into the XML document, so the parser requests the HTTP client to download that data from the server using links found in the document.

**`protected void parse(ParsingContext pc) throws RequestFailed Exception:`** This method sets the values of internal variables to the ones fetched from the XML document.

**`protected void parseScreens(ParsingContext pc) throws Request FailedException:`** This method is used to read the data specific to different puzzle screens. The data is fetched from the XML document where the current tag of the parsing context points to.

**`Protected void parseHints(ParsingContext pc):`**
This method is called to parse the data stored in the hints tag in the puzzle data xml document.

**`protected void parsePuzzleContent(ParsingContext pc) throws RequestFailedException:`** This method is called to parse the puzzle data. The implementation from the generic class does not do anything inside this method as it does not know how the data is being parsed. The child class needs to override this method and parse the data as required for the puzzle.

**`protected void parseMeta(ParsingContext pc):`**
This method parses the meta tag from the puzzle xml document.

**`protected void parseItem(ParsingContext pc) throws Request FailedException:`** This method is used for constructing a new object from the XML data.

**`public int useFreeTry()throws NoFreeTriesException:`**
This method decreases the number of free hints. If no more free hints are left, an exception is thrown.

**`public PuzzleHint useHint(PuzzleHint hint)throws NoMoreHints Exceptions:`** This method fetches a hint from the array of hints and returns back. The used hint is removed from the hints collection. The method Public boolean `canBuyPlayHint(int money)` checks if there are more coins left for the player to buy hints.

In some of the above methods a `RequestFailedException` is thrown. This represents an exception which occurs when working with the client. Such an exception is thrown by a GET or POST method when network operations cannot be completed. The main logic of the game does not care what exactly went wrong in network operations, such a type of exception is thrown when any problem with parsing or downloading of data occurs.

#### 3.3.1.5. NPC package

This package contains generic classes which are related to the non-playable character, like representing the NPC, the view of the NPC, the `npcviewevent` and the `npcviewevent` listener.

In the following subsection the goal, location, and how the specific puzzles are implemented are described in a brief manner. Every puzzle has an Activity with the name of the puzzle which extends the generic puzzle class and also another class which includes the logic specific to that particular puzzle. For example the Drag and drop puzzle has a class named `DragndropActivity` which extends the generic abstract class `PuzzleActivity`. The Drag and Drop puzzle also has another class named `DragndropPuzzle` which has the logic specific to dragging and dropping parts of images in order to solve it.

#### 3.3.1.6. Drag and Drop Puzzle

This is one of the three puzzles which have been implemented. As the name indicates the objective of the puzzle is to drag a piece of image present on the top of the screen and place it on the correct location on the picture present below it. The location in which the puzzle was implemented by the project group is the Town Hall (Rathaus) located in the city center of Bremen. Other locations in the city of Bremen or elsewhere, together with different content, can also be used to implement such a type of puzzle in future.

The puzzle is implemented by using the generic abstract class and also other classes which aid in developing the logic of the puzzle.

#### 3.3.1.7. Hotspot Puzzle

This is the second type of the three puzzles which have been implemented. The goal of this puzzle is for the player to find a certain place on the picture shown by touching at various places on the screen. In the hotspot puzzle implemented by the project group the location of this puzzle is the Bremen market place, and the picture of the Roland is shown to the player. The player has to locate the exact spot on the Roland which is related to the granting of market rights to the city.

This puzzle also extends the generic abstract class and also has two other classes. One class called `HotSpotPuzzle,` which has code related to the logic of the puzzle, for example, where the image is to be touched, what happens when a wrong spot is touched. The other class is the `HotSpotImageView` class which describes methods which show the area where the player should touch. Other locations or images can be used to implement another puzzle of the same type just as in the case of the drag and drop puzzle.

#### 3.3.1.8. Missingpart Puzzle

This is the last type of implemented puzzle. The goal of this puzzle is to find the part of the image which is missing from the screen by using the mobile device's built-in camera to take a picture. This picture is then sent to the server where it is compared with the answer picture. After the pictures are compared, feedback is sent to the player telling if the picture taken was right or wrong. The location of this puzzle, as implemented by the project group, is the Schlachte. Again the location can be not only the Schlachte but other places as well to implement another puzzle of the same type.

The puzzle has two classes: One is the class which extends the generic abstract class and the other one is the `MissingPartPuzzle` class which contains code to take a picture and send it to the server. The server receives the image and compares it using image recognition, and if correct sends the entire correct image to the client, and if wrong the player has to try again. The main logic of this puzzle is the image recognition part which is explained in section 3.3.2.

#### 3.3.1.9. Adding a new puzzle

If anyone wants to implement new puzzles there are two main classes to be created. One class extends the generic abstract class, and the other contains logic specific to that puzzle. The developer can also separate the view from the logic and create a new class for the view. It is up to the developer to decide how he wants to implement the puzzle. It is important to understand how to extend the generic `PuzzleActivity` class and override the functionality of the methods in such a way that will suit a specific puzzle.

Developing a new puzzle can be explained with the help of an example. If a developer wants to develop another puzzle of the type Missingpart puzzle, basically two classes have to be developed. One class extends the Puzzle class, it handles the initial communication part. While the player is moving to the location of the puzzle the downloading and parsing of data from the web server is done. Here, the initial image's width and height are downloaded as the player is going to the location of the puzzle.

The second class is the `Activity` class, which extends the `PuzzleActivity` class. Here the logic of the puzzle is coded. The `onPlayPuzzle()` method has to be implemented for initialization of the puzzle, i.e. preparing the holders and setting the image for the puzzle. This class makes use of the generic `NPC` class for various actions, like popping the NPC to guide the player in solving the puzzle, hiding the NPC when the player is solving the puzzle, providing respective feedback when the player solved the puzzle, etc.

### 3.3.2. Technique for Image Recognition[38]

[38]Alexander Tyapkov

This section describes the technique used for image recognition lying behind the Missingpart puzzle. The best way to describe the puzzle is to look at it from the user's point of view. Coming to the right location shown on the mobile device the user receives an image with a missing part. The goal of the puzzle is for the user to explore her surroundings and find the part of the image that is missing, then take a picture and eventually receive positive or negative feedback from the application about the correctness of the answer. The weak part of the puzzle's flow described above is the server side, which recognizes the image by comparison with the original. The stated task was to create a stable algorithm allowing to compare two images and to receive low-error response about the level of their similarity.

#### 3.3.2.1. Implementation

**Libraries and existing algorithms**

After the problem was defined the working team began to investigate available libraries which could help to solve it. Time limitation and lack of experience in image recognition techniques forced the team to find already existing algorithms and integrate them into the puzzle. While working on another type of puzzle, the jigsaw puzzle, where the player should connect pieces of a split image, the project team used the Python Image Library (PIL). Unfortunately, the functionality of this library covers only the basic operations for image processing. That was not enough for the project's purposes and the research was continued. Eventually it led to the Open Source Computer Vision Library (OpenCV), which is an open source library written in C and C++. The library has a rich functionality, and is considered to be one of the best libraries for working with images. Additionally it has the necessary set of methods allowing comparison of images using a variety of techniques. Two of such techniques were studied, and then integrated into the puzzle.

The first technique is based on the comparison of an image's histograms. As the OpenCV tutorial says "histograms are a classic tool of computer vision and find uses in many applications" (Gary Bradski, 2008) . Histograms are simply collected counts of underlying data organized into a set of predefined bins. They can be populated by counts of features computed from the data, such as gradient magnitudes or any other characteristic. The idea standing behind this technique is simple and based on the comparison of image histograms using different options. The comparison of histograms was used in our puzzle while counting the entropy of the images. That was done using PIL library.

Another method which was considered to be used in our algorithm is template matching. The tutorial mentions that "it is not based on the comparison of the histograms; rather, the function matches an actual image patch against an input image by sliding the patch over the input image using one of the matching methods". In order to use the described method we have imported the OpenCV library into our project.

**Implementation details**

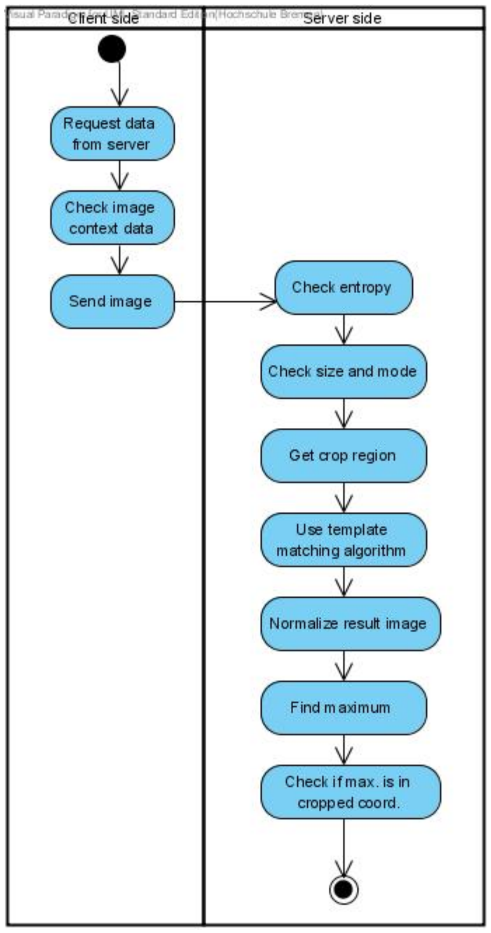


Figure 3.6.: The activity diagram illustrating the main steps towards image comparison.

The illustration of the developed algorithm using a simplified activity diagram is presented in figure 3.6). For simplicity the diagram shows both server and client side actions and does not show the decision boxes. Some of the steps were not programmed and should be considered only as a way to strengthen the algorithm.

Before the user can launch the puzzle, the mobile device, or in other words the client side, should request the necessary data from the server. This data includes the image in png format with the cropped area the user should find, location of the puzzle consisting of x and y coordinates, activation polygon and accelerometer data of the corresponding image. All the information about the puzzle is stored on the mobile device and no more requests are required until the image has to be checked.

Initially, the algorithm functions were split between the client and server sides. This allows executing some check procedures on the client side before sending the image to the server, what saves internet traffic and eventually user's money. At this step we are grabbing the accelerometer's data when the user takes a picture, then comparing it with the accelerometer's data attached to the missing part image, which was received from the server. The idea lying behind the comparison is based on the fact that the original image has only one set of accelerometer coordinates what means that one image can be made only from one position.

The implementation of this part of the algorithm causes another problem. All the images received from the server should have a set of attached coordinates, showing from which position the picture was made. In order to handle it a small application which loads the user's images from the mobile device straight to the server's image gallery was developed. In this case the creator of the puzzle can use the mobile device as a tool to attach the necessary context information to the images.

After the described procedure the picture sent from the client side is checked to see if it was taken in the correct location and position. This increases chances that the picture corresponds to the original one. Nevertheless, the checking of the image is continued on the server side. The first step to approve the image is to calculate the entropy and compare it with the original entropy. Entropy in this case is a measure of chaos which can be calculated using existing functions based on the comparison of the histograms. Comparison discards all dark or overexposed images, as well as all the rest of the images which do not have a correspondent level of entropy.

The next step is to prepare the user's image for further image comparison and check if the size and mode correspond to the original ones. If differences are detected the algorithms crop the user's image and changes the mode. In this case the possibility of a positive feedback decreases because of the size adjustment. Accordingly, the best results are shown when the size of the original image is equal to the size of the image saved in the database through the web-editor. The developed application allows making pictures on the mobile device, which helps to solve the problem, because the size and the ratio of original and the user's images are equal.

Coming back to the used algorithms, it should be noticed that the template matching algorithm needs a patch that will be sliding over the checking image. The patch is cropped from the original image and the image with the "hole" by overlapping them. Another solution was to not save the original image into the database, but only the patch. Nevertheless, it was kept for testing reasons. In order to proceed the patch should be passed to the image recognition function provided by the OpenCV library, which in turn returns a gray-scale image. Depending on the used algorithm the most white or black areas of this image show the regions with the biggest similarities.

Received answer simplifies the task. The gray-scale image showing the maximums and minimums allows to check the dispersion of these regions and to reason whether the patch is a part of this image or not. We have decided to simplify the algorithms according to the requirements of the puzzle and check only if the point of greatest similarity is situated in the coordinates of the patch, and therefore check the similarities between the original and the user's image.

## Testing details

A testing cycle was used while developing the algorithm. It means that the core functionality was checked straight after it was programmed. The core of the algorithm

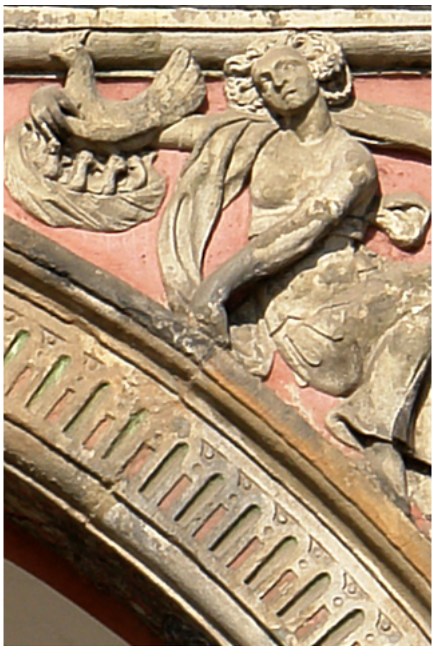


Figure 3.7.: Testing image

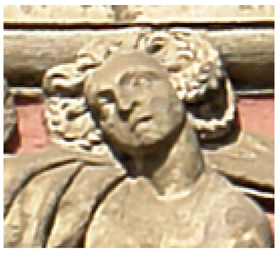


Figure 3.8.: Patch presenting the answer from a hypothetical user

are the functions responsible for the template matching algorithm, searching the maximum similarity and checking the coordinate's correspondence. All the additional components which make the image recognition algorithm stronger were tested in the same way, while they were being added.

The core functionality of the developed algorithm can be illustrated by a simple example. For testing reasons we will take any appropriate testing image and area cropped from it (figures 3.7 and 3.8). This area presents a hypothetical answer of the user received from the mobile client. By sliding the cropped image against the OpenCV function it returns a gray-scale image (figure 3.9) allowing to check if the patch is a part of the picture or not.

It should be noticed that the image matching function that is included in the OpenCV library provides a set of methods for image matching. The set of experiments showed that the normalized correlation coefficient matching method better fits to the algorithm

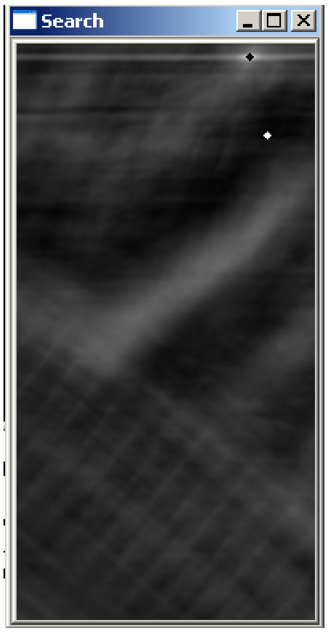


Figure 3.9.: Image returned after applying the function from OpenCV library with the point of similarity and difference.

used in *StreetDroids,* and also shows better results. All the later tests were done using this method.

The image returned after applying the function (figure 3.9) has a different resolution from the original and needs to be adjusted. Black and white points on the image show the points of highest similarity and difference correspondently. After calculating their coordinates it can be checked if they are situated inside of the cropped area. We assume that if the point of similarity is situated inside of the cropped area than it used patch is the part or the image.

## 3.4. Graphical interface

### 3.4.1. Technical Aspects[39]

[39]Felix Oey, Hima Bindu Vudathu

This section describes the technical aspects of the graphical interface in the Android client, which involve buttons, animation, orientation, picture format and drag and drop.

#### 3.4.1.1. Buttons

There are two types of buttons in Android: the regular button and the image button. Furthermore, both represent push-button widget and both are rectangle-shaped buttons. Another possibility to have a non-rectangle-shaped button is by using an `ImageView` and applying a listener class on the image. Therefore, three of them can be pressed or clicked by the user to perform an action. Thus, it gives the developer the freedom to handle a button event when a user press or click a button. (Google, 2010d)

The regular button displays a button with or without text on it. It represents a push-button widget where attributes like width, height, background color, margins of the button can be set. In addition, the size, font face, style, alignment, and color of the text can also be set.

Image button displays a button with an image instead of text. By default, an image button looks like a regular button with the default button background that changes color during different states. However, to remove the standard button background image the developer can define his own background image or set the background color to be transparent. Moreover, developers can also define a different image for each state to indicate whether the button is in a focused, selected, clicked, or pressed state. (Google, 2010f)

Image view displays an arbitrary image, such as an icon. Furthermore, the `ImageView` class can load images from a choice of sources (such as resource or content providers), taking care of computing its measurement from the image so that it can be utilized in any layout manager, and it provides many display options such as scaling and tinting. Moreover, to be used as a button, an action listener class must be linked with the image to perform the action. (Google, 2010g)

An event listener is an interface in the `View` class that contains a single callback method. The Android framework will call these methods when the object view to which the listener has been linked is triggered by user interaction with the item in the user interface. There are various event listeners such as `onClick`, `onLongClick`, `onFocusChange`, `onKey`, `onTouch`, and `onCreateContextMenu`. Each of them defines various events that can happen on an object.

**`onClick`** is an event that is called when the user either touches the object (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses the suitable "enter" key or presses down on the trackball. (Google, 2010n)

**`onLongClick`** is an event that is called when the user either touches and holds the item when in touch-mode, or focuses upon the item with the navigation-keys or trackball and presses and holds the suitable "enter" key or presses and holds down on the trackball for one second.

**`onFocusChange`** is an event that is called when the user navigates onto or away from the item, using the navigation-keys or trackball.

**`onKey`** is an event that is called when the user is focused on the item and presses or releases a key on the device.

**`onTouch`** is an event that is called when the user performs an action qualified as a touch event, including a press, a release, or any movement gesture on the screen within the bounds of the item.

**`onCreateContextMenu`** is an event that is called when a Context Menu is being built (as the result of a sustained "long click").

#### 3.4.1.2. Animation

Animation can be applied to views, surfaces, or other objects. There are alpha, rotation, scale, and translate animation. All of them are represented in the animation widget that handles tweened animations. (Google, 2010c)

Alpha animation controls the alpha level of an object, fading an object in and out. Moreover, this animation ends up changing the alpha property of a transformation, which defines the transformation to be applied at one point in time of an animation. (Google, 2010b)

Rotation animation controls the rotation of an object. Furthermore, this rotation takes place in the X-Y plane. The point to use for the center of the rotation has to be specified, where point (0,0) is the top left point in the screen. (Google, 2010k)

Scale animation controls the scale of an object. In addition, unlike the rotation animation, the point that is used in scale animation is the center point in the object. (Google, 2010l)

Translate animation controls the position of an object. Thus, the animation tweens an object from one point to another point in the screen. An x and y position have to be defined from the initial point of animation to the end point of animation. Hence, it will be a straight motion from the initial point to the end point in the animation. Moreover, a time value has to be set to define the speed of an animation. (Google, 2010m)

In this game, animation appears in the Non Playable Character (NPC) menu that appears to give information to the player along the game. That animation used is the translate animation where the character slides in and out of the screen.

#### 3.4.1.3. Picture format

There are several image formats that are supported in Android. JPEG(.jpg) image format is supported for both encoding and decoding, whereas GIF(.gif), PNG(.png), BMP(.bmp) are only supported for decoding. During the development of the game, Adobe flash (.swf) and scalable vector graphic (.svg) image format were not supported. (Google, 2010h)

#### 3.4.1.4. Drag and Drop

When an action takes place in the user interface, meaning that a user interacts with the screen, a listener listens to this action. Moreover, in the drag-and-drop puzzle, there are two main user interface listeners to handle the user event: the `touch` event and the `drag` event. In short, in the very top of the screen a listener of the class always stands by to monitor finger motion from the user. (Google, 2010i)

In the `touch` event whenever a user touches an object that is linked into this listener class (`MotionEvent.`*`ACTION_DOWN`*), the x and y coordinates where this action happened are firstly saved as initial parameters for later calculation.

In the dragging event, there is a parameter called *`DRAGGING_CONDITION_DISTANCE`* that is set into a certain pixel size to indicate whether the range of a motion is recognized as a dragging or just touching. Then, the listener checks whether the user's motion is surpassing this condition. Thus, if this condition is fulfilled, the event will be acknowledged as a dragging event. If not, it is just a mere touch event. In short, if the player touches the icon from the top panel and moves the finger down exceeding this *`DRAGGING_CONDITION_DISTANCE`* limit, the intention is treated as a dragging, so the dragging event of the piece starts.

The x and y coordinates of the piece that is dragged are also refreshed according to the user's finger position on the screen. Hence, the piece will move along according to the finger movement. Then, after the piece dragged into certain position, and the user release the touch, a listener senses this event (`MotionEvent.`*`ACTION_UP`*) and locates the piece on the last finger position when the user put the finger on the screen.

#### 3.4.1.5. Screen Orientation

*StreetDroids* has been implemented in the portrait mode orientation. For future work it is possible to extend the game to handle the accelerometer event orientation, both in the portrait and landscape modes. Therefore, another design layout for each screen would be needed. The designer just has to make another layout folder (for example layout-landscape) inside the resource folder and put the xml file with the same name from the layout folder in this other folder. In brief, each object positions in this other xml have to be relocated in the screen.

The layout for the camera class by default is in landscape mode, even the camera application from Android itself is developed in landscape mode. Thus, it is impossible to change this layout into portrait, because by doing this the image will be heavily distorted. In this special case the puzzle was implemented in landscape mode. However, further extension is also possible to have this puzzle work in portrait mode.

### 3.4.2. Design Aspects [40]

[40]Joatan Preis Dutra, Catalina Payán

This section explains how the principles and design guidelines of Human-Computer Interaction (HCI) and Graphical User Interface (GUI) were applied for the game *Street-Droids* in order to overcome the challenges presented when designing for small screens, and is organized as follows: In subsection 3.4.2.1 (Design Consideration for Small Screens) was elaborated on the problems faced by designers of applications for mobile devices. In subsection 3.4.2.2 (Design Concept for the Game) presents the guidelines offered by HCI and GUI theories which were used to tackle the design problems. It is discussed the research for the design of icons, menus, game interface, navigation and characters, experiments with colors and present the final choices. In section 3.4.2.8 it is placed the final considerations.

#### 3.4.2.1. Design Considerations for Small Screens

If on one hand current smart phones have computing power largely superior than high-end desktop computers of a couple of decades ago, on the other hand they are equipped with very small screens. Small screen size is, therefore, one of the major issues faced by designers of mobile applications. In this project, for example, mobile devices with 480x320 pixels of screen resolution was targeted. The main challenge was to find ways of displaying the needed information in a friendly manner, so that users have a pleasant experience while at the same time having complete access to the data and actions appropriate for a certain context. For this purpose, concepts of HCI and GUI design were applied to overcome usability issues and to explore the possible utilizations of the limited space at hand. The result was an iconographic and simple interface which was well received by the beta testers at the preliminary evaluation. It is possible to believe that the application of HCI and GUI theoretical design guidelines was essential to the development of a successful user-friendly interface with an intuitive navigation.

The interactive features present on smart phones, such as the touch interfaces found in increasingly affordable devices, give developers the opportunity to explore vast new possibilities. By using new technologies and devices (like the T-Mobile G1 Android[41]), new features are being developed, like context and location-awareness – one of the *StreetDroids* characteristics - which can be used to make games that are more connected to the real world and to the community around the user.

> *"As mobile phones move beyond telephony into areas as diverse as Internet access, personal entertainment, content creation, and interaction with so-called smart and pervasive computing environments, exciting new opportunities for intelligent auditory presentation behaviors arise. In recent pervasive-computing research, for instance, users intuitively navigated their way to undisclosed outdoor locations using a context-dependent, directionally adaptive auditory display (Etter & Specht, 2005). The underlying system uses global positioning data and a geographical information system to infer*

[41]The T-Mobile G1 was the chosen Android mobile device for the StreetDroids Project. Further information about this device is available in: http://www.t-mobileg1.com/

> *the mobile user's geographical context. Navigation cues are then rendered by adaptively panning and filtering music selected by the user to correspond with his or her direction of travel."* (Kortum, 2008 Pg 189)

The used approach to develop the design interface was based on HCI concepts since *"design, usability and interaction are recognized as the core issues in HCI"* (Ghaoui, 2006 Pg XIV). Furthermore, to work on small mobile screens is a challenging task. In order to enhance the interaction, studies on HCI and GUI were considered at the development process. For instance, the elements of the initial screen followed one of the principles of GUI design. According to Galitz, *"the array of alternatives available to the user is what is presented on the screen or what may be retrieved through what is presented on the screen - nothing less, nothing more"*(Galitz, 2008). From this principle, only what the user needs to know about the application and the available options should be presented on the screen, affecting how the design is developed.

Ronchi pointed that a well-developed interaction design could serve as bridge to fill the gap between man and machine, enhancing the interaction between them:

> *"The aim of interaction design is to close this gap (in man/machine communication) by bringing usability into the design process. This means developing interactive products that are easy, effective, and enjoyable to use from the users' perspective."* (Ronchi, 2009)

However, there are no easy rules to define which solution is the best when displaying content on the screen. Attention must be paid to aspects like intuitive usability, easy navigation, clear information, nice layout, harmonious colors and internationalization, also respecting the values of HCI/GUI in a well developed perspective. That is the base of the design concepts for the *StreetDroids* project.

*StreetDroids* was developed for Google's Android platform and, as mentioned, targets devices with 480x320 pixels screens. One of the features of the platform is the possibility of using the screen in a vertical or horizontal position. Some applications developed for Android can be used in both positions, which demands a flexible and adaptive design. That represents a change in the design paradigm since it is no longer possible to design for a single static position and view. It also makes it harder to predict patterns of user behavior because all the content might change according to the device's position. Taking this design challenge into consideration and looking for a solution that can clarify the structure of the game, the project fixed the vertical position for the game format. That choice considered also the most common position for handling a mobile phone. This decision not only facilitated the development, which was then carried out using a width of 320 pixels and height of 480 pixels, but turned the interactions clear and obvious for the user.

Once decided, it was necessary to keep in mind that the design for mobile requires special attention about the use and optimization for the small space available, where the choice of the displayed information is crucial for the navigation and performance. *"Designing for a mobile application is really quite a bit different than for desktop software. Limitations of the device itself, including screen real estate and user input methods, force us to make different choices."* (Bondo, 2009)

Another issue around screen usability is the visualization of elements and possibilities of navigation, where a minimum height and width should be considered. The displayed elements cannot be too small, considering that some users may have visual deficiencies. Also, the projected game has an outdoors context, which means that the lightness of the environment may interfere with the visualization as well. *"For small-screen interfaces, human factors designers face the challenge of displaying all the information they want to present, while making sure that what they present is not too small to be seen by the user."* (Kortum, 2008 Pg 324)

> *"While devices such as mobile telephones and MP3 players have continued to shrink, the problems with controlling and using these miniature devices have grown. From the physical ergonomics associated with using the systems to the navigation of tiny menus, the new systems have proven to be substantially different and more difficult, to use than their bigger brethren."* (Kortum, 2008 Pg 10)

These considerations were crucial for the design development of the mobile game. It is clear that the game could not be developed only by taking concepts of design of desktop applications and shrinking to a 320 x 480 pixels screen. It was sought to treat the mobile screen as an independent media, respecting its size limitations and exploring every pixel available in a user-friendly way.

The touch-screen feature is another factor that should be considered for screen design optimization, where the player uses fingers instead of a mouse or joystick. Fingers are less accurate and demand a larger area if compared to a pointer mouse.

> *"(...)the finger is not a mouse . On the desktop, a user can use a variety of input devices — such as an Apple Mighty Mouse, a Logitech trackball, or a laptop touchpad. But, on screen, the mouse pointer for each of these pieces of hardware is always identical in shape, size, and behavior. (...) Additionally, finger input does not always correspond to a mouse input. A mouse has a left click, right click, scroll, and mouse move. In contrast, a finger has a tap, flick, drag, and pinch."* (Wagner, 2008)

Some Android devices have a "trackball" (as the mentioned T-Mobile G1 (T-Mobile, 2009)) which works as a mouse. In the design process for the interactions, it was assumed that this hardware was not available, because not all mobile devices developed for the Android platform actually contain it. It was assumed users would use their fingers to interact with the game. In the next sections it is presented the solutions and concepts applied for the *StreetDroids* game.

#### 3.4.2.2. Design Concept for the Game

Albert Einstein once suggested to *"make everything as simple as possible, but not simpler"* (Kortum, 2008). Evidently he was not referring to design for mobile screens, but this axiom can be perfectly applied for it. An intuitive and attractive user interface is usually a familiar user interface. (Tidwell, 2006)

*"When you design user interfaces, it's a good idea to keep two principles in mind: 1. Users don't have the manual, and if they did, they wouldn't read it. 2. In fact, users can't read anything, and if they could, they wouldn't want to. These are not, strictly speaking, facts, but you should act as if they are facts, for it will make your program easier and friendlier."*(Spolsky, 2001)

These premises were applied when developing *StreetDroids* to support different types of open air scenarios. The main idea was to offer an intuitive interaction scheme. One of the features of the game is that users can create their own maps and characters through an online platform available at www.streetdroids.com. However, the design process started to be developed based on a default map and mission created using the Old City of Bremen as scenario, with a historical emphasis. This map choice, together with the Master Project's name (*mobileHIVE*[42]), were the inspiration elements for the visual elements, always aiming the user-friendly interaction.

*"The audience must be captivated, but at the same time, delivered an easy-to-use, well-polished title. The aesthetics of the game are important, but the functionality is equally critical. The app must perform flawlessly, or your judges will call you out."*(Hennessy and Kane, 2009)

Taking into consideration the aesthetics and functional aspects, the game starts with an iconographic main menu, where the user can access the main features and information about the game, such as Play Game, Language, Quit, Settings, Info/About and Help. Figure 3.10 shows a screen-shot of the main menu, where the icons are followed by the message: *"Please select one of the buttons"*. The logo of the game appears on top and disappears on the other screens when it is not needed, opening space for other elements.

### 3.4.2.3. Icons and Menu

Each button in the main menu has a hive capsule format, originally created as reference to the Master Project's name and logo. The hexagon format of each button fits in a well-balanced distribution on the screen, surrounding a centered image which symbolizes a mobile irradiating information, as shown in Figure 3.10. To enhance the meaning behind each button, it was chosen the use of icons.

*"The universal nature of icons avoids the idiosyncrasies of different languages. They also speed up the rate of input by removing inferable constituents of communication, such as prepositions. These advantages have made icons pervasive in modern computing systems and ubiquitous in communication and assistive aids."*(Ghaoui, 2006 Pg 298)

The menu shows two types of options: ENABLED (with ON and OFF status) and DISABLED actions. In Figure 3.10 the "Language" and "Settings" were disabled (not implemented, for technical reasons, in the first moment), and the others enabled. After the

[42]See "mobileHIVE" logo and description, at Appendix.

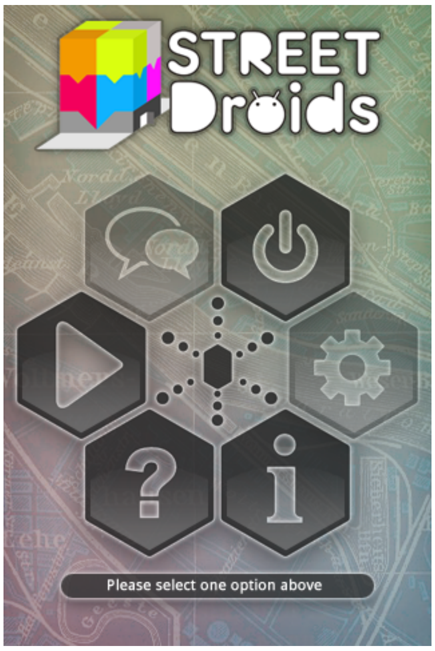


Figure 3.10.: Initial screen with the main menu and logo.

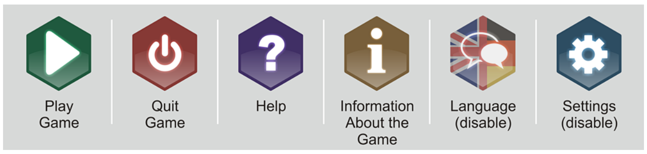


Figure 3.11.: Icons developed for the main menu.

user clicks in one of the enabled buttons, it changes to an ON status, with color and a centered white, bright icon, as shown in Figure 3.11.

The main navigation buttons were based on an iconographic concept, divided in three packages:

- Main menu icons (Figures 3.10 and 3.11).
- Puzzle icons (Figure 3.12). During a puzzle the user can ask for the Non-Player Character (NPC) help, which will offer some options as hints, information about the puzzle and option to quit.
- Navigation icons (Figure 3.13). These icons are used to help the player during the game, giving the status feedback about her performance, such as collected coins and items, and also offering the NPC help and accessing the compass.

Some research on the appropriate symbols to use for the icons was made, in order to get the proper metaphor for each action, maintaining the visual consistency for each icon group. The idea was to use the most well known concept for those actions and

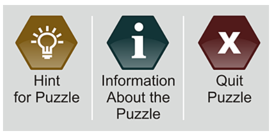


Figure 3.12.: Icons accessed by the NPC during a puzzle.

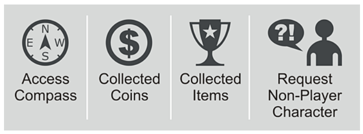


Figure 3.13.: Regular icons for the navigation during the game.

metaphors, allowing a meaningful interpretation even in a multicultural environment. Also, the research went further and also it was considered the way in which players interact and navigate with the game using their fingers on the touch-screen:

> *"User interface designs for touch screens must carefully consider the size of and spacing between touch-activated buttons and icons so that the user's inputs will be accurate. Usually, the larger the button, the easier it is for users to accurately point to it. But often, computer screen space is limited. Designs must trade off between button size and spacing that maximizes accuracy, and the ability to support the desired functionality for a given screen. (...) There was a significant difference among button/icon sizes. People performed best when it was equal to or bigger than 40*40 pixels."*(Sun et al., 2007)

This rule about the size of buttons was observed. The navigation menu in the bottom of the screen was directly influenced for this touch-screen factor, where the available buttons followed the minimum height of 46 pixels, with width changing from 99 to 56 pixels, according to the game scenario, as shown in figures 3.14 and 3.17a.

### 3.4.2.4. Colors

The composition of the design could be not considered complete without paying special attention to the selection of colors. The colors of the interface evoke answers and stimulation. *"One of the major challenges when working with color is finding a set of colors*

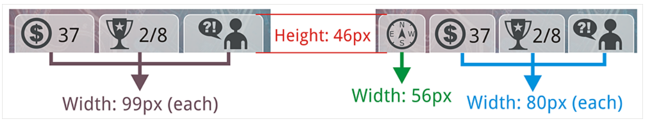


Figure 3.14.: Bottom Menus used on Game, during a Puzzle and Map, respectively. All buttons are above 40 x 40 pixels dimensions, improving the perception by the user.

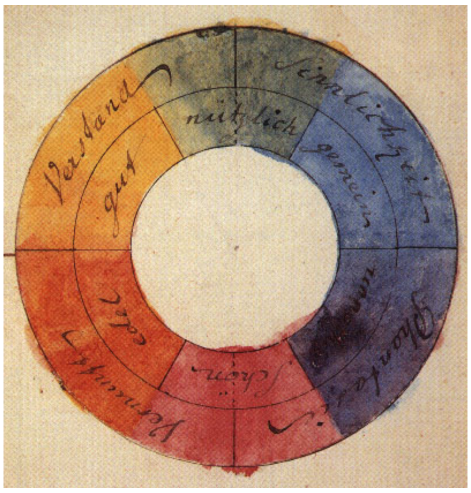


Figure 3.15.: Goethe's Color Wheel.

*that work well together. When colors look good together, the effect is often referred to as color harmony"* (Fox Pg 45). Sir Isaac Newton invented the first color wheel. He split white sunlight into red, orange, yellow, green, cyan, and blue beams, and then joined the two ends of the color spectrum together to show the natural progression of colors. Newton associated each color with a note of a musical scale. A century after Newton, Johann Wolfgang Goethe began studying psychological effect of colors (figure 3.15 (Irgtel, 2010)). Colors can be used to create illusions, sensations and luminosity in a layout, creating contrasts and harmonies in the screen. (Moore and Simpson, 2007)

The color wheel served as basis for further studies about the use of colors. One interesting result is about the proper combinations of colors, which can be verified using the wheel. By positioning a square inside it, the corners will point four colors which after mixed will result in Grey. (Fox, 2005 Pg 48). The Grey color can be considered as a neutral one, being ideal for background combinations. (KMBdesigns, 2010).

This balance of colors was used for the background colors at the *StreetDroids*' screen. This process is illustrated in figures 3.16a[43], 3.16b and 3.16c[44]. The square touches, in a clock-wise rotation, the orange, yellow-green, blue and red-violet areas. Extracting the tunes, they were applied in a soft version to the map that is the background of the game.

#### 3.4.2.5. Game Interface

Optimization of space, simplicity, metaphorical icons and symbols, and direct navigation were the key elements that guided the development of the game interface.

[43]Image modified from(Fox, 2005)Pg 48.

[44]The background image was taken from a selection on the map of the city of Bremen, dated from 1910, available at the online library of the University of Texas at Austin at http://www.lib.utexas.edu/maps/historical/baedeker_n_germany_1910/bremen_1910.jpg

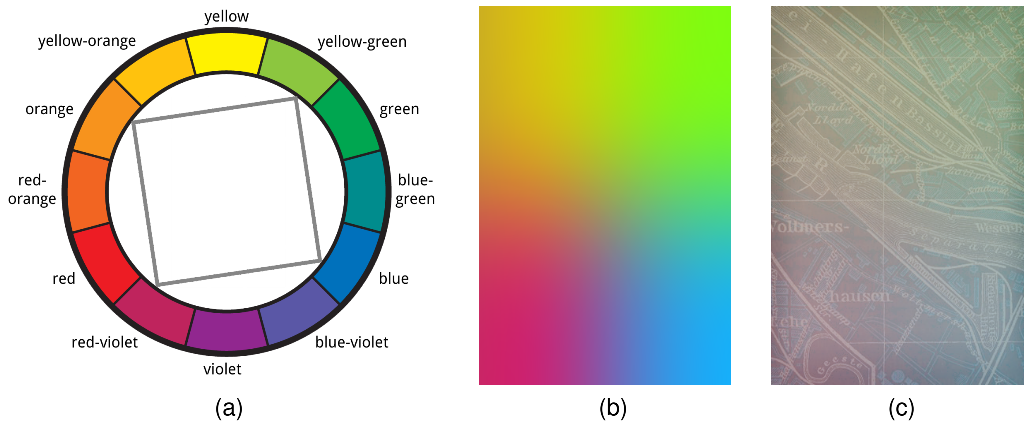
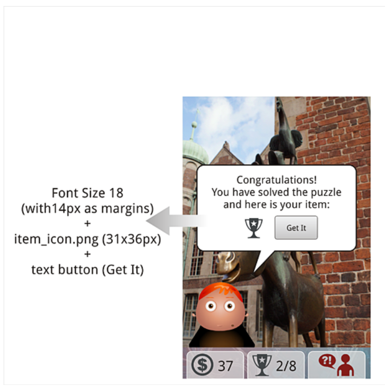


Figure 3.16.: (a) Selected Quadrant inside a Color Wheel. (b) Created color scheme based on the quadrant in Figure 3.16a. (c) Game Background with the colors applied above map layer with transparency property.

> *"Typically, a graphical user interface draws on a user's environment to provide a metaphorical representation of the user's tasks. A metaphor provides an analogy to concepts already familiar to the user, from which the user can deduce the system's use and behavior. Icons can express the metaphor directly, as graphical representations of the metaphorical objects. They may also directly represent a physical object. Icons are distinguished from other symbols on screens by the fact that they represent underlying system functions. Icons represent the objects, pointers, controls and tools making up the domain of an application and that users manipulate in doing their jobs. They can also represent status indicators used by the computer system to give information to the user and to mediate user interactions with software applications."*(ISO/IEC, 2000)

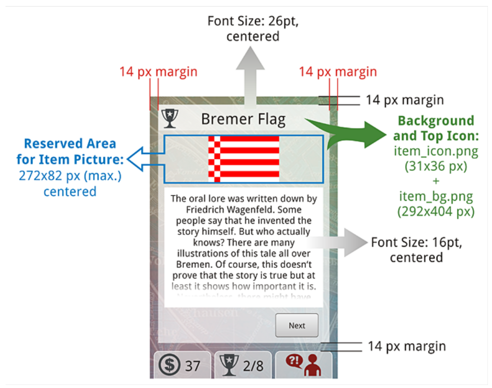
The metaphorical elements were not exclusive for the navigation buttons, but also applied to the entire interface, including background, NPC and action feedback during the game. One example is how the NPC "talks" to the player using the metaphor of comics' bubbles (balloons), as shown in Figure 3.17a. More than just offering a functionality, it is necessary to coordinate the graphic elements in order to provide a compatible and comprehensive visual rhetoric. In this way, the use of familiar elements helps to enhance the comprehension of the situation by the player. In this example, it is clear that the text displayed on comics' bubbles mean that the NPC is talking directly to the player. In Figure 3.17b, on the other hand, the feedback uses a different approach, showing a gained item with its description.

Scrollbars were avoided in the developed design. In some situations, however, they proved necessary, being introduced with a "fade out" effect, as shown in Figure 3.17b. *"To create a better mobile experience, follow these guidelines: - Simplify everything. Use clear, short, simple words for links, buttons, and menus"* (Frederick and Lal, 2009).



Congratulations!
You have solved the puzzle
and here is your item:

(a)



The oral lore was written down by Friedrich Wagenfeld. Some people say that he invented the story himself. But who actually knows? There are many illustrations of this tale all over Bremen. Of course, this doesn't prove that the story is true but at least it shows how important it is.

(b)

Figure 3.17.: (a) Tutorial example about navigation (and dimensions) after collecting a item. (b) Tutorial example about navigation (and dimensions) for visualizing a gained item.

It is important to point that one of the fundamental ideas of space optimization is that all required information should be displayed at the screen whenever possible.

> *"A user interface without a scrollbar is the best experience for the user. However, if the information is more than the display area, a vertical scrollbar is acceptable. Always avoid horizontal scrollbars. These provide a bad user experience in a small device and can be avoided using 100 percent-width."* (idem)

Despite the effort to provide meaningful items and other visual rhetoric components based on icons, text buttons are still necessary on the game interface. They are used for direct questions and actions, as shown in Figures 3.17a and 3.18.

> *"Labels and button labels should be clearly spelled out, with meaningful descriptions of the actions they will cause to be performed. Choices should be composed of mixed-case single words. Multiple words are preferred, however, to single words lacking clarity in their intent. If multiple-word labels are used, capitalize the first letter of each word (headline style). Use the same size and style of font in all buttons. The regular system font is preferred. Never change font style or size within buttons; these kinds of changes can be very distracting to the viewer. Center each label within the button borders, leaving at least two pixels between the text and the border."*(Galitz, 2008 Pg 408)

The text-buttons should use direct texts, expressing acts that could be easily understood by the player. Another reason to use simple and direct texts is the possibility to be translated in future language versions.

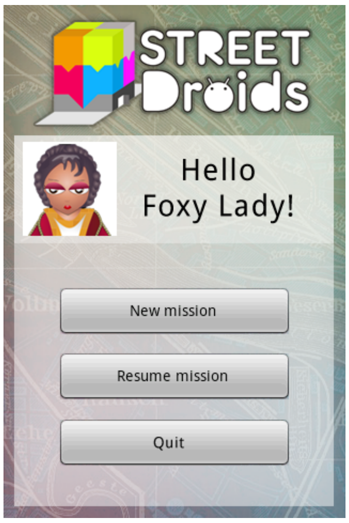


Figure 3.18.: Text buttons

#### 3.4.2.6. Navigation Design and Immediate Feedback

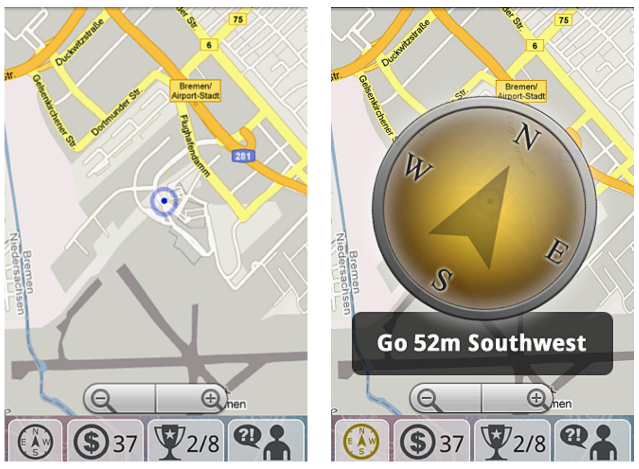
The use of icons and visual feedback can support the player to become aware of actions and follow consequences. *"You can improve the flow of your application by helping your user more quickly ascertain the meaning of your controls"* (Bondo, 2009 Pg 148). One way to enhance the controlling aspect is using immediate feedback on the navigation. This feedback can be done by image, text and symbols, as shown in Figure 3.17a and in the following examples.

One example is when the player accesses the compass during the game in order to find out where the next puzzle to be solved is located. Once the player clicks on the "compass button" on bottom screen, it immediately changes its color, showing that the function is active and displaying the compass with the needed information, as seen in Figures 3.19a and 3.19b.

Another subtle feedback is the color of the compass at the screen. It is available in three different colors: Green, when the user is close to the target of the puzzle (0 to 50 meters), Yellow if near (50 to 100 meters) and Red if far (above 100 meters), as shown in Figure 3.20.

Another example of immediate feedback presented on the game interface is when the player asks for help to the NPC during a puzzle. The icon (button) changes its color, as seen in Figure 3.21b, and after the choice is made, a floating-window shows up, describing the consequences of this choice and asking the user for confirmation.

Another way to give immediate feedback regarding the main menu is including label on the buttons as soon as they are clicked by the user.



Go 52m Southwest

(a) (b)

Figure 3.19.: (a) Map during the navigation toward the next puzzle. (b) Map during the navigation toward the next puzzle after the compass is accessed (with proper feedback).

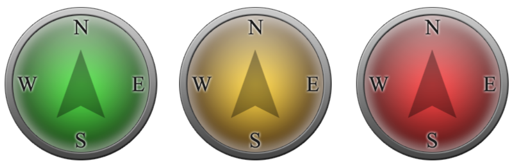
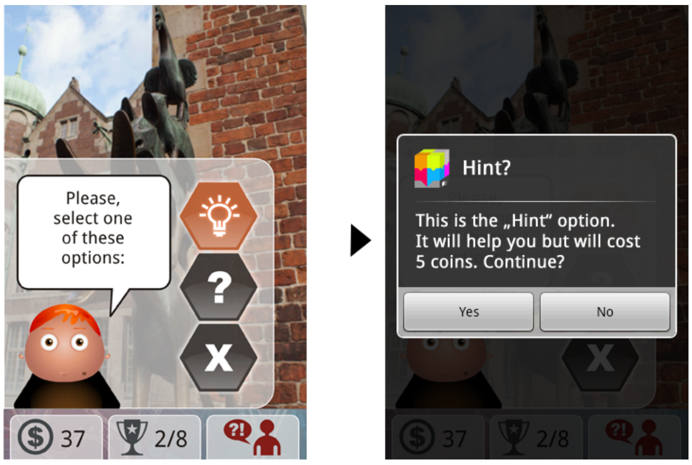


Figure 3.20.: The three stages of the compass, according with the distance from the next puzzle/target.

> *"One common way to enhance menus is by using icons. Originally, on desktop systems, icons were used to visually represent an object, or function, of the operating system (e.g. an icon to denote a document). Icons were used in menu systems to replace, or augment, text descriptions of functions. Pure replacement of text is rare – it's hard to pick an icon that unambiguously represents a function. More commonly, icons and text are used together in menus to reinforce an idea."* (Jones and Marsden, 2006)

This concept was followed with the labels in the initial screen, where a help button explaining the game and the used symbols is also available.

Observing the specific design developed for the *StreetDroids* as an academic learning process, it is evident that a careful theoretical review offered the proper support for efficient decisions. The iconographic navigation, allied with the immediate feedback, can be a powerful combination for usability and friendly navigation.



This is the „Hint“ option. It will help you but will cost 5 coins. Continue?

(a) (b)

Figure 3.21.: (a) Accessing the NPC during a puzzle. (b) Feedback about the chosen action.

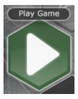


Figure 3.22.: Play Game button with label.

### 3.4.2.7. Characters

Designing for mobile games is a complicated task, mainly because of the physical limitations of the small screens and the technological challenges of different mobile platforms. One of the main goals in video game design is to entertain and engage the user. The entertainment aspect involves several aspects of design, including game story, pacing, challenge level, and game mechanics (Desurvire et al., 2004). However, it is also important that game designers pay special attention to the usability issues (Pinelle et al., 2008), because a failure in the usability game interface can interfere with the experience of the user and therefore have a negative effect on the quality and success of the game.

Our everyday life is filled with interaction that we make with characters real or fictitious, and the satisfying result of these interactions relies upon in the understanding of the character. People interact with characters in all aspects of their lives by watching a film, reading a book, playing a game, etc. Different researchers use a variety of colorful and evocative terms to describe the relations between the users and the figure we see in the game, for example: believable agents, life-like computer characters, synthetic characters, creatures, anthropomorphic agents, bots, NPCs, artificial intelligences (AIs), and avatars (Hayes-Roth and Doyle, 1998). They can transform a simple transaction into a memorable social experience by using their aesthetic properties and their behavior with both positive and negative currents.

One of the aims in game artificial intelligence is making more believable and social characters (Bailey and Katchabaw, 2008), in our case a NPC. One of the reasons for having believable social characters is the inclusion of the suspension of disbelief required to immerse a player in the game. Characters promote realism and create interesting game situations; they also support the game designer in the challenge of making a convincing game experience, help in the interactivity, player choice, and replayability (Reynolds, 2004).

In order to make the game experience more believable, the best method for designing a character is by emphasizing the use of physical properties (design) and physical behavior (Tychsen et al., 2008). This is achieved by adding personality to the character, making a balance between general properties of each character, and keeping the relatively generic properties, in order to avoid a negative impact of the personality on the player. The design of the NPCs can be divided in different components: character personality, integration, appearance, etc.

There are numerous and varying approaches to character design (Tychsen et al., 2008), characters can be predefined, or intentionally made for the interpretation of the player allowing the game story to be open to a variety of character personalities. Normally mobile games are defined by a cast of NPCs who act as enemies or partners, and help players providing challenges, offering assistance and supporting the storyline (Laird and van Lent, 2000). Also the conversational skills reflect certain capabilities that differ qualitatively from those who use the conventional natural language (Hayes-Roth and Doyle, 1998). In order to have a more engaging game the NPC must be a good conversation partner because it is the character who makes the story, and not the other way around.

In *StreetDroids*, the NPCs play the role of support characters, they can be merchants, tradesmen, guards, etc. who support the storyline of the game by giving the user tasks, hints, items or help. In this case the technology used to create these characters is based on the reflexive behavior (Merrick and Maher, 2006). This is a pre-programmed response rule-based to the state of the environment were only a recognized state will produce a response and define the behavior – a reflex without reasoning.

For the integration of the *StreetDroids* characters in the game many components were considered in the design, for example the location and background, and a clear definition of their appearance by giving special attention in their representation (how do they look), and their behavior (how do they interact in the game, their role). These differences help the player in the identification of personality templates. But having an interesting NPC is not enough in a game, players appear to engage more when they can use the characters components, by manipulating them, creating their own, personalizing the story and promoting a character-based play.

*StreetDroids* game support all these components with a generic NPC that can be transformed in many others by applying few changes in the physical appearance. The basic form for a *StreetDroids* character consists in a head, upper body and arms. All of them suitable in a grid composed by a circle and a square, see Figure 3.23 In the *StreetDroids'* characters everything the player sees is changeable: hair style, hair color, skin tone, eyes style, eyes color, mouths, eyebrows color, head shape, nose size, nose color, fashion style and hand color; see Figure 3.24

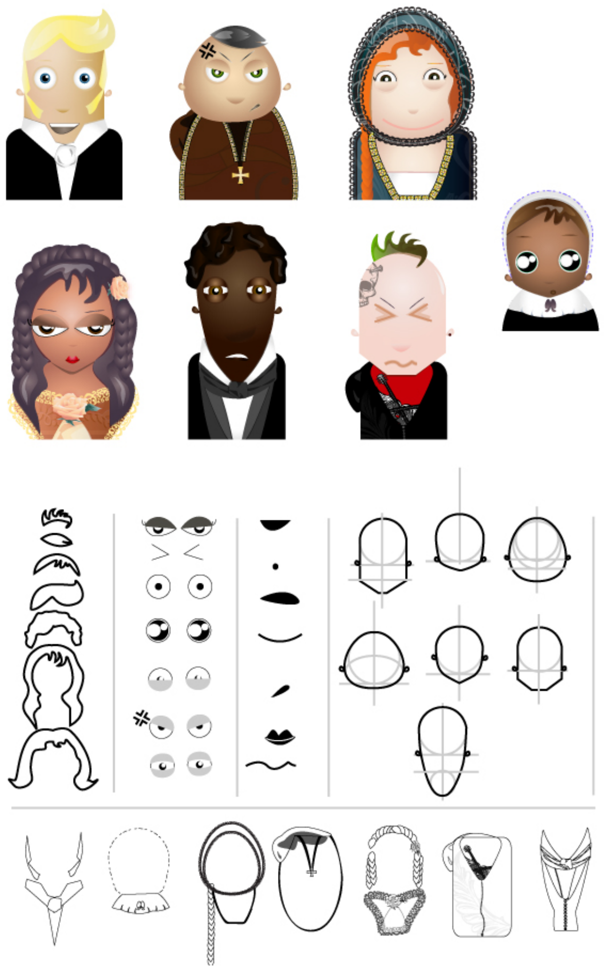


Figure 3.23.: Basic form of *StreetDroids* characters and their parts.

In order to achieve and maintain believability, the behavioral components for the NPC in *StreetDroids* are given by the vast number of graphic elements that can be use for personalizing them; they give the user and the game many possibilities like:

- A large repertoires of different behaviors(Hayes-Roth and Doyle, 1998) to cover a large range of situations, defined by different facial expressions.
- A normal variability in the expression in order to appear not robotic but friendly.
- Different facial expressions that can be used in different contexts giving different interpretations to the players.

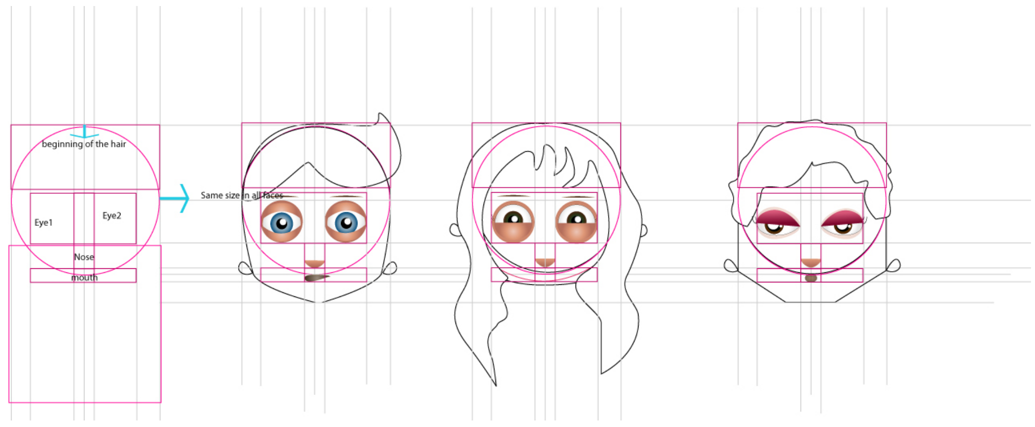


Figure 3.24.: Characters' changable parts.

- Idiosyncrasies in the choice of individual characters allowing the player to distinguish each one from all others.

The behavioral and believability aspects are given mainly by the facial expressions, even if they look friendly for the user and easy to talk with, the player can change and transform the facial expression and give the NPC another context and a different interpretation of the situation. The player has now the possibility to interact and create a game where he can be in control of the story along with the NPC that lives in it, enabling the creation of many other useful and enjoyable interactive environments as seen in Figure 3.23.

#### 3.4.2.8. Final Considerations

Planning good interface design for mobile screens is not just a matter of providing colorful buttons and fancy graphics. For game development, *"a well-designed product based on a team effort with a simple, user-friendly interface developed within a reasonable time frame will be successful"*(Pedersen, 2003). That was one of the guidelines used to support the learning process during the *StreetDroids'* design development. It was not a "trial and error" process, as commonly can be observed in game development. Each aspect was deeply analyzed based on the theoretical frameworks and translated for each specific case.

In that context, *ergonomics*, and *internationalization* are important keys for the design development in this project. The use of icons and symbols is one of the research areas whose results were reflected in the solutions for the user interface. In a "visual era", as the world is now experimenting, GUI allied with HCI are more than just important fields of research. They can define the success or the failure of a project. If the final product does not hold the users' attention, you will lose them.

To conclude the design overview, a general evaluation of the product (see the results at section "4.2 Evaluation") showed that the design aspects of *StreetDroids* received a positive approval by the consulted users. This indicates that the used approach to design development resulted in an attractive product. It is possible to believe that this is a reflection of two main aspects of the used approach, the choices made in consonant with theoretical foundations and the observation of a meaningful, coherent and well implemented aesthetic language.

## 3.5. Web Platform implementation

### 3.5.1. Purpose of the Web Platform[45]

[45]Cristina Botta, Till Hennig

In contrast to the mobile-frontend, which is used for gameplay, the purpose of the web-frontend is to enable the creation of content, view appendices of existing information and build a community to engage users in exchange of content, information and knowledge. These features are sourced out of the mobile client for the reason that a desktop browser offers greater comfort and more interactivity possibilities than a mobile-client for example mobile devices have a limited screen resolution and limitations in terms of data input (in many cases no QWERTY-keyboard).

The community is the most important aspect of the web-frontend, and with it we want to incentive people to show what they can do, learn from each other, and allow *StreetDroids* to exist beyond the project. The members and their contributions are the most important asset the game could have, since they can keep the game alive, feed it with new content, and maybe even have new ideas on how to use game elements in ways not predicted by the developers. They have the potential to make the game evolve. Users can also benefit from the feedback they receive, and from the sharing of knowledge and interests that such a community allows.

In the paragraphs bellow the main elements of the web-frontend are briefly described. The concepts for community and collaboration can be found in section 2.2.6, “Collaboration”. This section comprises a more practical explanation on the purpose and tools available.

#### 3.5.1.1. User Profile

The user profile, as in any other website, is used as a central for the user, where she can add or update information about herself, like avatar, location, about. Players also need a user profile in order to access the elements they have created for the game, the stories and puzzles they have played, their playing statistics, and their inventory.

#### 3.5.1.2. Inventory

The inventory is used to save the items collected throughout the game. The information attached to the items is introduced as short texts during the game-play and this should be sufficient to play and understand the story. The player should not be forced to read a long text on the small display of the mobile device. Through the web-interface the player has the possibility to see more detailed information on the collected items. Especially interesting topics can be marked (“bookmarked”) by the player to find it again more easily in the inventory later.

#### 3.5.1.3. Rankings

Members of the community can see how others are doing in the game and measure themselves against the them. There will be different possibilities to be listed in the game rankings. Examples are:

- Who was the fastest?

- Who collected the most items?
- Who has played the most Story Maps?
- Who has created more (or the best) content?

#### 3.5.1.4. Search and Sort Content

A system of tags will be implemented, so players can find what they want. Elements can also be put in more than one category with this system. This is important, because as the community grows the amount of elements available will be to big for the users to look for them in a simple list. Sorting the created content is also very important on the web interface since members should be able to edit the content to create new things from it. Created elements should be sortable not only by tags, but also by how they rank in the community. Some examples of how this functionality could be used are: the most played puzzles, the most used characters, the most interesting story maps, etc.

#### 3.5.1.5. Create Content

A major component of the game is that the user is be able to create content for the game on his own. This will be done through what we call editors, which will only be available via the web interface. This means that he can create the following parts of the game:

- story maps/missions
- puzzles
- characters
- items

All created parts are re-usable by everyone. This means, the player can use existing puzzles to build his own story map as well as building a story map from scratch. It is also possible to only create a single puzzle or only a character, that then can be used by others.

In the next section the implementation and technical details of a puzzle editor will be described.

### 3.5.2. Technical Details and Structure of Editors[46]

[46]Till Hennig

The part of the web-frontend that empowers the user to easily and flexibly contribute to the game's content is what is described by the term *editor*. The front-end, i.e. the interaction interface for the user with the editors, is implemented in HTML and Javascript, which is generated and validated by server-side technologies. Each creatable entity requires its own editor, with the most prominent one being the puzzle editor. In the following, the Puzzle Editor will be introduced and discussed in regards to its function and the level of completion of the prototypical implementation.

Technically, the `Puzzle` is also the central concept, as it is the entity that is referenced by the most other models. It requires the most complex editor since it provides a framework for embedding customizable forms of inputting data into a common set of forms. This framework provides common functionality (picking a location, picking or creating a character, etc) as well as the base for implementing customizable puzzle editors.

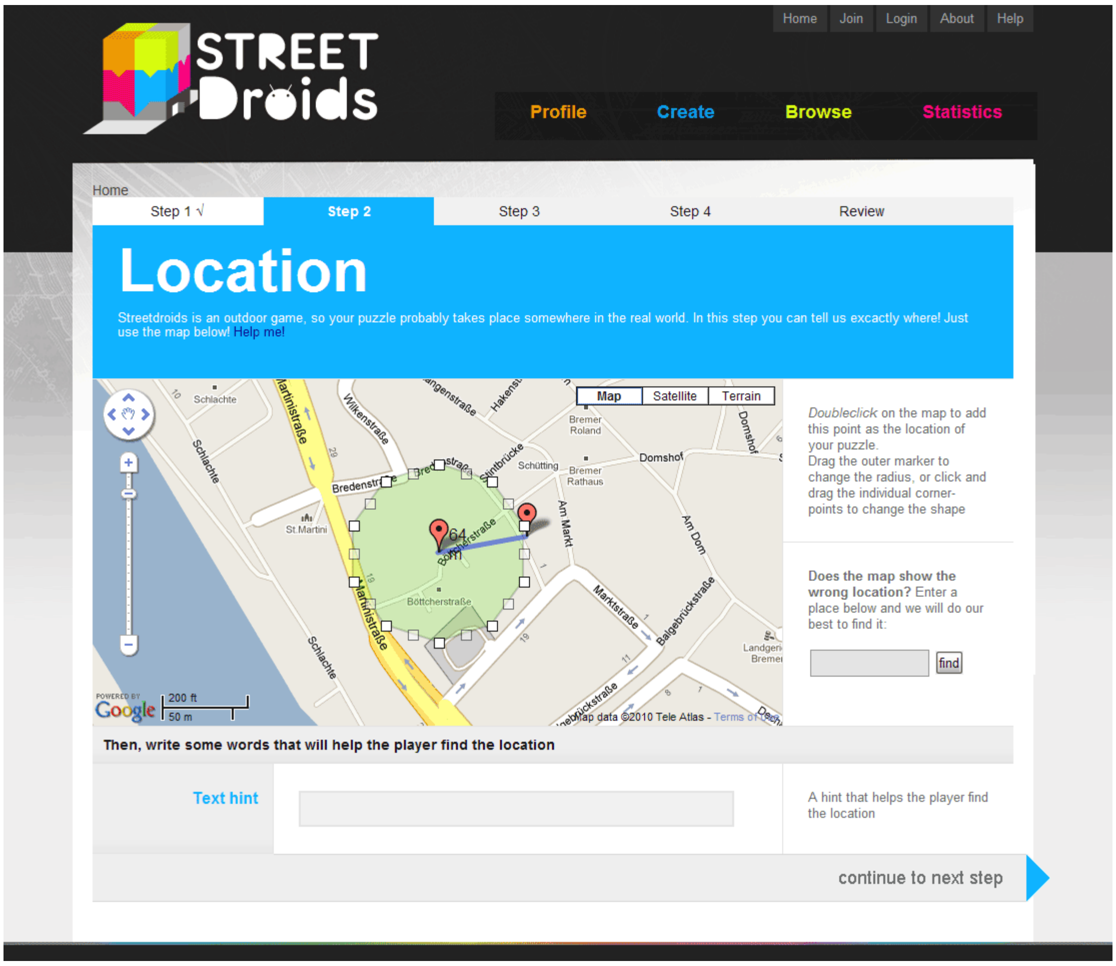
#### 3.5.2.1. Workflow from the user perspective

The puzzle editor follows the wizard pattern. A "wizard", in the context of computer science, is described as "[...] a user interface element that presents a user with a sequence of dialog boxes that lead the user through a series of well-defined steps. Tasks that are complex, infrequently performed, or unfamiliar may be easier to perform using a wizard." (Wikipedia, 2010b) Because the process of creating a puzzle is extensive and requires the input of complex information that is not always directly related to the other information, it is advisable from a usability standpoint to split the form into logically grouped units, called steps.

The user proceeds through the steps of the puzzle creation process in the following order:

1. User selects the type of puzzle he/she wants to create
2. User inputs general information about the puzzle (such as the title, the language of the following information)
3. User selects the location of the puzzle by placing it on a map (figure 3.25)
4. User inputs puzzle specific information
5. User selects background imagery, the non-playable character, introduction texts, greetings and other customizable values
6. User reviews the inputted data
7. Successful puzzle submission is confirmed, user is given options to proceed

The steps are made visible in the user interface by a tab-based timeline, indicating the current step by highlighting it, as well as the ones that have already been completed and those that are ahead. This allows the user to gain an understanding of where in the process he is and what is expected from him in the upcoming steps.



Location

Streetdroids is an outdoor game, so your puzzle probably takes place somewhere in the real world. In this step you can tell us excactly where! Just use the map below! Help me!

*Doubleclick* on the map to add this point as the location of your puzzle.
Drag the outer marker to change the radius, or click and drag the individual corner-points to change the shape

**Does the map show the wrong location?** Enter a place below and we will do our best to find it:

Then, write some words that will help the player find the location

Text hint

A hint that helps the player find the location

Figure 3.25.: Step of the puzzle creation process where the user chooses the puzzle's location.

### 3.5.2.2. Workflow from an architectural perspective

When the user accesses the puzzle editor an instance of `PuzzleWizard` is created. A “wizard” as a noted before, describes a user interface approach. It is also an infrastructural construct, that "glues" multiple steps of input data together. It allows the integration of customizable elements at different stages of the puzzle creation process. These so called "plugins" implement functionality, by overwriting the default implementation, for which the wizard offers interfaces. While the interfaces need to be well-defined on one side, to guarantee a seamless integration into the wizard, they need to be flexible enough to be able to accommodate a wide array of use-cases and easy implementation.

### 3.5.2.3. Workflow from an implementation perspective

The puzzle editor is assigned the URI `/editor/puzzle/` in the `urls` module which is mapped to the view `core.views.puzzle_editor`: This function renders a template which prompts the user to select a puzzle type.

After a valid selection has been made, an instance of `EditorWizard` is created. `EditorWizard` is a subclass of `django.contrib.formtools.wizard.FormWizard` (Django) an abstract class which is supplied by the framework. It validates and passes on the already collected information and has anti-tampering measures built-in to prevent malicious code from being executed.

An `EditorWizard` requires a list of individual Form instances (subclasses of `django.forms.Form`), which are groups of input values, and the implementation of the `done` method, which defines the code that is executed once all steps have been completed. This method creates a XML document of all information and sends it as an HTTP POST request to the webservice.

The list of `Form` instances and the `done` method are means for customizing the a puzzle editor - these two attributes depend on the puzzle type that has been selected.

**The forms**

The list of forms consists of forms, which are common to all puzzle types and those that are specific to the selected puzzle type. The common forms are implemented in `core.editor`, the custom forms need to be placed in the `editor` module of the puzzle application and included in the attribute `EDITOR_FORMS`; this attribute is imported and injected into the list of common forms at runtime.

The order and name of the instantiated Form classes as processed by the `EditorWizard` is as follows:

1. `MetaForm` - collects general information about the puzzle (name, language, etc)
2. `LocationForm` - collects information about the location of the puzzle
3. `Forms` imported from puzzle application
4. `StoryForm` - collects information about the puzzle which will be embedded into the storyline of the mission (NPC, item, feedback texts, etc)

Each `Form` class defines the Fields that are required to be filled in by the user. This definition includes options for data validation, type-checking, help texts, etc.

In addition, each `Form` can overwrite the default template used for rendering by supplying a HTML template with the name of the form in the `/templates/editor/` directory of the puzzle application. This allows each `Form` to fully customize its appearance to include customized UI elements. This has been taken advantage of in the case of `LocationForm`, where a map has been embedded for easy selection of coordinates. While the appearance does not have any resemblance to the default form the server-side implementation is the same.

**The done()-method**

The `done` method serializes the user's input data into the StreetDroids XML format. It is called after all forms have passed validation.

The `EditorWizard` can only serialize the common parts, i.e. the data from `MetaForm`, `LocationForm` and `StoryForm` and not the data from the forms which have been injected by the puzzle. For this reason the function `create_xml_from_input` is imported from the `editor` module of the puzzle application.

The `done` method renders the template `master_create.xml` from the `xml` template directory. That template contains a placeholder tag, called `puzzle_data`. Inside this tag the contents returned from `create_xml_from_input` will be placed.

As suggested by the name, the function is expected to generate an XML fragment; by convention it does so, although not required, by rendering `create.xml` from the `templates/xml` directory of the puzzle application. The only requirement is, that the XML is well-formed, the layout is adaptable to the structure needed to model the puzzle's specific data.

#### 3.5.2.4. Diagram

For a more technical documentation of the internals, please refer to the inline code documentation and/or the technical documentation in the appendix.

#### 3.5.2.5. Outlook

At the current state it is possible to create puzzles with the editor. Functionality which is not available, but required for a real world deployment, is the possibility to edit puzzle information. This is true for the puzzle creation process (users cannot go back from the review screen to a specific step to change information) as well as changing an already created puzzle. Editing information means loading it first, changing it, and then saving the changes. For simple data types (such as strings) editing options could be easily implemented. More problematic to implement would be editing of complex puzzle data (draggable A should not by placed at (x:5/y:3) but at (x:3/y:5)) because that would require loading the existing data into the editor and recreating the user interface so the user can edit it.

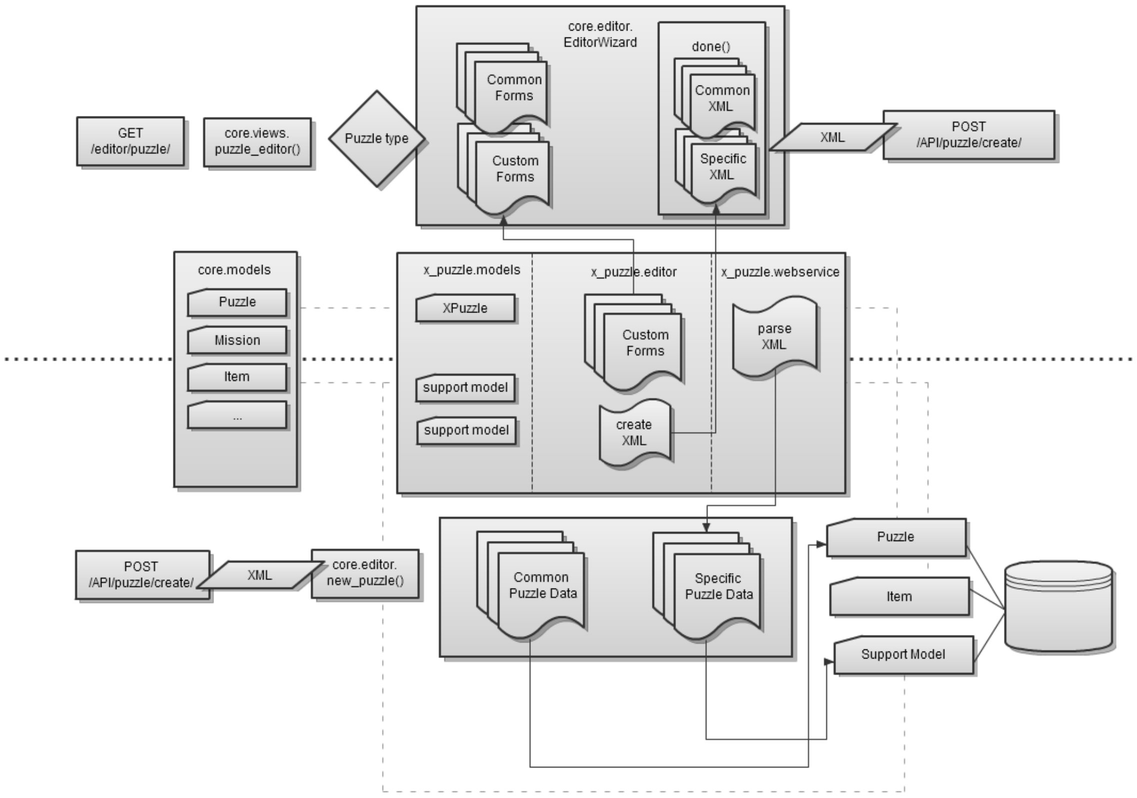


Figure 3.26.: this figure needs a caption and a reference in the text

While the feasibility for said functionality is definitely given, the development would require extensive conceptual effort to consistently enable loading and editing of complex data as well as the manpower to actually realize the functionality for the existing puzzle types.

## 3.6. Issues and Solutions

### 3.6.1. Downloading Time Issues[47]

[47] Yarik Sheptykin

#### 3.6.1.1. Problem statement

The *Streetdroids* project software development consists of two phases: the game prototype development and the final game implementation. The game prototype was designed and build to prove the game implementability in the chosen environment and check if there are any weaknesses in the game concept or in the software architecture. One problem found during the prototype testing was the long pause in the game play between reaching the location and the puzzle launch. The root of the problem was in the puzzle's content downloading. According to the software concept the content for a puzzle should be retrieved from the game server right before the puzzle starts. Testing in the emulator showed quite a good speed of the network transmission rate but the live testing proved the opposite. The testing happened in the real environment, thus the system was influenced by many conditions, one of them being the weather. Under different circumstances the GPRS data exchange speed used to slow down significantly, therefore the puzzle's launch delay increased times up. This was a noticeable drawback in the implementation concept.

#### 3.6.1.2. Investigation

An investigation of the problem was started by analizing the data being transmitted. An average puzzle content package has the approximate size of 1 megabyte. The average GPRS data rate shown by the emulator is around 100 kilobit per second (GPRS provides data rates between 56-114 kbit/s). With such a data rate 1 megabyte (8 megabit) of data needs around 70 seconds to be completely downloaded. Nevertheless, as mentioned above, in the real play environment the GPRS data rates might drop lower and consequently delay the puzzle launch. Tests proved that content downloading might last up to 140 seconds which means that data is transferred with the lowest rate. There might be a number of reasons for this phenomenon, ranging from an access point overload to the player's location peculiarities.

#### 3.6.1.3. Solution

There were two suggested ways for solving this problem. First, the puzzle content size could be reviewed and compressed in size. There are many parts in the game where such an optimization would be possible. For example, in one of the drag-and-drop puzzles all the image resources are stored in an uncompressed PNG format. Though there are places where PNG images are required because of their transparency support (Boutell, 2003), in this particular example it is not needed, and therefore images could be converted to JPG, which needs less size. Though this approach is very helpful for the given example it might not be suitable for other content types, therefore another solution was proposed. It was suggested to shift the puzzle's downloading process to the system background, and execute it while a player is looking for the puzzle location. Under the normal game flow it is expected that more than 2 minutes are needed to get from one puzzle location to another. This time slot could be successfully used to download the content for the next puzzle. Between these two solutions the second was prefered and implemented in the final game client. Evaluation of this downloading strategy proved, as expected, to speed up the puzzle launch.

To all the benefits of the second approach some drawbacks have also to be added. The same problem might arise again if the puzzle locations are placed too close to each other. If the player reaches the place faster then the puzzle content is downloaded the puzzle launch will be delayed. Another drawback appears from putting a download process in the system's background, which is done by threading the application process. Since the child threads have a lower priority compared to the main thread they get less processing time and therefore are executed slower. Despite the given disadvantages none of them has much impact in the system under normal game play circumstances, which is often the case.

### 3.6.2. Memory Leaks[48]

[48] Vahe Markarian

Upon integration of an increasing number of components to the game problems, like frequent game crashes, started occuring. In-depth debugging measures were taken in order to determine what caused the application to crash. At first, debugging by toggling breakpoints everywhere in the code was tried, but it proved to be unsuccessful in the detection of the problems. Later, a tool that comes with the Android SDK, DDMS (Dalvik Debug Monitor Server)[49], was used to debug thoroughly. DDMS monitors for threads and heap updates in an application. While observing changes in heap size, a huge increase in heap size was noticed when opening the navigation activity to show Google Maps on the screen. The growth of heap size is normal since navigation activity requires memory to be able to download map data. Clearly, navigation wasn't the main reason for the application to crash, because navigation activity relies mostly on the Google Maps library, which is quite stable in other systems. Therefore, the problem was using a lot of resources in the game besides having navigation activity accumulating even more resources. Afterwards several successful attempts in spotting and fixing some components helped increasing the system's stability. For example, the use of a compass in the navigation activity was dramatically increasing the heap size. In order to rotate the compass the device's accelerometer was sensing the rotation angle relative to North several times per millisecond. While rotating, the compass was being re-drawn on the screen with a different rotation angle. The compass had three different states, and each state had a different color. In addition the three different compass colored images were being used by the compass as many times as the sensor was checking for rotation updates. As a result, the compass was using extra image resources in vain. The problem was fixed by setting the compass to use only one of the three images at a time.

Despite several successful attempts to make the system more stable, there were times where the application did crash again. Researching for a possible solution led to many hypothesis on the several issues that could be the reason. First of all, inadequate layout structuring and improperly handling UI view references could trigger such issues. Subsequently, bad resource management throughout the application would result in such problems as well. Either way, it would take a restructuring of the entire application to resolve this issue, which is likely not the best solution because of time constraints.

[49] `http://developer.android.com/guide/developing/tools/ddms.html`

# 4. Results

## 4.1. Prototypes

### 4.1.1. First Prototype (Paper Prototype)[1]

[1]Catalina Payán

Paper prototyping is a usability testing method useful for any human-computer interface. It is a mockup (Spool et al., 1998), that helps to clarify requirements and enables screen and draft interaction designs in order to simulate and test them. Paper prototyping can be considered a method of brainstorming, designing, creating, testing, and communicating user interfaces. It is a tool for every academic researcher or usability specialist. This method does not depend on a platform and can be use for Websites, Web applications, software, handheld devices, etc. (Snyder, 2003).

Paper prototyping is also a method used by many game developers. Giles Schildt, director of the game development at Austin-based *Steve Jackson Games*, argue that changes are a necessary part of game design, and as earlier as we can made the changes, the easier and cheaper it will be. *"A paper prototype costs 'effectively nothing', and if it's done before specialized art or programming it can shave off the final cost of the project."*(Henderson, 2006). According to Carolyn Snyder's article on paper prototyping in the IBM site (Snyder, 2001), paper prototyping is especially useful for gathering data about problems that can be presented when you are developing the software, in this case the game. She described some considerations to take in count when you are in the developing stage.

- **Concepts and terminology:** How to make the target understand and feel familiar with the game.
- **Navigation/workflow:** How is the sequence of the game, in other words the application of the game mechanics, how the user will know where to go and will not feel lost in the game.
- **Content:** What information is going to be use in the game, if it is concrete, useful and reaches the objectives of the game.
- **Page layout:** Even if it is earlier to test the final design it is important to know if the users can find the information they need.
- **Functionality:** How functional is the game, and if the flow chart is easy to represent in the game sequence without complicating the user abilities.

#### 4.1.1.1. *StreetDroids* paper prototype

For the *StreetDroids* game development, the paper prototyping method was used according to the user expectations and needs; by explaining in detail the screenshots, menus, dialog boxes, interactions, etc. that suited better the performing of the tasks already discussed in the mechanics of the game. The team explored different metaphors and design strategies, and decided the initial design of each individual screen by testing the architecture of the game. The paper prototype was used to improve different areas of the game, for example: team communication, implementation, and explanation of the game. Moreover the paper prototype developed a brainstorming were the team collected and visualized ideas for the game mechanics and the appearance of the interface, by testing the legibility of the design and identifying the main navigation, clickable elements, etc. see Figure 4.1

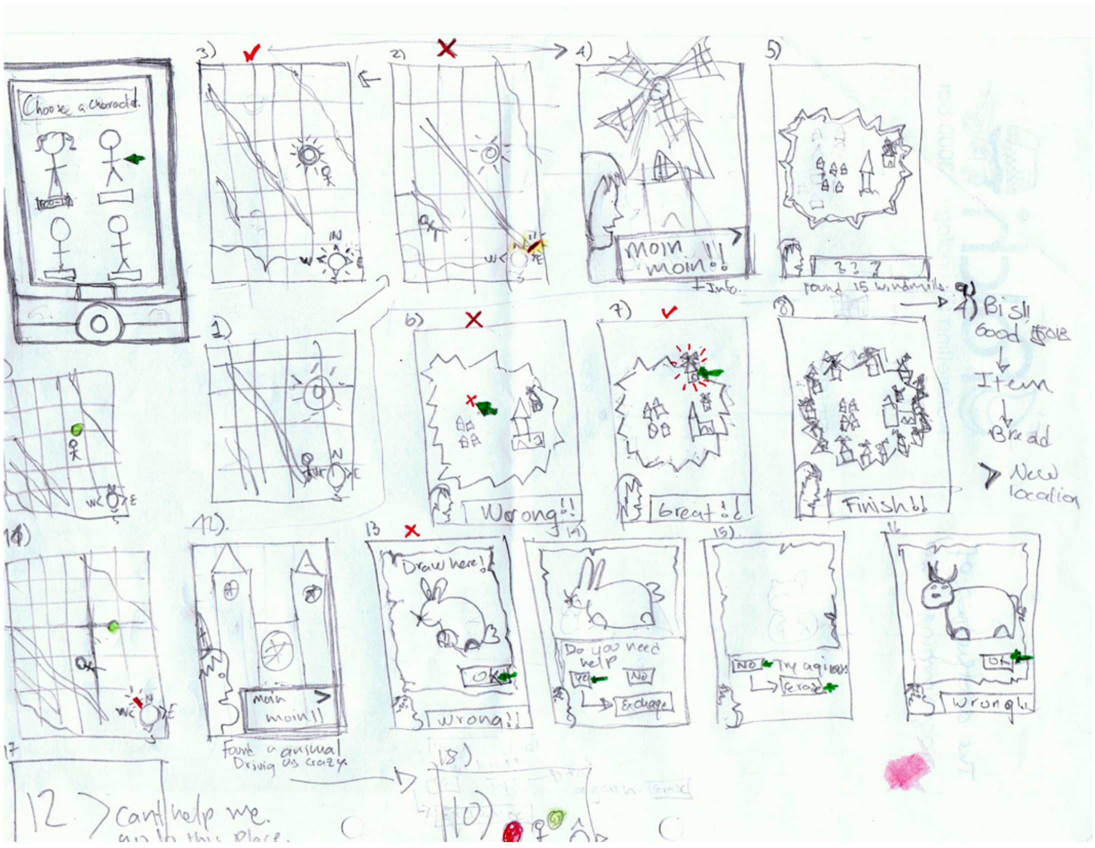


Figure 4.1.: Storyboard of the puzzle activity structure.

After the creation of the paper prototype a usability test was conducted in order to simulate the behavior of the interface, as a result the group had its first approach to the game play and could unify the different ideas about the concept and the mechanics of the game. A paper usability-testing works similar to any other usability-testing session. The test started with a team member playing the role of the mobile phone, manipulating the pieces of paper and simulating how the interface would behave. The other part of the team were the testers or the expected audience, they had to perform realistic tasks by interacting directly with the prototype, “clicking” by touching the prototype buttons or solving the tasks.

In the usability test some ideas form Giles Schildt (Henderson, 2006)were taken into considerations in order to improve the results of the paper prototype:

- Taking notes about what happens. Which are the most common problems?
- Don’t get emotionally attached to the mechanics, because they can change depending on the user’s response.

- Don’t be afraid to make changes.
- Don’t argue with the testers.
- Listen for “first-person” comments such as “I think” or “I like,” and pay special attention to those who say “I’m confused”.

The usability test with the paper prototype gave positive results, the team discussed about the problems solutions, marked on the prototype were a user attempted to interact with the interface or were the interaction was not clear, asked the users about their game expectations and saw the reactions to the game challenges. It helped to improve not only the mechanics of the game but also the use of the content in the game like the information the user should have in order to accomplish the task. On the other hand, it gave quick feedback of the game and detected the usability issues early in the design process before the investment of a lot of development effort, see Figure 4.2

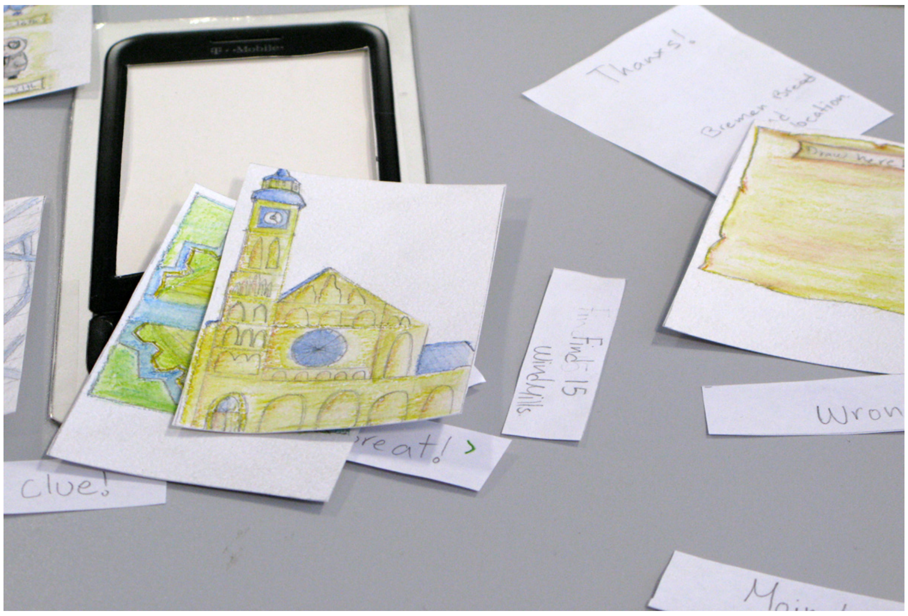


Figure 4.2.: Paper prototype.

To make the experience of working with a paper prototype as useful as possible, it is ideal to have a technical view. Paper prototypes do not demonstrate the technical capabilities, so it is important to have a person who understands the technical constraints. It is also a good idea to have a graphic designer because he may find problems that could influence the visual aspects of the design.

### 4.1.2. Final Prototypes and Their Application[2]

See demo videos by Sven Hamann and Nils Thies.

## 4.2. Evaluation[3]

[2]Sven Hamann, Nils Thies

[3]Dema El-Masri, Isabella Lomanto, Nils Thies, Jana Wedekind

### 4.2.1. Objectives

#### 4.2.1.1. What do we want to achieve with the evaluation

When designing software applications, the goals are mainly to have an interface which is easy to learn, use and master (Desurvire et al., 2004). Whilst, the goal of game design goes beyond that, requiring it to be playable, enjoyable and not just having a usable interface. The main reason for that is that games evaluation assesses additional properties of the gaming experience such as the game mechanics and the story. The game market being the competitive market it now is, has required game designers and industries to invest a lot of time in finding new ways to design and evaluate games (Lee and Im, 2009).

The increase of this market's size, the advances in technology, and the diverse game-platforms require re-establishment of terms as evaluation, and associating it with new terms as playability, satisfaction, and enjoyment, rather than merely usability. There are many heuristics that have been applied to games in general and to mobile games specifically, yet there remains a need to integrate these heuristics in to a model which fits to *StreetDroids* context and platform. Also because, the traditional usability heuristics cannot be directly applied to games (Korhonen and Koivisto, 2006). Game designers create games to be challenging, therefore, the player does not know what to expect, and works towards goals defined by the game designers (Sharples, 2009), and evaluation tries to find out how efficiently users have achieved these goals (Sweetser and Wyeth, 2005). For this reason more aspects than usability have to be applied to game evaluation. Not to mention to mobile games such as the one at hand *StreetDroids*.

The evaluation started with the development of heuristics fitting the game, platform and technology at hand. These heuristics were adopted from a number of previously done projects and researches, and are thoroughly handled in the methodology section of the evaluation. The aspects of the game to be evaluated were the game play, usability, and game mechanics. Mobile device features were also an important part of the evaluation, as it is the goal of game designers to have players focus on the game and enjoy their time rather than struggle with a control or with the technology. Aside from focusing on the game play and enjoyment, a more challenging aspect was to be evaluated, namely how well the concept fits into the mobile context. Designing games in general is a challenge, and designing them to fit into small screens and numerous interaction techniques for mobile phones poses only a bigger challenge.

How much the mobile context affects the tasks that the user performs was evaluated, if it made things easier or more difficult, for reasons associated with the device or the game. *StreetDroids*, being a location-aware game played outdoors, also required to include in the evaluation the question of how enjoyable an outdoor scenario is for players; for the current content that scenario is the Old City of Bremen.

A main goal of the evaluation was to verify whether the game mechanics allow players to engage in a fun exploration of an environment, through which they could acquire some knowledge. In the current content at hand these learning aspects have been applied to the Old City of Bremen. This raised other questions as to whether players were able to learn about the history of Bremen by exploring the urban environment of the city. More specific questions in mind were focused upon in the evaluation as well, such as: how easy the technology was?, were the instructions of the game clear or not?, was the game enjoyable and attractive or not?

*StreetDroids* has a mobile and web end, the mobile end was tested repeatedly by the project group and later on evaluated by a group of international students successfully using qualitative and quantitative methods. The web end on the other hand, was only tested by the project group themselves. Nevertheless these tests did focus, as on the evaluation, on applying heuristics as usability, debugging it, developing the navigation in it, and constantly improving the user interface´s (UI) design and functionality.

To allow for all these aspects to be covered in the evaluation at hand, a methodology was developed containing heuristics covering the different criteria falling underneath usability, explorability, playability and satisfaction in games. Usability remains a base aspect to base an evaluation on, focusing on the UI for example. Whereas explorability is an aspect related to the fact that the game is a location-aware game, and exploring and learning from the surrounding environment is a main goal of it. Playability and satisfaction are closely related to games evaluation in general, as the goal of games is for the players to enjoy themselves and have fun.

#### 4.2.1.2. What can be tested

The mobile end of the game has gone through many testing sessions by the project group, since a paper prototype of it was made in the early stages of the project. In those early stages, the team worked basically on further-tuning the game mechanics and navigation of the system to make it consistent. A rough design was made for the interface and was later on adopted for the design stage for Non-Playable Characters (NPC's) and the general navigational UI.

A prototype was created for *StreetDroids* by the end of the project´s first semester. It was basically a number of quizzes which would be activated upon reaching a specific location on the map. The development of this prototype, as well as in the case of any other prototype, helped point out problems and solve them in earlier stages rather than having to deal with them in more advanced stages of the game development. Yet these problems found in the prototype, and working on solving them, posed many difficulties to reach the project´s goals on time, and therefore some features were dropped.

Another prototype was developed, consisting of one mission with two puzzles. This was the prototype used in the evaluation process, even though initial aims were at having the subjects play several missions in one story, this was not realized. At that time the web end was being developed as were the puzzle, character, and mission editor. These features were not evaluated because they were not complete.

#### 4.2.1.3. Methodology

To conduct the evaluation a one-time afternoon session with six subjects was organized. The subjects were between 22 and 29 years old, which is older than the defined target group for the test scenario, but for the whole concept also other age ranges are interesting. It was planned to make another evaluation with an improved prototype and preferably with a school class in the desired age range for the history map of Bremen. However, given the lack of time and human resources in the project, it was not reasonable to do another evaluation. Due to the fact that English-speaking subjects were needed, the group proved to be very international, consisting of the following nationalities: American, Colombian, Italian, Polish, and Russian. The group was composed of three female and three male subjects. One subject left during the play-testing because of personal reasons.

The methodology consists of three different parts, which include a questionnaire, play-testing, and a focus group discussion. The two latter were video-taped in order to be able to later analyze the activities in depth. During the analysis the observation notes were considered as well. It was planned to additionally use some logging on the mobile devices, that could give information about problems that occurred while playing. Unfortunately, it was very difficult to connect the information of the log to what was actually happening in the video, as there has been no logging of the current location of the player.

Before starting to play, the subjects were posed some general questions about their usage of mobile games and their interest in location-aware games. After they had finished the game some more questions were asked to get immediate feedback if the subjects liked the game and could imagine to play it or a similar game again. The results of the questionnaire are documented in 4.2.3.1. Apart from that, the given answers were considered in the analysis in relation to the defined criteria, whose results are noted down in 4.2.3 Results.

After answering the questionnaire the subjects began to play with the prototype. They had an hour to explore the city center of Bremen. The subjects started not too far away from the actual puzzle locations, which they did not know. The players were told basic information about the game, but they were not told how for instance a puzzle would work. This was done on purpose to investigate critical issues of the usability and the game mechanics. The video of the play-testing was conducted in regard to identify breakthroughs and breakdowns, following the methodology suggested by Mike Sharples. "*Breakthroughs are observable critical incidents which appear to be initiating productive new forms of learning or important conceptual change. Breakdowns are observable critical incidents where a learner is struggling with the technology, is asking for help, or appears to be labouring under a clear misunderstanding.*" (Sharples, 2009) In our case these breakthroughs and breakdowns were applied both to learning and to the game experience, in other words to the fulfillment of the goals and objectives of the game. The incidents can either be used to describe unexpected reactions of the user or the system, as well as to verify anticipated behavior. Thus, using this methodology allowed us to identify critical incidents that needed further analysis. "*A critical incident is an event observed within task performance that is a significant indicator of some factor defining the objective of the study.*" (Pan et al., 2004). It was planned to relate these information to a log that was created on the mobile devices, but as mentioned earlier it was difficult to find out which information relates to a certain incident documented in the video.

As a final step the players participated in a focus group discussion. To get the focus group started and in order to get an understanding of the emotional reactions of the subjects towards the game, the so called Product Reaction Cards[4] (Benedek and Miner, 2002a,b) were applied. It was decided to film the session, as the video helps recalling the statements of the subjects. During the discussion it was possible to talk about certain situations that were observed throughout the play-testing, as well as to address parts of the game that only exist as a concept yet, as for example the collaboration aspects. With the help of the video, a short summary of the questions and statements was written down in a transcript. This was the basis for the analysis of the focus group according to the below in *4.2.2* defined criteria. The results were also compared with the answers from the questionnaire and the notes of the observers. As guidelines for the analysis the methodology suggested by (Krueger and Casey, 2000) and (Rabiee, 2004) was used.

### 4.2.2. Criteria

#### 4.2.2.1. Usability

Most probably the best known of aspects when it comes to evaluation. In traditional software products, usability could be defined as the effectiveness, efficiency, and user satisfaction in a specified context of use (Wiberg et al., 2009). Games however, are most enjoyable and fun when they provide sufficient challenge for a player, also a feeling of engagement. The challenge can be, for instance, in learning the game, solving problems or discovering new things. (Sharples, 2009) The applicability of traditional usability heuristics on games can be questioned by game designers (Liljedal, 2002), due to the fact that usability focuses mainly on the interface and disregards game play. Usability in games goes further than the user interface to include elements as game play, and game mechanics. In the case of *StreetDroids*, in addition to it being a game played on a mobile device, it is also location aware. Requiring the user to interact with the surrounding environment, and maintaining coordination at all times, another aspect in which the game at hand differs from a traditional one and requires a wider variety of usability concerns (Pinelle et al., 2009).

Pinelle et al. (2009) in (Liljedal, 2002) published game usability heuristics that are based on game reviews and they have been validated in a preliminary study. These heuristics are used to evaluate game usability (user interface) and there are no heuristics concerning game play issues (Liljedal, 2002).

Game researchers started to develop heuristics which would include both usability and game play issues, to assist game developers in discovering playability problems in the game design (Liljedal, 2002). Clanton in Kurosu in: (Lee and Im, 2009) for example offers a way to encapsulate the different usability issues of games into three areas: game interface, game mechanics and game play. The interface is the device through

[4]Developed by and © 2002 Microsoft Corporation. All rights reserved.

which the player interacts with the game. That being a mouse or a keyboard or so. It also includes the software controls that players use to set up their games, move through the game, save and exit the game. Game mechanics on the other hand are the physics of the game, being a combination of animation and programming. They are used to describe how players interact with rules like how the player is supposed to move throughout the game, strategies, and game states. Game play is the process by which the player reaches the goal of the game. The previous three aspects developed by Clanton relate to the game being both functional and satisfying and require both design and evaluation Kurosu in: (Lee and Im, 2009). Thus, assessing the usability of games taking into consideration the user interface, game play and additional aspects that are associated with games when speaking of their usability as game mechanics.

As explained above, combining usability and game play for mobile games would require additional aspects other than the traditional usability ones focusing only on the user interface, this was also applied in *StreetDroids*. That meant combining the user interface, game play, game mechanics, and the device's usability, all up to some point. Another reason for that is that other criteria than usability were defined in the evaluation and measured. In games in general, a thin line divides criteria as usability and playability, yet it was best for the evaluation to break down the criteria, allowing for more specific analysis and definition of aspects under each of the criteria to be measured. Therefore usability in the case at hand remained a specific area with the aspects of readability, interaction, feedback and help falling underneath it. The other criteria of evaluation (playability, explorability, and satisfaction) take on more aspects regarding the enjoyment and satisfaction the player experienced while playing the game and will be discussed in the following parts.

When developing the usability criteria, the main aspects kept in mind were the user interface and the controls of the game; those controls being the physical ones that the device provides the user with or the game controls through which the user navigates in the game (e.g. exiting, saving, etc, also the game's feedback system, the help system and the readability).

Some of these aspects as the feedback and help system can be applied to games in general, being a mobile or computer-based game. Assessing the game's mechanics by evaluating the way in which it provides the user with immediate and precise feedback or help. That can be related to the game's controls; the game giving the player feedback when pressing on a specific button in the game or on the device which triggers a specific reaction from the system. The precision and relativeness of this feedback to the user is amongst the aspects to be measured and evaluated. The help system, quite similar to the feedback system, an aspect to be measured by evaluating the precision of the help provided by the system to the player once he requests it.

More specific aspects closely related to the mobile game evaluation are the other two criteria of usability, namely: interaction and readability. Interaction, in particular dealing with the special interaction techniques the android device offers the player: mainly the touch screen. Readability, taking on a criteria for itself under usability, is also closely related to the mobile device. The device's 3.5 inch display screen presents a challenge for designers to fir their concept, design, and text into this mobile context.

Another challenge regarding the readability is the accelerometer the device provides, this feature allows the phone to switch display modes between portrait and landscape when the user simply rotates the device 90 degrees. In doing so, the text and graphics also have to switch modes posing yet an additional challenge on the designers to make the game functional and appealing in whichever mode it is used in, and changeable instantly.

#### 4.2.2.2. Playability

There is not much consensus on a definition for playability, however one can identify two distinctive approaches for understanding it. On one hand it is understood as an instantiation of usability for games (Fabricatore et al., 2002), (Desurvire and Wiberg, 2009), and on the other hand as the quality of gameplay and player experience (Järvinen et al., 2002), (González Sánchez et al., 2009), (Sweetser and Wyeth, 2005). Although playability is closely related to usability, it should go much further as "*Experiencing gameplay does not equal experiencing regular product use*" (Nacke, 2009). This means that usability alone is not enough to evaluate and design the player experience. A much extensive model and specific heuristics are needed. A model that takes in consideration the player enjoyment and concentration, the types of challenges, the manner of storytelling and the degree of emotion when players play video games.

For the purpose of the evaluation of *StreetDroids* we defined playability as set of methods and criteria for designing and evaluating a product's quality of gameplay or interaction (Järvinen et al., 2002), (González Sánchez et al., 2009). Concretely we followed the approach to playability proposed by Järvinen et al., which focuses on the gameflow. For that reason we took the model for evaluating games developed by game researchers Penelope Sweetser and Peta Wyeth as the base for our playability evaluation. For this model, the researchers used the concept of *flow* by Mihaly Csikszentmihalyi as the foundation to organize and synthesize different game heuristics into a concise model that focuses on player enjoyment in games(Sweetser and Wyeth, 2005).

The concept of flow as explained by Csikszentmihalyi does not refer to the common conception of flow used in game development, which relates flows to a certain fluency that allows for player satisfaction. Csikszentmihalyi's flow theory is based on the idea that elements of enjoyment are universal, no matter from what activity enjoyment is derived from. The flow experience is described as a state of mind in which a person is freely and effortless giving full attention to a particular activity in order to achieve a goal. Sweetser and Wyeth explain that *"the key element in flow is that it is an end in itself – the activity must be intrinsically rewarding and autotelic"* (Sweetser and Wyeth, 2005). For them this is specially true when applied to games, because for a player the main reason to engage in the activity is the experience it will produce. Sweetser and Wyeth add to their arguments for applying Csikszentmihalyi's theory to games, that flow activities produce a feeling of being in a different or new reality, which they affirm is a familiar sensation for gamers. The result of Sweetser and Wyeth study is the GameFlow model consisting of eight core elements, see Table 4.1 (Sweetser and Wyeth, 2005).

The GameFlow model offered us the basis to structure our playability model and

| Elements | Description |
|---|---|
| Concentration | games should require concentration and the player should be able to concentrate on the game. |
| Challenge | games should be sufficiently challenging and match the player’s skill level. |
| Player skills | games must support player skill development and mastery. |
| Control | players should feel a sense of control over their actions in the game. |
| Clear goals | games should provide the player with clear goals at appropriate times. |
| Feedback | players must receive appropriate feedback at appropriate times. |
| Immersion | players should experience deep but effortless involvement in the game. |
| Social interaction | games should support and create opportunities for social interaction. |

Table 4.1.: GameFlow Elements and Description

criteria. However, some adjustments were needed in order to have a model that would fit the characteristics of our game. In particular we wanted to give special attention to the player’s concentration and involvement in the game and the player´s ability to learn and master the game. For this reason we took into account the playability model proposed by González Sánchez et al. and introduced in our model the element of *learnability*, which included the two elements *challenge* and *player skills* from the GameFlow model. That helped us to focus on this matter more deeply during the evaluation despite the limited time and resources. Another change we made to the GameFlow model was to leave out the feedback element. We made this decision because this topic was covered with the usability criteria we explained before. This changes produced a model with six main elements and ten main criteria, see Table 4.2.

Our adaptation to the GameFlow model was applied in our evaluation paying special attention to the play-test observations and the results of the focus group. In detail the main elements and criteria of our model were described as follows:

**Concentration:**

We define it as the player’s ability to concentrate and maintain the focus on the game. The game should call the player’s attention and motivate her to stay on the game and master it. We identified two main criteria for evaluating concentration: *Attractiveness*: which refers to the game capacity to provide meaningful and pleasant stimuli to the player. *Appropriate workload*: this means that the player should not be burdened with too many tasks or swamped with stimuli. Task and stimuli should be balanced so

| Element | Criteria |
|---|---|
| Concentration | - attractiveness<br>- appropriate workload |
| Learnability | - skills<br>- challenge |
| Control | - meaningful interaction |
| Clear goals | - clarity<br>- consistency |
| Immersion | - emotional involvement |
| Social interaction | - group awareness<br>- communication |

Table 4.2.: Playability model for *StreetDroids*, main elements and criteria

that the player can concentrate without feeling overloaded.

**Learnability:**

We take González Sánchez et al. definition of learnability as *"the player's capacity to understand and master the game's system and mechanics"* (González Sánchez et al., 2009). The game should offer the player the possibility to learn the game rules and objectives as well as its ways of interaction. For this element we take into account two criteria: *Skills*: which means that through this learning process the player should acquire the necessary skills to master the game. *Challenge*: which refers to the appropriate balance between the game's difficulty and the player's skills.

**Control:**

We understand control as the player's ability to sense that her actions or choices influence the game. The game mechanics as well as the interface, should support the player and allow her to achieve this sense of control. We identify meaningful interaction as the main criteria to evaluate control. *Meaningful interaction*: refers to the quality of choices that the player can make and how she perceives that these decisions affect and influence the game. For this type of interaction adequate feedback and a sense of control of the game interface is needed, as well as coherent game mechanics that allow for emergent behavior.

**Clear goals:**

We take Sweetser and Wyeth definition of clear goals and understand it as the game capacity to provide goals that are coherent, consistent and understandable to the player. These clear goals should also be given at appropriate times. For this reason we use *clarity* and *consistency* as the main criteria to evaluate this element. *Clarity*: refers to the manner in which the goals are presented and their coherence. *Consistency*: means that clear goals should correspond to the appropriate moment in the game.

**Immersion:**

We follow Sweetser and Wyeth definition of immersion as the player's capacity to get effortlessly and directly involved in the game. The game should create a believable environment that allows the player to get emotionally involved in the game. Our main criteria for evaluating immersion is: *Emotional involvement*: which refers to the player's ability to become less aware of her reality outside the game and become deeply involved in the game's reality. Since the game world in *StreetDroids* overlaps with the real world, we relate immersion to the game's ability to offer a mixed reality experience where the player can establish an emotional connection with the real world.

**Social Interaction:**

We define it as the player's ability to communicate and interact with other players inside and outside the game. The game should support communication and interaction between players whether they play together in a collaborative fashion or competitive one. Even if players play individually, the game should also offer the possibility for social interaction and group awareness. We identify two main criteria for this element: *Group awareness*: which means that the game should support player-to-player interactions that allow players to feel that they are part of a group or community. These interactions can be part of the game as competition or collaboration, or can be outside the game as social or online communities. *Communication*: refers to the game capacity to provide different channels of player-to-player communication (e.g. face-to-face communication, chats, online communities).

#### 4.2.2.3. Satisfaction

Satisfaction poses the question if the player enjoyed the game, therefore the player's emotions while playing or during later interviews is observed (Sharples, 2009). The game should allow players to have a fun experience that provokes positive emotions in the player. González Sánchez et. al divide satisfaction further into fun, disappointment and attractiveness (González Sánchez et al., 2009). Their definition of the three terms is as follows: "*Fun: the main objective of a video game is to entertain, hence a video game that is no fun to play could never satisfy players. Disappointment: we should ensure that players do not feel so disappointment or uneasy when playing a video game that they abandon it altogether. Attractiveness: this refers to attributes of the video game that increase the pleasure and satisfaction of the player.*" Hence it can be inferred that the main objective of evaluating satisfaction is to detect if the game is fun. Second is to find out if an enjoyable experience is created that the player is interested in keeping on playing.

For the evaluation we were especially interested in finding out if the players enjoyed...
...the whole game concept.
...the outdoor scenario.
...the exploration of the city with a mobile device.

One way to assess satisfaction, is the use of the mentioned Product Reaction Cards. They are designed by Microsoft and are described in detail in the paper (Benedek and Miner, 2002a). These cards regularly include 118 cards with adjectives such as confusing, personal, and frustrating. For our purposes we narrowed it down to 60 adjectives, with which reactions towards the technology or the game experience were observed. Further, during the focus group questions about satisfaction were posed to the subjects, in order to get a better understanding of what they liked about the game and what they felt frustrated about. An example strongly related to the player's emotion was the question about anyone being scared because of playing next to the tram tracks.

#### 4.2.2.4. Explorability

Explorability is a novel approach in mobile learning, stating that the players will learn about and with the help of their surroundings. To achieve explorability games should offer sufficient possibilities for spatial exploration in a meaningful way. This means a game should enable the player to actively engage with his environment by exploring the game world. The player should be motivated to make a connection between the real world and his knowledge gained within the game world. Similarly, Sharples (Sharples, 2009) describes that while measuring effectiveness, one must take into account the context in which the learning occurs and the intended aims of the activity.

We propose, that the interaction with the environment as well as with other players can create an immersive experience for players, which can motivate and enhance learning processes. Further, explorability can be defined as the ability of a game to support a more complex and meaningful exploration, which allows the player to relate emotionally with the environment. Explorability as evaluation criteria for pervasive, mixed reality or location-aware games should measure or indicate the extent to which a game enables the player to experience the game environment as both space and place.

Concerning the explorability the questions below were the main ones to answer with the help of the evaluation:

- Is an interaction with the environment happening?
- Did the game support a new way to explore the city?
- Was factual knowledge about the visited places remembered?

### 4.2.3. Results

#### 4.2.3.1. The questionnaire

The questionnaire was used to get a general impression of the subjects and their view on mobile games. The questionnaire included several closed questions, which partly were answered before playing the game (*Questions 1-8*), and partly afterwards (*Questions 9-15*).

1. Do you play games on your mobile phone?
   Answers: 2x yes, 4x no
2. What kind of mobile games do you usually like to play?
   Sudoku, Arcade, Guide, Puzzle 2x, Word, Action
3. Do you like to go outside for playing a mobile game?
   1x yes, 4x no, 1x maybe
4. Have you ever played a game like ours before?
   1x yes, 5x no
5. If you did, which one did you play?
   1x: Maybe? I wrote a mobile audio-online game
6. Did you use another mobile learning game before?
   6x no
7. If so, which one?
   No answers, as no one played another mobile learning game before.
8. Are you interested in creating your own games/ content for a mobile game that you could play outside?
   2x yes, 4x no
9. Would you play such a location-aware game again in the future?
   4x yes, 2x no
10. Were the rules of the game clear to you?
    2x yes, 4x no
11. Did you experience any difficulties playing the game?
    5x yes
12. Can you describe in a few words what caused the problems?
    *Subject 1*: Sometimes I was staying in the target place, but it was saying I hadn't reached it or sometimes the GPS was slow.
    *Subject 2*: The puzzles were not appearing where I reached the right places.
    *Subject 3*: Bad GPS signal, slow internet puzzle download
    *Subject 4*: GPS/Tracking difficulties, unclear puzzle rules
    *Subject 5*: Clicking? Accuracy of the GPS module, finding locations by using map module
13. Did you learn something about Bremen?
    5x yes
14. If you did learn something, can you give us some examples?
    *Subject 1*: About Rathaus like a place to manage something in the city and the

representation up the globe.
*Subject 2*: The Roland has different symbols which represent something like the ring
*Subject 3*: The Rathaus building and the Roland statue
*Subject 4*: Story behind founding, history of flag
*Subject 5*: I can remember some facts about the sailors coming to Bremen in medieval times.

15. Did you feel you could explore the city by playing the game?
3x yes, 2x no
Comments:
*Subject 1*: By improving it yes.
*Subject 4*: yes, if future versions fixes technical issues

#### 4.2.3.2. Product Reaction Cards

| | Subject 1 | Subject 2 | Subject 3 | Subject 4 | Subject 5 |
|---|---|---|---|---|---|
| 1 | Innovative | Attractive | Frustrating | Innovative | Personal |
| 2 | Approachable | Creative | Engaging | High Quality | Attractive |
| 3 | Slow | Fun | Friendly | Confusing | Slow |
| 4 | Stressful | Confusing | Confusing | Frustrating | Engaging |
| 5 | Frustrating | Slow | Fresh | Difficult | Confusing |

Table 4.3.: Table showing the results of the Product Reaction Cards

#### 4.2.3.3. General observations

The results of the play-testing and the focus group can be classified in two parts: technical and game play related issues.

**Technical problems**

1. *GPS is not responding/ slow GPS-Connection*
The low performance of the GPS and the processing time of the incoming data causes an enormous distraction. For the subjects it was the most distracting aspect of the field test and therefore a critical item of the game to be fixed. The maps to navigate to the puzzles were slowly responding and misleading.

2. *Rarely responding Touchscreen*
The field test took place during a weather period with temperatures under the freezing point. The touch screens of the devices didn't respond every time they were attended to. In this case the problem cannot be distinguished between the cold or a general problem of the touch screen. In warmer environments the touch

screen of the HTC G1 phone has the same problems but not that often. This fact is the second critical item to be fixed.

3. *Puzzles crashing*
   The version of the software prototype used during the field test was never tested in outdoor conditions. The data exchange and ratio was not stable. The applications were restarted a couple of times. This problem did not affect every subject. In the main the test it was not present in a critical way, but it is a critical item to fix too.

4. *Preferences of the G1 Menu*
   Beside the game menu the preferences of the G1 phones have a menu for the overall preferences like WiFi-Connection. This preferences have to be adjusted sometimes especially after crashes or for trouble shooting. The subjects did not have any experience with the phone and therefore had no clue about the phones preferences. This problem cannot be completely fixed by the adjustment of the game. The phones preferences (or the users) have to be modified for this.

5. *Puzzle download duration*
   Waiting times were observed during the field test caused by downloading the puzzles' data. The gameplay was freezing because there was no indicator or information for the players of what the current status was. This critical item can be fixed with: a. faster downloads, b. information for the user to know what is happening (progress bar) or c. a mix of both.

6. *Display Size*
   The size of the screen in connection with tasks using images and/ or precise movements was also a technical issue. In connection with the problem mentioned in 2. the item is critical for the gameplay. It has to be commented that after a while of using the device the player handled the tasks better. A learning effect of using the devices was observed here.

7. *Trackball moves unintentionally*
   The standard trackball on the G1 Device was activated unintended from time to time by the users. This could be forced by the new way of using this device or a design specific problem. In any case, this problem could be solved by deactivating the trackball when is not needed.

**Game play related problems**

1. *Puzzles had no introduction*
   The first playable prototype missed to have an introduction for every puzzle. It was sometimes not obvious what to do for player at first sight. On the other hand the subjects explored the puzzles by themselves without external help in a few seconds. Concluding this facts introduction texts are not needed for easy puzzles.

But to assure that every player understands the task of the game an introduction have to be implemented.

2. *Missing highlighting on buttons*
   During the game phase it appears that some players had problems to recognize highlighted buttons. In this case the green highlighted button for the help of a NPC and scrollable text boxes.

3. *Misleading map symbols*
   Symbols used during the navigation with a map (GoogleMaps related) showing the position of the next puzzle and the player were rarely distinguished. That leaded some of the subjects in the wrong direction. The problem was boosted by the fact described in the technical problem *1) GPS not responding/ slow GPS-Connection*.

4. *Text Clues were not clear*
   The texts provided to communicate hints were sometimes not clear to the subjects. These texts have to be proved and changed if applicable. The information has to be clear and explicit. In relation to B. the texts have to be as short as possible to fit without scrolling.

#### 4.2.3.4. Usability

Usability results were retrieved mostly from the play-test session, were most of the difficulties appeared, and players commented upon, either on location which was caught on tape and analyzed later, or in the focus group session where the players engaged in an open discussion about their experience with the game.

Amongst the product reaction cards results - presented in 4.2.3.2 - appears the words frustrating and confusing quite often. The subjects expressed their confusion constantly during, and after playing the game. The instructions were not clear, neither how to solve the puzzle itself nor the overall goal was clear either. In the focus group following the play-test session many of the subjects recommended that an example of the puzzle being played could be presented to make it clear what the goal is.

Technically related problems were also encountered by the subjects as:
... "*i was in the right location but the puzzle did not appear*"
... "*clicking was annoying, had to click many times to them work*"
... "*GPS did not work so good*"

Such problems are related either to the phone itself having to have to touch the screen many times until the user gets a response. Other reasons are the GPS sensitivity which created a sense of confusion both in the testing sessions and the evaluation sessions. Many of the subjects were very sure that they were in the correct location and were simply waiting for the puzzle to appear, but that did not happen on many occasions, making them rethink if that location was really correct or not, and basically not knowing what to do.

The design of the user interface which is the NPC character design, game design with menus and arrangement of elements as well of the animation of menus and controls of the game received very positive comments:
... “*nice graphics*”
... “*pictures were nice and professional*”
... ”*application looked professional*”

**Readability**

The subjects remarks about this aspect were not positive, represented in the following:
... “*readability not good*”
... *“too much text”*
... *“no scrolling bar is better*”
This implies the subjects dissatisfaction with the text, being the size of it – which did require the player to scroll down to read it- also the size of the text did not fit well into the mobile context, as a result of the device’s small screen’s size and having the small screen also fit a number of features all together (e.g. text, NPC’s and graphics) which played an important role in increasing the readability problem. On the other hand other subjects argued that the game has learning aspects applied to it, and the amount of text for that reason is reasonable. Another suggestion regarding readability by the subjects was to allow the player himself to decide whether he wants to read more or not, simply by putting a limited size of text on the screen accompanied with a “read more” button, and use small chunks of text rather than long passages.

**Interactions**

Underneath this aspect interaction issues fall, represented by the device itself and the game controls from within. From the product reactions cards results retrieved the main adjective describing this is slow. This and other remarks have been directly expressed by the subjects when speaking of the interaction process, and is clear through the following comments:
... “*clicking was annoying, had to click many times to get them work*”
... “*afraid to touch buttons on the phone*”
... “*could not hide the NPC*”
The device at hand can be interacted with in two ways, one are the keyboard and controls when sliding it open, this feature is not supported in *StreetDroids*. The other way, that was the chosen one in *StreetDroids,* is to use the touch screen for control. From the previous comments it is obvious that the phone posed some struggles on the subjects when attempting to click or interact within the game. Some complained that it was too slow and they got no reactions after clicking once or twice on a specific location on the screen and had to do that repeatedly until it actually gets touched and functions. The repeated touching of the screen, some even said they started touching the screen everywhere to get some sort of a reaction, caused a sense of confusion for the subjects. It had not given them any evident clue that something is happening in the

game based on their touch, so they simply kept on clicking elsewhere until something happened. The screen was not very sensitive to any touch, and on many occasions the same area had to be touched a number of times for it to be activated.

The size of the display of the phone posed some problems. Some subjects believed it would be a lot better to use a touch pen instead of the fingers because touch pens are more precise and smaller than the finger and might therefore decrease or eliminate the problem presented above.

Subjects also expressed their fear to touch any of the buttons on the phone itself while playing the game, fearing that it might turn off the phone, exit the application instantly, place a call or any other phone-related action. They assumed the phone buttons were only phone-related but actually they were made by the game designers to control the game as well, one of those buttons for instance being a “quit game” button, after which a confirmation message also appears.

Amongst the difficulties that appeared in the play-test regarding the game controls is that many subjects complained about not knowing how to make the NPC disappear -after he gives a clue or talks with the player- they tried touching the text bubble in which the NPC’s text appears as well as on any location on the screen. It was in fact designed to disappear by touching the small NPC icon itself that appears in the menu at the bottom of the screen. Related to that, another problem appeared being that the game menu which is at the bottom shifted to the bottom about another 50%. that made the menu half visible and most probably not understood and not functional.

**Feedback**

Interaction having been described above did at many times make it very unclear for the player to know whether he is getting feedback or not because of the low sensitive of the screen, that affected the feedback system of the game negatively and required the players at many points to repeatedly touch the screen on any location seeking a form of reaction from the game. Other than the device’s touch screen capabilities was also the GPS problem. Players were at many times in the correct location – where a specific puzzle is supposed to appear- but yet did not get it. This also can be related to the results of the product reaction cards using words in describing their experience as slow and frustrating. Puzzles take some time to download upon reaching the designated location and when this is happening no indication is given to the player that he is or not in the correct location. GPS also created confusion when one subject got the puzzle at a specific point “x” and another subject was in the same point “x” and did not get the puzzle.

**Help**

In the beginning of the play-test many subjects were confused about what to do, what to specifically follow on the screen, and during the focus group discussion they suggested that a system of help could be helpful when on the map navigating from one puzzle to the other. This help could be a:

... “*arrows to show which way to go*”

... "*make the street to be followed marked in red*"

In the game the player navigates from one location to the next by receiving hints, interpreting them and then proceeding to this location. But some of these hints could stand more than one answer. One hint asked the player to proceed to an administrative building as in ...*"You said you assume that people know where x building is. I wouldn't assume anything*". Some subjects said they skipped the instructions given at the beginning of the puzzle and tried their luck solving it on their own because they did not like to read a lot. Yet doing so placed them in a hard situation and many of them had to go back to the instructions of the help for a clearer insight on the goal of the puzzle. The help system in general was a point most of the subjects commented on saying there was a clear lack of instructions, not knowing what to do in the beginning.

#### 4.2.3.5. Playability

Playability results were mainly retrieved from the observation of the play-test session and the analysis of the focus group. The results from the questionnaire were indirectly considered; while the results of the product reaction cards were included in the analysis of the focus group as the testers had the chance to comment on their selection of the product reaction cards during the focus group.

**Concentration:**

As can be seen in Table 4.3 the majority of the subjects expressed that they felt attracted or engaged in the game. During the focus group the subjects used adjectives as innovative, friendly, fun and attractive to describe the game, however they also expressed their frustration specially with the technical problems that occurred. Despite these problems the players were able to maintain their interest in the game and to keep their focus. As one player explained: *"My first one* [adjective] *was frustrating because of all the restarting and not having the right location, but I could see behind all that and I thought this [the game] would be cool (...) that's why my second one* [adjective] *was engaging because I really wanted to find out, although there were things preventing me."*

While observing the play-test these type of reactions were evident. We could see that the players kept trying to find the right location or to launch a puzzle even when the system response was very slow or misleading. Even after having to restart the device they did not give up and gave the game another chance. We attribute this behavior to the attractive stimuli provided by the game that was worth of attention. In particular the players said they *"enjoyed the friendly graphics"* and that the game looked professional. As for the workload, the major issue were the technical problems that consumed much of the players' time and interfered with their concentration.

**Learnability:**

The learnability of the game was not very well rated and was perhaps one of the major problems for the game playability. Players expressed that they felt frustrated and confused because there were no instructions, or directions on how to use the game. To the question *One thing that you think that is missing in the game?* the majority answered that they were missing clear instructions. Some expected a long manual, and some others said they would never read a long manual or even try a separate tutorial. *"When you start the game you don´t know anything about items or hints (. . . ) you are not going to read the help file when you open it, you just wanna turn it on and use it."* To this issue one subject suggested that *"Maybe for each puzzle you can do an example in the beginning were you show what you want us to do".* During the play-test we could observe that these problems with the game learnability, which involved the game interface as well as the game content.

On the challenge criteria we could observe that many of the players had difficulties understanding the directions or hints to find the locations. Moreover, they expressed that the challenges in the game felt unbalanced, as they felt that finding the right location of the puzzle was much more difficult than solving the actual puzzle. On this topic one player suggested that we *"Make it easy to find the place and then make sure that the player has to find something else at that place (. . . ) or maybe the longer it takes give better clues."*

**Control:**

More play-testing would be needed to know if the players were able to feel control in the game, because the results of the evaluation where not conclusive for this criteria. There were many usability issues related with the controls of the interface i.e. difficulties with pressing buttons or understanding the menu options, and problems with location feedback. This usability issues made difficult to determine if the players were able to sense that their actions and choices have a meaningful influence in the game because the game flow was interrupted many times by the technical problems.

**Clear goals:**

As the learnability of the game was very low it was no surprise that the game objectives and goals were not clear for many of the players. During the play-test we could observe that the subjects found out very late that they could obtain an item after solving a puzzle, and if they did so, they did not pay much attention to it. When asked during the focus group if they noticed the item obtained after solving a puzzle, all of the subjects answered that they did not care much about it and many said that they skipped it without reading the information attached to it. It was also observed that some players had troubles to find the hint or help option offered by the NPC, which made it more difficult for them to finish the puzzle. One player even asked *"Do you receive an item after you read the text* [that comes with after you solve the puzzle]*? (...) I thought that was an additional information."* This shows the lack of clarity and consistency in how the goals of the game are presented (or not presented at all). From this we concluded that since the players were not aware of the rewards or help options, they did not get a full sense of achievement after playing the game.

**Immersion:**

During the focus group many of the players described the game as engaging and attractive, and they also affirmed that they could explore the city and appreciate it in a different way. However, there are no conclusive results for the criteria emotional involvement. Especially because the prototype used during the play-testing did not offer a full game mission, it only offered the players a couple of puzzles to try. For this reason the players did not have the chance to know the full story that the concept of the Bremen map included. This limited the players' possibilities to get emotionally involved with the game environment. Immersion was then analyzed in a more general way.

During the play-testing it was observed that immersion, as concentration, was also affected by the technical problems that occurred such as the slow download time of the puzzles or the incorrect or delayed location feedback. These issues were discussed in the focus group and the players agreed that even though they were attracted by the game they experienced frustration because of the technical problems that occurred. However, the players affirmed that if the problems were solved the game experience would be different. One particular subject said: *"...if everything was not slow, but very fast without interruptions the game is very interesting because it seems like a treasure hunt, and if there is no delay, no errors, it is extremely interesting."*

Besides the technical difficulties, that many of the players experienced in one way or another, some of them complained that they felt interrupted with the amount of text they received. They affirmed that the game would be more interesting if there was less text to read and if it was offered in a brief and more attractive manner. They suggested presenting the text as some sort of headlines with the option to read more if wanted. *"To me it would be more interesting if the game had the cool stuff, the most interesting things like you have on the cover of a magazine. Short hints* [information] *about the place and then you can read something more if you are interested."* Other players were not bothered by the amount of text and enjoyed the additional information. However, they said that they could agree with the *"headline"* option as long as they had the chance to get the additional information when they wanted.

**Social Interaction:**

This element from the playability model was not fully evaluated during the play-test because the prototype used did not include any of the collaboration features and the online community was also not tested. Nevertheless, during the focus group the subjects were asked questions that allowed us draw some conclusions as well as to collect suggestions that could help improve the design of the collaboration in the game and in the online community.

When we asked the subjects if during the play-test they considered asking people on the street for help to solve the puzzles or find the locations, there was a general agreement that it was not necessary. One player affirmed that she felt the game was very personal. Others said that asking for help to someone outside the game would not be productive because you could get many different answers and other people do not know what is going on in the game.

As for asking help from other players, many said they did it. Those who asked other players for help explained that they did it because they were confused and did not know what to do, specially in the beginning. During the play-test we could observe that, as the game started and the players did not have clear instructions, they got together in small groups and tried to find out what to do and how to do it. They discussed and tried different strategies until they figured out the way to find the location or how to solve a puzzle. However, in the end subjects were playing individually an only asked for help when there was a serious technical problem. To this situation one player added that *"*[in that situation] *it was ok that we had the group, but if you are alone the game has to be just clear and well described in the beginning or otherwise it would be very frustrating."*

During the focus group we explained the subjects our concept for a collaborative puzzle and we asked if they would like to play with strangers. The majority said that playing together with friends would be fine but not with strangers. One of the answers was that: *"It's too weird (. . . ) Because then someone knows where you are, they could stalk you".* Another subject said that *"this is a very personal game, and just maybe if you get some help from people on internet, people that have already played this game (...) it would not be so important, just another option, something extra in the future (...) like chatting but not physically meet, you don´t say to someone you don´t know 'hey would you like to join'."*

In this point the players were very clear and they all agreed that they would not like to meet face-to-face with other players they do not know. Some did not want to play with strangers at all. However, they could consider playing with strangers if they could use a more impersonal way of communication or if they could hide their location. One subject proposed that *"maybe one player can communicate information from the place he is to another player that is not there, so they can share information."*

Finally we presented our concept for an online community where players could create their own games, as Story maps, puzzles or missions. We explained them that there would be editors for many of the game components and that, despite some limited options, they could create new content by making new components for the game or edit existing ones. The overall reaction was that they liked the idea and could see that it could be very interesting for some people. However, many said that they will probably not create any content themselves.

#### 4.2.3.6. Satisfaction

After the play-test the subjects had the possibility to choose the adjectives from the product reaction cards. In Figure 4.3 the results are displayed in a tag cloud[5] that shows which adjectives had the most occurrence. The complete list of used adjectives can be seen in the Table 4.3 in 4.2.3.2.

[5]Made with `http://tagcrowd.com/`

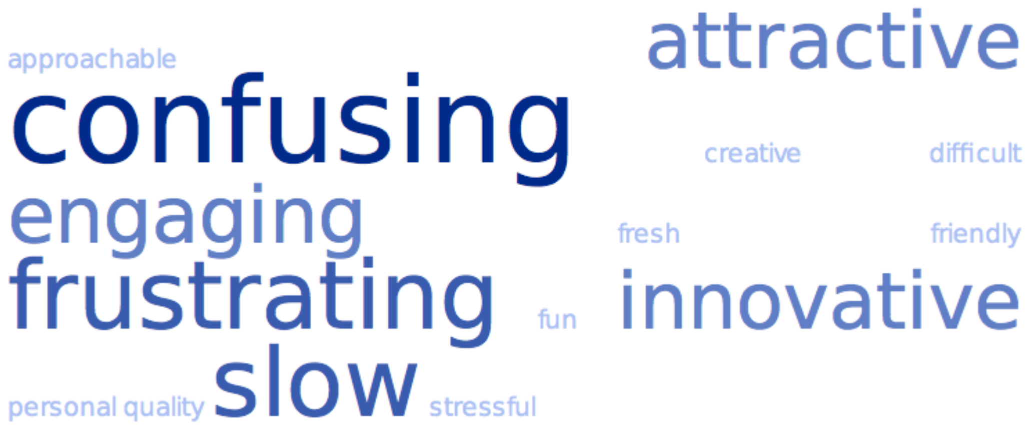


Figure 4.3.: Tagcloud with the used adjectives in the Product Reaction Cards.

On the one hand, the first impression shows that many of the players had the feeling that the game is frustrating, confusing and slow. On the other hand however, the subjects also mention that the game is innovative and attractive. In the following the results will be analyzed in more detail in order to get a better understanding why the subjects chose the named adjectives.

**The game concept**[6]

The subjects were interested in the general game idea (*Question 9*), although only one of them had some experience with similar games before (*Questions 4-7*). In the focus group the subjects mentioned as well that they like the game idea, but they were not satisfied with the technology. Too many problems occurred during playing, which hindered them in experiencing all the possibilities the game has to offer. One subject said for example: "*All the restarting and not having the right location (. . . ) I could see behind all that stuff and think wow this should be cool if it did what I wanted it to do.*"

Players were mainly frustrated, because...

. . . the phone reacts too slowly.

. . . the GPS position is not accurate.

. . . puzzles are not loaded.

. . . the application crashes.

Right from the beginning of the focus group it was getting obvious, that the whole group was frustrated due to technical problems. Hence one of the subjects mentioned: "*(...) I had technical problems, but i really wanted to do that, so i tried once, the second time... and unfortunately every other time was even worse with the location and stuff like that, so it was more frustrating (. . . )*" The disappointment of not being able to load

[6]The questions in brackets refer to the questionnaire in *Part 4.2.3.1*

all the puzzles for example was high. Only one of the participants had the chance to try out the Hotspot Puzzle. Yet, all the subjects were eager to unlock the second puzzle and tried over and over again. Specifying one of the problems and yet his interest in playing one participant stated: "*If everything was not slow, very fast, very quickly, without interruptions then it is very interesting because it seems like a treasure hunt. And if there is no delay there is no pauses, there is a lot of time, something like that, it is extremely interesting.*"

One of the main complaints of the subjects was the lack of instructions. They said they would have preferred to get more information on how to play the game (*Question 10*), especially how to solve the puzzles. Nevertheless, they were still able to play the game without much instructions.

Players in general agreed that due to the novel concept the game has to offer, it is...

... engaging.

... innovative.

... personal.

As (González Sánchez et al., 2009) mention the main objective of a game is to entertain, to bring fun to the player. One of the players used the successive phrasing: "*(...) It seems like a really powerful platform you guys have and that you can do a lot with it and it was a lot of fun. I could see that it could be fun (...)*" The overall impression drawn from the observation is that the subjects were excited about playing, as they were very ambitious in finding the correct locations and in solving the puzzle. It therefore can be concluded that they can imagine to play *StreetDroids* or a similar game again (*Question 9*).

Another aspect mentioned by (González Sánchez et al., 2009)is the topic of attractiveness, which is an aspect of a game that increases the satisfaction of the player. Everyone agreed that the graphics of the game are of high quality and they were very enjoyable. One statement representing these feelings towards the game was the following: "*(...) that's why my second one [adjective] was engaging.... cause it was really like... i really wanted to find out but there was things like preventing me from getting to that cool (...) game stage*." Another subject also decided for the adjective engaging, "*(...) because anyway if you like history if you like things like that, you just try to search the place and learn everything. (...)*".

**The outdoor scenario**

Before playing we asked the subjects, if they like playing a mobile game outside (*Question 3*). Only one of them answered with yes, four of the subjects did not like the idea at all, and one could not decide. Nevertheless, after playing the game four of the subjects said they could imagine to play such a game again, and only two answered with no (*Question 9*). One of the biggest problems the outdoor scenario presents is the dependency on the weather conditions. A problem related to that is the touch screen of the G1 which can only be used without gloves. As during the evaluation the conditions were not ideal, it was about -12℃ and very windy, most of the participants were wearing gloves to cover their hands, which complicated the handling of the mobile device considerably. It therefore needs to be said, that in order to have a better understanding of how well the outdoor scenario works, some play-testing in different weather conditions would be necessary.

**The exploration**

In the questionnaire that was posed directly after playing the game, most of the subjects had the feeling they could explore the city, at least under the terms of having less technical difficulties. Although in the beginning almost none of the players could imagine to enjoy playing outdoors (*Question 3*), they enjoyed exploring the city center of Bremen while playing. For more details on exploration aspects take also a look at *4.2.3.7 Explorability*.

#### 4.2.3.7. Explorability[7]

In the beginning the subjects were asked whether they themselves had the impression that they were able to explore the city, to evaluate if an interaction with the environment was happening as planned and if the player really explored their surroundings in order to solve the puzzles. Three subjects answered with yes, two with no, but those two mentioned that they could imagine being able to explore the city, if the technical problems they were experiencing while playing were solved.

Throughout the game the players receive textual hints that are connected to something they can see in real life. This supports their understanding of the space they are acting in. The group did not agree on the question whether more or less text would be better to support the learning outcome. Some of the subjects stated that they would prefer less information, similar to a headline of a paper, and only in case the player is interested to show more information. Other players said that they think the amount of text was adequate, especially because the player is supposed to study his surroundings.

One difficulty concerning the explorability that occurred during the discussions, was that the subjects said they had problems with the instructions. For them it was not always clear where to go, as for example one of the hints asks them to go to an administrative building, but for them it was not obvious which building could be meant by that. One of the subjects mentioned the following: "*You said you assume that people know where x building is. I wouldn't assume anything.*". Having trouble finding the locations is problematic, as the players might explore something else but the desired location and the prepared information do not fit to that.

They all agreed that the instructions to find the place should be easier, because it was one of the most frustrating points for them while play-testing. If they knew better where to go to, they could focus more on looking at their surroundings while walking to their destination and concentrate on the puzzle, where they also have to take a close look at buildings for instance. One of the players put it into the following words: "*You have to be there to know the answer, but maybe make it really easy to get there like...okay here is the building but then really inspect the building (...)*" This means that the hints need to

[7]Written by Jana Wedekind

be revised and probably several play-test sessions would be needed in order to make sure that they are understandable by people with different backgrounds.

It was important to find out if the subjects could remember factual knowledge that they were presented with during the puzzles. As the players for instance have the possibility to learn something about a building at-site, the chances that they will connect this knowledge better with the real object are higher. In order to understand if the subjects learned new facts, we asked them in the questionnaire, as well as in the focus group what they remembered. Everybody agreed that they learned something about the history and the city of Bremen. One subject mentioned this: "*I can remember some facts about the sailors coming to Bremen in medieval times.*". Another one said "*I learned something regarding the architecture of the Rathaus. (. . . ) And the statute... the Roland one.*" This shows that the subjects could indeed connect their knowledge to some specific places they visited.

One argument that supports the theory of exploration while playing is that subjects proposed to have the possibility of making a pause in between. One subject for instance was curious about the market and wanted to stop and eat there. For the players exploration means taking the time to discover. It became very obvious during the discussion, that the subjects rejected the idea of playing this kind of game on time. As one player said: "*You will lose the objective of the game, because if it is like according to time, you won't read... you will just hurry up... and the idea is to discover the city and the details. (. . . )*" They all preferred to have time to explore the city, and might even take a break in between playing to relax and enjoy the surroundings.

After the analysis it became obvious that it would be interesting to implement a functionality that allows the user to receive information about buildings or points-of-interest on-site while playing in real-time. This was suggested by some of the subjects and it could indeed improve the exploration of the players. This information could include additional facts, which are not only related to the next puzzle, but support the individual interests of the player. Moreover, this would take into consideration the interest that the subjects showed in exploring the space given in more detail and at their own pace.

## 4.3. Future Work[8]

[8]Cristina Botta, Isabella Lomanto

The concept is finished, but still open for various adaptations in the future. Due to time constrains, only a prototype, which still lacks many of the planned features, was developed. Furthermore some major technical problems were noticed during the evaluation process, which would have to be resolved for the game to be usable in a real world scenario. Additionally some game-play related problems were also observed, and if solved would greatly improve usability and enjoyment of the game. More information about those problems can be found in section 4.2.3.3, “General Observations”.

In the next paragraphs further possible improvements in the areas of content, collaboration, design, technical implementation, and evaluation will be discussed.

**Content**

One of the main characteristics of the game is that the user can actively contribute to the game by creating content. In the prototype of the game, basic characters and elements that the player can re-use were developed. Future work could include the development of further characters and objects. In addition, more *Story Maps* could be implemented and offered to the user.

Additionally, not all content researched for the Bremen history *Story Map* was implemented, and only three puzzles were developed from it. Future work could involve the further development of missions and puzzles for this scenario. The original plan was to have four missions comprising different periods of Bremen’s history, each containing at least five puzzles.

**In-game Collaboration**

During the concept phase of the collaborative puzzles the appropriate research in the field of collaborative games creation was done. During the design phase the necessary algorithm was structured explaining step-by-step the player’s actions and the logic of the puzzles, which should be programmed on the server side. Eventhough the concept and design phases are finished the technical implementation has not been started. A clickable prototype, which offers a common ground for both, designers and programmers, and simplifies further work, was developed. A proposal developed for the server-client communication and the logic algorithm should simplify the programmers’ work, but it does not exclude possible difficulties. These difficulties can influence some steps of the algorithm, but nevertheless will not change it drastically.

**Web Platform**

The web platform was only partially developed, and there is room for improvements also in the already implemented part. User profiles were not fully implemented and only a basic layout, create account, login/logout, and some basic statistics are working. Some sorting and display functions of the created elements exist, but more sofisticated means would be needed, for example, sorting and searchin by specfic types, tags, creator, etc. Adding a forum would also be necessary for a community oriented website. It was not also not possible to test the current layout and design for usability, which would be very important for a platform that has a large user base in mind.

**Editors**

Only two puzzle editors were fully implemented, for the Drag and Drop, and for the Hotspot puzzles. Development for the Missing Part puzzle, Characters, and Mission editors were started, so future work would include finishing those. Development of the items editor was not started. A functionality that was planned, and could be developed in the future, is the possibility to edit the already created elements.

**Indoors Implementation**

The initial implementation is for outdoor use only, but the general concept is applicable indoors as well. Nevertheless, the used technology has to be adapted, as for now GPS is used for locating the user, which is not a feasible solution for indoors. It has to be explored to which extend the technical requirements can be re-used, and which parts need to be rebuild.

**Client Architecture**

One of Android's peculiarities, which has been mentioned in section 3.2.2.1, "Activities", is the activities concept of the mobile platform application. The activity structure used in the final application design concept lacks some screens defined in the game mechanics document. This happened because during the game's concept implementation all features were prioritized, and only the ones with high priority made their way to the final prototype. Nonetheless, while developing this activities model a possibility for further development was always considered. Therefore, there is already a place reserved for those features which were not implemented yet, but could be included in future releases. The architecture which is currently in use has evolved during the project's development, and it is possible to continue this evolution if further development happens.

**Adding New Puzzles**

The possibility to implement new puzzle types exist through the creation of two main classes. One class extends the generic abstract class, and the other contains the logic specific to that puzzle. There is also the possibility to separate the view from the logic and create a new class for the view. It is important to understand how to extend the generic `PuzzleActivity` class and override the functionality of the methods in such a way that will suit a specific puzzle.

**Technique for Image Recognition**

During the project time all the steps of the algorithm related to the server side were implemented. All the functions are commented and described, and the use of the functions inside of the algorithm is clear. Nevertheless, the functions on the client side were implemented and tested only as a prototype and were not included in the puzzle. At the time of this writing the final integration of the puzzle has not yet been finished. One of the possible steps towards improvement could be to implement the comparison of the accelerometer's data on the client side to the contextual information of the original image provided by server. The deployment of these steps will influence the core structure of the server side and should be done jointly. It also influences the mobile part and forces the improvement of the application for uploading images to the web-server from the mobile device. Another possible direction can be the final integration of the puzzle into the structure of the project, testing, and further evaluation.

**Augmented Reality**

One of the possible expansions of this project is the implementation of augmented reality puzzles to create an even more engaging experience. For example, overlaying graphics of how a place used to look in the past over a current image presented on the screen, or showing extra information about the locations. More ideas of possible uses for AR in *StreetDroids* can be found in section 2.2.5, Mixed Reality.

**Design Considerations for Small Screens**

An important issue regarding future work on the design interface is the adaptation for different screen resolutions. The Android platform is available for devices with several different screen resolutions (Google, 2010j), and with different aspect ratios. It is not enough to simply scale the game to different resolutions, it is needed to adapt for different space proportions.

If the game is to be used by several kinds of users, it is important to consider the age factor. Discount-age applications have been developed (BBC, 2010), together with some criteria for the product interface design, such as simple and minimalist start menu and large icons and fonts.

> *"Adaptable interfaces, where the user can adapt a generic interface for his or her disability, are frequently put forward as a solution for the dynamic diversity exhibited in older users discussed in this section. Some people might say that those with different needs are supported by systems that allow the users to configure the interface to their own requirements. Microsoft Windows, for example, offers several adaptations, which can be invoked by the users to help them use the interface."*(Pirhonen et al., 2005)

Another issue for future implementations is the translation of the game, towards the internalization of *StreetDroids*. To this end, it is necessary to use only simple instances of English text, whenever possible. *"Simple English text will be easier and less expensive to translate."* (Galitz, 2008 Pg 571). As a recommendation, German should be the first European choice (after English) [idem] and, in this context, *StreetDroids* was placed in a positive environment.

**Evaluation**

Due to the limitations of time and availability of human resources the results of the evaluation can only be called preliminary. An improved concept and further playtest sessions with more than six subjects would be necessary to obtain more conclusive data. At that time the web platform was being developed, as were the puzzle, character, and mission editor. These features were not evaluated because they were not complete. However, for the evaluation process to be complete, and for the concept to be validated, all parts would have to be tested together.

# 5. Conclusion[1]

[1] Cristina Botta, Isabella Lomanto

Due to the limitations of time and human resources in the project the results showed on this report can only be called preliminary. The created concept proved to be very complex and difficult to develop. Time restrictions, the size of the team, and the different levels of knowledge were equally hard to manage. Additionally, exercising more constrain during the concept phase could have helped to keep the features more manageable.

Despite all the difficulties, the team was able to develop working prototypes of the most important concepts. Unfortunately, only the mobile client prototype could be tested since the web platform was not complete at the time of the evaluation. However, with the results of the evaluation we can assume that the original research question was answered in a positive manner.

Even though, *StreetDroids* is not necessarily a learning game, the evaluation showed that the subjects actually learned about the given content and connected it to the visited places. The interaction with the environment as well as with other players can create an immersive experience for players, which can motivate and enhance learning processes.

As mentioned in section 4.3 Future Work there are different ways and directions in which the game could be expanded. An improved concept, fixing of the technical problems, and further play-test sessions with more subjects would be necessary to obtain more conclusive data. However, the results at this point already show that spatial expansion can be achieved by using the presented game concept. In contrast to similar games at present, *StreetDroids'* objective is not only about using today's technology to lead the players to certain locations, but to make them aware of their surroundings. This objective was reached as the game allowed the players to connect the game world to the real world, to join digital content with real life objects as part of a pervasive gaming experience. Moreover, the evaluation showed that besides offering a meaningful way of interacting with their environment, the subjects also enjoyed this exploration and could imagine playing this game or a similar one again.

# Bibliography

Rup - role: Stakeholder, 2003a. URL http://rup.hops-fp6.org/process/workers/wk_sthld.htm.

Rup - artifact: Vision, 2003b. URL http://rup.hops-fp6.org/process/artifact/ar_vsion.htm.

S. Akkerman, W. Admiraal, and J. Huizenga. Storification in history education: A mobile game in and about medieval amsterdam, 2008. URL http://www.ilo.uva.nl/homepages/wilfried/docs/CAE_2008_online.pdf. [Accessed: 04.05.10].

ApacheSoftwareFoundation. The jakarta commons httpclient, February 2008. URL http://hc.apache.org/httpclient-3.x/. [Accessed: 2010.04.01].

Carmelo Ardito, Maria Francesca Costabile, Rosa Lanzilotti, and Thomas Pederson. Making dead history come alive through mobile game-play. pages 2249–2254, 2007. URL http://doi.acm.org/10.1145/1240866.1240989. [Accessed: 04.05.10].

Christine Bailey and Michael Katchabaw. An emergent framework for realistic psychosocial behaviour in non player characters. In *Future Play '08: Proceedings of the 2008 Conference on Future Play*, pages 17–24, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-218-4. doi: http://doi.acm.org/10.1145/1496984.1496988. [Accessed: 04.08.10].

BBC. Bbc news | technology | new pc to encourage older users, 10 2010. URL http://news.bbc.co.uk/2/hi/8352606.stm. [Accessed: 10.01.10].

J. Benedek and T. Miner. Measuring desirability: New methods for measuring desirability in the usability lab setting, 2002a. URL http://www.microsoft.com/usability/UEPostings/DesirabilityToolkit.doc. [Accessed: 04.01.2010].

J. Benedek and T. Miner. Product reaction cards, 2002b. URL http://www.microsoft.com/usability/UEPostings/ProductReactionCards.doc. [Accessed: 03.26.2010].

James Bennett. Writing reusable applications, 2008. URL http://media.b-list.org/presentations/2008/djangocon/reusable_apps.pdf. [Accessed: 2010.03.29].

I. Blecic, A. Cecchini, and P. Tronfio Rizzi. An innovative perspective for gaming simulation in making dead history come alive through mobile game-play. pages 337–348, 2002.

Joachim Bondo. *iPhone User Interface Design Projects*. Apress, Inc. New York, 2009.

Thomas Boutell. Portable network graphics (png) specification, November 2003. URL http://www.w3.org/TR/PNG/. [Accessed: 03.28.10].

Tim Bray and Jean Paoli. Extensible markup language (xml) 1.0, November 2008. URL http://www.w3.org/TR/REC-xml/#sec-cdata-sect. [Accessed: 2010.04.01].

BremenOnline. Bremen.de - fascinating facts about bremen, 08 2009. URL http://bremen.de/10293754. [Accessed: 08.26.09].

Sean Casey, Ben Kirman, and Duncan Rowland. The gopher game: a social, mobile, locative game with user generated content and peer review. In *'07: Proceedings of the international conference on Advances in computer entertainment technology*, pages 9–16, New York, NY, USA, 2007. ACM. URL http://doi.acm.org/10.1145/1255047.1255050. [Accessed: 04.05.10].

Elizabeth F. Churchill, Les Nelson, Laurent Denoue, Jonathan Helfman, and Paul Murphy. Sharing multimedia content with interactive public displays: a case study. pages 7–16, 2004. URL http://doi.acm.org/10.1145/1013115.1013119. [Accessed: 04.05.10].

Roger L. Costello. Building web services the rest way. URL http://www.xfront.com/REST-Web-Services.html. [Accessed: 2010.03.29].

Claudia Dappen. Bremen entdecken und erleben. 2008.

Hugh Davies. Place as media in pervasive games. In *IE '07: Proceedings of the 4th Australasian conference on Interactive entertainment*, pages 1–4, Melbourne, Australia, Australia, 2007. RMIT University. ISBN 978-1-921166-87-7. doi: http://delivery.acm.org/10.1145/1370000/1367963/a7-davies.pdf?key1=1367963\&key2=2496201721\&coll=GUIDE\&dl=GUIDE\&CFID=83814451\&CFTOKEN=34442824. [Accessed: 04.01.10].

Bernie DeKoven. *The Well-Played Game*. iUniverse, February 2002. ISBN 0595217907, 9780595217908.

Heather Desurvire and Charlotte Wiberg. Game usability heuristics (play) for evaluating and designing better games: The next iteration. In *OCSC '09: Proceedings of the 3d International Conference on Online Communities and Social Computing*, pages 557–566, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02773-4. doi: http://dx.doi.org/10.1007/978-3-642-02774-1_60. [Accessed: 04.04.10].

Heather Desurvire, Martin Caplan, and Jozsef A. Toth. Using heuristics to evaluate the playability of games. pages 1509–1512, 2004. URL http://doi.acm.org/10.1145/985921.986102.

Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, 2001. ISSN 0737-0024. doi: http://dx.doi.org/10.1207/S15327051HCI16234_02. [Accessed: 04.05.10].

Software Foundation Django. Form wizard documentation. URL http://docs.djangoproject.com/en/dev/ref/contrib/formtools/form-wizard/. [Accessed: 2010.03.29].

Paul Dourish. Re-space-ing place: "place" and "space" ten years on. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 299–308, New York, NY, USA, 2006. ACM. ISBN 1-59593-249-6. doi: http://doi.acm.org/10.1145/1180875.1180921. [Accessed: 02.04.10].

T. M. Duffy and D. H. Jonassen. *Constructivism and the Technology of Instruction: A Conversation*. Lawrence Erlbaum Associates, Mahwah, NJ, USA., 1992.

Carlo Fabricatore, Miguel Nussbaum, and Ricardo Rosas. Playability in action videogames: a qualitative design model. *Hum.-Comput. Interact.*, 17(4):311–368, 2002. ISSN 0737-0024. doi: http://dx.doi.org/10.1207/S15327051HCI1704_1. [Accessed: 04.01.10].

R. Fielding, UC Irvine, and J. Gettys. Hypertext transfer protocol – http/1.1, June 1999. URL http://www.ietf.org/rfc/rfc2616.txt. [Accessed: 2010.04.01].

B. Fox. *Game Interface Design*. Thomson Course Technology PTR, Boston, 2005.

G. R. Frederick and R. Lal. *Begining Smartphone Web Development*. Apress, Inc. New York, 2009.

W. O. Galitz. *The Essential guide to User Interface Design: An Introduction to GUI Design Principles and Techniques - 3rd edition*. John Wiley & Sons, New York, USA, 2008.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, Amsterdam, 1995.

Adrian Kaehler Gary Bradski. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

C. Ghaoui. *Encyclopedia of Human Computer Interaction*. Idea Group Reference, 2006.

José Luis González Sánchez, Natalia Padilla Zea, and Francisco L. Gutiérrez. From usability to playability: Introduction to player-centred video game development process. In *HCD 09: Proceedings of the 1st International Conference on Human Centered Design*, pages 65–74, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02805-2. doi: http://dx.doi.org/10.1007/978-3-642-02806-9_9. [Accessed: 04.01.10].

Google. Google xml document format style guide, 2008. URL http://google-styleguide.googlecode.com/svn/trunk/xmlstyle.html. [Accessed: 2010.03.29].

Google. Android developers: Activity, 2010a. URL http://developer.android.com/reference/android/app/Activity.html. [Accessed: 03.31.10].

Google. Android developers: Alpha animation, 2010b. URL http://developer.android.com/reference/android/view/animation/AlphaAnimation.html. [Accessed: 31.03.10].

Google. Android developers: Animation, 2010c. URL http://developer.android.com/reference/android/view/animation/Animation.html. [Accessed: 31.03.10].

Google. Android developers: Button widget, 2010d. URL http://developer.android.com/reference/android/widget/ButtonWidget.html. [Accessed: 31.03.10].

Google. Android developers. http://developer.android.com/index.html, 03 2010e. URL http://developer.android.com/index.html. [Accessed: 03.16.10].

Google. Android developers: Image button, 2010f. URL http://developer.android.com/reference/android/widget/ImageButton.html. [Accessed: 31.03.10].

Google. Android developers: Image view, 2010g. URL http://developer.android.com/reference/android/widget/ImageView.html. [Accessed: 31.03.10].

Google. Android supported media format: Image format, 2010h. URL http://developer.android.com/guide/appendix/media-formats.html. [Accessed: 31.03.10].

Google. Android developers: Motion event, 2010i. URL http://developer.android.com/reference/android/view/MotionEvent.html. [Accessed: 31.03.10].

Google. Supporting multiple screens | android developers, 03 2010j. URL http://d.android.com/guide/practices/screens_support.html. [Accessed: 03.25.10].

Google. Android developers: Rotate animation, 2010k. URL http://developer.android.com/reference/android/view/animation/RotateAnimation.html. [Accessed: 31.03.10].

Google. Android developers: Scale animation, 2010l. URL http://developer.android.com/reference/android/view/animation/ScaleAnimation.html. [Accessed: 31.03.10].

Google. Android developers: Translate animation, 2010m. URL http://developer.android.com/reference/android/view/animation/TranslateAnimation.html. [Accessed: 31.03.10].

Google. Android developers: Handling ui events, 2010n. URL http://developer.android.com/guide/topics/ui/ui-events.html. [Accessed: 31.03.10].

The Hanse. City league the hanse, 2009. URL http://www.hanse.org/en/the_hansa/. [Accessed: 09.04.09].

C. Hardy. Activity theory - an introduction, n.d. URL http://osiris.sunderland.ac.uk/~cs0car/hci/3_con_at.htm. [Accessed: 09.28.09].

Barbara Hayes-Roth and Patrick Doyle. Animate characters. *Autonomous Agents and Multi-Agent Systems*, 1(2):195–230, 1998. ISSN 1387-2532. doi: http://dx.doi.org/10.1023/A:1010019818773. [Accessed: 04.08.10].

J. Henderson. The paper chase: Saving money via paper prototyping. may 2006. URL http://www.gamasutra.com/features/20060508/henderson_01.shtml. [Accessed: 04.08.10].

O. Hennessy and C. Kane. *iPhone Games Projects*, chapter Starting with a Game Design Document: A Methodology for Success, page 132. Apress, Inc. New York, 2009.

Adrian Holovaty and Jacob Kaplan-Moss. The django book. http://www.djangobook.com/, 2009a. URL http://www.djangobook.com/. [Accessed: 03.27.10].

Adrian Holovaty and Jacob Kaplan-Moss. *The Definitive Guide to Django: Web Development Done Right*. Apress, Berkeley, CA, second edition edition, 2009b.

Andrew Hunt and David Thomas. The pragmatic bookshelf. list of tips. http://www.pragprog.com/the-pragmatic-programmer/extracts/tips, 2000. URL http://www.pragprog.com/the-pragmatic-programmer/extracts/tips. [Accessed: 03.27.10].

H. Irgtel. H. irgtel - color systems: Johann wolfgang goethes's farbenkreis, 03 2010. URL http://www.uni-mannheim.de/fakul/psycho/irtel/colsys/GoetheFarbkreis.html. [Accessed: 03.23.10].

11581-1:2000(E) ISO/IEC. *Information technology - User system interfaces and symbols - Icon symbols and functions - Part 1: Icons - General*. ISO, Geneva, 2000.

Henry Jenkins. *Textual Poachers: Television Fans and Participatory Culture*. Routledge, New York, 1992.

Henry Jenkins. *First Person: New Media as Story, Performance, and Game.*, chapter Game Design as Narrative Architecture., pages 118–130. The MIT Press, 2004.

Henry Jenkins. *Convergence Culture: Where Old and New Media Collide*. NYU Press, 2006. ISBN 0814742815, 9780814742815.

D.H. Jonassen and L. Rohrer-Murphy. Activity theory as a framework for designing constructivist learning environments. *ETR&D*, 47:61–79, 1999.

M. Jones and G. Marsden. *Mobile Interaction Design*. John Wiley & Sons, Ltd., 2006.

Aki Järvinen, Satu Heliö, and Frans Mäyrä. Communication and community in digital entertainment services. prestudy research report, 2002. URL `http://tampub.uta.fi/tup/951-44-5432-4.pdf`. [Accessed: 04.03.10].

Jacob Kaplan-Moss. Snakes on the web, 2009. URL `http://jacobian.org/writing/snakes-on-the-web/`. [Accessed: 2010.03.29].

V. Kaptelinin. Activity theory: implications for human-computer interaction. pages 103–116, 1995.

Eric Klopfer. *Augmented Learning: Research and Design of Mobile Educational Games*. MIT Press, 2008.

KMBdesigns. The meaning of the color grey / gray, 03 2010. URL `http://www.kmb-designs.com/colors/grey.html`. [Accessed: 03.23.10].

A. Kofod-Petersen and S.A. Petersen. Learning at your leisure: Modelling mobile collaborative learners, 2008.

Hannu Korhonen and Elina M. I. Koivisto. Playability heuristics for mobile games. pages 9–16, 2006. URL `http://doi.acm.org/10.1145/1152215.1152218`.

P. Kortum. *HCI Beyond the GUI: Design for Haptic, Speech, Olfactory, and Other Nontraditional Interfaces*. Morgan Kaufmann, 2008.

R.A. Krueger and M.A. Casey. *Focus Groups: A Practical Guide for Applied Research*. Thousand Oaks, CA: Sage Publications, 3rd edition, 2000.

K. Kuutti. Activity theory as a potential framework for human-computer interaction research. In B. A. Nardi, editor, *Context and consciousness: Activity Theory and Human-Computer Interaction*, pages 17–44. MIT Press, Cambridge, MA, 1996.

J. Laird and M. van Lent. Interactive computer games: human-level ai's killer application. In *AAAI National Conference on Artificial Intelligence*, pages 1171–1178, 2000. URL http://doi.acm.org/10.1145/1178823.1178828. [Accessed: 04.08.10].

LearningTheoriesKnowledgebase. Cognitivism at learning-theories.com - learning theories knowledgebase, 2010. URL http://www.learning-theories.com/cognitivism.html. [Accessed: 09.21.09].

Jinah Lee and Chang-Young Im. A study on user centered game evaluation guideline based on the mipa framework. pages 84–93, 2009. URL http://dx.doi.org/10.1007/978-3-642-02806-9_11.

Pierre Levy. *Collective Intelligence: Mankind's Emerging World in Cyberspace*. Perseus Books, Cambridge, MA, USA, 1997. ISBN 0306456354. Translator-Bononno, Robert.

A. Liljedal. Design implications for context aware mobile games. 2002. URL http://www.enactive.com/publications/liljedal-thesis.pdf.

S. Lundgren and S. Björk. Game mechanics: Describing computer-augmented games in terms of interaction, 2004. URL http://www.cs.chalmers.se/~lundsus/lundgren_bjork_game_mechanics.pdf.

J. McGonigal. *The Puppet Master Problem: Design for Real-World, Mission Based Gaming*. MIT Press, 2007.

Kathryn Merrick and Mary Lou Maher. Motivated reinforcement learning for non-player characters in persistent computer game worlds. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 3, New York, NY, USA, 2006. ACM. ISBN 1-59593-380-8. doi: http://doi.acm.org/10.1145/1178823.1178828. [Accessed: 04.08.10].

Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. In *IEICE Transactions on Information Systems*, volume Vol E77-D, No.12, December 1994.

Markus Montola. Exploring the edge of the magic circle: Defining pervasive games. In *CD-ROM Proceedings of Digital Arts and Culture. Copenhagen*, pages 1–3, 2005. doi: http://users.tkk.fi/mmontola/exploringtheedge.pdf. [Accessed: 01.04.10].

E. K. Moore and P. A. Simpson. *The Enlightened Eye: Goethe and Visual Culture*. Editions Rodopi B.B., Amsterdam - New York, 2007.

MountainGoatSoftware. Learning scrum - the product backlog. URL http://www.mountaingoatsoftware.com/scrum/product-backlog. [Accessed: 28.03.2010].

Lennart Nacke. From playability to a hierarchical game usability model. In *Future Play '09: Proceedings of the 2009 Conference on Future Play on @ GDC Canada*, pages 11–12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-685-4. doi: http://doi.acm.org/10.1145/1639601.1639609. [Accessed: 01.04.10].

C Nutt and M Kumar. Panel: Why user-generated content matters for games, june 2008. URL http://www.gamasutra.com/php-bin/news_index.php?story=19029. [Accessed: 04.05.10].

OpenHandsetAlliance. Open handset alliance. http://www.openhandsetalliance.com/, 2010a. URL http://www.openhandsetalliance.com/. [Accessed: 03.20.10].

OpenHandsetAlliance. Open handset alliance: Overview. http://www.openhandsetalliance.com/oha_overview.html, 03 2010b. URL http://www.openhandsetalliance.com/oha_overview.html. [Accessed: 03.20.10].

B. Pan, G. Gay, J. Saylor, H. Hembrooke, and D. Henderson. Usability, learning, and subjective experience: user evaluation of k-moddl in an undergraduate class. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 188–189, New York, NY, USA, 2004. ACM. ISBN 1-58113-832-6. doi: http://doi.acm.org/10.1145/996350.996392.

Celia Pearce. *Communities of Play: The Social Construction of Identity in Persistent Online Game Worlds*. The MIT Press, 2009.

R. E. Pedersen. *Game Design Foundations*. Wordware Publishing, Inc., Plano, Texas, USA, 2003.

D.C. Phillips and J.F. Soltis. Perspectives on learning. Teachers College Press. URL http://www.funderstanding.com/content/behaviorism. [Accessed: 09.20.09].

David Pinelle, Nelson Wong, and Tadeusz Stach. Heuristic evaluation for games: usability principles for video game design. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1453–1462, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: http://doi.acm.org/10.1145/1357054.1357282. [Accessed: 04.08.10].

David Pinelle, Nelson Wong, Tadeusz Stach, and Carl Gutwin. Usability heuristics for networked multi player games. In *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 169–178, New York, NY, USA, 2009. ACM. URL http://doi.acm.org/10.1145/1531674.1531700.

A. Pirhonen, H. Isomäki, C. Roast, and P. Saariluoma. *Future Interaction Design*. Springer Verlag London, 2005.

M. Pivec, A. Koubek, and C. Dondi. Guidelines for game-based learning. 2004.

M. Prensky and S. Thiagarajan. *Digital game-based learning*. McGraw-Hill, 2007.

F. Rabiee. Focus-group interview and data analysis. In *Proceedings of the Nutrition Society (2004)*, volume 63, pages 655–660, 2004. URL http://journals.cambridge.org/action/displayFulltext?type=1&fid=902312&jid=PNS&volumeId=63&issueId=04&aid=902300. [Accessed: 01.20.10].

Josephine Reid. Design for coincidence: incorporating real world artifacts in location based games. In *DIMEA '08: Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*, pages 18–25, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-248-1. doi: http://doi.acm.org/10.1145/1413634.1413643. [Accessed: 01.04.10].

Klaus Renzel and Wolfgang Keller. Client/server architectures for business information systems. 1997. URL http://www.objectarchitects.de/ObjectArchitects/papers/Published/ZippedPapers/renzel.pdf. [Accessed: 03.30.10].

B. Reynolds. How ai enables designers, apr 2004. URL http://gamasutra.com/phpbin/. [Accessed: 04.08.10].

Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly Media, 2007.

Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike. *Android Application Development: Programming with the Google SDK O'Reilly Series*. O'Reilly Media, Inc., 2009.

A. M. Ronchi. *eCulture: Cultural Content in the Digital Age*. Springer-Verlag Berlin, 2009.

RubiconConsulting. The apple iphone:successes and challenges for the mobile industry. a study of iphone users, 03 2008. URL http://rubiconconsulting.com/downloads/whitepapers/Rubicon-iPhone_User_Survey.pdf. [Accessed: 01.04.10].

K. Salen and E. Zimmermann. *Rules of Play: Game Design Fundamentals*. MIT Press, 2003.

M. Sharples. Big issues in mobile learning. Report of a workshop by the kaleidoscope network of excellence mobile learning initiative, University of Nottingham, 2007.

M. Sharples. Methods for evaluating mobile learning. In G.N. Vavoula, N. Pachler, and A. Kukulska-Hulme, editors, *Researching Mobile Learning: Frameworks, Tools and Research Designs*, pages 17–39. Peter Lang Publishing Group, 2009.

M. Sharples, J. Taylor, and G. Vavoula. Towards a theory of mobile learning, mobile technology: the future of learning in your hands. In *Proceedings of the MLEARN 2005*, 2005.

SkyhookWireless. Application developers survey: iphone, android, palm, symbian, j2me, rim, windows mobile, 2009. URL http://www.locationrevolution.com/stats/SkyhookDevelopersSurvey2009.pdf. [Accessed: 01.04.10].

R. E. Snow. Abilities and academic tasks. In R. J. Sternberg and R. K. Wright, editors, *Mind in Context: Interactionist perspectives on human intelligence*, pages 1–38. Cambridge University Press, New York, 1994.

Carolyn Snyder. Paper prototyping sure, it's low-tech, but this usability testing method can help you sidestep problems before you write your code, nov 2001. URL http://www.snyderconsulting.net/us-paper.pdf. [Accessed: 04.08.10].

Carolyn Snyder. *Paper prototyping: the fast and easy way to design and refine user interfaces*. Morgan Kaufmann, San Fransisco, CA, USA, 2003.

J. Spolsky. *User Interface Design for Programmers*. Apress, Inc. New York, 2001.

Jared M. Spool, Tara Scanlon, and Carolyn Snyder. Product usability: survival techniques. In *CHI '98: CHI 98 conference summary on Human factors in computing systems*, pages 113–114, New York, NY, USA, 1998. ACM. URL http://doi.acm.org/10.1145/286498.286560. [Accessed: 04.08.10].

SQL. BNF grammar for ISO/IEC 9075:1992 - database language SQL (SQL-92). http://savage.net.au/SQL/sql-92.bnf.html, 2010. URL http://savage.net.au/SQL/sql-92.bnf.html. [Accessed: 03.21.10].

K. Squire. Content to context: Videogames as designed experience squire. *EDUCATIONAL RESEARCHER*, 35:19–29, 2006.

X. Sun, T. Plocher, and W. Qu. *Usability and Internationalization - HCI and Culture*, chapter An Empirical Study on the Smallest Comfortable Button/Icon Size on Touch Screen, page 615. Springer-Verlag Berlin Heidelberg, 2007.

P. Sweetser and P. Wyeth. Gameflow: A model for evaluating player. *ACM Computers in Entertainment,*, 3(3), jul 2005. URL http://doi.acm.org/10.1145/1077246.1077253. [Accessed: 01.02.10].

T-Mobile. The t-mobile g1 with google phone, 2009. URL http://www.t-mobileg1.com/. [Accessed: 09.14.09].

TeachingSupportServices. Learning objectives: A basic guide, 2003. URL http://www.tss.uoguelph.ca/resources/idres/learningobjectives1.pdf. [Accessed: 09.21.09].

J. Tidwell. *Designing Interface - Patterns for Effective Interaction Design*. O'Reilly, Sebastopol, 2006.

Joshua Topolsky. Mobile os shootout: iphone os 3.0 enters the fray, 2009. URL http://www.kmb-designs.com/colors/grey.html. [Accessed: 01.04.10].

Anders Tychsen, Michael Hitchens, and Thea Brolund. Character play: the use of game characters in multi-player role-playing games across platforms. *Comput. Entertain.*, 6(2):1–24, 2008. ISSN 1544-3574. doi: http://doi.acm.org/10.1145/1371216.1371225. [Accessed: 04.08.10].

L. Uden. Activity theory for designing mobile learning. *International Journal of Mobile Learning and Organisation 2007*, 1(1):81–102, 2007.

R. Wagner. *Professional iPhone and iPod touch Programming: Building Applications for Mobile Safari*. Wiley Publishing Inc., 2008.

Charlotte Wiberg, Kalle Jegers, and Heather Desurvire. How applicable is your evaluation methods - really? analysis and re-design of evaluation methods for fun and entertainment. In *ACHI '09: Proceedings of the 2009 Second International Conferences on Advances in Computer-Human Interactions*, pages 324–328, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3529-6. doi: http://dx.doi.org/10.1109/ACHI.2009.54.

Wikipedia. Scrum (development), 2010a. URL `http://en.wikipedia.org/wiki/Scrum_(development)`. [Accessed: 28.03.2010].

Wikipedia. Wizard (software), 2010b. URL `http://en.wikipedia.org/w/index.php?title=Wizard_(software)&oldid=347750342`. [Accessed: 29.03.2010].

Terry Winograd. Architectures for context. *Hum.-Comput. Interact.*, 16(2):401–419, 2001. ISSN 0737-0024. doi: http://dx.doi.org/10.1207/S15327051HCI16234_18. [Accessed: 04.05.10].

José P. Zagal, Jochen Rick, and Idris Hsi. Collaborative games: lessons learned from board games. (1):24–40, 2006.

# A. Visual Documentation

Other design aspects not covered by the final report. This section serves as visual documentation of the *mobileHIVE's* project, based on voted contests made by the project members.

## A.1. mobileHIVE

### A.1.1. Name of the Project: Chosen by votes, at 29.05.2009

**Name's** concept: Bees are having a very social nature and are building communities, which is what we want the users to create by using our game. Their hive resembles our busy group that tries to develop this game.

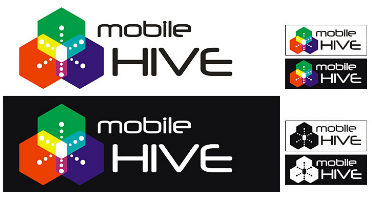
**Authors:** Jana Leoni Wedekind, Cristina Laura Botta

### A.1.2. Logo of the Project: Chosen by votes, at 05.06.2009

**Logo's** concept: In this logo three hive "capsules" are combined suggesting the union of colors, which can be also a metaphor for the union of different skills backgrounds that are presented in our group. In the center, the result of this colors' union is another hexagonal form in white color, which irradiates content for all corners, symbolizing the mobile aspect of our project.

**Author:** Joatan Preis Dutra

**Logo:**



## A.2. StreetDroids

### A.2.1. Name of the Game: Chosen by votes, at 08.10.2009

**Name's** concept: Androids on the street.

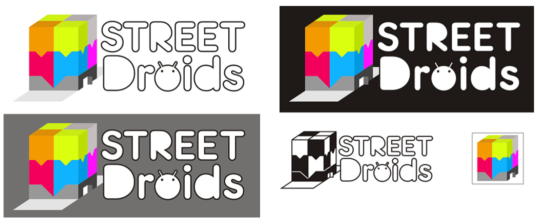
**Author:** Isabella Lomanto

### A.2.2. Logo of the Game: Chosen by votes, at 05.11.2009

**Logo's** concept: Depicted are building blocks, which have an identical shape, but differ in color. They are stacked and interconnected with each other to form a new unit which fits perfectly into a foundation. The foundation has a door, which is opened - signaling at the openness of our platform. To make the connection to the wordmark and the environment the game is played in, the building-like structure is multidimensional and situated on a street leading from the fore- to the background.

**Author:** Till Hennig

**Logo:**



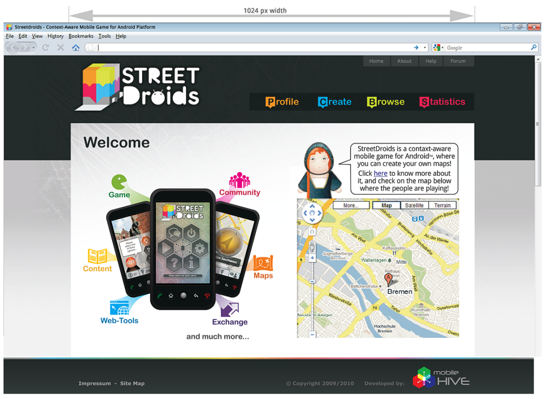
## A.3. WebDesign

### A.3.1. WebDesign model for the Game: Chosen by votes, at 19.11.2009

**Design's** concept: Androids on the street.

**Author:** Joatan Preis Dutra

**Original** Concept:


1024 px width
Streetdroids - Context-Aware Mobile Game for Android Platform
File Edit View History Bookmarks Tools Help
Google
Home About Help Forum
STREET Droids
Profile Create Browse Statistics
Welcome
Game
Community
Content
Maps
Web-Tools
Exchange
and much more...
More... Map Satellite Terrain
Bremen
Impressum - Site Map
© Copyright 2009/2010 Developed by:
mobile HIVE

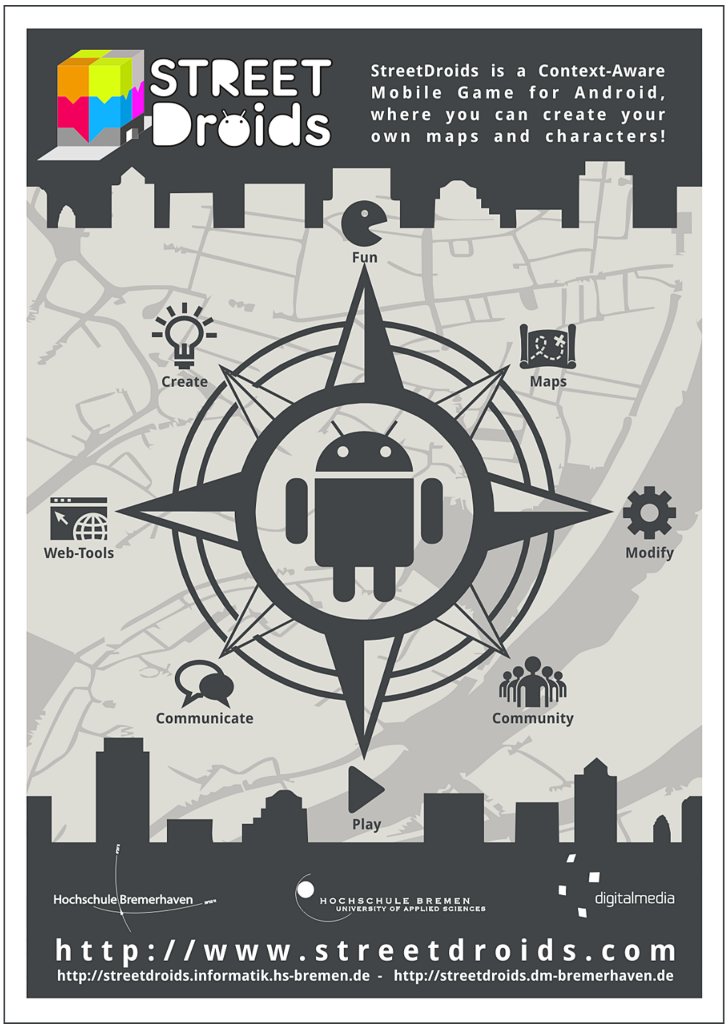
StreetDroids is a contaxt-aware mobile game for Android™, where you can create your own maps!
Click here to know more about it, and check on the map below where the people are playing!

## A.4. Poster for promotion

### A.4.1. Poster for the Game: Chosen by votes, at 05.02.2010

**Author:** Joatan Preis Dutra

**Original** Concept (A2 format):


STREET Droids
StreetDroids is a Context-Aware Mobile Game for Android, where you can create your own maps and characters!
Fun
Create
Maps
Web-Tools
Modify
Communicate
Community
Play
Hochschule Bremerhaven
HOCHSCHULE BREMEN UNIVERSITY OF APPLIED SCIENCES
digitalmedia
http://www.streetdroids.com
http://streetdroids.informatik.hs-bremen.de - http://streetdroids.dm-bremerhaven.de