Social Voxel Level Editor

Rabie Khodr Jradi

DTU

Kongens Lyngby 2013
IMM-M.Sc.-2013-77
In this project, a tablet-sized prototype has been developed. The application is a platform for gamers to create, discover, play and share user generated content for a hypothetical game. All user generated content is made of cubes called voxels.

Behavioural science and persuasive technology, along with existing related games in the market today (e.g. Minecraft, Vox and Lego) have been used for the analysis of user needs and for the conceptualization and design of the prototype application. The platform attempts to encourage the user to engage in the content creation and further increase the creative experience of creating content using a social dimension.
Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Informatics.

The supervisors of the thesis are Jakob Eg Larsen and Michael Kai Petersen from the Department of Informatics and Mathematical Modelling at the Technical University of Denmark.

Lyngby, 15-August-2013

Rabie Khodr Jradi
I would like to thanks each of the test participants for your valuable feedback, and to Thor Helms and Kasper Roos for your time and feedback on this report.

Furthermore, thanks to my supervisors Jakob Eg Larsen and Michael Kai Petersen for being supportive and flexible in the idea of the game platform itself and letting me define a master’s project based on that.

And finally, a big thank you to my brother, mother and father for their motivational support throughout the project and your endless supply of delicious Arabic food! You have saved me a lot of cooking time and done everything you could so I could focus on my education (and this thesis).
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td>Preface</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>v</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Game platform Concept and Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1 Challenges and Risks</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Goal</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Scope</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Report Structure</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Glossary</td>
<td>4</td>
</tr>
<tr>
<td><strong>2 Related Work</strong></td>
<td>7</td>
</tr>
<tr>
<td>2.1 Human Behaviour</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 Neurophysiology</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 Behaviour and Attitude Change</td>
<td>8</td>
</tr>
<tr>
<td><strong>3 Similar Games</strong></td>
<td>13</td>
</tr>
<tr>
<td>3.1 Vox</td>
<td>13</td>
</tr>
<tr>
<td>3.2 Lego</td>
<td>15</td>
</tr>
<tr>
<td>3.3 Minecraft</td>
<td>16</td>
</tr>
<tr>
<td>3.4 Gameglobe</td>
<td>19</td>
</tr>
<tr>
<td>3.5 Worms</td>
<td>20</td>
</tr>
<tr>
<td><strong>4 Persuasion and User needs</strong></td>
<td>23</td>
</tr>
<tr>
<td>4.1 Motivating the user and persuasion techniques</td>
<td>23</td>
</tr>
<tr>
<td>4.1.1 Notifications as a Trigger</td>
<td>25</td>
</tr>
</tbody>
</table>
## Contents

4.1.2 Meta-gaming ........................................... 26

4.2 User Needs ............................................. 26
  4.2.1 Feedback on Similar Games ....................... 26
  4.2.2 Usage Pattern of Tablet Devices ................. 28
  4.2.3 The Target User .................................. 29
  4.2.4 Scenarios ........................................ 31
  4.2.5 Use cases ....................................... 32

4.3 Multiplatform Implications ............................... 34

4.4 Voxel Editors .......................................... 34
  4.4.1 Editor Types ...................................... 35
  4.4.2 Editor Learnability ............................... 35
  4.4.3 Editor Discoverability ............................ 37

5 Design .................................................................. 39
  5.1 Conceptual Paper Prototypes ......................... 40
    5.1.1 First Paper Prototype ............................. 41
    5.1.2 Second Paper Prototype ......................... 48
    5.1.3 Third Paper Prototype ............................ 54
    5.1.4 Choosing a Solution ............................... 57

5.2 Computer-based Prototype ............................... 57
  5.2.1 Sequential Flow ..................................... 58
  5.3 Tablet Prototype ........................................ 60

6 Implementation ............................................. 63
  6.1 Datastructure for a Prototyping Framework ......... 63
  6.2 Software Components .................................. 65
  6.3 Using the Prototyping Framework ..................... 66

7 Evaluation ..................................................... 69
  7.1 Heuristic Evaluation of Concept Prototypes ........ 69
    7.1.1 Method ........................................... 70
    7.1.2 Findings .......................................... 71
  7.2 Empirical Evaluation of Computer-based Prototype 72
    7.2.1 Method ........................................... 73
    7.2.2 Findings .......................................... 74

8 Discussion .................................................. 77
  8.1 Results ................................................ 77
    8.1.1 Iterations .......................................... 77
    8.1.2 Evaluations ........................................ 78
    8.1.3 Tablet Implementation ............................. 79
  8.2 Future Work ............................................. 79
    8.2.1 Alternative Content Discoverability .......... 79
    8.2.2 Future Evaluation of Product .................... 80
Companies such as Mojang known for the computer game Minecraft, and Lego have had success targeting the human creativity and imagination, by offering a simple way to construct objects and whole worlds from bricks. Minecraft is especially an interesting case as it takes a Lego-like game into the virtual world and offers even less possibilities with cubes\(^1\) only in one size. Even with this simplification there are many examples of complex constructions and worlds that users have created.

The lessons learned from these two examples are that simple 3D shapes in the correct context may be enough for a user to have fun. For computer game developers this means a lot less resources/money is needed to create highly polished 3D models. This is especially beneficial for independent game developers since these usually don’t have a lot of resources and therefore developing a game might suddenly be feasible by a couple of coders alone.

With the variety of ways for indie developers to distribute their games to a large audience we see today such as App store, Google Play, Steam, etc. and the availability of low cost (multi-platform) 3D game engines such as Unity3D, the barrier has been lowered to a level where it is actually possible for these developers to attempt competing with the large AAA game companies on the game market. Finally

\(^1\)referred to as voxels
In the following section I will present to you the concept and motivation behind an indie game for which is the basis of this thesis.

1.1 Game platform Concept and Motivation

The basic idea and the long term goal based on the work done in this project is to build an online platform where users are able to contribute with the actual content of the game. We want to offer a platform that possibly will support multiple different game genres, each with their own game mechanics, along with the tools necessary for the users to build worlds and constructions that will be part of their own world they want to create within a genre. When a user have created their own world, they are able to distribute this through our platform (central service) in order for other users to discover, play, get inspired, and further share it with their friends.

The idea behind distributing the responsibility of content creation stems from the frustration observed by gamers that crave for more content in the games that they really like, and that the collective mass of users will be able to eventually create content that is far more exiting compared to a handful of game designers. What we hope to achieve with this, assuming that we have the critical mass required to actually create some creative content, is that the users will never run out of new worlds to experience. This is very similar to how other content centric services work, such as app stores such as iOS, Android, etc. and entertainment services such as Spotify, Netflix, etc. The success of these and other services rely very much on the existence of new undiscovered content that is able to satisfy the users’ need for new experiences, with the aim of not letting the user get bored and eventually look for other sources of entertainment.

The content that users are able to create would be based on coloured voxels. The benefit is both that development of a voxel-based game needs much less resources compared to highly polished 3D models, and that the nature of the cube shape lets the user abstract away from the small details of a polished 3D model and lets them instead fill the missing visual gap with their imaginative power of the brain itself. When a kid picks up a wooden stick on the playground and starts to shoot fictive bad-guys with the stick, we are witnessing how the brain is able to imagine a more immersive world by abstracting away from the fact that the kid is holding a wooden stick and not a functioning gun with all its mechanical details.

One of the most important criteria for the success of this platform is to ensure that the users are motivated to create the content we hope for, and therefore it
is very important that a large part of the platform is built around rewarding the
users whenever they create content, and especially create content that others
like. Collaborative creation with friends and other rewarding mechanisms is
therefore needed, and possibly also offer extrinsic rewards such as money for
the authors of popular content.

The reason to eventually offer multiple genres and different game mechanics is
so users can create and play worlds that fit their liking.

1.1.1 Challenges and Risks

As you have probably been thinking while reading, this concept is quickly be-
coming a big project that contains many risks and a lot of development time.
And even if we are able to offer a compelling platform there is no guarantee
of success. There is a large risk that the service in relation to the economics
may be a bad business. One example of this is Lego which had to completely
shutdown and discard their Massively multiplayer online game (MMOG) known
as Lego Universe despite actually creating a game that had very positive feed-
back by its target audience and having spent an enormous amount of money
in development costs.

1.2 Goal

The goal of this master’s project is to create a mobile prototype for a voxel level
editor which is to motivate the users to create content for a voxel-based game.

The emphasis of this project is on:

• Prototype a frontend that is able to offer useful tool to the users so they
  are able to create the content they like.

• Utilizing social and other persuasive elements to motivate the users to
  create content.

The frontend is to be developed for tablet-sized devices, specifically 7 inch tablets
such as the Nexus 7.

1.3 Scope

The development of the platform will assume that the game mechanics and gameplay of the content that exist in the platform itself already exists, therefore this project is constrained to look at the platform merely as a toolbox and not actually a game.

In addition the emphasis is on prototyping and for this project it means to compromise on the implemented functionality in order to focus more on the design process.

1.4 Report Structure

Apart from this chapter, Chapter 2 presents the theoretical work which this thesis is based on, followed by a presentation and analysis in Chapter 3 of existing games on the market that is similar to the concept of this project. Chapter 4, which is the second part of the analysis phase, looks at the concept from the perspective of the related work of Chapter 2, the similar games of Chapter 3, and finally the analysis of user needs.

Based on these analysis chapters, Chapter 5 presents prototype solutions for how the knowledge may be applied. Chapter 6 then presents the work of implementing a prototyping framework along with an Android application. Chapter 7 presents the heuristic and usability evaluations conducted on the developed prototypes, followed by a discussion of these results and future work in Chapter 8. Finally, Chapter 9 summarizes all the previous chapters, especially the discussion in Chapter 8 and states to which degree the project goal have been achieved.

1.5 Glossary

**Voxel** A pixel in 3D space, i.e. a coloured cube.

**Massively multiplayer online game (MMOG)** A genre of games where the users play together/against each other over the public internet.

**User generated content** A term used for e.g. services and games that rely on the users to create the information that actually makes the product worth using.
1.5 Glossary

**In-game** The 3D voxel environment that a user may build structures.

**Sandbox** The general term for any type of in-game environment where the user is able to control his/her avatar freely (within the rules of the game mechanics).

**Independent game developer (Indie developer)** Are creators of games that do not rely on funding from video game publishers.

**LAN** A "local area network", which interconnects a group of computers that are physically close to each other, such as in a home or school.

**Non-player character (NPCs)** Is any character that is never controlled by a player.

**Horizontal prototype** Has the purpose of showing the UI of the screens that make up the application, thus a horizontal prototype does not implement any of the available features. It gives the broad view of an entire system.

**Vertical prototype** Focuses on implementing a prototype of the functionality for single selected use case. It gives a narrow, vertical slice view of a small part of the system.
Related Work

This chapter will present important background knowledge that his project relies on.

2.1 Human Behaviour

Since the fundamental motivation for the game concept and the goal of the project is to create an engaging play/building experience, it is important to understand the underlying behavioural mechanisms in humans so that the platform meet the users expectations.

2.1.1 Neurophysiology

Current research[Sch98] in the field of neurophysiology basically states that human behaviour is reward driven with the release of the chemical compound known as dopamine. Dopamine is basically what motivates us to get out of bed in the morning and seek out rewarding behaviour with the intent of releasing more dopamine. Dopamine is released, when we eat and drink, reproduce, socialize, learn new skills, etc. Positive reinforcement is felt for any behaviour
Figure 2.1: Our needs, prioritised in a pyramid shaped model

that contributes to the fulfilment of any of the human needs and negative reinforcement when seeking behaviour that is damaging to your chances of survival. Figure 2.1 is a model that attempts to illustrate the priority of our needs for what a fulfilling life requires. This model is useful to get the very basic understanding of the kind of role that our platform has in terms of targeting a certain need. Our platform concept obviously attempts to target the top layer of the model, where we have needs such as creativity or problem solving (for winning in game). Figure 2.1 is known as "Maslow’s hierarchy of needs" and this may help us explain why we are rewarded for a given type of behaviour.

2.1.2 Behaviour and Attitude Change

In extension to the knowledge we have about the reward-driven behaviour of humans, B.J. Fogg have conducted research in the area where computers and persuasion overlaps. He labels this field "Captology"\cite{Fog02}. Out of his work he has created what he refers to as "Fogg’s Behaviour Model" (FBM)\cite{Fog09}. FBM is a framework that may help designers in the development of interactive products that has the intention of persuading the user to a certain behaviour. FBM basically postulates that behaviour is not alone the result of motivation, but a convergence of three factors that must be present simultaneously:

1. Motivation
2.1 Human Behaviour

Motivation is present when we are in need. Ability explains how well a person is capable of doing some behaviour, or how simple it is. A trigger is anything that activates a motivated person with the ability to actually do a certain behaviour. It is something that tells people to perform the behaviour now.

Figure 2.2 illustrates the concept of FBM, as a two-dimensional system with ability and motivation is seen on the x- and y-axis respectively. Unlike Maslow’s, Fogg do not attempt to structure the motivational subcomponents. The figure shows that the best chance for achieving a target behaviour from a person is if both motivation and ability are high. If a trigger is present in exactly that moment, the behaviour would then be likely to happen. Note also that the model says that motivation and ability may compensate each other. Figure 2.2 illustrates the possibility for these two elements to compensate each other with the curve marking an activation threshold. As an example, a student may have a high motivation for getting a master’s degree and that would compensate for any disability (e.g. reading or concentration problems, not having the time it takes, etc.) that this student might have.

Each of the three elements (motivation, ability and trigger) has a number of subcomponents that explains them further. Motivation is composed by three
core motivators:

- Pleasure/pain
- Hope/fear
- Social acceptance/rejection

Ability is present when a behaviour is simple to do, so in that sense without simplicity there is no ability. Elements of simplicity are:

- Time
- Money
- Physical effort
- Brain cycles
- Social deviance
- Non-routine

Any behaviour that requires a high amount of the listed simplicity elements contributes to lower the ability of that person to perform the target behaviour. And finally three elements of triggers as seen by Figure 2.3. A spark is the type of a trigger that is useful when the person isn’t motivated. As an example, a movie trailer may function as a spark for that person to go to the cinema and see the film even if the motivation where low to start with. On the other scale of the axis, a facilitator is used when a person has high motivation but low ability. In the example of the student, a facilitator here could be a mentor or personal teaching assistant that helps the student making the goal of getting a master’s degree an easier task to overcome. The facilitator therefore makes a target behaviour easier to do. The third type of trigger is a signal and is a reminder for the person who is both high in motivation and ability. The challenge with a signal is thus a question of timing, so that all three factors may converge. Let us say the student has an exam next morning and therefore goes to bed early and sets the alarm. When the alarm signals, the student has hopefully slept all the hours needed (ability), and doesn’t want to miss the exam (motivation) and is therefore very likely to wake up on the signal of the trigger.
Figure 2.3: Three types of triggers exist.
The concept behind Social Voxel\(^1\) (SV) is not a completely innovative idea, but inspired from several other games, social services and other content serving software. In terms of innovation, Social Voxel may be seen as an incremental step.

The games described are all important for designing a platform which builds on existing solutions known already in order not to reinvent the wheel. Basically these games are used to get inspiration and ideas for what kind of features games contains, how the UI is designed and how to increase users desires to participate in content creation for games. Positive and negative elements of functionality will be exemplified, but solutions (the \textit{how}) for possible enhancements and ideas for features are reserved for Chapter 5.

\section*{3.1 Vox}

Vox\(^2\) is also an indie developed RPG for PC in its alpha stage, which is extremely customizable in every aspect.

\(^1\)For simplicity the platform is referred to by the name "Social Voxel"
\(^2\)http://www.vox-game.com/
The current release of Vox (v0.37.2 in the time of writing), is very unstable and buggy so it is hard to actually find out how it is solving the interesting challenges of customization and multiplayer capability. Currently multiplayer is not available so the only thing to look at is how customization is possible for avatars, weapons/items and non-player character (NPCs). The editor used for all of these is the same, with a slight difference for how the avatar editing works. Figure 3.1 shows the initial screen of Vox’s avatar editor.

The user is basically able to choose among prefabricated available body parts (head, body, legs, etc.) and then choose one of these parts to edit at a time by clicking the pencil icon. Clicking on this icon leads to the same editor used for editing weapons and the rest of the mentioned types. This editor can be seen in Figure 3.2.
In this editor the user is able to perform changes to each individual voxel. The user is able to remove, add and change colors of voxels, through the tools and color chooser on the left panel. The problem with this panel is the color chooser is not very intuitive and that the tools (seen by the radio button in the top section) could be supported with icons so that the user is able to recognise what the effect of the tool will be with a quick glance in contrast to now where the user must read the label to understand what the tools do.

The positive about this 2-step editing of the avatar is that full controls are not initially available unless the user specifically presses the pencil icon. This is an example of progressive disclosure, which is basically to hide away (more complex) information to secondary screens [NB12] p. 59. The danger of letting the user is able to fully edit their avatar right away for any new user who have just installed the game, is that they might get discouraged to play the game if the editing is too much of a challenge. In relation to FBM this would be an example of low ability, which will not lead the user to perform the behaviour (assuming that their motivation isn’t very high). It would be more suitable if the detailed editor was a feature that was given to users that has proven themselves capable, either through successful behaviour in gameplay itself or through some kind of point system.

3.2 Lego

Lego has existed before computers were used for gaming and is still enjoyed by a lot of kids and adults. The concept behind Lego is to construct whatever you like from different shaped (typically squared) plastic bricks that fit together in many different combinations. The more pieces you have, the more possible combinations you have to create the thing you want. Lego is an example of how to balance of freedom (the number of combinations), shape (size of the plastic pieces) and colors in a way that positively rewards with creativity and spatial learning skills to the player.

Although Lego is based on physical plastic blocks that fit together in varies ways, this product is in important inspiration to the basic concept of creating everything ranging from imaginative figures to massive real world representations of almost anything. The primary use case is usually centered on the building process while actually using the creations (e.g. playing with them) is a secondary objective.

The main weakness in physical blocks is clearly the limited resources to create

[^1]: http://www.lego.com/ds-dk/
whatever the user might like and that it is asocial both in terms of the building process but also that sharing or playing with other’s creations has a high barrier. It is exactly this barrier that the SV platform will be able to address with a central service since it is a software product where all creations are virtual.

3.3 Minecraft

Minecraft is mainly inspired from Infiniminer. It is a voxel-based sandboxing game that lets user play together by letting the user setup their own LAN or online servers. Minecraft’s main gameplay is based on survival and exploration by finding resources in the world that can be used for crafting new tools and build constructions in order to survive through the dangerous "Creepers" that attempts to kill, especially during the night. Minecraft is an example of how a game with no real objective becomes fun because of the player’s ability to build creative structures for surviving the Creepers. Minecraft is an important source for inspiration since it also offers a mobile gaming experience (both mobile and tablet devices), and therefore demonstrates one way to solve any design issues there is to navigate and create things in a voxel world. Unfortunately it is also the only game that has a mobile port of the games mentioned.

Minecraft attempts to leverage some of the mentioned barriers by letting the user create and play in a virtual world, but even though it has proved to be a very successful game there are still potential to lower the social barriers and lead/persuade it’s users to engage to the virtual world even more.

Minecraft is an important product to consider for several reasons. It has proven that it is possible to create an engaging virtual sandbox environment. More importantly it has even simplified the number of different building blocks to roughly a single, one sized block while still proving that the users’ creativity is able to build impressive creations that are very similar to Lego in complexity.

Minecraft is very similar to the proposed concept of this project but as stated, but lacks a deeper integration of social features in the system and to reward motivated users with the means to take larger control in the building process (i.e. offer more advanced tools and combinations). In other words, increase the capability of these users.

Unless the user installs a third party modification, the game itself does not offer any customization capability for the user’s avatar, weapons, NPC etc. The game focuses solely on letting the user modify individual voxels when the user

\[https://minecraft.net/\]
3.3 Minecraft

Figure 3.3: First-person view of the in-game Minecraft world

is inside the voxel world. Figure 3.3 shows a snapshot of Minecraft’s in-game voxel world.

The user may navigate the avatar to move in the voxel world by using the displayed controls on the left and use swipe gestures with the right hand in order to look around in all directions. This is the typical way to control an avatar in a first-person game and imitates the way it is done in the PC also, so it may be assumed that the user is familiar with the interaction.

The only problem with this style of navigation is that the left buttons usually have a fixed position on the screen, but this is not very helpful in a touch device since the user have no way of knowing whether he is actually pressing the buttons. On a PC keyboard the user does not have this problem as physical buttons allow the user to sense if he is pressing it. The consequence with this solution is thus that the user might get frustrated if he e.g. gets killed by a creeper in the game, because he wasn’t able to run away. In a stressful scenario the user would have time to focus on where his finger is located relative to the controls. A solution to this usability problem could be to replace the arrow buttons with a joystick widget, and the user is then able to press anywhere on the left side of the screen to activate the widget in the position of the finger, which would enable the user to slide his finger in any direction resulting in movement of the avatar.

When examining Figure 3.3 a bit closer, the center of the screen has a cursor and the voxel that this cursor points at is highlighted. Tapping anywhere on

\footnote{Note that for readability and simplicity reasons, the user will always be referred to as male. Constructs such as (s)he, he/she or his/her is simply an annoyance.}
the screen (not on any of the buttons though) will result of a new voxel added on top of the highlighted position. A user may possibly get confused at first and expect that the voxel should be added at the position where he is tapping at, but once the user notices the center cursor and highlighted voxel, it will be clear for the user how the interaction of individual voxels work.

As opposed to a standard tap, a long tap on the screen will initiate the removal of any highlighted voxel, and continue to remove any other highlighted voxel until the user releases the tap gesture. Finding out that the one has to long tap for removing voxels may not be as intuitive as hoped for and that is the general usability problem with gestures, which in contrast to visible UI buttons is not discoverable. This is trade-off between clogging the screen with all kinds of buttons for each possible action and tool, and on the other extreme having to memorize all the gestures needed to successfully use the application.

When the user wants to start a game, they are only able to choose between two options, as shown by Figure 3.4. Clicking on any of the two will launch the procedural generation of a 3D world, but unfortunately the user has no means to control any parameters that affects the output. In SV this is considered an important part of the creation process, i.e. if the user wants to create his own world and not play already available content, as a quick way to create a large part of the basis for the rest of the creation process. Chapter 5 shows how this may be solved in different ways.
3.4 Gameglobe

Figure 3.5: An example of the editing tools available in GameGlobe

3.4 Gameglobe

Gameglobe[6] demonstrates how creative user generated content doesn’t need to be based on voxels. The content is polygon-based and gives a much more polished experience. The game makes it possible to shape the contour of large landscapes relatively effortlessly with the tools it offers, and published the created worlds for others to play on the platform.

It would also make sense to offer a way to manipulate (add/remove) larger portions of voxels in SV by letting the user choose between different sizes (in a "voxel unit scale") of brushes. This could drastically reduce the time and amount of interaction needed to create a large crater or mountain when the user is running around in a voxel world.

Figure 3.5 a snapshot of the editing environment. The figure shows a sphere shape that is controlled by the user’s avatar. This sphere and the other available options below is a tool to alter the landscape. Naturally these tools may be scaled in size by the user so that the sphere’s volume increases thus affecting a larger portion of the landscape at a time.

In addition a copy/paste-like feature of blocks could lower the time it otherwise
would take to recreate a certain structure. As an example the user might build a small hut and want to build an entire village out of that hut. Without a way to copy/paste this group of voxels it would take a long time to build. In Gameglobe there is the notion of a "props" which is a list of prefabricated 3D models of objects that can be put in the world as the user sees fit. Figure 3.6 shows the props selector menu. As it can be seen by the figure all of these props are high-polished 3D models. A similar concept could be used in SV, but instead with voxel based objects, otherwise the platform will not be able to utilize the creativity of the users building their own props.

3.5 Worms

Worms is a 2D turn-based game for PC where each player controls a group of small worms. These worms have a range of different weapons and the objective is then to kill the other team’s worms before they defeat you. A battle takes place on a 2D platform/landscape with the worms standing on it.

Even though the game itself is fairly simple, the thing that continues to make Worms fun to play is that the 2D platform where the battle takes place is procedurally generated. Each battle is thus always a different experience for the user. The interesting thing is that the user is able to specify parameters for this generator and the game succeeds in balancing the amount of freedom and control.
**Figure 3.7:** Procedural editor of the game Worms: World Party

Figure 3.7 shows a snapshot of the procedural generator editor where the user is able to specify the output of the generator in a fairly simple way. The user is able to switch between different generated noise-maps that specify how the contour of the platform should be, other parameters such as water level, and even directly draw the landscape with a brush tool.

---

Note that the red integer labels are normally not part of the editor. Source of image: [https://tinyurl.com/lt6bw32](https://tinyurl.com/lt6bw32)
Chapter 4

Persuasion and User needs

In this chapter will analysed different successful persuasive designs used in the social services and some games, analyse the user needs.

4.1 Motivating the user and persuasion techniques

Persuasion through use of intrinsic and extrinsic rewards could both be used in the platform, though it is known that intrinsic rewards generally are a more powerful motivator [Wei11] p. 125. Extrinsic rewards is basically anything physical that is of value that the user may get, while intrinsic rewards is intangible to measure such as the feeling of being appreciated or recognised by others.

Intrinsic rewards for letting the user create content in the system could be given through a point system that is similar to StackOverflow. StackOverflow

Figure 4.1: Shows how a profile name in StackOverflow is connected to a point system
is basically a collaborative Q/A website for people having problems in any sort of mathematics, physics, programming etc. StackOverflow stand out from other similar sites by being collaborative (like Wikipedia) and in the way profiles are connected to a point system that functions similar to how levelling systems work in games. The service uses the point system as a measure of the engagement of the user. The degree of engagement is simply a number labelled "reputation". Figure 4.1 shows an example of how this looks like. The word reputation itself is immediately associated with the social acceptance motivational element and since the amount of reputation a user has also is public for all others to see, it makes the point system important to maintain. Getting more reputation thus functions as an intrinsic reward. In SV, points would be given each time the user successfully does a behaviour which would contribute to the content.

Letting users be able to love/like/favor each other’s content (i.e. mark it as favorite) which results in the creator earning points is a way to offer intrinsic rewarding for a behaviour that is associated with creating content. This kind of functionality would be similar to Facebook’s "like" button, and it is evident that getting likes is the digital version of getting social acceptance. The point system will be discussed further in Section 4.4.2 regarding how users can learn to use a tool that allows more freedom as they progress. The intention of a "love" feature is that impressive content created by a user is eventually "loved" by many so that having a large number of "loves" is in itself a motivation for users to create additional content.

As a supplement to the point system, it could be considered to reward highly popular content with real money, which would make the platform resemble the idea of app stores. The danger this kind of extrinsic motivation is that the user may become reluctant to perform the behaviour later on unless money is again a motivator [KAFH01]. If money should be used as a motivator, it makes less of a problem if the reward comes unexpectedly to the user. It has been observed [CA10] that people are more unconscious motivated if they previously have experienced an unexpected reward after performing a behaviour, and would therefore increase the behaviour when doing it a next time. So a solution could for example be that SV uses a probability function for calculating if a user is to be monetary rewarded for some content based on a random factor that also depends on how many users that have "loved" this users’ content. In this way, the system would be able to motivate the users who are able to produce popular content by using both intrinsic and extrinsic rewards in a way that does not backfire since the user cannot understand the exact pattern for how rewards are handed out.

As an alternative to real money it could be considered to use a virtual currency that is integrated to the game itself. This kind of solution depends on how the gameplay is designed so that the currency fits in, in a natural way.
4.1 Motivating the user and persuasion techniques

In order to attract users to SV, the user must be motivated enough to invite his friends to the platform. This is a one-time behaviour similar to what is seen in other services such as Facebook, LinkedIn and Dropbox. Dropbox rewards the user with extra free hard disk space when successfully referring to friends. An intrinsic reward could be used in SV by giving the user additional points if they invite their friends.

4.1.1 Notifications as a Trigger

In order to remind the user about the existence of the application it could use the notification system in the OS itself that both Android and iOS offer, to show a non-interrupting push message about some important event or friend activity in the system. If the system tracks that the user usually plays the game together with some specific friends the system could give a notification in the notification tray of the device that these friends are online and have joined a game.

A study [BT13] for a personal informatics app that experimented with unobtrusive notifications to act as a trigger for the user to enter data for tracking food consumption, resulted in more than 5x increase in data collection, simply as an effect of showing an icon on the top bar (notification tray) of the mobile device. Same idea could be applied here.

Using notifications would basically function as a trigger for the user to re-enter the system and is what is needed if both the user’s motivation and ability is present. The ability in this context is simply how easy it is for the user to enter and continue with whatever the user was doing in the system last time. This may be achieved by ensuring a persistent system state whenever the user exits the application and by letting the user enter the app in an easy way by simply tapping the notification itself. The timing of the trigger is crucial for the notification to become persuasive instead of annoying. The key is to find out when the user’s motivation is high.

The naive solution would be not to consider this aspect at all and just present a notification each time relevant information is present. The naive solution has obviously the disadvantage that the user might easily get annoyed by notifications when in fact the user is not interested in this information at all. The next step is to utilize the user’s points as a simple metric to understand the motivation of the user through the engagement the user have had previously. Finally, the design of the notification trigger could be based on a multifaceted function that attempts to estimate the user’s motivation based on the points, the time of day and days of the week that have been recorded as the user’s
playing pattern and how often the user interacts with the particular friend for which the notification is about.

4.1.2 Meta-gaming

The construction process may be seen as a kind of meta-game, since there is no real objective in this stage for user except using creativity to create a world for others and him to play. Such kind of meta-game is seen in many other sandboxing games also, from Grand theft Auto to The Sims. In both of these games the player might deviate from the main objective, and begin to explore hidden content or unintended outcomes of the sandbox environment. As a user would be able to cooperate with other users they might start constructing things that do not contribute with content that other might want to play. For example two friends might decide to tease a third friend by attempting to lock him inside a huge structure in case that person is away from the game for a moment. This is an important characteristic for sandbox games/platforms since the more fun a user might have while in the creation process, the more likely they associate content creation with a positively rewarded behaviour. The sandbox environment may thus in itself contribute to achieving the target behaviour.

4.2 User Needs

This section will now focus on establishing the user needs in order

4.2.1 Feedback on Similar Games

The first step to understand the needs of the user is to look at the current games that exist similar to SV. Unlike Chapter 3 that analysed the games themselves, this section will instead look at the games from a user’s perspective in terms of what needs they have.

Many systems such as games, websites and online media services usually have some kind of feedback loop for the user so that the user easily can complain about bugs, request features, ask questions or simply for praising. Some of these systems are implemented so the feedback is sent directly to the developers, without the possibility for other users to read them. App stores are for example by design open for others to read these reviews.
4.2 User Needs

As a designer and developer of the SV system, this functions as a cheap information resource to get a basic understanding of the users’ needs. The consideration to keep in mind with this is that the quality of the feedback may vary greatly and a lot of the data is simply unsuitable for use and therefore some filtering must be applied. One great advantage of this information source is that it can be assumed that most of these comments originate from the type user that SV attempts to target.

The users’ comments may be divided into the following categories:

1. Feature requests
2. Bugs and updates that break the application
3. Usability issues and experience issues
4. Praise

The sum of the user comments in the demo and paid versions of Minecraft on iOS and Google Play Vox’s forum and GameGlobe’s forum is enormous which would be impossible for a human to go through, so the approach have been to skim through parts of the comments that seemed most useful. The problem is clearly that the outcome of the approach depends on a subjective opinion of the reviewer. One could think of using a program to filter the dataset based on a linguistic analysis similar to how spam filtering works which could give some very interesting observations, but doing this kind of analysis would risk using too much time on that task alone.

The observed feedback is based on the app stores and forums mentioned above and the following list is the result of the applied approach:

- Online play with family and friends
- New NPC, animals, creatures and more content
- Autosaving of created constructions/worlds, i.e. data loss prevention

This list is clearly very condensed, but as much of the feedback is either related too much specifically for the games themselves, i.e. gameplay this would be
out of scope to discuss here. Other ideas from the users are also given, but some of these are simply out of scope or not popular enough to be feasible for development.

As described earlier in Section 1.1 one of the central aspects of SV is that the content is created by the users themselves. Depending on the quality and the amount of "producing" users of the system the need for new (hopefully fun) content would be satisfied by distributing the content creation to the users themselves, i.e. empowering the motivated users to create any missing content themselves. The goal of such a feature would be to prevent boredom of consuming the same content all the time (exactly the problem with Netflix-like services), and create a stronger ownership and relationship experience between Social voxel and the user as he establishes a bond to his created content.

4.2.2 Usage Pattern of Tablet Devices

Developing applications for mobile devices adds additional constraints to as these devices are used differently compared to traditional applications. The many different contexts in which mobile device may be used as a result of their portability means that there is a much higher risk for the application to lose the users attention. Even when the user is focusing his attention to the device, the installed apps themselves compete against each other to some degree, for attention through the use of notifications, audio including alarms and music, etc. Finally the device itself has some hardware related limitations, most important of these is typically a short battery life. All these factors shape the user's device usage pattern and the applications must be designed with these factors in mind in order to meet the users' needs.

For mobile phones this means many short bursts of app consumption. [FMK+10] studied 255 owners of mobile phone devices found that that the pattern of their daily use of apps was between 10 to 200 short bursts of app consumption of 10-90 different apps. A single burst would span between 10 to 200 seconds.

For tablet devices a similar pattern emerges, but because they usually don’t fit the user’s pockets, the context in which these devices are used is different and thus the different pattern. A study [MGW12] conducted on 33 tablet users showed that they mostly used the device for fun and relaxation purposes such as watching videos, social networking, gaming and looking up cooking recipes. The typical context is at home on the couch, bed or in the kitchen, for an average duration of 36 or 48 minutes in weekdays and weekends respectively. In particular the study showed that most people playing games would be on the couch, at home or in the car.
Finally a study [PRG+12] on a MMOG game called Parallel Kingdom released on both tablet and mobile phone devices for iOS and Android, showed an average of 10 minutes burst of gaming. The study itself is large compared to [FMK+10] and [MGW12] with over 600,000 unique users distributed across more than 100 countries therefore it contributes to a more realistic picture of the real timespan that a tablet application may expect to run in a single burst. Note though, that the gameplay has an impact on the ability to generalize the observed average burst times to other games, but the study is still useful to give an insight of the usage pattern.

These studies indicates that it is a good idea for tablet applications to consider the same kind of requirements, regarding short start-up time, minimizing the time it takes to actually start to play/create and being able to continue from previous state after sudden interruptions. Not having these features in SV could lead or contribute to a frustrating user experience.

Note that the longer bursts seen in tablets fits well with the choice of developing the SV platform for this kind of device compared to a mobile phone because it is expected/hoped that the creating content is a process better suited for longer periods of interaction. Compared to mobile phones, the screen size of a tablet also gives a better overall user experience [NB12] p. 15.

4.2.3 The Target User

In order to better determine the user needs of this system, it has to be made clear what characterises the typical user, and thus personas will be presented here.

Humans are intelligent in different ways and are typically not equally intelligent in each form. While one person may understand and solve complex mathematical or logical problems another is maybe more capable of understanding music and its patterns. [Kos10] presents the following forms of intelligence which originates from the work done by Howard Gardner in [Gar93]:

1. Linguistic
2. Logical-mathematical
3. Bodily-kinesthetic
4. Spatial
5. Musical
6. Interpersonal
7. Intrapersonal

Why is this relevant? As the type of the game concept and tool is fundamentally dependent on some degree of spatial perception (basically because it is a voxel-based building environment) this is a very important characteristic for the typical user of SV. [Kos10] claims that there is a tendency of males having a higher degree of spatial perception while females (generally speaking) are more emotionally intelligent. Based on this generalization the gender of the typical target user of the system is thus male. However because there is an increasing amount of female gamers [12] [Ale09] and since it is known that intelligence isn’t just based on genetics only but may be increased through training and experience throughout life, it is expected that the future user of SV thus is likely to include female players as well. The success of games such as on social networks (e.g. Zynga’s FarmVille on Facebook) where approximately 60% of players are females [Su10] [Par11] also helps increase the exposure of gaming as a behaviour. This may motivate to explore other types of games, some of which might train the user’s spatial perception.

SV does not purely rely on spatial perception since it is the goal of the system also to provide a higher level of user experience through interactive play (e.g. coop) with other real players, which means that some degree of social skills is also necessary. Again generally speaking, females tend to have an advantage in this area, and therefore the complexity of the target user increases as well.

4.2.3.1 Personas

With the analysis of the target users, fictional persona descriptions will now be given for the expected SV stereotypes.

**Bob** is 27 years, shares an apartment with a friend, and is at his finishing stages of acquiring a university master degree. He is the casual kind gamer that likes to relax in his spare time during the weekdays by playing games and watching video clips on his iPad and occasionally plays online PC games on his laptop with friends. The weekends are usually reserved for socializing with friends.

**Alice** is a 18 years old girl, currently studying at highschool. Her favorite subject at school is mathematics. She is one of a handful of dedicated female
4.2 User Needs

gamers. She usually plays on her desktop PC when she isn’t interrupted by homework, and likes to play MMOG types of games, but also games such as Minecraft. Alice has a younger brother about her own age. Her family recently bought an Android tablet for entertainment and relaxation purposes during spare time at weekdays and weekends. Her brother whom she occasionally plays games with has just discovered SV on Google Play and has also installed the game on PC.

James is 35 old guy, works in a full time position as a physiologist and is in a relationship with a girl that is a couple of years younger. James were a hardcore gamer when he was younger but today he doesn’t have much time to play as he would like to. Luckily his girlfriend occupationally also play video games, sometimes with together with James so unlike some of his friends, his girlfriend don’t mind watching him play games. On average he plays about 1-2 hours per day, usually before bedtime.

They have both a high interest in movies and when they have friends visiting in the weekend when they haven’t work they sometimes play a bit of multiplayer console games all together. All in all James and his girlfriend spend quite a bit on digital media each month. They recently brought a desktop PC for gaming and the recent generation iPad.

4.2.4 Scenarios

Now that we have defined who the users are and the typical context of their gaming consumption, it is possible to present scenarios that describe how these users would use SV.

4.2.4.1 Normal Weekday

James is heading home after having finished work for today. His girlfriend has arrived earlier from work also and is already started cooking dinner. When James arrives he attempts to help her in the kitchen but he is told to find something else to do until dinner is ready. With nothing else to do, James picks up his tablet device and lies down on the couch to kill the time. As he unlocks the screen, he notices a system notification, and immediately recognises the icon to be the SV platform. Without thinking too much about it he automatically taps the notification which opens the SV app and greets James with the frontscreen. As he currently doesn’t feel like joining the game a friend of his is currently is
Figure 4.2: Shows the possible set of user’s interaction with the SV platform playing, he clicks on the recent world he have been building on and dives into the voxel world. 20 minutes later he is suddenly disturbed by some voice. He remembers that he is actually quite hungry and joins girlfriend for dinner who were calling for him.

4.2.5 Use cases

This section will now summarize all the features and user needs that have been elicited through use cases. A basic overview can be seen by Figure 4.2 which describes all use cases that is in the boundary of the system (the SV platform).

Note that the actor labelled User is not connected to the use cases in the diagram.
as the diagram would become unreadable. This actor should be connected to all use cases, except the Reward with SV points use case. The SV platform is the actor itself for that use case.

Even though it is not in compliance with UML notation, the use cases of Figure 4.2 are grouped together into related entities. In terms of SV as a voxel editing platform, the content creation group is the primary set of use cases. The Discover content use cases are related to letting the user be able to filter in the created content in order to make sense of large amounts of content. Play only contains two use cases and the main reason is that these are more related to the gameplay aspect of the platform. Finally the profile contains use cases for persuading the user to engage in the system.

Notice that there is none of the use cases of Figure 4.2 that include features for chatting or messaging with friends. There are three reasons for leaving out this kind of functionality where people can text each other:

1. Typing on mobile devices is generally an error-prone and tedious task and would thus lead to a bad user experience.

2. Similar games on the market don’t have this kind of functionality which would suggest that the corresponding user need is not existing or has a low priority.

3. And finally in the case of two or more users that needs to communicate there are numerous other competitive services that is able to fulfil this kind of need such as TeamSpeak, Skype, Google Hangout, etc. This of course is not relevant for people playing together that are physically close enough for normal communication.

If it should be discovered that this is a wrong decision, alternative solutions could be to consider communication through a fixed number of prefabricated options that covers the types of emotions that the user is likely in need to express. Solutions could be similes and/or avatars expressing through body language (including chants or dances). Communicating with other users through the user’s avatar has the benefit of lowering the social anxiety barrier, which allows people to express their emotions with less filtering.
4.3 Multiplatform Implications

Users perceive cross-platform support as less of a value proposition and more as a common feature. One of the great advantages of using Unity3D as a development platform is precisely that it leverages the work of dealing with each platform independently by offering a software layer between native Android, iOS, etc. operating systems and the layer in which the applications (including our game) reside. But with all such middleware software, it is not a silver bullet, since extra awareness to usability issues (and UX) must be given now that a single application may be deployed to several OSs at once. This naturally puts additional constraints on the designing of the interface unless each platform is handled independently. For a small indie team that might not be feasible. In any case compromises must be taken.

To illustrate one possible usability issue, reading Google’s design guidelines for Android raises the following criteria for consistent navigation in Android apps:

App supports standard system Back button navigation and does not make use of any custom, on-screen "Back button" prompts. ... All dialogs are dismissable using the Back button. ... Pressing the Home button at any point navigates to the Home screen of the device

Since users on the Android platform would be familiar with this kind of interaction and iOS users are used to back-buttons on the upper left corner, this is an example of a usability issue. Despite their differences they also share many of the same design patterns so a reasonable UX is still feasible with a single shared design.

The most important factor that justifies a single design for both OSs, is the fact that the application is going to use a low level UI, meaning that there is full control for each pixel on the screen. This is a natural consequence of using Unit3D as development platform. In contrast high level UI is standardized widgets that implement a certain design pattern, which may be implemented different across two OSs.

4.4 Voxel Editors

This section will now analyse the voxel editors in more details.

4.4 Voxel Editors

4.4.1 Editor Types

We have seen how users in other games can create content in different levels of detail and control through the editors of the games. To recite, Worms offers an editor for procedural world generator, while GameGlobe offers quick manipulation of smaller part of the world at a time and finally, Minecraft offers only an editor which enables individual voxel manipulation.

In a voxel universe it would make sense to offer different kind of granularity for creating content, meaning that it make sense to let the user be able to quickly generate a large world with very little work and in the other end also be able to control every detail of this world voxel by voxel. In addition, an intermediate editor could be defined which lies in the middle on the "voxel control scale" which offers the user the ability to control a group of individual voxels and then create multiple instances (clones) of these. The benefit for such an editor would be the ability to shape a specific part of a generated landscape. This leads us to define at three kinds of voxel editors:

- A procedural level editor, which gives little control for each individual voxel.
- An in-game level editor, giving complete control of each voxel.
- and a prefab Level editor that controls voxels by groups.

Note that even though we distinguish between three kinds of editors, it is not necessary noticed by the end user as separate editors. The decision for whether each of these should be designed as tools for their own or even whether the end user actually have a need for three kinds of editors, is basically a design question which will be presented in Chapter 5.

4.4.2 Editor Learnability

At this point we have identified the user need of being able to construct objects in different size scale, from adding individual blocks to generating a complete landscape with in just a few steps. One of the challenges that must be dealt with, when considering the design of such interfaces that support such interactions, is that the users have different levels of expertise/former experience with a tablet touch device and different levels of creativity and spatial capability to imagine constructions before building it. This naturally leads to a challenge on
presenting the user with the tools for building, matching the complexity of the user’s capabilities.

One way to understand how to progress from this realization is to design the interface in a way that works very similar to how many games attempt to teach its game mechanics to its users/players: In incremental steps by being confronted with harder and harder puzzles in each level. If the required bar of expertise is too high, the user will not be able to engage with the system in a constructive way that rewards and probably reject using SV while offering a too simplistic tool would on the other end quickly become too boring for the user if it doesn’t offer enough power to manipulate and build constructions of his taste.

Different approaches to solve this problem could be through introductory tutorials (text and/or video clips) or through an integrated point system (levelling system). Requiring the user to see and learn from tutorials requires greater care since this solution assumes that the user is motivated enough for spending the time to learn a new behaviour. It is safe to assume that most users are inherently lazy [Wei11] p. 132. The latter approach integrates the learning process of using the building tool through the interface in a non-interruptive way (if design in a good way), by distributing the learning over longer timespan.

A very important reason for letting the user have the tools to create more detailed and complex structures as they progress is that depending on their existing spatial intelligence he will experience this as a learning progress that allows the user to improve their creative capabilities. If the tools allow the user to learn by constructing objects, the user’s brain will reward itself for succeeding in this behaviour and will strengthen the association of that behaviour with the positive reward felt at that moment. In the best case, this will make the user come back for more.

StackOverflow is an example of such a levelling system as mentioned earlier, where the user is able to unlock functionality as you progress and prove to the system (and other users) that you are capable and/or motivated enough to accumulative points. The levelling system serves several purposes, it is not only used to control the learnability of the system, but also to motivate the user into a certain behaviour as they see fit. This is achieved by the way the system rewards you with points (e.g. answering good questions), but also attempting to motivate the user to become more engaged through offering "badges" (intrinsic reinforcement). If you for example have visited the website once a day in 100 consecutive days you achieve a certain badge for that achievement. GameGlobe has a similar persuasive feature where you are rewarded with 100 gold coins for each new day you login to the system.

A common persuasive technique to motivate users is by showing a progress
Figure 4.3: LinkedIn’s way of persuading behaviour

To summarize, offering creator control when creating worlds and editing your avatar will be features that are unlockable as the user proves himself capable and motivated by accumulating the required points in a point system to unlock that feature. Additional motivation for reaching the goal of unlocking an "unlockable" could be through the use of status indicators such as exemplified by Figure 4.3.

4.4.3 Editor Discoverability

In order to make the editors simple to use the user must be able to explore all commands/features (all possible interactions) through the GUI. This is the one big advantage in GUI’s compared to command line interfaces [NN10], since the user do not have to remember each exact command. The problem now with touch-based devices is that the interactions with the GUI are through gestures. Unless the user is equipped with a manual/tutorial, the user will have to guess which gestures to use in the different interfaces. With this reasoning in mind it would make sense to minimise complex gestures when possible, unless there is some kind of indication for the user, that a certain type of gesture is possible (e.g. a swipe) or that the user has a clear perceived affordance.

What this means in terms of the editors in the system (the avatar editor, the
procedural level editors and the in-game level editors) is, that the first available editors, when the user is new to SV or have low points and thus have not unlocked the advanced editors, must require minimum gestural interaction. On the other hand when the user has proven his engagement in the system (by accumulating more points) the user will be able to unlock the advanced editors which naturally offers more detailed control of the content creation, through more complex gesture interaction. This gives the design two advantages:

1. New users do not meet a steep learning barrier when they try the system

2. Since we know that an unlocked editor means an engaged user we may assume that this user is more motivated to use some more time on learning how to interact with the system and in return of that time investment, the user is provided with more advanced tools (i.e. the editors) to accelerate their creativity.

Giving motivated users better tools is an example of "putting hot triggers in the path of motivated people" - B.J. Fogg.
This chapter will now present suggestions on solutions for how such the SV platform could be designed that solves the needs and challenges discussed in Chapter 4 and the analysis of similar games in Chapter 3.

The challenge in the design phase is to find the optimal (or close to optimal) solution for a given problem and since there essentially exist endless combinations of designs that all solve the same problem there is a need for a solution strategy that must be followed in order to progress. The task of designing may thus be generalized to a search problem with the goal of finding the state that satisfies the user needs with the best user experience. The state-space is in this case different design solutions, and the solution strategy to reach the goal is to use knowledge of previous solutions (related games), well-known design principles (heuristics) and usability evaluation (user feedback) to progress in the search from less desired states/prototypes to the more optimal. This process is the essence of what we know as parallel and iterative design process [Nie11].

Figure 5.1 illustrates the individual iterative steps, from conceptualization to a final tablet prototype running on an Android device. The first part of the prototype design consists of the proposal of three different low-fidelity horizontal paper prototypes. Based on the heuristic evaluations conducted on these prototypes, the prototype with the best result has been picked and merged with the good ideas of the remaining concepts. From this, a new computer-based hori-
Figure 5.1: Graph illustrating the design process. pX, where X is an integer, is used as a unique identifier for each prototype.

Horizontal prototype have been build using Axure\(^1\). Usability evaluations conducted on this prototype have been used to further improve the design, user needs and the persuasive elements. Finally a native Android implementation based on the findings of the computer-based prototype has been developed with the purpose of being able to more accurately evaluate the parts of the application where voxel editing is present.

5.1 Conceptual Paper Prototypes

As mentioned earlier only Minecraft has a tablet version of their game, so even with the analysis of user needs and the kind of behaviour the platform is targeting, there is relatively little known in terms of how these features might be designed on a tablet device. In other words it is still a "greenfield" area in terms of design. Therefore it is important to attempt several different suggestions first and then compare the positive and negative parts of the solutions. The concept prototypes serve as a way to experiment in the early stages of development.

Three full horizontal paper prototypes have been created in a one-to-one scale to the dimensions of the screen real estate of a Nexus 7 (1. generation tablet), 15.4 cm $\times$ 9.5 cm in width and height respectively, assuming that the tablet is in landscape mode.

\(^1\)https://www.axure.com/
5.1 Conceptual Paper Prototypes

The three sets of original scanned image files is accessible at:

https://www.dropbox.com/s/7bbt3oqmuhw91hf/paper_prototypes_1-3.zip

The following sections will only use a subset of these to present the most important decisions on content and navigation for how they solve the problems in the concept prototypes. Note that the interfaces shown in this chapter is not necessarily a correct representation of the dimensions since the figures have been scaled to fit the page and that each image of an interface has been cut to the relevant part. Again, it is referred to the original scans as they are true with respect to dimensions and scale.

5.1.1 First Paper Prototype

The initial screen, as shown by Figure 5.2, that the user is presented with in the first concept, simply consist of two buttons that represents the main use cases of SV. The user may either create or play content. The advantage of this design is that the functionality of SV is evident from the first screen, but at the cost of having the user to make a (conscious) decision on what to do here and the fact of having one additional interaction step.

Clicking on the create button takes the user to the screen seen in Figure 5.3.

Common for the subsequent screens for both create and play "branches" of interfaces is the use of a top bar that consists of an optional back-button, a title of the current visible screen and buttons for notifications, messages and a shortcut to the user’s profile screen.

Figure 5.4 shows an example of the notification and message icons expanded. The notifications contain any events that the user’s friends are involved in, while the messages is for specific events directed to the user himself such as recommendation of a map, an invitation for a game or messages from the SV platform. When new messages are received the two icons change in color and a small integer is shown on top to indicate new notifications or messages. This is similar to how Facebook works. In addition since these two icons are placed away from the main content of the center, any changes in their icons is more likely to be noticed since it would be in the peripheral vision of the user [Wei11] p. 6.

The create screen on Figure 5.3 basically offers three ways for the user to create
Figure 5.2: p1: The initial screen displayed when opening the app

Figure 5.3: p1: Shows the possible actions for creating content in SV
5.1 Conceptual Paper Prototypes

Figure 5.4: This snippet illustrates the idea of the notification and message icons placed in the top bar of the app.

Figure 5.5: A long press on a content tile brings a similar context menu:

- Recommend to friends
- Invite friends to play
- Unpublish world

content: A shortcut to generate a world with random parameters, a custom world, and finally for editing the user's avatar. To the right, a panel is visible so that the user can continue the creation process with the most recent generated worlds. This is a panel for quick access to the user's content.

Common for the concepts are the decision of using tiles for content. The tiles are useful for visual content and, in this case, a snapshot of the world is used both for increased enjoyment when browsing, but also as a quick way for the user to identify a certain world. A single tap on a tile will take the user inside the world itself while a long press brings up a context menu as seen by Figure 5.5.

The context menu lets the user invite friends to collaborate, recommend the world to others, or publish/unpublish the world. Using context menus give the advantage of hiding away less frequently used actions, which reduces the risk of overloading the screen with information. On the other hand, the user must first discover that the long press works on these tiles. Considering that collaboration in the creation process is part of how the system attempts to increase the target behaviour, hiding the "invite friend" action might work against that goal, which is a clear disadvantage of this solution.
Figure 5.6: Shows the customizable options the user have in the standard procedural generator editor.

Figure 5.6 shows the screen of the custom world editor, which is basically an interface for a procedural generator of worlds. In this prototype the world editor contains both a real-time preview of how the world will look if the user presses the generate button. On the left side of the interface, the user is then able to specify parameters to control the output of the generator. The controls are separated into three categories, as seen by the tabs for controlling landscape contour, colors, and "prefabs", where "prefabs" are SV's label for prefabricated groups of voxels that similar to "props" in GameGlobe.

Figure 5.7 shows the remaining two tabs. Each of the tabs lets the user choose between four different colors and prefabs that will be used in the world. For prefabs, the user can browse his gallery of favorite prefabs for quick access, while a full browsing of prefabs is hidden away.

The standard world editor, as seen by Figure 5.6, offers similar features as what is seen in similar games but leaves out many of the specific decision for the user concerning aspects such as the chosen colors are used, where the prefabs are spread out in the world and how the contour of the terrain is shaped. This enables the user to quickly create the basis for a unique world, which may then be altered later on when generated.

As an alternative, the user will be able to unlock a more advanced editor when...
Figure 5.7: Snippets of the two other remaining tabs of the procedural generator editor

Figure 5.8: The advanced world editor only available for the user when enough points are collected for it to unlock
he has collected enough points in the system, where this editor instead will enable the user to decide himself the things that the standard editor otherwise did. Figure 5.8 shows the interface for the advanced editor. This editor focuses on giving a more immersive experience as it takes much of the screen real estate. The user is therefore more encouraged to interact with the 3D preview. Similar to the standard editor, this editor offers three tools to manipulate the world, which are changing landscape contour, colouring, and adding prefabs.

In contrast to the standard editor, the advanced editor is based on gestural interaction for editing. It is the intention that the user is supposed to interact with this interface by pinch-to-zoom and using two fingers to pan around in the plane of the terrain. Any action done with one finger depends on the type of tool selected:

- The color spray paints the nearby voxels at the position of the finger and may be moved around the terrain while drawing

- Tapping with the prefab tool selected adds a single prefab at the position currently selected

- A tap-and-hold followed by move up or down raises or lowers the contour of the terrain on the position of the finger when the terrain manipulator tool is selected.

If the user presses the avatar customization button shown on Figure 5.3, the interface seen in Figure 5.10 is shown. Here the user is able to change the user’s avatar as he sees fit by browsing through lists of prefabricated body-part. A real-time interactive 3D preview of the avatar enables the user to see the end result directly.

The final interface of interest in concept 1 is the play screen which appears if the user chooses "Play other users’ content" use case when the app starts. Figure 5.9 shows the play screen, where the user is able to scroll through a list of published world tiles. The list is filtered by the taps on the top, which makes it possible for the user to look through recently released content, least and most popular content, and finally the user’s own favorites. Concept 1 attempts to suggest worlds that the user may find interesting to play by showing some of the content his friends have been playing.
5.1 Conceptual Paper Prototypes

Figure 5.9: p1: This Screen is where games can be found. Tiles are used for content and filtering through tabs

Figure 5.10: p1: A user may edit his character by selecting between different body parts and see the result immediately
5.1.2 Second Paper Prototype

In this prototype, the initial extra step has been cut out, and instead there have been used tabs as a way to navigate between the interfaces. Figure 5.11 shows how the first screen of the interface looks like. Cutting off the initial screen means that the designer must choose between the two main use cases of the platform. In this prototype, the choice has been to show the create world tab, when the application is opened, since this will increase the chance that the target behaviour is achieved.

Notice that the top bar now always shows the user's current status in the platform in terms of rewarded points and unlocked editors. Similar to StackOverflow this solution acts as a continues reminder for the kind of behaviour that the platform awards the user for. Accumulated points are called "prestige" in SV, as a synonym term to StackOverflow's "reputation", so it functions as a motivational element for social acceptance/rejection through the kind of association there usually is with the word. Clicking on prestige presents the user for the list of actions he may take to get rewarded with additional points. The list can be seen in Figure 5.12. Notice that these actions are related to social and editing activities in SV. Apart from using the top bar navigation, the interface seen in Figure 5.12 also lets the user browse through a list of unlockable items. These unlockables can be seen in Figure 5.13.
5.1 Conceptual Paper Prototypes

Figure 5.12: p2: Shows an example of the type of actions that would result in an intrinsic reward through prestige points

SV attempts to achieve the target behaviour by motivating the user to perform social and editing activities with points, which in turns unlocks additional capabilities for editing, thus creating a "reward loop".

The world generator, as seen in Figure 5.11, is even simpler than the interface presented earlier in concept 1 Figure 5.6. The possibility to select prefabs to be added randomly in the world is removed, as it may make more sense for the user to be able to control that in an in-game environment instead. In addition, the control for the contour of the terrain is also simplified from two sliders to a single slider that determines the amount of mountains, and height of mountains. Finally the 3D preview of the world is hidden away to the following step in generating a world. The second step can be seen in Figure 5.14. As a whole, these changes reduce the amount of decisions that the user has to make and declutter the screen that otherwise was seen in concept 1. Finally the benefit of having a separate interface for the 3D preview is that the experience when interacting with it is more powerful as it allows for more details on the screen.

Instead of the solution seen in concept 1 where the top panel had an icon for notifications and messages as well, this prototype solves the problem by having a panel on the right combining the two. The panel can be swiped or tapped, which reveals an activity feed as seen by Figure 5.15. This panel is always visible in the prototype so that the user in any time is able to respond to system messages.
Figure 5.13: p2: Unlocked achievements can be browsed and locked items may be read about and anticipated

Figure 5.14: p2: A separate interface for a preview of selected parameters when generating a world
5.1 Conceptual Paper Prototypes

Figure 5.15: p2: An alternative way to display messages and notifications in a partially visible drawer, with the intent of getting attention or friend activities. The panel is partially visible so that any new notifications or messages are easier noticed since the content changes accordantly with a new item on the top of the feed. The idea is that the new item should blink for a short while so the change cannot be missed.

The drawer seen on Figure 5.15 is inspired by Soundrop.

The consequence of using tabs is the less amount of screen real estate available for the actual content of the interface. This is of course less of a problem for tablet devices compared to mobile devices, but in this case since the application is viewed in landscape mode, it may give the feeling of a disproportion between height and width. In order to utilize the screen real estate in the width, the second tab where the user is able to browse for content to play, is made for horizontal scrolling of content instead of the typical vertical scrolling. Figure 5.16 shows how the resulting interface looks like.

As it can be seen, the problem of using tabs as a way to structure the content and navigation is evident as soon as nested tabs are needed. Much of the height is lost with having the top bar and two additional rows of tabs, since there also must be some padding between each row of tabs in order to avoid hitting the wrong tabs when the user attempts to navigate. Notice that concept 2 have

---

2Sounddrop is a third party app for Spotify, which is a music service that makes listing and creating music playlists a collaborative social experience. It offers a collection of genre-specific live chatrooms. When joining a chatroom, a user is able to add additional tracks to the room's playlist and people are then able to upvote each track so that tracks with the highest amount of votes get played first.
Figure 5.16: p2: Browsing other's content is restructured to horizontal scrolling high amount of chrome in height

made actions visible on the content tiles directly, in contrast to concept 1 that used a context menu. These icons fits naturally along with the button to mark content as favorite, i.e. "love" it.

Figure 5.17 shows the controls available when the user enters a world, i.e. the in-game environment.

The top bar of the in-game environment is a two-tap panel that holds quick access for the recent used colors and prefabs. The ideas are that the current selected tab determines the construction "mode" the user is in. In both of these two modes a button is available for selecting other colors or prefabs.

Tapping on any of the recently used prefabs, results in a prefab added to the landscape on a point of the landscape that the in-game character is currently looking at. The idea is then that the user will then be able to scale, translate and rotate the prefab by using gestures directly on the prefab itself:

- 2-fingers up or down panning for vertical translation in the direction of the landscape's normal vector.
- 2-fingers left or right panning for rotating the prefab around the same normal vector.
Figure 5.17: p2: The interface of an in-game environment when a user enters a world

- Pinch-to-zoom for scaling
- 1-finger panning in any direction for translating the prefab around the plane of the landscape.

In block mode the user may add a voxel at the point where the in-game character is looking at by a single tap on the screen or remove a voxel by long pressing.

The joystick in the in-game interface should be visible to start with and then slowly fade out so that the user understands how to navigate the avatar. In contrast to the arrow keys seen in Minecraft, the joystick is activated the user presses on the left side of the screen. In that way the user doesn’t need to know if he is hitting the correct spot on the screen in order to invoke a movement of the avatar. The only exception is when a user is adding a prefab, which could invoke the interaction with that prefab instead of moving the avatar if the user presses on top of a prefab that is being added.

In the in-game interface the user is able to jump with his character by using a dedicated jump button on the right side of the screen. The problem noticed in other similar games on tablets such as Minecraft is that jumping is on the same side as the movement control (whether it is a joystick or arrow-keys) which makes it harder for the user to run around while jumping. On the right side,
we have the swipe gesture to orient the character’s view in the desired direction and angle, which has lower priority than keeping the character moving during a jump.

### 5.1.3 Third Paper Prototype

The third concept supports vertical scrolling better since the top chrome has been eliminated. Again the create world use case is in focus which is evident by the initial screen seen in Figure 5.18. The third concept also borrows the idea from concept 2 with partially showing an activity panel on the right side, which contains recent activity from friends and SV messages.

The procedural editor seen in Figure 5.18 presents 3 steps for creating a landscape. It enables many different combinations of prefabs, colors and types of landscapes but the user have little control of the detail of these components themselves. The user is first able to alter the shape of the chosen world when the map is generated, but this is not a problem in this concept since the user already would have an idea of the result since the landscape are samples of the outcome.

Instead of using tabs or having an additional step before the user can create
5.1 Conceptual Paper Prototypes

Figure 5.19: p3: Shows the main navigational options of the app.

or discover content as in concept 1, these are hidden away to a menu on the left side of the application. Figure 5.19 shows For consistency reasons the first of these buttons on the slider is "Create", since this is the initial screen shown when the user opens the app. Using this kind of menu is often seen in both iOS and Android for containing buttons for navigating between different parts of an app.

Figure 5.20 shows the interface for browsing the playable worlds. Content is presented the same way using tiles with images throughout the UI for consistency and actions belonging to each of these maps (i.e. share, invite and love) are grouped on the side of each image with enough spacing to other content so that the user is able to create a relation between the actions to the map with little cognitive load. The new type of "tabs" that is used in this prototype (Figure 5.20 is an example of that), free up more of the screen real estate but there is the danger that the user might not be familiar with how to interact with the UI element because of the user may not have a clear perceived affordance. For Android users this UI element may be familiar as it is a dropdown menu used in Android v4.0 and higher.

The avatar editor in concept 3 is made even more simpler compared to concept 1, by not having a 3D preview of the full avatar but instead use three slider widget to hold the different combinations of body parts that the user can choose between. Figure 5.21 shows this interface. The user can then horizontally swipe through the different options and the currently selected body parts are...
**Figure 5.20:** p3: Content discovery with large preview tiles with quick actions to share, favor, or invite friends to play

**Figure 5.21:** p3: The default avatar editor available
the highlighted parts in the center. Notice that some of these body parts could require a certain level of accumulated prestige to unlock, as a motivator through anticipation.

5.1.4 Choosing a Solution

The presented prototypes have attempted to solve the design challenges of SV in different ways. The preferred attempt to design SV would be a solution that is both shallow with regard to the possible navigation depth and is able to comply with usability heuristics in the best way in order to strive for a useful solution. A shallow navigation structure is preferred as not to disorientate the user [NB12] p. 26. By useful, it is mean a solution that both provides the features that users need combined with how easy and pleasant these features may be used [Nie12b].

When comparing the navigational depth of the concepts, it is seen that concept 2 and 3 are the shallowest. Concept 2 even requires less interaction for the user to navigate between content discovery and content creation with the use of tabs. However the solution shown in concept 3 is preferred for its use of a navigation menu on the left and dropdown menus instead of tabs as these requires much less of the screen real estate. Especially the use of nested tabs in concept 2 is seen as a problem with regard to the amount of chrome it takes.

The heuristic test conducted on these designs also shows that the solution presented in concept 3 has a better average score on the 13 heuristics used. Therefore concept 3 is the basis for the design of the computer-based prototype that will be presented in this chapter. More details on the heuristic evaluation is presented in Chapter 7, and additional information on the heuristics used can be found at Appendix A.

5.2 Computer-based Prototype

The prototype can be viewed online on a web browser at:

http://share.axure.com/KGPULW
It is recommend to use Firefox if viewed on PC, or using a full screen browser app if viewed on a Nexus 7 tablet. The tablet dimensions have been designed to fit the Nexus 7.

Compared to the paper-based prototypes, there has been a shift in the choice of main scenario, from content creation to the discovery and play of content. In the computer-based prototype this is evident from the fact that the first screen to meet the user is "discover and play". The design decision may be seen as a setback in terms of achieving the target behaviour of letting the user create content. However, from feedback received, the subjects independently reported that they found it strange not to offer games to play as the first thing when the app is opened.

This may result in a slight setback in relation to the diversity of the content. As Alex Norton also warns about in his blog post [Nor13], users might be less open to try out new ideas if they already have already seen some of the content others have created. On the other hand, depending on the gameplay, the average user might just want to be entertained with playing created content, while creating it would be suitable for the more curious type of user.

5.2.1 Sequential Flow

This section will now document the main flow of a content creation scenario.

When the application is open the interface seen by Figure 5.22 is shown for the user. Notice that the design mainly is inspired by the solution of concept 3.

By clicking on the menu button, the panel will slide out and reveal, the interface seen in Figure 5.23. Compared to concept 3, the profile button is not located in the menu as this is always accessible through the top bar of the application. Instead the user now has the capability to manage and browse his friends and other users, which was hidden away in the profile interface in earlier concepts.

In the interface seen by Figure 5.24 the user can now choose the overall contour of the landscape along with colors for land and water blocks and prefabs that should be distributed. This interface is the same as seen in concept 3.

When the user finally generates the world, he is taken into the in-game environment as seen by Figure 5.25. Here the user is able to move around in the

[https://play.google.com/store/apps/details?id=it.automated.android.browser.kiosk]
Figure 5.22: p4: The initial play screen

Figure 5.23: p4: The left menu expanded
world, add individual blocks of different colors, add prefabs, and create a prefab by marking a group of voxels.

5.3 Tablet Prototype

The .apk file for running the application on Android device can be found here:

[https://www.dropbox.com/s/d2zrbhsbjahj8ib/game_mobile.apk](https://www.dropbox.com/s/d2zrbhsbjahj8ib/game_mobile.apk)

Note that this application is build for Android v4.1 devices and higher!

And an .exe version of the same application may be found here:


Note that although it is possible to run and play on a PC, the interface is developed for touch devices, so therefore it is tricky to control using mouse as input.

The idea with implementing a native app was to demonstrate a thin vertical slice of the main scenario, which was creating a world. The prototype seen by
Figure 5.25: p4: The in-game environment

Figure 5.26: p5: A snapshot of the SV prototype running on a Nexus 7 device
Figure 5.26 shows the result of the prototype. The user be able to control a first-person avatar on a flat plane build from cube-shaped polygons, by using a joystick on the left side of the screen and directing the orientation of the avatar with swipes on the right side of the screen. In addition the user is able to add new blocks of three different colors on the plane, by tapping on the screen. The user is able to choose between the three colors by clicking on any of the three grey buttons placed on the top of the screen. Finally a red cursor have been implemented that indicates were a user will be adding a block if the screen is tapped. This is seen in the figure by the flat red surface placed in on the cube that lies in the center of the viewing direction. Only when the cursor is visible, will the user be able to add any blocks.
This chapter will discuss the design of a prototyping framework that have been developed and a simple proof of concept tablet implementation for an in-game editing environment.

For reference on the scripts presented in this chapter, the whole Unity project is available for download here:

https://www.dropbox.com/s/46nybix6moifrz/UnityProject.zip

6.1 Datastructure for a Prototyping Framework

Since prototyping is a compromise between time spent on development time and achieving multiple iterations, there has been no emphasis on a solid design of software architecture for the prototyping itself. On the other hand, a framework that supports multiple prototype implementations has been designed. Figure 6.1 shows a UML class diagram of the framework’s architecture.

As seen by the diagram, a prototype implementation basically comprises of implementing the IGestureStrategy and IUIStrategy interfaces. These two
Figure 6.1: UML class diagram of a prototyping framework
interfaces are each part of their own strategy pattern and are used to define interfaces for concrete implementations of a prototype that needs to handle gestures from the user and setting up the UI.

A prototype in this framework is thus defined by a double strategy pattern with a VoxelEditor class as the context of these two strategy patterns. A strategy pattern is basically a dynamic way of defining behaviour during runtime, in contrast to the strict structure of inheritance.

In order to implement a prototype, one must subclass VoxelEditor, and during initialization of this subclass, it would need concrete implementations of the IGestureStrategy and IUIStrategy interfaces as parameters. The benefit of this architecture is achieved in the dynamic nature of the strategy pattern, since this allows for multiple different implementations of behaviour for handling gestures and UI that may be swapped out as needed. For each new iteration, the procedure is simply to create a new pair of IGestureStrategy and IUIStrategy implementations and pass instances of these instead of any previous implementation. The disadvantage of this design is the risk of having duplicate code if a new iteration only has small differences. On the other hand, the previous implementations are useful to have as documentation of how the iterations has developed over time and in that sense, the choice of compromise justifies.

6.2 Software Components

The implementation of the native android application relies on two helper libraries on top of the Unity3D engine middleware software. Figure 6.2 illustrates in which layers these components reside, including the implementation itself and how they interact with Unity3D so it can be deployed on Android.

The two helper libraries are TouchKit and UIToolkit respectively. TouchKit offers an extensible abstraction implementation for handling multi-touch gestures, while UIToolkit offers a rendering efficient implementation of standard UI widgets and components that is common for mobile applications.

While Unity3D of course has their own API to access input and UI, these are not developed for mobile applications which are limited in computational power. Disregarding its multiplatform capability, Unity3D’s main advantage is in handling the mathematical calculations such as raycasting, physics and rendering of 3D space.

1 https://github.com/prime31/TouchKit
2 https://github.com/prime31/UIToolkit
The developed application is designed for mobile deployment, in particular for Android, but it is possible to run it on PC as well. The following sections will now present the prototype component in greater details.

6.3 Using the Prototyping Framework

The section will present how the framework have been used to implement design of the in-game environment as described earlier in Chapter 5 Section 5.3. The description of the implementation is based on the class diagram seen in Figure 6.3.

As seen by the figure, the implementation of the in-game environment consist of two concrete classes of the IStrategyGesture and IStrategy interfaces, called InGamePrototypeGestures_1v1 and InGamePrototypeUI_1v1, respectively. In addition a concrete class of the VoxelEditor abstract class, called VoxelEditorInGame is needed.

InGamePrototypeUI_1v1’s main responsibility is to setup the UI elements and handle the actions that input related to these UI elements must perform, such as implementing a joystick, and buttons to change the selected block color.
Figure 6.3: Compressed UML class diagram of the SV native implementation
In order to implement this responsibly the \texttt{InGamePrototypeUI\_1v1} uses the \texttt{UIToolkit} library.

Similarly, the \texttt{InGamePrototypeGestures\_1v1} class uses functionality from the \texttt{TouchKit} library to implement the responsibility of recognizing the user’s gestures and delegating that event so the corresponding action is performed. \texttt{InGamePrototypeGestures\_1v1} listens for a tap gestures to invoke the adding of a voxel in the world, and listens for swipes in any direction of the right half of the screen in order to orient the direction that the in-game camera looks at.

The in-game environment prototype needs to be able to move the avatar of the playing user, and that is the responsibility of the \texttt{InGameCharacterMovement\_1v1} class. This class uses an in-build Unity component, called \texttt{CharacterController}, to affect a physical object in the 3D world with gravitational forces and movement. Note that the class diagram does not show otherwise important components such as the camera object. This object is attached to the physical object of the avatar so that affecting the avatar with movement and forces, also automatically affect the translation of the camera in 3D space. \texttt{InGameCharacterMovement\_1v1} is associated with to the \texttt{InGamePrototypeUI\_1v1} class since it needs to know if the user have used the joystick since last frame in order to move the avatar’s position in the world. For similar reason \texttt{InGameCharacterMovement\_1v1} is associated with the \texttt{InGamePrototypeGestures\_1v1} so that swipes results in moving the camera’s direction correctly.

The \texttt{BlockContainer} class is simply a singleton container that manages adding and removing of blocks present in the in-game world. When the application is started, the \texttt{BlockContainer} creates a flat landscape of cubes. Whenever a request for adding a block is received, it uses the in-build raycasting of Unity to calculate which block on the screen the new block must be added on.

Finally \texttt{ApplicationStateController} maintains a list of \texttt{VoxeEditor} instances which it may switch between at runtime to switch the interface of the application to a completely other editor. This can be done by using it’s \texttt{SwitchEditor()} method.
There are basically four ways in which it is possible to evaluate a user interface [NM90]:

1. Formally, by some analysis technique
2. Automatically, i.e. a computerized procedure
3. Heuristically
4. Empirically

In this project the latter two approaches have been used in different stages of the iterative development process as they are both the most cost-effective. These will be described now.

### 7.1 Heuristic Evaluation of Concept Prototypes

Heuristic evaluation has been used early in the development process on the three design suggestions. It is recommended to have between 3-5 evaluators for each
heuristic evaluation in order to identify a satisfactory proportion of all existing usability issues. By using multiple evaluators, their aggregated findings increase due to their different perspectives. About 75% of the usability issues is expected to be found when using 5 evaluators \cite{Nie95b, NM90}.

Heuristic evaluation is an informal way of evaluating the usability of an interface and is based on the fact that an evaluator with domain knowledge/expertise is able to find usability issues in an interface simply through inspection.

The goal of the conducted heuristic evaluation was to find usability issues early in the process, with little resources.

Appendix A contains detailed information on the used heuristics, and the results.

### 7.1.1 Method

The Interactive Heuristic Evaluation Toolkit \cite{heu} have been used along with the 10 usability heuristics presented in \cite{Nie95a} to ensure that all relevant aspects of usability heuristic for tablet devices in the entertainment category has been considered during the test.

Heuristic evaluation on the three concepts was conducted with two evaluators, one of them being myself. The procedure for the evaluation was the same for us both, except since the other evaluator wasn’t familiar with the interface to start with, he was given the interfaces of a prototype before commencing with the evaluation itself. I asked the evaluator to describe what he was seeing on the interfaces in order to make sure he understood them and to answer any of his questions that he might have had in relation to any about an interface. In the process, after he was familiar with a prototype, I was not allowed to say anything in order not to bias the results. Optimally, he should have taken his own notes so I didn’t need to be present at all in order to be 100% that no kind of bias was possible as a result.

When he was familiar with a prototype, all papers for a prototype was laid out on the table and he would then inspect each of them for each of the heuristics. I was sitting with my laptop and taking any notes that he had for the individual interfaces and a heuristic would end with him giving a score between 1-3. 1 being poor/lacks support, 2 being Average/partial support and finally 3 for good/The app supports this heuristic fully. When a score was given for all heuristics for a prototype, that would conclude the evaluation and the process was then executed for the two remaining prototypes as well.
7.1.2 Findings

The total average score from both evaluators showed that concept 2 was the worst of the three with 1.77 points in a 1-3 scale, while the third concept scored best with an average score of 1.96.

The heuristic evaluations reviled many small issues and others more critical problems. The critical ones were mostly related to lack in "Help users recover from errors" and "Help and documentation".

The following highlights the some of the findings from both evaluators to each of the three concepts.

**Concept 1**

The labels of the tabs in the discover & play interface seems confusing, as the user might think that 0-25 and +500 numbers refer to top 25 and button 500 rated worlds instead. Also there are too many different tabs.

The avatar interface has a very confusing finish button. It doesn't make sense to use the back button as a finish button also.

The notification icon on the top bar may be too hard to press correctly.

It seems strange that the procedural world generator does start as a flat landscape in the 3D preview.

**Concept 2**

Again there is a problem with the labels of the tabs which may confuse.

The list of unlockables and prestige are too cluttered with their buttons. The user might accidentally press the wrong ones.

The word "cap" in the list of unlockables may be confusing for the user to understand. "Limit" may be a better word.

There is missing an indicator for the right activity panel if new notifications occur.
The prototype is more confusing to navigate, possibly because of the use of tabs.

Some interfaces are missing a back button.

Concept 3

It doesn’t make much sense that the create interface is shown first. The play screen is possibility more important for the user.

The jump button in the in-game interface is not clear.

The labels of the drop downs for the user’s own favorite content must be relabelled, to e.g. "My favorites".

There is a general lack of error prevention, and confirmation dialogues.

7.2 Empirical Evaluation of Computer-based Prototype

Appendix B contains full descriptions on the test subjects, the tasks used, transcript summaries, and the results.

The goal of the empirical evaluation is:

1. Identify usability issues.

2. Find out if users’ expectations are meet in terms of offering the tools that they need.

3. Assess the effectiveness the persuasive design.

In other words, the aim is to feedback in order to test the hypothesis - The project goal described in Section 1.2

Three participant where used in the empirical evaluation.
7.2 Empirical Evaluation of Computer-based Prototype

7.2.1 Method

For evaluating the computer-based prototype, the think-aloud method \[\text{Nie12a}\] was used, where the test participants were encouraged before the test itself began, to say whatever they came to think of and describe at all times what actions they were currently doing to solve a given task. Since the prototype itself is made with Axure, it was necessary to upload the application to AxShare, install a full screen kiosk app\[\text{1}\] on the tablet, and use that app to display the prototype. In this way it was possible to replicate the feeling of the app actually being installed on the device.

Before a test began, the test participant where told that the test was in no way about judging their capability to use the application. The objective of the test was only focused on judging how well the prototype itself was designed. In addition they were also told that they were free to withdraw from the test in any given time. Finally they were asked if it was OK to record the whole session with a video camera. The camera itself was placed between the participant and the tablet itself so that it was recording how the user’s hands were interacting with the screen. The tablet has put on a stand so the participant didn’t need to hold the device in the camera’s view. A snapshot from one of those recordings can be seen in Figure 7.1. This allowed for a more careful analysis of the recordings later on without having to note down the user’s actions during the test itself along with timestamps.

Finally parts of the test required the user to interact with a 3D world or other

---

\[\text{1} \text{https://play.google.com/store/apps/details?id=it.automated.android.browser.kiosk}\]
interactive 3D models, which was not possible to implement in a Axure prototype, so instead the user where specifically asked to tell how they imagined the interaction while showing it on the screen, even if the system didn’t respond to their actions. This was needed for the in-game play and create environment and the advanced world editor. They were only asked to do this after their first attempts of interaction in order not to bias their intuitive behaviour.

### 7.2.2 Findings

All participants manage to solve the given tasks, and all of them within two minutes. Compared to the context here tablets are used, time is not a critical factor as compared to a task on a mobile phone, so this is a relatively good result.

One critical issue was discovered which was evident with the advanced world editor. 2/3 participants didn’t understand how to use this new editor, because of the missing 3D preview of the complete world, missing icons for especially the lower/raiser tool, and missing small indicator of the currently selected color or prefab when the corresponding tools are used. These things together lead the participant to be confused about whether they still say the perspective from an in-game view and secondly all participants though that the color and prefab selectors on the bottom of the screen was objects that were draggable.

Surprisingly all three participant said at some point during the test, that they liked the greyish color of the interface, and that it was not something that would become annoying to look at over longer periods.

When the participants tried to plant a prefab in the in-game environment they all understood the controls on the left for positioning the prefab, but all confused the joystick controlling the character with being related to moving the prefab as well. This is considered a problem as the prefab controls are very close to the joystick controls and therefore the participants think they are related.

Two out of three attempted to use the physical back button at some point during the test, which resulted in the closing of the Axure prototype instead of going a step backwards, as they otherwise expected. This is not a problem with the design of the application itself, but an implication of using a kiosk browser app for displaying the prototype on a tablet. The kiosk app closes if the back button is pressed.

Participant 1 noticed the drawer but didn’t know what it was for and though it looked confusing so he didn’t want to press on it. This may not be a prob-
lem itself since the other two participants used it several times, but none of them noticed the red notification icons on the drawer, indicating new messages. This would be a problem for SV since much of the persuasive design relies on motivating messages of different kinds in the drawer.

Participant 2 was confused about whether the app would be able to save the building progress so he published the world, just to be sure. This may be related to the "Exit" label in the in-game menu that could be solved by e.g. renaming to "Exit & build later".
From the tests conducted in this project we only have a small indication through the feedback of the participants for whether the prototype delivers a satisfactory user experience. In order to be able to validate such a hypothesis, there should have been conducted user evaluations on an actual implementation of at least the "creation" use case in order to get a better understanding of the experience of interacting in the 3D environment.

8.1 Results

The following sections will discuss the results of this project.

8.1.1 Iterations

Unfortunately there computer-based prototype have not been evaluated more than once so that several iterations could be made. This is quite a problem as the intended development progress was iterative! On the other hand several paper prototypes has been created so that some degree of parallel prototyping have been achieved.
8.1.2 Evaluations

Overall the evaluations reviled much positive feedback, with only a few real problems. Many of the experienced problems are related to the limitations of prototyping using Axure, such as making a horizontal slider that snaps to position when choosing a landscape contour in the create editor, or the color chooser widgets that are not interactive.

The interfaces that required more complex gestures such as the advanced level editor and the in-game editor, the participants gave a mixed result on the degree of understanding how these are supposed to work. This was mainly visible in the advanced level editor where 2/3 participant though that the block sprayer and prefab sprayer tools was to be used with drag-and-drop instead of painting with the finger of the 3D world. When the users was in the in-game environment they all understood surprisingly well how it was supposed to work, even though the Axure prototype was very limited regarding the implemented interactivity. The only serious problem where all failed/was confused was how to use the prefab control panel that is given when a user attempts to add a prefab to the landscape. They all though that the joystick was related to moving the prefab in the scene. The likely reason for this behaviour is because they are placed close together on the interface and therefore the user thinks that their functionality is related. Finally the translation buttons (up and down) in the prefab controller panel was not clear for the participants. They didn’t know if it was up and down orthogonal to the landscape or along the plane of the landscape.

With regards to their feedback on if the application gave them the necessary tools their overall comments where positive, except participant 1 who expected to be able also to control physical elements such as gravity when creating worlds, and being able to give behaviour to living creatures and objects. He wanted a platform that supports more than a static environment.

Finally, when asking the users if they felt motivated to create content in the game the answer was not a clear yes. 2 of the participants understood the idea with points and unlockable where one of them directly said that they felt motivated by the fact of the unlockable objects. Whether they would actually perform the target behaviour when they try the app themselves in a natural environment (i.e. not in a test setup) cannot be concluded. At least not with the number of test participants used for the usability evaluation.

The feedback received was overall surprisingly good in the sense that most of the critical usability issues was due to the limits of Axure for imitating 3D worlds and complex gestures.
Using Wizard of Oz method in some way to imitate the parts where Axure fails could maybe have solved the issue of evaluating interactive 3D editors, but the real solution would have been to prototype these in a native app. This would give a much more realistic opportunity to assess/evaluate the voxel editors. A tablet implementation would make UX evaluation easier since the user would get immediate feedback from the device itself when using gestures, instead of simulating the 3D world using some kind of Wizard of Oz method.

8.1.3 Tablet Implementation

Unfortunately the current state of the Android application is very limited in functionality and doesn’t implement a vertical slice to demonstrate the sequence of steps for a chosen use case. The current state is merely a proof of concept that the developed prototyping framework, works as intended in addition to demonstrating how Unity3D and the helper libraries may be used to create prototypes that imitates a voxel world capable of recognizing gestural interactions and display of UI elements inside a 3D engine.

A satisfactory implementation of a native application should have been able to show how a user might navigate through the application, and for example choose generate a new world of his choice and finally be able to manipulate the world in an in-game environment. That kind of functionality would demonstrate a tool that fulfills the application flow for the "create world" use case.

8.2 Future Work

This section will discuss areas where the SV platform needs more work based on the current status.

8.2.1 Alternative Content Discoverability

An idea for an alternative way of presenting the content created by the users could be to have a separate world with entrances to the games of the users, with the goal of making the content discovery an exploitative experience itself. This feature is inspired by the way content is made available in platform games such as Super Mario, or the way the user has to navigate in a fictive skyscraper to get to different chatrooms in Netstationen (the website have been taken down).
80 Discussion

Figure 8.1: Super Mario is an example of alternative content discovery, here with a 2D world map

Figure 8.1 illustrates this idea (though it is a 2D world) with points of interest on the map that indicates new content to play.

8.2.2 Future Evaluation of Product

In smaller scale user evaluation, the front-facing camera on the tablet could be considered to be used to record the user’s facial expression in addition to recording how the user is physically interacting (i.e. screen touches) with the tablet. The facial expression of the test participant could then be used in an analysis to establish the emotional response to the user experience. In order to quantify the validity of this analysis, it should be accompanied with a debriefing of the test participant where a questionnaire is used in an interview. The interview could simply be to let the user to give a value (e.g. from 1 to 5 where 1 is low and 5 is high) to the degree that different types of emotions such as anger, joy, anxiety etc. were experienced.

In this project, the evaluation has been focused on a relative few test participants, but with the benefit of highly detailed feedback. By evaluation based on usage metrics from data collecting in each device, it would open up for the opportunity to generalize on usage behaviour. A large scale evaluation for the

future product could be done as seen in e.g. [PRG+12], where a tracking program (similar to Google Analytics) is added to the game itself in order to collect data on how the user interacts with the game.

8.2.3 Integrating Gameplay

Obviously Social Voxel cannot be considered as a game with the absence of gameplay, so in its current state it needs to be integrated with a complete gameplay in order for the system to evolve beyond being just a persuasive collaboration tool.

With regards to this project, the absence of a clearly defined gameplay, have also had an impact on the increased difficulty for defining a clear problem definition early in the process.

The work done in this project can be broken down into, the research, analysis and design of a front-end prototype for a mobile framework/tool, that in the future will support a gamification dimension as well. The goal of the gamification would clearly be to drastically increase the fun factor to go beyond a virtual collaborative construction tool.

Gameplay itself can be a powerful facilitator for increasing the creative engagement [LS12]. Minecraft is a great example of that where users create some impressive fortresses to protect themselves during the attacks of the creepers at nights. Gameplay is the cause of this.

[Kel13] talks about how powerful gamification is, as a tool to control behaviour. It states that "fun is the sugar of behaviourism". With that in mind, it is clear that the integration of gameplay may itself be used as a persuasive element.

8.2.3.1 Avoiding Abusive Persuasion

Given the knowledge about how people may be persuaded in general and in software system such as games, it is crucial to acknowledge that this kind of "technology" must be used with care. Scientists like B.J. Fogg and Ramin Shokrizade have proposed their own models on how to persuade people to a certain behaviour through persuasive design (FBM) or through fun in gameplay [Sho13b] [Sho13a] respectively. Since humans to some extent are predictable in terms of what makes people decide to do a certain behaviour, it is important as a developer and designer of a persuasive platform not to abuse this knowledge.
by accident or with intent. There are examples of games that rely on a business model which is integrated to the gameplay itself and use persuasive design to make money through features that resembles gambling. Such designs would be both unethical and damaging for the gaming experience. Persuasion through gambling may lead to addiction and is problematic for the younger users whose brains are not yet fully developed.

8.2.4 Improving First Impression

Currently the developed prototypes have had little to no focus on the use of a color scheme, and beautiful icons. An overall theme is needed and this would undoubtedly contribute to a more pleasant UX. The main reason that this hasn’t been part of the prototyping is simply because the choice of an overall look and feel of the frontend is very dependent on the story and setting of the gameplay itself. These two elements must be connected otherwise the user might have problems with relating (establishing trust) to the platform. According to [SBFH04], establishing trust is affected by the first impression of "look and feel", navigation, colors and text size. It is worth mentioning that this study was conducted on websites related to health, but studies like these could justify resources spend on improving the visuals of tablet applications as well.

For similar reasons for using resources on improving the visual, the feedback through audio is an important factor, especially considering games. Sound is a way to integrate triggers in the platform as sound attracts attention and may also be used for non-visual feedback in situations with long loading times. As an example, long loading times may occur when a user wants to play and thus clicks on a game that needs to load. On mobile devices it is recommended to use non-visual feedback for delays of more than 4 seconds [RO03]. Here sound and/or vibration could be used as a signal type of trigger to let the user know when the loading is over, since long loading times may distract the user from the application.

8.2.5 Beyond Mobile

There is the potential to push the UX of collaboration tool into other areas such the Kinect or Oculus Rift where the interaction empowers the experience of virtual reality (the second point in Fogg’s functional triage) depending on the gameplay that is to be incorporated. A voxel-based virtual reality with

\[\text{http://www.oculusvr.com/}\]
the opportunity to manipulate physical areas of the world could prove as an
interesting alternative way to interact with Social Voxel that hasn’t really been
seen before in other of the related games.

These examples of interaction platforms offer an alternative user experience as
a supplementary for interacting with SV as a tool for creating and playing in
immersive voxel worlds. The reason why these technologies should be limited to
supplement the primary device (tablet or pc) are simply because of the limited
amount of time that users would want to wear a helmet (Oculus Rift) or wave
with his arms (Kinect), before they either get bored or exhausted. A review
of the Oculus helmet claim that half an hour of use currently result in headache or
dizziness, which clearly isn’t an experience that SV should be associated with!

In relation to the future development of a usable platform the focus would clearly
be in the mobile space and on the challenges that will come when PC users are
playing the same games as well. If the gameplay relies on competing against
each other, the PC user would have an unfair advantage because of more precise
and faster interaction through mouse and keyboard and thus the implications
must be considered. In terms of gameplay this may be solved if the game is
cooperative instead of competitive.

8.2.6 Backend

The backend component of SV have not been addressed at all in this project,
though the use case diagram presented in Section 4.2.5 may give a very basic
understanding of the kind of functionality a backend must be able to handle.

One of the main challenges that have the largest effect on the UX, is the syn-
chronization and handling of concurrent users that play and create together
across countries. This problem may be divided into several small subcompo-
nents which would make a possible solution easier to manage. The very first
step is to decide on how the technicalities of server(s) is setup, and the most
feasible solution for this challenge would probably be to use server virtualization
services such as Amazon Web Services and Google App Engine. These kinds
of services is the common choice for indie developers of online games, as they
relieve much of the developers resources better used on the actual game itself,
and may scale dynamically depending on the popularity of the game.

The second part of the challenge with backend development is implementing
an efficient network protocol that supports the requirements of the platform.

[http://www.eurogamer.net/articles/2012-09-05-oculus-rift-impressions-its-amazing-until-it-makes]
Unfortunately an iterative approach for developing protocol prototypes would not suit this component. It is probably more feasible to analyse and design an efficient architecture to start with, after a requirements elicitation step.
The goal of this thesis was to develop a persuasive tablet prototype for a platform that is able to support users in collaborating, sharing and playing together through the tools offered by the prototype itself. The intent with the persuasive design of the platform is to increase the user’s behaviour for creating content.

Three horizontal paper prototypes were created in order to attempt different solutions for designing the platform, and tested using heuristic evaluation. Based on the findings of the evaluation, there have been developed a horizontal computer-based prototype using Axure. Usability evaluation has been conducted in order to test the usefulness of the prototype and finding out if the persuasive design works as expected. The prototype was tested on a Nexus 7 device.

The computer-based prototype demonstrates how users could use the given tools to create worlds with other friends and share them on a online platform. Persuasive techniques have been utilized to attempt to motivate the users to increase their behaviour of creating content for the platform. Based on the test conducted in this project, it is not possible to conclude whether they actually work.

Finally a vertical prototype of an Android application was developed using Unity3D with the goal of implementing a vertical slice of the create use case. Unfortunately the implementation is very basic and only supports an in-game
voxel environment with few features. The goal of using the Android prototype for testing the interactive 3D editors or demonstrating a vertical slice was thus not achieved.
Appendix A

Heuristic Evaluation

A.1 Heuristics

The following is the recommended questions generated by the Interactive Heuristics Evaluation Toolkit for tablet systems in the entertainment category that the evaluator asked himself for each of the given heuristics.

1. Visibility of system status
2. Match between system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recover from errors
10. Help and documentation

11. Navigation

12. Structure of information

13. Extraordinary users

A.2 Results

Score system:

<table>
<thead>
<tr>
<th>Points</th>
<th>Means:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>poor/Lacks support</td>
</tr>
<tr>
<td>2</td>
<td>Average/Partial support</td>
</tr>
<tr>
<td>3</td>
<td>Good/full support</td>
</tr>
</tbody>
</table>

Evaluator 1:

<table>
<thead>
<tr>
<th>Heuristik #</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Average: 1.69  1.77  1.85

Evaluator 2:
## A.2 Results

### Total Average:

<table>
<thead>
<tr>
<th>Heuristic #</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Average:**

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>p3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1,77</td>
<td>2,08</td>
</tr>
</tbody>
</table>

**Total Average:**

<table>
<thead>
<tr>
<th>Average:</th>
<th>1,69</th>
<th>1,77</th>
<th>1,85</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average:</td>
<td>2</td>
<td>1,77</td>
<td>2,08</td>
</tr>
<tr>
<td>Total average:</td>
<td>1,85</td>
<td>1,77</td>
<td>1,96</td>
</tr>
</tbody>
</table>
Appendix B

Emperical Evaluation

B.1 Tasks

The following sections present the questions and tasks given for a empirical evaluation of SV

B.1.1 Before the test

- Age
- Sex
- Occupation
- Have you had a smartphone before?
- Have you had a tablet?
- Have you played games on smartphones and tablets?
- Have you tried to play any voxel games on these devices or any platform?
B.1.2 During the test

1. You are at home and have some time to spare before you need to leave for an appointment. You decide to try this new app called Social Voxel that your friends have been talking about. You want to join a game with a couple of friends, what do you do?

2. You have now played for some time and really feel like making a game for your own after seeing what others have made. How do you proceed?

3. Now that you have asked your friends to join, proceed with adding some objects to the world. How would you put a tree object in the world?

4. Let’s assume you have finished creating your game, now is the time to personalize your own character avatar. How can you do that?

5. SV have noticed that you have been active in the platform and awards you with points enough for the advanced game editor to be unlocked. Try to make a game with the new editor.

B.1.3 Debriefing

• Did the application comply with your overall expectations?

• Can you possibly mention the 3 good and bad things about this application?

• Do you feel that SV motivates you to create your own games?

• Does the platform give you the abilities to need to create things you like?

• Are there any features that you feel missing?

B.2 Test Transcripts

B.2.1 User 1

B.2.1.1 Before the test

23 years old, male, student, has smartphone but no tablet, played Minecraft before on PC
B.2 Test Transcripts

B.2.1.2 During the test

Task 1  The user immediately attempts to scroll the game list and then taps the first game. Since the app is slow to load with no immediate feedback he taps a second time. He spots the menu immediately and presses, finds the invite friends button and then invites a couple of friends.

He is wondering if he is able to see if the friends respond to the invitation, by some feedback on the screen. He is not able find any feedback as the system doesn’t support that yet.

Task 2  He presses the back button of the device as he hopes to step back in the app, but the browser shuts down in this attempt.

I intervene and resets the browser to the same state again. He now attempts with clicking the menu button, and then exits the in-game environment. He proceeds with clicking the create world button in the menu which takes him to the correct place. He attempts to click on the slider on a particular landscape and expects it to choose it but nothing happens. He attempts to scroll in the create world several times. He stars with choosing colors first. He is not sure what prefabs are. He asks if it is some kind of package. He clicks generate several times as it is slow to load. He is now in the in-game create environment.

Task 3  The user understands that the joystick is for controlling the character and that adding blocks would be added to the center of the screen.

He doesn’t understand what the jump button does.

He clicks the prefab button in the drop down menu and adds a tree. He understands the control on the left but is not sure of the outcome for the vertical up and down buttons.

Task 4  Clicks on settings in the in-game menu, but then decides to exit the game.

He discovers the username in the top, and understands that he has a list of recent maps that are under creation.

He finds the edit avatar button and successfully completes the task.
He comments on the slider bar of the avatar, saying that he thinks the pictures to the far left and right should be cut in half so that it is understood that the user would have to slide horizontally for choosing a new body part.

**Task 5** He clicks around in the toolbox on the left side.

He doesn’t understand the landscape lower raiser tool.

He doesn’t realise that the pro editor shows a preview of the whole world, i.e. that the character isn’t placed inside it. After a while he is told about what the placeholder for the 3D preview should contain and then he understands the interface a lot better and even thinks that it is a nice way to make changes on the larger scale while still being able to make the small details when generating the world.

**B.2.1.3 Debriefing**

**Did the application comply with your overall expectations?** He would expect more elements in the in-game environment and controls for the physics of the world. He doesn’t think that being able to add static blocks and prefabs is enough. He expects a lot more fine grained details for things such as adjusting gravity, making objects breakable, and adjusts the speed of things in the world.

He expects that the platform should support an unlimited playground for whatever the user may like to do.

He says that his expectations would certainly depend on what is friends have told him about such an app.

**Can you possibly mention the 3 good and bad things about this application?**

1. He likes the menu icon on the top left. It is big enough and he knows it from other apps on his Android phone.

2. The colors of the interface are nice to look at. The interface colors are easy to comprehend.

3. Create a world editor was nice and easy to use because of the integer numbers indicating the steps. The fact that all steps are visible on one
screen and not having to go through many small steps to generate a world was good.

1. He thinks that the panel on the right is confusing, even though he feels tempted to click on the arrow that open the drawer

2. He doesn’t like that the app closes when using the physical back button of the device. On his mobile phone, this button usually backtracks in the app and doesn’t close.

3. He thinks the app lacks an optional tutorial for how to control the character and manipulation of objects in the in-game environment.

Do you feel that SV motivates you to create your own games? Partially, Since the start screen doesn’t really indicate that he can create worlds.

Does the platform give you the abilities to need to create things you like? Yes, but I would like to be able to configure more details such as gravity.

Are there any features that you feel missing? There is no gameplay so the application feels undone. Otherwise I would think it could be nice if there is a tutorial in the app itself as a game created by you, developers, where the user must attempt to create a world with the minimum set of tools as a way to get familiar with the system.

B.2.2 User 2

B.2.2.1 Before the test

22 years old, male, student, has smartphone and tablet (nexus 7), played Minecraft before on PC.

B.2.2.2 During the test

Task 1 He clicks on the menu to start with, but then attempts to scroll the content. He discovers that a friend and three others is online in a game and clicks on that tile. He completes the task with success.
Task 2  Exits the game, clicks on menu, goes to create world interface. He understands the slider, and chooses colors. He adds colors to all earth and water. He chooses three prefabs and generates the game. He completes the task with success, and then attempts to move the character with the joystick.

Task 3  He first clicks on the menu button but realises that it is the incorrect thing to do. He clicks the dropdown, chooses a tree prefab and then attempts to click on the prefab controls. He understands the controls except the translation buttons, as he is unsure which direction it would be moved to. He completes the task by finishing the addition of the prefab.

Task 4  He clicks the menu button several times, clicks publish world and exits the world. He is confused about whether the application is able to remember the current building progress so he chooses to publish the world. When back in the front, he quickly finds the avatar editor to be located inside the profile view that is accessed through the top bar. He completes the task with success.

Task 5  The user clicks around on the three radio buttons and seems to understand the purpose of the tools. But when he is asked to show on the screen how he has imagined that they should be used, he tries to drag-and-drop prefabs and blocks instead of the intended way of a single tap to select, before drawing/spraying prefabs or blocks on the 3D world. Just after he does this, he actually talks about an alternative way to interact that exactly matches the intended way. With a little confusion the user actually has a pretty good idea of how it should be used, which is quite surprising as the prototype gives very little help in the current state.

B.2.2.3  Debriefing

Did the application comply with your overall expectations?  Overall he seems to be pleased with what he says regarding his initial expectations.

Can you possibly mention the 3 good and bad things about this application?

1. The information on the content of the tiles are good.
2. He likes the how the content is structured

3. Finally he is think it is a convenient way to add prefabs to the world.

1. The horizontal slider in the create world is annoying because the individual images do not snap to the place of the selector when a sliding gesture is released.

2. It is annoying that the color chooser on the create world do not work as expected. It always chooses a brown color.

3. He finds it annoying that there is no save function in the in-game environment. He thinks that he must publish a world in order not to lose the progress. He is in doubt if the application makes a backup itself.

Do you feel that SV motivates you to create your own games? Yes, somewhat. He notices the unlockables and says that he feels motivated by these unlockables.

Does the platform give you the abilities to need to create things you like? Overall yes.

Are there any features that you feel missing? No, not really.

B.2.3 User 3

B.2.3.1 Before the test

23 years old, male, student, has smartphone, no tablet, never played Minecraft or the like.

B.2.3.2 During the test

Task 1 He clicks the first content tile, clicks menu and invites a couple of friends.
Task 2  Clicks on menu, tries to access the settings, which fails and afterwards exits the game. He proceeds to the create world interface. He doesn’t understand how the slider works. He attempts to press different landscapes several times without luck. He proceeds with choosing colors, and tries to add a tree to the landscape before generating the world.

Task 3  He attempts to navigate using the joystick and asks why it doesn’t work. He clicks on the dropdown, selects the prefab panel and clicks on a recent prefab to add a tree to the landscape.

Task 4  He exits the game using the menu, clicks on the profile view, but actually goes away again, as he doesn’t realise that he can edit his profile, he continues to search in the friends list, goes away, clicks on the settings, goes away, goes back to the profile view, and then discovers the edit avatar button. He succeeds in the task but it was harder for him compared to the others, but he did manage to finish it in time.

Task 5  He clicks on the generate button, clicks around on the tool box, attempts to drag tree prefabs to the world along with blocks. drag-and-drop is the way he expects.

B.2.3.3 Debriefing

Did the application comply with your overall expectations?  Yes, but I haven’t really played Minecraft so I am not sure what to expect.

Can you possibly mention the 3 good and bad things about this application?

1. He thinks it is cool to have his own avatar
2. It is nice that you can add prefab into the world
3. The procedural world editor

1. The missing interactivity in the app
Do you feel that SV motivates you to create your own games? Yes, definitely. If the interaction worked, he would like to build, but the only problem is that he would not know what he should build.

Does the platform give you the abilities to need to create things you like?

Are there any features that you feel missing? No, not anything I can think of!
Appendix C

Attributions

The following is a list of hyperlinks to the source of any icons used in the computer-based prototype, in order to comply with the license requirements. I want to thank each of the authors for distributing their work for others to use. The order of the list is completely random.

- http://www.iconfinder.com/icondetails/115727/512/bookmark_favorite_heart_love_valentines_day_icon
- https://www.iconfinder.com/icondetails/49856/256/open_padlock_unlocked_unsecure_icon
- https://www.iconfinder.com/icondetails/49855/256/closed_padlock_icon
- http://www.iconfinder.com/icondetails/99683/256/i_alt_share_icon
http://www.iconfinder.com/icondetails/107175/512/circle_color_facebook_icon

http://www.iconfinder.com/icondetails/107170/512/circle_color_twitter_icon

http://www.iconfinder.com/icondetails/107180/512/circle_color_google_icon

http://www.iconfinder.com/icondetails/103174/512/list_menu_icon

http://www.iconfinder.com/icondetails/103172/512/add_plus_icon

http://www.iconfinder.com/icondetails/103181/512/close_cross_delete_remove_icon

http://www.iconfinder.com/icondetails/103184/512/check_checkmark_ok_yes_icon

http://www.iconfinder.com/icondetails/103345/512/settings_icon

http://www.iconfinder.com/icondetails/171495/512/exit_right_icon

http://www.iconfinder.com/icondetails/134216/512/hamburger_lines_menu_icon

http://www.iconfinder.com/icondetails/43246/128/email_letter_mail_message_monotone_round_sobre_icon

http://www.iconfinder.com/icondetails/172458/512/arrow_down_icon


http://www.iconfinder.com/icondetails/43304/64/add_monotone_plus_icon

http://www.iconfinder.com/icondetails/103459/512/back_controls_play_icon

https://upload.wikimedia.org/wikipedia/commons/thumb/9/94/ Stick Figure.svg/500px-Stick Figure.svg.png

http://www.iconfinder.com/icondetails/103296/512/arrow_full_up_icon

http://www.iconfinder.com/icondetails/118619/512/plus_icon

http://www.iconfinder.com/icondetails/172180/256/email_icon

http://www.iconfinder.com/icondetails/49848/256/media_random_shuffle_icon
• https://www.iconfinder.com/icondetails/118743/512/alt1_arrow_up_icon
• https://www.iconfinder.com/icondetails/49866/256/undo_icon
• https://www.iconfinder.com/icondetails/118618/512/alt_plus_icon
• https://www.iconfinder.com/icondetails/118642/512/alt_minus_icon
Bibliography


[heu] Interactive heuristic evaluation toolkit.


[Nor13] Alex Norton. Designing a game to behave in ways for which it was not designed. [http://www.altdevblogaday.com/2013/04/09/designing-a-game-to-behave-in-ways-for-which-it-was-not-designed/], April 2013.


[RO05] Virpi Roto and Antti Oulasvirta. Need for non-visual feedback with long response times in mobile HCI. In Special interest tracks and posters of the 14th international conference on World Wide Web, WWW ’05, pages 775–781, New York, NY, USA, 2005. ACM.


