A Mixed-Initiative Approach for Assisting Blind and Visually Impaired Users with the BlindBits Level Editor

Areen Said

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juli 2016

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, July 14, 2016

Areen Said

Contents

Acknowledgments

I would like to thank a few people that helped me throughout this work. First, I want to thank my supervisor Christoph Schaufler for his support and advice. Thanks to DI Georg Regal and Mag. Elke Mattheiss from the Austrian Institute of Technology for their support and help whenever I needed it. A big thanks to Mag. Erich Schmid from the Bundes-Blindenerziehungsinstitut and all the students that volunteered and made this work possible. Last but not least I would like to thank my family, who also helped greatly in their own ways.

Kurzfassung

Blinde und sehbehinderte Schüler erhalten Orientierung und Mobilitätstraining um allgemeine Navigationsfähigkeiten und bestimmte Routen zu erlernen um mehr Unabhängigkeit zu erlangen. Um dieses Training zu fördern versucht ein Projekt namens BlindBits die Wegfindung der Schüler in einer Schule für Blinde in Wien, das "Bundes-Blindenerziehungsinstitut", zu verbessern. Um dies zu erreichen, wird eine Art Schnitzeljagd Spiel für das Handy entwickelt mit dem Zusatz von einem Editor, der die Schüler ihre eigenen Geschichten und Aufgaben für die App erstellen lässt. Da dies ein mühsames Unterfangen sein kann, wird ein Assistent eingeführt, die die Arbeitszeit für die Erstellung eines Spiels stark reduziert. Durch die Verwendung von Graphgrammatiken wird die Spielstruktur aufgebaut, die darüber entscheidet wie und wo die Wege hinführen und wo sie enden. Ergebnisse von einem Testlauf haben gezeigt, dass die Erzeugung mit dem Assistenten im Vergleich zur manuellen schrittweisen Erstellung die Arbeit erheblich erleichtert und beschleunigt.

Abstract

Blind and visually-impaired students receive orientation and mobility training to learn general navigation skills and specific routes and become more independent in their daily lives. To foster this training a project called BlindBits aims to improve the wayfinding of students in a school for the blind in Vienna, called the "Bundes-Blindenerziehungsinstitut". To achieve this, a scavenger hunt-like game app for the smartphone is developed with the addition of an editor, that lets the students create their own stories and tasks for the app. Since this can be a tiresome endeavor, an assistant is introduced that completes a lot of the work in order to reduce the time necessary to create a game. By using graph grammars, a game structure is built that decides how and where the paths should lead to and where they should end. The results show that the assistant developed in this study does indeed ease the building of games compared to manually setting up the whole work.

Chapter 1

Introduction

1.1 Motivation

Blind and visually impaired people cannot rely on visual cues as sighted people, thus finding their way in an unfamiliar building can be a significant challenge. Therefore, schools for the blind try to assist pupils in learning general navigation skills and specific important routes by providing orientation and mobility (O&M) training classes. Learning the concept of how to get from one place to another is crucial for independent wayfinding, but learning a certain route is a tedious process of repeatedly walking the very same route and sequentially memorizing landmarks and orientation points. To counter that, games are introduced as playful approaches for orientation and mobility training to foster motivation and therefore learning, since they tend to be fun.

This is why a project called *BlindBits* was started, which tries to support O&M training in its own playful way. After the game idea was set, two requirements were defined: to create a level editor, where the users can create their own designed levels; and a game app on a smartphone, where the created levels can be played on. During the development some problems arose when trying to find a way to make it accessible and easy to use for the target group. Some levels ended up not being playable because of ill-designed games, e.g., some necessary items were required to continue, which were not obtainable. Another issue was the amount of work that had to be put into it to create a level. Though it was interesting and exciting in the beginning, this turned out to be tiresome for some.

1.2 Objective

The objective of this thesis is to create a digital assistant, that supports the user by removing the tedious work of creating games from scratch. Additionally, finished games should be verified as being solvable before they

get published. In order to achieve this, most of the work should be done automatically by the assistant, like setting up the structure and providing most of the game specific-information, while leaving the creative part, which is to come up with a story, to the user. Since the user is still managing the whole game, errors can occur. To prevent such games being published, the assistant will inspect them for mistakes and point the user to the fields that need to be resolved first.

1.3 Structure

Chapter 2 gives some insight about the purpose of O&M training and playful learning, with several example games that have already gathered some knowledge in this field. After that, Chapter 3 explains the BlindBits project in detail, how it was executed and what design methods were used. In Chapter 4, different procedural content generation methods and their usage are discussed. The implementation of the BlindBits project and the assistant is described in Chapter 5. Chapter 6 shows and evaluates the test results. Finally, Chapter 7 concludes with a summary and an outlook of this work.

Chapter 2

Orientation and Mobility Training

2.1 Definition

The purpose of O&M training is to help blind or visually impaired people to maintain their independence when walking. Orientation is the ability to identify the location and have knowledge of one's position in it. Mobility is the capability of moving in an orderly, efficient and safe manner through the environment [23]. Those two skills are needed by anyone to travel from one place to another. However, the blind and visually impaired need to compensate for their reduced or nonexistent visual information by learning to use their other senses, such as hearing and touch, to enhance their navigation competence.

2.2 Playful Learning

O&M training is crucial for independent wayfinding, but the continuous learning of specific routes and behavior rules can become tedious. Hence, playful approaches like computer games are used in some research studies to foster motivation and learning. Research in this field has shown that the use of computer games to support blind people in O&M training does indeed produce motivation and commitment for learning [20]. Those games support the construction of a mental map of an existing building or space, which later helps the blind to navigate in the real physical world. Ways to achieve this could be the use of the player's own body movement only (*MOVA3D*) [17], or in combination with a Wiimote Controller in order to interact with a virtual environment, as seen in the game *MovaWii* [16]. Both games use the clock orientation system [18], where turning the own body's axis around influences the direction the player looks at. Another approach is an *Audio-*

based Environment Simulator (AbES), a tool that allows users to navigate through a virtual building by using a keyboard, in which it plays auditory cues while the user walks through [15]. The blind can thereby gather enough spatial information of a building to convey it to the real world. Based on AbES, a new game named *Audio Haptic Maze* was created that expands the idea by adding a haptic device [19]. Each distinct object receives a haptic texture to represent the shape, and thus a player can feel and identify them in the virtual environment.

2.3 Tools

2.3.1 Audio Game Maker

Only a few approaches are dedicated to user-generated games. The *Bartiméus Accessibility Foundation*[1] supported the development of software called *Audio Game Maker*, which enabled blind people to create their own computer games without any programming knowledge. Users could create events and place sounds by choosing one from a library that was provided or add one from his or her own soundpools. The interface was completely acoustic and had no graphical UI so sighted users had no advantages or could not aid the blind. The goal was to increase the number of computer games for the visually impaired; however, it seems that it is no longer offered on the official website.

2.3.2 Blastbay Game Toolkit

The Blastbay Game Toolkit is also like the Audio Game Maker software, which allows the creation of audio-only games without prior knowledge in programming, even though it would be beneficial. With its own scripting language, which is based on the C++ programming language, and its own engine, it aims to simplify the development severely. In addition, it comes with numerous auxiliary modules and comprehensive documentation. Also included are orchestrated music loops, which are available for use in their own games. Through this tool a few audio games for the blind were released, making it more successful than the developer had expected but not financially viable. Thus, in 2014 he changed the software from being commercial software to freeware, dropping official support and only updating it when he has time.

[1] http://www.audiogamemaker.com

2.4 Examples

To get a better insight into some of the games noted in Section 2.2, they are described here in more detail.

2.4.1 AbES

The goal of the game is to prove that when users interact within a virtual environment, they can create a mental map of the building, which helps them to navigate later in the actual physical environment [6]. At first, the users learn the keystrokes used to navigate in the virtual world (see Fig. 2.1). Then they hear sounds that represent objects and their relative location. To make the virtual environment interesting, game elements are implemented. The player has to find and collect all the jewels that are hidden in different rooms and avoid colliding with a monster that can take the jewels away and hide them in the building again. Each step approximates one step in the real world and turns are 90° which makes the movement discrete. Opposed to continuous movement, this helps the user to understand exactly and far he or she has traveled. For the test, three blind participants were asked to play the game. The building that was used was unknown to the testers and they were also not aware of the overall purpose of the project. The presenters had tasks for the users that they had to solve within a certain time limit. The next step was to let the participants perform the same tasks in the real world, where they found themselves for the first time. It turned out that they were even faster at solving the tasks in the physical world than in the digital world, which proves the concept of the writers. The main advantage over standard O&M training is that the spatial land of unknown places can be learned in a safe and controlled manner. Since the target group were totally blind students, the visual representation of the game shows a top-down view of the map, although the audio feedback is in 3D. This means that, when the enemy is approaching from the left side of the player, he or she will also hear it as if the monster is coming from the left. The purpose of the visuals is to make it easier to control the movements of the players.

2.4.2 Audio Haptic Maze

The Audio Haptic Maze is based on the AbES software, with the addition of a haptic device. With the haptic device "Novint Falcon"[2] a user can control a 3D cursor inside the virtual environment, which can be seen in Fig. 2.2. Any object in the game can receive a texture, which is used to represent distinct objects. If the user "touches" a wall with the 3D cursor, the haptic feedback of that object's texture will be different compared to that associated with touching a door. This works with different vibration strength feedback and

[2] http://www.novint.com/index.php/novintfalcon

(a) (b) (c)

Figure 2.1: In column (a) the floor plan of the actual building can be seen. Under column (b) the digital version of the map was created that serves as the virtual game world. In column (c) items and the enemy can be seen with the addition of objects getting labeled.

the controllability of the cursor. For instance, sliding the cursor over an icy surface will make the cursor of the Novint Falcon start skidding whereas touching over a rough surface will be harder. With this, a player will be able to identify shapes within the rooms and therefore create a more precise mental map. For feedback for actions like walking or turning, vibration in the direction of the user's movement is applied. The authors came to the conclusion that the haptic interface is as effective as the audio interface for O&M purposes.

2.4.3 MOVA3D

In the video game MOVA3D, the player has to navigate through a virtual environment in order to find a certain number of pocket watches on the map. Once the user finds and grabs one, he or she has to keep it for thirty seconds while running away from enemies who try to steal it. The navigation movement is discrete, where each step is 40 cm and each turn represents 30° in the real world. The game has 3D graphics which are used to motivate and provide extra cues for low-vision users; however, the software was developed in a way that it is possible to navigate without using the visual cues (in Fig. 2.3(a) the visual representation can be seen). The most significant elements of the environment were put into the video game to make it possible to create an accurate mental representation of the building. Stereo sound that

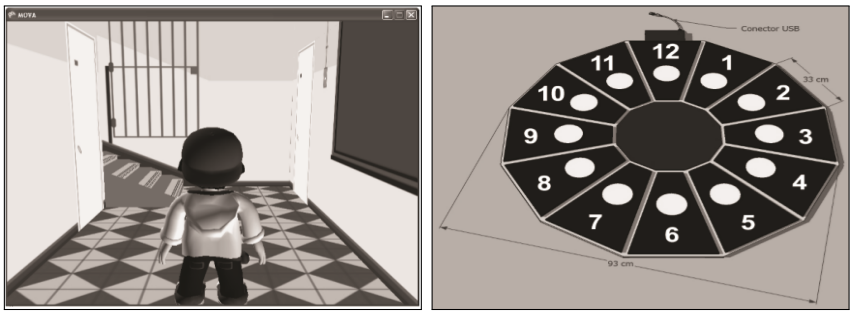
Figure 2.2: The "Novint Falcon" emulates haptic feedback. A user can control a 3D cursor inside the virtual environment.

emulates 3D sound was used to allow players to detect each of the elements in the video game. The navigation works with a normal keyboard or a "Digital Clock Carpet" (DCC). The DCC consists of a round wooden structure divided into 12 cells (see Fig. 2.3(b)). The Sections correspond to keys that the user can press with their feet. It also has tactile cues that allow the user to recognize where they are standing on the carpet. The DCC uses a clock orientation system for navigation, meaning that if the user needs to turn by 90° he or she would have to turn to 3 o'clock and press it. Thus, using this device does not allow the player to see the display device at all times while playing.

A test with 24 children was conducted with the age range from 7 to 14 years, with seven of them blind and the rest partially sighted. The results have shown that users could transfer what they have learned in the game to real world tasks.

2.4.4 Legend of Iris

The Legend of Iris plays in a virtual world, which has its focus on conveying O&M skills instead of teaching an existing building. The game took its inspiration from the popular Action Adventure Puzzle game "The Legend of



(a) (b)

Figure 2.3: The visuals of the game "MOVA3D" can be seen in (a) and the hardware used for the DCC is displayed in (b).

Zelda".[3] It promises to deliver more fun and a high replay-value compared to the other O&M teaching games mentioned above, by providing a story and lots of puzzles [2]. Some tasks in the game are, for instance, following moving objects, while having background noises or avoiding moving objects to teach focused hearing. In order to create a more realistic acoustic environment head-related transfer functions (*HRTFs*) are used. HRTF describes how sound is received by the ear, whether directly or indirectly as a reflection, and takes the linear distortion in the signal through the ear, the head and the torso into account. HRTF describes the common filtering effect the head, torso and the outer ear have on the frequency distribution of an incident acoustic signal. To give a more natural feel of hearing the environment, the virtual reality headset "Oculus Rift"[4] can be used to locate objects aurally by moving the head. The game uses continuous movement to be faster in navigating. Naturally, this results in users being lost in the world and not knowing where they actually stand in a room, making it impossible to create a mental map; although that is not the intention of this game, but rather to teach users certain skills which can be used in real life, such as focusing more on their hearing. A screenshot from the game can be seen in Fig. 2.4.

[3] http://www.zelda.com/
[4] https://www.oculus.com/en-us/rift/

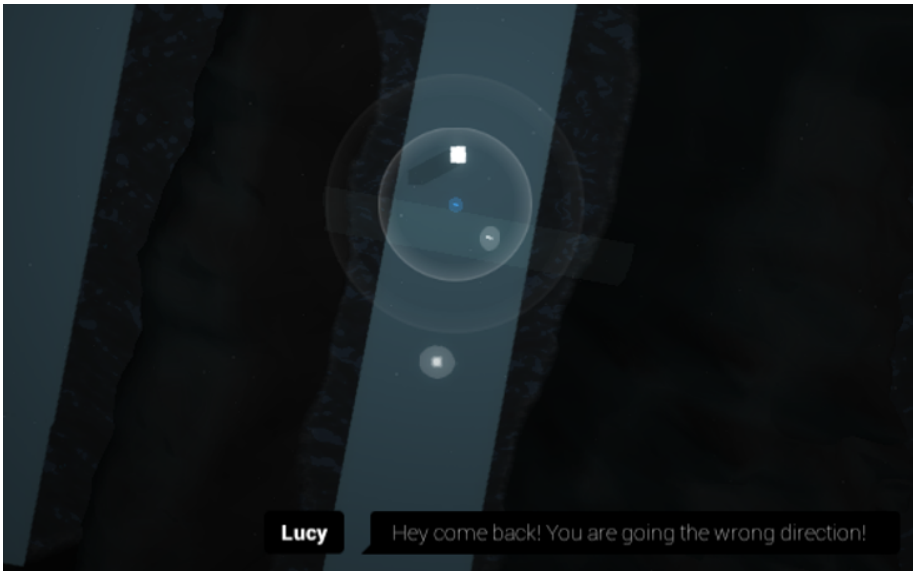


Figure 2.4: The visuals of the game "Legend of Iris". In this screenshot the user walks on a bridge and avoids evil spirits.

Chapter 3

The BlindBits Project

The BlindBits project is coordinated by the *Austrian Institute of Technology* (AIT) and is carried out in cooperation with the *Bundes-Blindenerziehungs-institut* (BBI) and the *University of Applied Sciences Upper Austria*. The latter is responsible for the implementation and technical issues whereas the BBI, a school for the visually impaired and blind, provides feedback from the position of the target group, which is crucial for the execution of the project. The purpose is to complement the O&M training of the BBI and support the learning of the school building routes by using a digital educational game.

3.1 Design Method

Regarding the design phase in developing a game for blind people, a model is presented where the end user takes on an important role [21]. This end-user-based design might lead to a higher acceptance rate of the participants and reduce the probability of redesigning the software. That is why this project implements a user-centered design process, which means that students work with the research team in workshops and provide feedback on the development of the game and the level editor in all the project stages. This way the implementation will be shaped together with the students by having multiple test cycles to ensure a product that fits the needs of the blind and visually impaired.

3.2 The Game

The game type itself is like a scavenger hunt, which uses the BBI building as a playground. To be able to play it, an Android-powered smartphone with the BlindBits app installed is needed.[1] Essentially, the goal is to navigate from one place (in this game it is the door of a room) to another and by

[1] Future releases to other platforms are possible.

Figure 3.1: A student using the game to scan a NFC-tag.

doing so, fulfill all the tasks that challenge the player on his or her way. At the beginning, the player has to run the app and choose a game they want to play. Those games are a collection of predefined, as well as user-generated levels. The app will then ask the player to position himself at a specified start point. From there on, the player will be faced with multiple tasks, which include navigating to different locations within the school building. Depending on the level, different paths in the storyline can be taken, where the player has to solve puzzles and gather items in order to achieve the goal of the game.

3.3 Game Mechanics

The main element of the game mechanic and thus also of the editor is an event. An event is set at one concrete location (e.g., the cafeteria) that initiates a speech by a non-player character (NPC) (e.g., the waiter) or a dialog with that character (e.g., by answering a question asked by the character). Moreover, interactions and the collection of items happen at an event. The following events can occur during the game:

Dialog: This is the standard event that initiates a dialog with a NPC. It can be used for progressing the story or providing information about certain elements in the game.

Quiz: The player will be confronted with a question and has to choose the right answer from several possibilities.

Choice: The app works similar to the Quiz-Event, with the difference being that the chosen answer determines the progress of the story.

When starting an event, a sequence of actions (hints, dialog options, fetching an item, etc.) is triggered. Events can only be started if the preconditions are fulfilled. This could be the possession of a certain item (e.g., gold) or

having finished a predecessor event (e.g., event “talk to the janitor” finished) or both. All events have a postcondition to mark that an event is finished. As a postcondition the player can receive a concrete item (e.g., gold) or a token (e.g., event finished). The possession of an item or token is then used as a precondition for successor events.

3.4 Level Editor

As mentioned above, the students can create their own scavenger hunt games, which can then be played and shared with other students. The same tool is used by the team behind this project to create some example games to produce a small collection that can already be played by the students and give some inspiration for how a game can look. Hence they can write the whole story, choose which and when events should occur and what items are needed. As the users are blind, which could result in disregarding certain steps while creating a level, an assistant is included in this tool, which this thesis will revolve around. This should ensure playable levels by finding design errors and providing help in correcting them when needed. It should also aid users by giving advice on what to do next.

3.4.1 Text-Based Editor

When the planning for the level editor started, the platform was chosen quickly: a desktop PC with Windows 7 or higher. The argument is that most students are trained to use a PC with adjusted contrast and/or screen readers. Therefore using a keyboard for writing stories and creating events seems more practical with a desktop. Windows is the main OS used in the BBI, so the tool can easily be installed on their devices. The editor is text-based and can be operated with the keyboard only using the default Windows Text-To-Speech (TTS) API for reading out the UI elements and its contents. There is also a graphical UI implemented for the pupils with low vision, which can be used with a mouse. In the menu, they can change the contrast, scale the size of the UI and enable or disable the TTS reader. During the first tests and after some feedback, new ideas came forth about how the editor can be expanded. More information on the development of the editor can be read in Section 5.2.

3.4.2 3D-Model Editor

One of the additions is the virtual model of the school building, which the students can walk through with the keyboard to test their created games without the need to do so in reality. Furthermore, editing features are implemented to allow direct modification of a game, so users can fix errors or change events while testing the game in the 3D world. This feature is

supported by audio cues to enable non-visual handling of the editor as well. One step in the game roughly equals one step in the physical world. Acoustic feedback is provided when coming close to a wall and certain objects or rooms trigger distinctive sounds that allow them to be identified. The used sounds need to be chosen carefully and in consultation with the target group, as those can get annoying or even be confusing for the users. For example, a constant knocking with increasing speed when approaching a wall was assessed very negatively by the pupils. As a help function, information about the current position can be enquired by pressing a key. Distance in meters and compass directions turned out to be useless, as pupils usually do not have an appropriate conceptualization of this information. Providing information about the current floor and which rooms are ahead of the player seemed to be more suitable.

Alternatives

Another idea that was brought up and also realized later is to have the editor on phones. Most visually impaired pupils use their smartphones regularly and are familiar with making text input. With a mobile version, pupils could create games independently from their location. Input would be handled by swiping and tapping. For TTS output the native TTS engine would be used.

Chapter 4

Procedural Content Generation

4.1 Definition

Procedural content generation (PCG) is the creation of game content with limited or indirect user input using algorithms [28, 30]. The term "content" can refer to many things such as levels, maps, items, quests, weapons, etc. This definition, however, is not universal and excludes common applications of search and optimization techniques like artificial intelligence (AI). The most common uses of PCG in games have the purpose of providing more content and worlds than could be reasonably expected for a human designer to manage, given the time constraints during design, the limitation of artists and hardware constraints for storing large, detailed worlds [4, 10]. These tools mostly require just a human to press a "generate" button to fulfill the task like in *Civilization* [34], or they generate content automatically when needed while exploring the world, as seen in *Minecraft* [31, 27]. Thus, the aim of PCG is to reduce the scutwork for designers, while giving them more opportunities for creativity.

The range of fields that can gain from PCG is vast, e.g., in movies, sound, textures and so on. This thesis, however, focuses on PCG in the gaming field. In the following Sections some approaches for content generation are discussed.

4.2 Types of PCG

4.2.1 Online and Offline Algorithms

One of the first things that has to be considered when using PCG algorithms is whether the generation is carried out offline during game development or online, and if the algorithm has to work during the runtime of a game. An

(a) (b)

Figure 4.1: This figure shows the landscape in The Elder Scrolls: Oblivion [32] (a) and a randomly generated world map in Civilization IV [34] (b).

example of offline PCG is when a fictional world like landscapes (such as hills, trees, plants), locations of treasure chests and even dungeons are created automatically, for example in the game *Oblivion* from the *Elder Scrolls* series [32], which are later modified by a human designer before the game is shipped. The online approach works differently, where for instance the game world is created afresh, whenever a new game is played. Therefore, each world can look different depending on the parameters set by the developers. The turn-based strategy game series *Civilization* [34] allows that method, where unique maps can be generated. Screenshots of the mentioned games can be seen in Fig. 4.1.

4.2.2 Randomness in PCG

All content-generated algorithms expand content of an already existing representation of it. Adding randomness to it is not necessary but can add unexpected structures and increase replay value, e.g., Minecraft. Some algorithms might only need a seed number as input, while others might use a multi-dimensional vector of parameters, which defines the properties of the generated content. For example, when using the same parameter values, the algorithm should create the same content each time. Changing just one value should return completely different results.

4.2.3 Stochastic and Deterministic approaches

Another design question is whether the content generation should be stochastic or deterministic. Stochastic algorithms generate new content every time they are used, while the deterministic approach produces the same result. Considering the seed parameter as mentioned above in Sec. 4.2.2 would mean that all algorithms are actually deterministic; therefore, they are disregarded on purpose to differentiate between those two approaches.

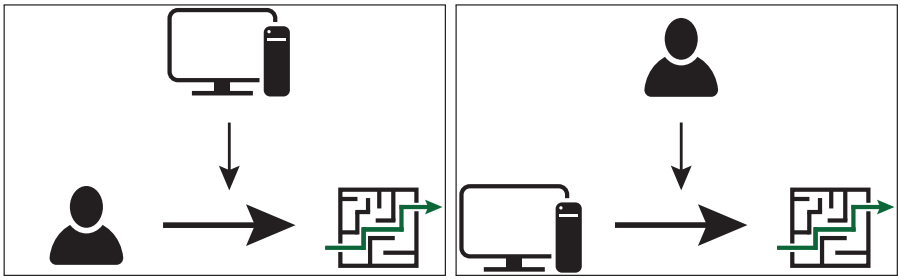


Figure 4.2: (a) Human creates content, the computer tests if it breaks any design constraints and presents alternatives. (b) Computer generates content, the human makes changes to the solution to get the style they want.

4.3 Mixed-Initiative PCG

Mixed-initiative PCG is somewhat similar to regular PCG, with the distinction that the mixed-initiative PCG tools automate only a part of the process, requiring human input to complete the operation [11], and thus using an offline approach. There are two ways how this could be implemented (see Fig. 4.2): first, the human designer has a goal, where he needs the computer to aid him in realizing it; and second, the computer generates content autonomously, with the human to evaluating the results and improving when necessary.

There are two areas in this field that will be looked into more deeply: mixed-initiative interaction and computer-assisted design.

4.3.1 Mixed-Initiative Interaction

Allen et al. [3] mention in their article that mixed-initiative interaction describes an interaction between computer and human in which initiative is shared. This style of interaction can be seen as a conversation. An agent and a human can take turns in controlling the conversation, while the other works to assist it. They can also work independently from each other, and aid the other when needed. The interaction should adapt the style depending to the current problem. They mostly are about computers being able to help the human with the design process.

4.3.2 Computer-Assisted or Computer-Aided Design

Computer-aided tools have been described as “the designer’s slave”, but this can also be the case for advisors, when certain requirements are not met [12, 29]. Computers already perform very efficiently as so called “slaves”, so the next step is to improve their role as an advisor or “colleague” [13]. In Lubart’s

article [33], he mentions that the most ambitious vision of human-computer interaction for creativity involves a real partnership. This idea comes from the field of artificial intelligence, where computers behave like humans and cannot be distinguished from them. An example of creative thinking would be to rely on random or semi-random search mechanisms to generate new ideas, in case one is stuck in his or her thought process. Computer-assisted design can be useful in game editors, helping non-programmers in creating levels. In some cases modding games with the level editors leads to new game titles like *Counter-Strike*.

4.4 Genetic Algorithms

An evolutionary search algorithm is stochastic, which is inspired by biological evolution, such as reproduction, mutation, recombination and selection. The idea behind this is to get individual solutions, that get evaluated. Depending on the fitness function each candidate gets a rating. The *fittest* individuals get the chance to *reproduce*, which hopefully results in better solution candidates due to *mutation*, whereas the least fit ones are removed from the *population* of candidate solutions. The hardest and most important step is to find the right fitness function. This is problem-dependent, since it rates how well an individual solution can solve a problem. Depending on the nature of the problem, the population size may contain many solutions.

They usually act on the following pattern [9]. At the beginning, multiple solutions are generated randomly to create a search space with all possible solutions. Solutions might be *seeded* in areas where they are likely to be found. Then, certain steps will be repeated until a termination condition is reached. First, solutions will receive a fitness value, evaluated by a fitness function. Depending on the method used, all solutions may get rated by the fitness function or only a random sample of the population, although as it relies on the size, the process can be time consuming. The best solutions are selected for reproduction. The next step is to create new solutions by the combination of genetic operators called *recombination* or *crossover* and *mutation* by using a pair of solutions as "parents". The crossover is responsible for inheriting the characteristics of its "parents" and mutation allows the development of new ones to get "offsprings", which are not just determined by their inheritance. Finally, after these processes new results exist that are different from the first population and the process repeats, if the termination condition is not fulfilled. The flowchart can be seen in Fig. 4.3.

4.5 Brute-Force Search Algorithm

Also called exhaustive search, is a method that simply iterates through all the possible configurations. It checks all the possible arrangements, which

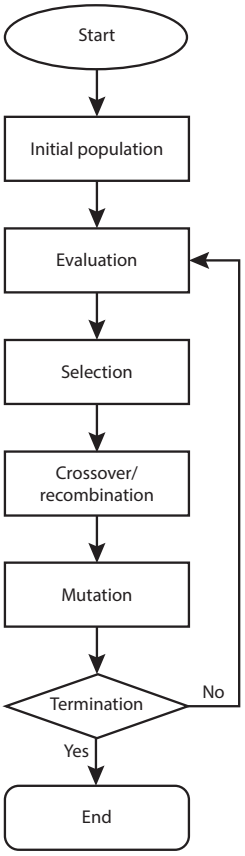


Figure 4.3: Diagram of a genetic algorithm.

means that it will find a solution if it exists. Therefore, depending on the size of the problem, a brute-force search might quickly go out of proportion. So, either powerful enough hardware is needed or heuristics[1] are implemented to limit the problem size.

4.6 Generative Grammars

Generative grammar is a theory that comes from linguistics by the linguist Noam Chomsky, where grammar is considered as a well-defined set of rules to describe a language, consequently being capable to generate infinite numbers of grammatically correct sentences [5]. A generative grammar typically consists of symbols, which are letters from the alphabet that represent certain things, depending on the usage, and a set of rules. A rule defines

[1] A function that estimates the exact solution to rank paths in search algorithms.

what symbol can be replaced by what other symbol which has the form $symbol(s) \rightarrow newSymbol(s)$. Therefore, grammars replace what is on the left-hand side (LHS) of the arrow with what is on the right-hand side (RHS) of it. Symbols that cannot be replaced are called terminals and are usually represented with lowercase characters, while non-terminals use uppercase characters. A given string at the beginning will be checked repeatedly to try and apply rules when possible, until either all the symbols are replaced with terminals, or until a certain amount of derivations is reached, or a desired result is reached. Which design pattern is implemented depends on the use case.

There are two popular methods replacing symbols, namely *sequential* rewriting and *parallel* rewriting [26]. In sequential rewriting, the string gets checked from left to right and starts rewriting a symbol when a rule can be applied without looking at the following symbol. In parallel rewriting, as the name suggests, all possible symbol changes are rewritten simultaneously.

Graph grammars need something to work with so it can expand, and thus that 'S' is often used as the start symbol [7]. For instance, a set of rules might look like this:

$S \rightarrow AB$
$A \rightarrow Bb$
$B \rightarrow A$

Applying those rules with the start symbol 'S', the derivations will look like this:

S
AB
BbA
AbBb
BbbAb
AbbBbb
...

As can be seen, there will always be two non-terminal symbols present in the string, and thus the rewriting will never stop if not restricted with an iteration limit. Choosing the right rules to achieve the desired results is challenging and time-consuming, and needs a lot of testing.

Grammars can be either *context-free* or *context-sensitive*. They are context-free, if the LHS of every rule consists of just one single nonterminal symbol. Hence the name *context-free*, since the rules will be applied if a symbol in a string matches the LHS of a rule, no matter in what context the symbol occurs. This is as opposed to *context-sensitive* grammars, where the occurrence of a symbol is significant. Here is an example of a context-free grammar

$$A \rightarrow BC,$$

where the occurrence of 'A' alone is needed to be replaced by 'BC', and the context-sensitive grammar

$$aAb \rightarrow BC,$$

where 'A' must exist between two terminal symbols 'a' and 'b' to be replaced by 'BC'. Generative grammars are not restricted to strings and can also be used on different types such as shapes (see Sec. 4.6.1), tile maps and graphs (see Sec. 4.6.2).

4.6.1 Shape Grammars

Shape grammars are useful for generating 2D and 3D spaces [7]. Originally they were introduced for generating paintings (2D) and sculptures (3D) by George Stiny and James Gips in the 1970s [25], but then they found their way into computer science. Like string grammars, shape grammar shapes are replaced with new shapes. A shape rule consists of two labeled shapes, with one shape on each side of the arrow [24]. Labeled shapes in turn consist of two parts: a shape and a set of labeled points. The labeling helps to differentiate aspects of the shapes. Non-terminal shapes have so-called markers and are therefore on the LHS of a rule. They are similar to labels, but are actual symbols attached to a shape. Those labels and markers restrict the rules on where they can be applied and help to locate and orientate new shapes. In order to get terminal shapes, the markers and labels must be removed. The alphabet used can consist of words that describe certain shapes, like a wall or a door. These are used as symbols, compared to the letters in string grammars.

A way to employ shape grammar is to first find a symbol in the target space and then look for a rule that implements this symbol. After that a possible position to use the rule is located, based on the relative fitness, as one location might be better to use than another.

4.6.2 Graph Grammars

Graph grammars, which consist of nodes and edges, are useful to represent missions, since they are easier to convey structures like nonlinear and explorative adventure games, where the missions contain dead-ends, locks and keys, and multiple paths that lead to the level goal [7, 26]. In graph grammars, one or several nodes and interconnecting edges can be replaced by a new structure of nodes and edges. Many different types of graphs exist, e.g.,: directed, undirected, simple and multi graphs (as shown in Fig. 4.4). Depending on those graphs, the rules have to be defined accordingly. If a context-free graph grammar is used, the LHS of a rule is a single node, otherwise it is a graph.

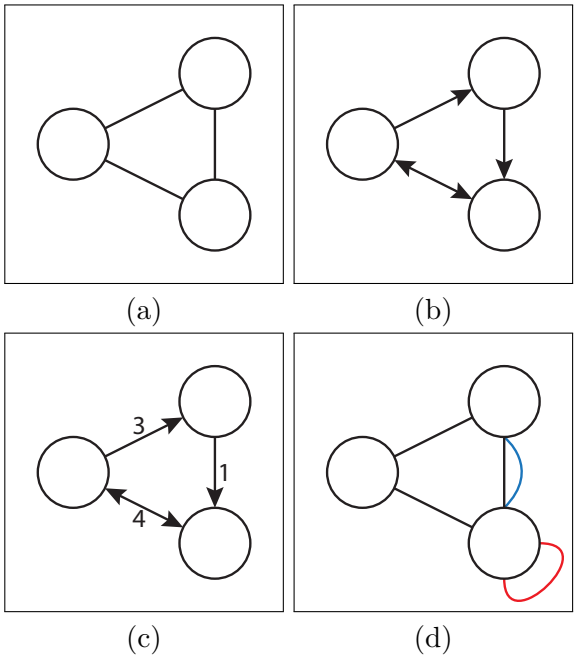


Figure 4.4: Different types of graphs are shown here: (a) an undirected and simple graph, where multiple edges and loops are not allowed and edges form a set, with each edge being an unordered pair of distinct vertices; (b) a directed graph, where the edges have a direction associated with them; (c) a weighted graph, where a number is assigned to the edges representing a value depending on the usage, e.g., lengths or costs; and (d) a multigraph, where two or more edges can connect the same two vertices (blue edge) and where loops can be permitted, which are edges that connect a vertex to itself (red edge).

4.7 Examples in Games

4.7.1 Minecraft

The second most popular game in history[2] *Minecraft*, from the developer Mojang, relies heavily on PCG [31, 40]. As mentioned in Section 4.2.1, Minecraft uses an online approach, where every new game started creates a new world around the starting point of the player. The further the player goes in a direction, more parts of the world are generated. The founder of the game, Markus Persson, wrote a blog post [38] about the terrain generation, saying that he uses 3D Perlin noise with linear interpolation to generate flat areas and smooth hills. However, that article was published in 2011 and much has changed in how terrains work since then; therefore, this technique might be

[2] As of 21st September, 2016

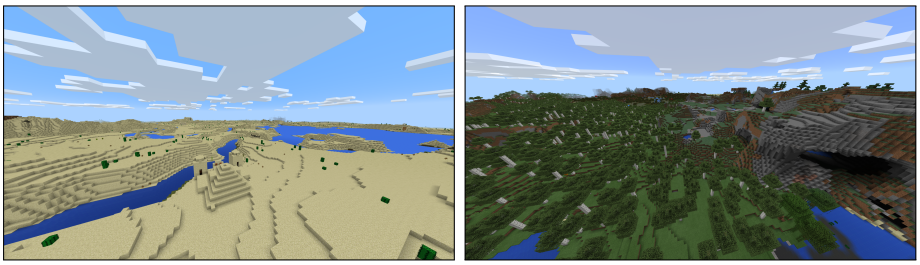


Figure 4.5: Automatically generated landscapes in "Minecraft".

Figure 4.6: The 96 KB game ".kkrieger", where everything is generated with procedural methods.

obsolete now. Some examples of terrains generated in Minecraft are seen in Fig. 4.5.

4.7.2 .kkrieger

The game *.kkrieger*, which came out in 2004 is a first-person shooter that impresses with its small size of 96 KB. This is possible because of the extensive use of PCG methods. Every texture, mesh and sound in the game is generated with procedural methods. A screenshot of the game can be seen in Fig. 4.6.

4.7.3 No Man's Sky

No Man's Sky is an action-adventure survival video game which was released worldwide in August 2016 [35]. It involves exploring a massive universe and its planets, with their mostly unique flora and fauna. Everything can be

Figure 4.7: One of the populated planets in "No Man's Sky", although most of them will be entirely barren.

named, from plants to animals, planets and even stars, but only once; therefore, being the first player discovering places has its benefits. However, the numbers of planets is so huge, that finding a planet that has been visited before is very unlikely.

The main feature of No Man's Sky is that the whole universe, including the stars, planets, lifeforms, ecosystems, and the behavior of the space-bound factions is created through procedural generation using deterministic algorithms and random number generators (as explained in Sections 4.2.3 and 4.2.2). Hello Games uses a 64-bit seed number to create all these features, which enables a virtual universe which contains over 18 quintillion planets [36]. In Fig. 4.7, an inhabited planet can be seen, with generated landscapes and animals.

4.8 Examples in Tools

4.8.1 SpeedTree

SpeedTree[3] is used for generating vegetation by entering a few parameters like the branch length, angle, tree height, etc. So, an artist spends most of the time defining the rules in order to receive the desired trees, grass and plants. After that is done, SpeedTree can generate an infinite number of trees which meet these parameters. It has been used in many major Hollywood

[3] http://www.speedtree.com/

Figure 4.8: An example screenshot of the Game "The Witcher 3" using SpeedTree for the vegetation.

movies and AAA-games[4] like, for example, "The Witcher 3" seen in Fig. 4.8.

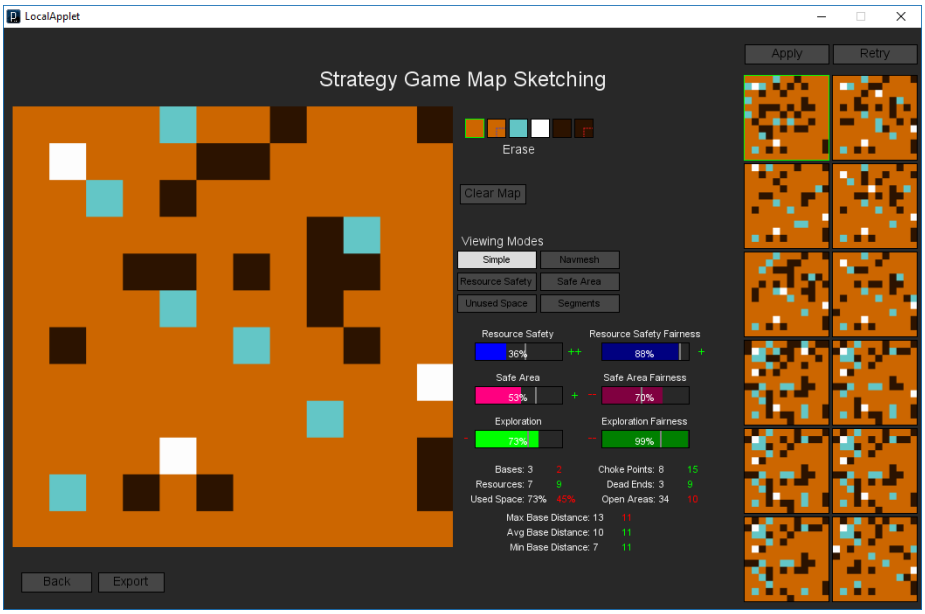
4.8.2 Tanagra

With Tanagra, a human can work together with a computer to create a level for a 2D platformer game [11, 22]. It has a mixed-initiative approach, where a human designer and the generator work together through iterative cycles, combining their strengths. The generator produces a lot of levels quickly and ensures playability without the need for human testing, while the human focuses on his or her creativity and judges the levels' quality. A combination of reactive planning and constraint programming allows Tanagra to respond to designer changes in real time.

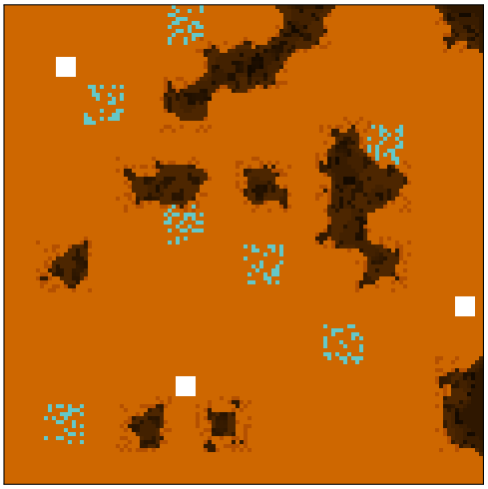
4.8.3 Sentient Sketchbook

The Sentient Sketchbook is a tool that helps the designer to create low-resolution strategy game maps [37, 12]. With a graphical interface, a user can edit the maps to his liking, while the tool tests for their playability. To fulfill the wish that a computer should act like a colleague, it also suggests alternative maps to the user's current designs to enable new unthought map ideas. It uses genetic algorithms, implementing a feasible–infeasible novelty search, which allows the real-time generation and presentation of alternatives

[4] A term used for high budget games by large studios.



(a)



(b)

Figure 4.9: The Sentient Sketchbook interface can be seen in (a) with some sketches created by a human on the left side and map suggestions from the generator based on the user's sketches on the right side. A high resolution map from the sketch in (a) can be seen in (b).

maps, using the users' input for presenting more suitable solutions. The interface can be seen in Fig. 4.9.

Chapter 5

Implementation

The first Section of this chapter covers the used frameworks and functionalities that were needed to realize the project. In the second Section, the implementation of the used algorithm will be described.

5.1 Framework

The Assistant, BlindBits Editor and most part of the smartphone app are written in the programming language C# using the *Unity Game Engine*.[1] There are several reasons for this choice, which are already existent knowledge about the engine, the extensive functionality and support on various platforms and the ease of use. Both 2D and 3D features are needed for this project to handle the UI and 3D school building. One other feature of the engine had a considerable influence on its selection, and that is the portability to multiple OS platforms. With just minor code changes the PC editor can be ported to Android or iOS. Since Unity supports C# and JavaScript (Boo support dropped with Unity 5, but the compiler can still compile it) and the API is the same regardless of the language, the choice for C# is purely personal preference. The IDE is *Microsoft Visual Studio Community 2015*,[2] which provides native support for Unity. Some of mobile OS's distinct features, like NFC scan and TTS, which were needed in this project, are not supported in the Unity API. Therefore, Java is used to create a wrapper class with all the Android specific methods that are needed for this project. After that, Unity can access the code using the Java Native Interface (JNI), with more details explained in Section 5.3.2.

[1] http://unity3d.com
[2] https://visualstudio.com

5.2 Editor

To start working on the assistant, the editor had to be finalized. Otherwise it would make no sense, since the design and features of the editor changed continuously throughout the project, getting feedback from the students after each iteration step. As stated above, Unity was chosen since it has many advantages that can be used for the project. However, it also has its limitations, with one of them being the lack of API for an operating system's TTS engine, which is an essential component in this project. Therefore, a DLL[3] is used to access the speech engine from the Unity project. This resulted in imitating most of the default TTS features regarding what is said and how.

5.2.1 Game Data

The game data is saved in a XML file thus making it easily readable by any text editor or browser. The name consists of the PC account user name and the game name. Therefore, they are also accessible for blind users to read and even edit with their TTS software without the need to use the Game Editor. In the editor there is a publish button to upload the games onto a server and make them playable on the phone.

5.2.2 User Interface

The order of the UI elements is arranged like most other video games, meaning they are listed from the top to bottom of the screen. There was no need to create a special design, since the blind rely on the navigation handling itself, which is built like other applications they use on the PC, and visually-impaired users have the familiar interface.

5.2.3 User Interface Element Adjustments

Since the target group consists of visually impaired and blind people, the UI had to be adapted to aid them accurately according to the navigation handling. Every action gives the user some kind of audio feedback, which will be explained in detail later. However, independently from all the special cases of audible reactions, if the focus of an element changes to another, the name of the new focused element will always be read out, stopping any previous ongoing feedback. There are four types of UI elements that had to be adjusted to be of use: radio buttons, list boxes, drop-down lists and text blocks. Navigating from one UI element to the next works by pressing the *Tab* key and when navigating in reverse, the *Shift* key has to be pressed

[3] A *dynamic link library*, which is a library that contains code and data that can be used by more than one program at the same time.

additionally. The navigation uses the carousel movement, however without the visual effect, which means when reaching the last element of a menu and trying to jump to the next the first element on top of the screen is selected. The same is the case in the other direction, resulting in selecting the last element when the first element was selected before. When active, the purpose of the UI element and its type is read out. Then, depending on the element, the handling works differently, as described in the following subsections:

Radio buttons

Once a radio button field is selected, the TTS engine will read out the current selected button plus the element type and its status; for instance, “simple event mode – radio button – active”. Navigating through the radio buttons in the same field works by using the *up* and *down* arrow keys, which also automatically selects the focused one and deselects the other, since only one radio button from a group can be selected at a time.

List box

If a list box is selected, the elements inside can be opened by pressing the down arrow key. Once inside the list, navigation works with the up and down arrow keys. Pressing the *Home* key selects the first element, while the *End* key selects the last one. When trying to go beyond those ends, a signal sound plays. Another method to speed up the navigation is to press the initial letter of the word that is being searched for, which will jump to the first word with that initial letter. With the *Enter* key the toggles can be selected and deselected, which will trigger the speech to read out the current item in the list and the item type, with the addition of telling the user if it has been selected or deselected; for instance, “book – control field – deselected”. To close the list box it is enough to just press the *Tab* key to jump to the next element, or by pressing the *Alt* key and up arrow key simultaneously, which will close the box but the focus will remain on the list box element.

Drop-down list

Drop-down lists are similar to list boxes, with the difference of just being able to pick one element from the list. Therefore, selecting an element will close the drop-down list, with the full information given to the user: the purpose of the element, its type and the selected item, e.g., “location: drop-down list – gym hall". There is an additional type of drop-down lists which have the ability to add an entry to the list. If this is possible, the first entry will be the trigger called “Add Person”, for example, which switches the list to an input field. When the new entry is written there are again two ways to

proceed, with both of them marking the new entry as selected: pressing the *Enter* key to still have the focus on the drop-down list element or pressing the *Tab* key to jump to the next element.

Text

The adjustments in text fields are about the handling of the speech output. The following list is the behavior expected from TTS software:

Key echo: Every pressed key of the alphanumerical block is spoken.

Word: When a word is finished being typed in by pressing the *Space* key, the whole word is read out. It should also be spoken when using the *Ctrl* key with the *left* or *right* arrow key, which lets the cursor jump over words.

Deleting: When a character is being deleted with the *Backspace* key, the removed character has to be read out. However, when a character is deleted with the *Delete* key, the new character next to the cursor is read out.

Cursor position: When moving to the left or right, the character under the cursor is spoken.

Text: When moving up or down in the text field, the whole text is spoken, including the punctuation.

Normally, the speech will read out sentences just using the words, using special characters only for the pronunciation, as intended like a human does. To enable the speech to recognize punctuation and read it out, the string is converted into an XML string. Since the speech has some key word commands that it should look out for, the XML tags `<spell>` and `</spell>` are added before and after the special character. This will force the TTS software to read them too.

5.3 Mobile

As discussed above, the game is an app running on an Android smartphone (see Sec. 3.2). Another requirement for the phones is to have a NFC chip, since the app relies heavily on that technology. Like the editor, the game is made in Unity with the benefits explained in the Framework section (see Sec. 5.1).

5.3.1 User Interface

Again, the UI is like in the editor from top to bottom. Of course again, every input must give a feedback by sound, and due to the fact that it is a smartphone vibration feedback was also added.

Menu Control

The navigation works through swipe. So going through a list of elements, for example when selecting a game, works by swiping up or down with one finger with the TTS informing of the current selected element. Tapping once repeats the information to give the user the feedback of where he or she is currently positioned. Double-tap confirms the selected element and executes it, e.g., loading a game.

Game Control

When the game starts, the user immediately gets confronted with the first information about the game (where the player should position himself). If he missed the information and wants to hear it again, a simple swipe-up is needed to make the Speech Assistant repeat the text. This works throughout the game. For the case of Quiz and Choice events, when the user's input is needed, there is more to consider. In this case, the user hears a dialog or a question, where he or she has a few options to choose from. Swiping up repeats the dialog or question as expected. But the handling of the answer options is similar to the menu control, which is tapping once for repeating the option and tapping twice for confirming. However, to differentiate the game selection and the menu, to go through the options the user has to swipe left or right. In case the user wants to stop the game and go back to the main menu, swiping up with two fingers shows the option to go back.

5.3.2 Java Native Interface

There are OS features needed for the app that cannot be accessed with the Unity Engine API. Thankfully there is the Java Native Interface (JNI), which is a framework that enables the ability to write native Java methods and access them in a project with a different programming language. With the JNI, Java objects can be created and updated, methods can be called, class information can be obtained and more. Unity provides helper classes such as *AndroidJNIHelper* and *AndroidJNI* to access this functionality in an easier way. The Unity documentation [39] states the following information about those helper classes:

> AndroidJNI is a wrapper for the JNI calls available in C. All methods in this class are static and have a 1:1 mapping to the Java Native Interface. AndroidJNIHelper provides helper functionality used by the next level, but is exposed as public methods because they may be useful for some special cases.

With that functionality, a plugin was written in Java to access the standard Android TTS engine, the NFC-Scanner and phone vibration methods.

5.4 Assistant

The first objective of the assistant was to implement a control feature that inspects if all the postconditions in each event are set and if they are all used in other events as a precondition. Besides, it checks if there are empty fields in an event, like no person is set or no dialog is given. This is crucial to ensure a fully working game; otherwise, it could easily lead to dead-ends, which could be seen in previous tests without that functionality. This control comes to action when a user is trying to publish the game to the server, in order to enable others to play it. When the verification fails, the assistant jumps to the incomplete or incorrect event, telling the user what is necessary to fix it. No special algorithm was required for implementing this feature, since it is a simple check-up process.

The more complex part is the automatic generation of levels. To come up with a suitable algorithm, the requirements for the level generation had to be defined, which are the following:

Controllability: The users should be able to set parameters on how the game levels should somehow appear in the end. Therefore, a flexible approach is needed that aids the user in creating the game structures they want.

Non-linearity: The editor allows users to create games with multiple ways to solve a game or even reach dead-ends by design. This has the benefit of giving the game designer more freedom in producing game structures and stories. The designer can also use this to reward the player in certain ways, like risking to take a route which might seem hard, but that actually becomes a short-cut. It also gives the game a replay-value for motivated gamers, to explore all possible ways to reach a game's end.

Realism: Since the users are a part of the process, the way the games are generated should be understandable.

5.4.1 Used Algorithm

Graph grammars are used for this task. As mentioned in Section 4.6.2, graphs are helpful in representing level non-linear structures clearly and naturally. They are flexible in the way they are created by adjusting the rules to one's needs.

5.4.2 Rules

The hardest part of graph grammars is to use the right rules in order to create the game levels as desired. Since not all rules are equally important, some should be performed more often than others, thus all rules have a probability value associated with them. The higher the number, the more

likely it will get chosen over others. Two rules can have the same probability value.

5.5 Methods

5.5.1 Graph Replacement

Checking for a match of an LHS of a rule and a target graph is not as trivial as finding a match in string grammars (see Sec. 4.6). Since graphs are non-linear, in no particular order and in this project also context-sensitive, an algorithm had to be developed.

There are two categories on how the rules should be applied: the *algebraic* and the *algorithmic* approach.

Algebraic Approach

The algebraic approach was invented by Ehrig et al. [8] in the early 1970s. For the system to know which nodes of the LHS and RHS belong together *morphism* is used, which gives a relation between nodes from both sides. Every node that exists on the LHS but not on the RHS will be removed from the graph, and every node that appears in the RHS but not in the LHS will be added to the graph, while the nodes that have a match stay. This technique is used to *glue* the nodes to the graph, since all new nodes that should be added to the graph are automatically connected to it by the ones that remain. The algebraic approach is again divided into more sub-approaches, with the two most common ones being the double-pushout (DPO) and the single-pushout (SPO) approaches [1]. When a node in a graph gets deleted, the edges that were connected to that node will end up pointing to nothing instead. These edges are called *dangling edges*. The DPO approach specifies that when a node gets deleted, all edges connected to that node must also be deleted. The SPO approach on the other hand is more simple, accepting the deletion of any nodes since dangling edges will be deleted automatically.

Algorithmic approach

The algorithmic approach has many mechanisms for how to add new graphs. In this work, we will investigate two: the node-label controlled (NLC) and the edge-label controlled (ELC) mechanisms.

In the NLC approach, new nodes get connected to the graph with node labels with the LHS only having one single node called the *mother node*. When a match is found, all edges that connect the nodes to the mother node get deleted. Those nodes are called *neighborhood nodes*. When the graph from the RHS is added to the graph, new edges have to be added to

connect this new graph to the target graph. For this to happen, connection instructions are provided, which are ordered pairs. These however can only connect to neighbors of the mother node. For example, if an instruction is *(a,b)*, then each node with the label 'a' of the graph, that was a neighbor of the mother node will be connected to each new node with the label 'b'. A disadvantage is that all rules use the same set of connection instructions [1].

The ELC model was presented by Main and Rozenberg [14], and is influenced by the NLC approach. Instead of a mother node there is a *mother edge*. All edges connected to the source and target node of the mother edge are called *neighborhood edges*. When a match is found, the mother edge, the source and target node and all neighborhood edges are deleted from a graph. Now the RHS of the rule, which is called the *daughter graph*, is added, which is not yet connected to the graph. Like in the NLC approach, there are connection instructions which inform how new edges are introduced between the target graph and the daughter graph.

5.5.2 Used Method

Since the algorithmic approach needs connection instructions, which make them less controllable, the algebraic approach is chosen for implementation in this project. Because of that, context-sensitive matching is used, even though it is more computationally intensive. Also for applying the rules, the single-pushout method will be used, as taking care of dangling edges by just removing them appears to be sufficient for the task. Since the order of the game is important with having pre- and postconditions, directed edges are used. Loops are not allowed, since an event cannot target itself. This game has two kinds of events, one being a simple event and one a complex event. Then there is the start, that does not have a precondition and the last node, which ends the game. Additionally, a goal node can have a maximum of two edges connected to it, since there is just one alternative postcondition list.

To start off, a match has to be found. The flowchart diagram in Fig. 5.1 shows how to find a match from a subgraph of the RHS in the graph. It also shows the case for when there is only one node on the LHS, which can be ignored for our project, since all rules consist of graphs. The process works as follows:

1. Find the first node from the pattern in the graph.
   (a) If not found, then stop.
   (b) Else if the first node is found, find the second node from the pattern in the graph.
      i. If not found, then stop.
      ii. Else if the second node is found, check the edge between those two nodes.

A. If edge not found, go to (b).
B. Else if edge is found and match is complete, return indices.
C. Else if edge is found and match is not complete go to (b), using the second node here as the first node in (b).

To get a better understanding of the process, the steps to apply a rule are described. For this example, the rule from Fig. 5.2 will be used on the graph from Fig. 5.3(a). All steps are labeled, and thus can be referred to in the same Figure.

1. Choose a rule and match the graph from the LHS with the nodes from the target graph. If found, the selected nodes for replacement will be labeled according to the LHS of the rule (Fig. 5.3(b)).
2. Remove all edges between the selected nodes (Fig. 5.3(c)).
3. Replace the labeled nodes with their equivalent nodes on the RHS of the rule (Fig. 5.3(d)).
4. Add any nodes to the graph that exist on the RHS, but do not have an equivalent on the LHS (Fig. 5.3(e)).
5. Add the edges as specified by the RHS of the rule (Fig. 5.3(f)).
6. Finally, remove the labels that were added in the first step (Fig. 5.3(g)).

5.6 Converting the Graph to a Game

When applying this algorithm and executing it graphs are built, with one example shown in Fig. 5.4. The start and end node can be seen, which mark the beginning and end of the game. In between are several simple event nodes and one complex event node. The complex event splits into four paths, with two of them leading to the end goal and two to a dead-end. Now that the graph generation works, it has to be converted into a playable game file. For this a converter class is made that checks the type of node and its connection to other nodes, and thereby gets the information that is needed. To make it simple at first, every node has a postcondition of being visited, besides the end node and possible dead-ends which lead to the end of the game. Every source node from an edge is a precondition to the target node of that same edge. When the file is created, it can get loaded already by the editor. However, besides the connection of the events to each other, every field is empty that would turn the game into a real game. Of course, theoretically that would be enough for the users to use it and start filling in the gaps. But that is not enough to convince a user to use that generated game instead of just making their own from scratch, since it provides no real benefit. They still have to get the information of how the structure of the game is set, getting the numbers of events that need to be done until the end is reached

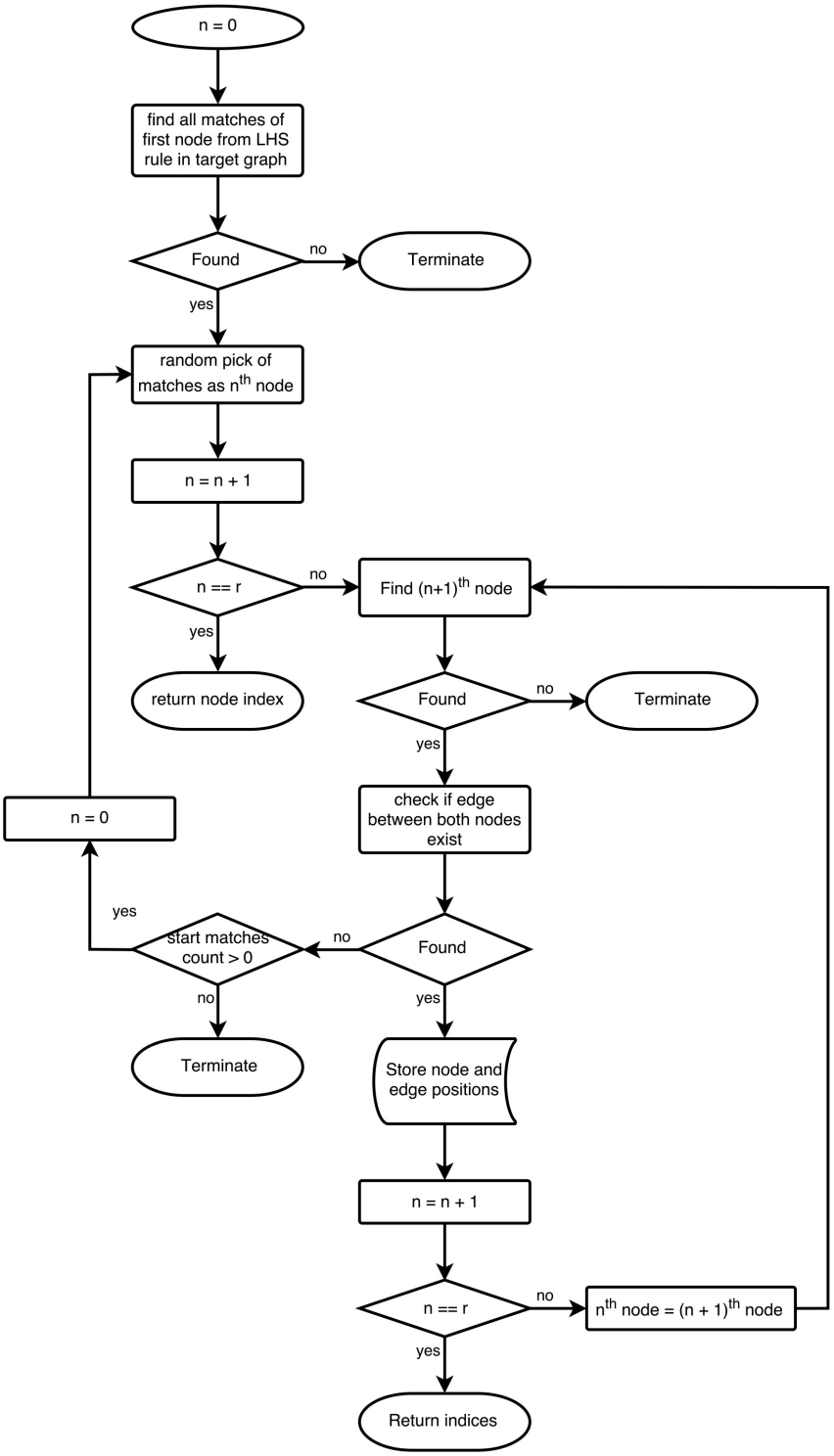


Figure 5.1: Diagram of the graph matching algorithm.

Figure 5.2: A representation of a context-sensitive graph rule.

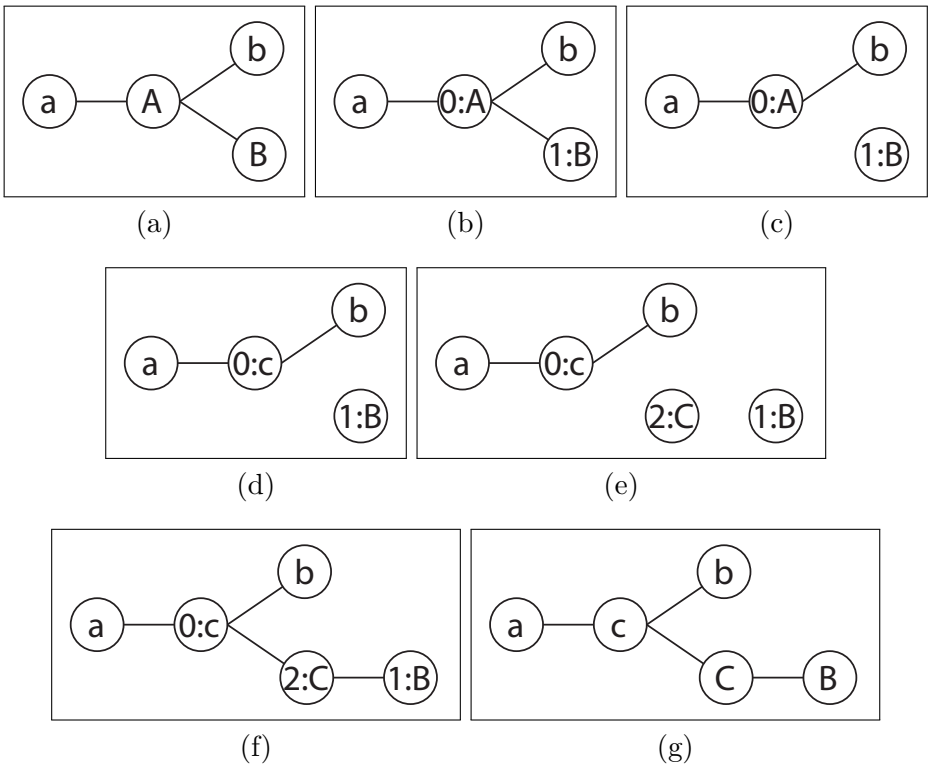


Figure 5.3: The process of graph replacement by using the rule from Fig. 5.2 on the graph seen in (a) is shown from (b) to (g). The graph in (g) is the final result.

and so on. This is even a drawback, since it gives the user limitations on how long the game story should take. So it is not enough to just create the game structure, but it is also necessary to enter some helpful information into the empty fields to support the user being creative in making a game.

To let the users have more say in how the levels are created, the option of difficulty was discussed. There are two ways to do this; by creating difficulty options like easy, medium and hard, or the alternative of letting the user fill in the parameters. The latter would not make that much sense, since giving an exact number of how long the game should be makes it seem like the user already has an idea of a game, which would make this automatic generation

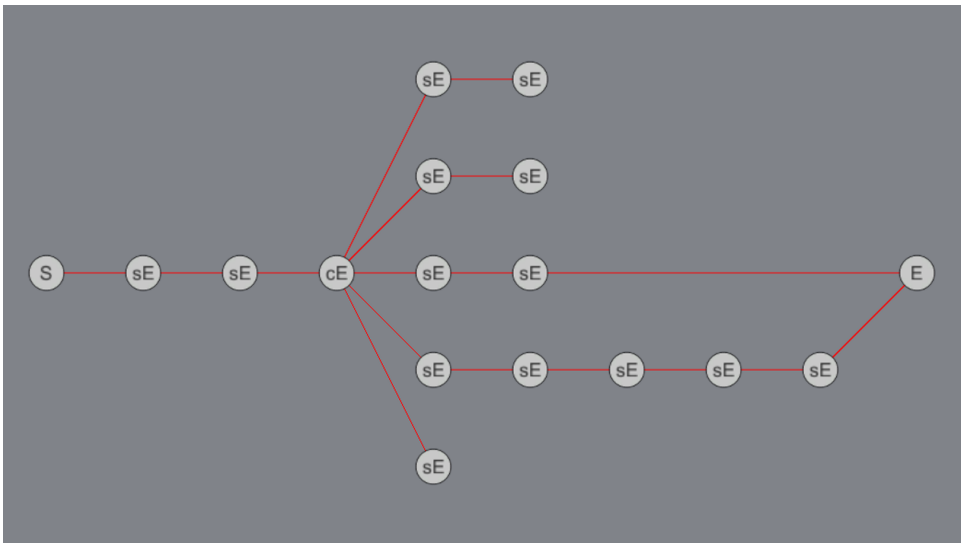


Figure 5.4: A randomly generated graph in the game editor. The letter "S" stands for "Start Node", "sE" for "simple Event", "cE" for "complex Event" and "E" for "End Node".

useless. The first idea with the difficulty modes appears to be more fitting by giving the user freedom to some degree. It was decided that the easy mode would create a maximum of 10 events and 1 complex event, medium up to 15 and 2 or 3 complex events and hard with up to 20 events and 4 complex events. The harder the setting, the furthermore the locations can be from each other.

5.6.1 Filling the Gaps

The easiest way to fill in the gaps would be to use random places, people and items. A creative person might even end up creating a story around it, but that seems rather far-fetched. A better idea came up when thinking about the creation of the difficulty setting. It is not enough to just depend on the size of the events. The school building is 4 floors high, so having the distance between the doors as a factor for difficulty seems sensible. The closer the rooms are, the easier it is to finish the game. Taking this into account, the starting room can actually be chosen randomly and starting from that origin, other rooms can be randomly set by considering the distance value. Then, according to the locations, people and items are chosen that seem to fit to those places. To make this possible, a database must be set, with each room having a set of most possible people and items. An example would be the IT classroom. It will be checked regarding which people are most likely to be there, which is the teaching staff and with the most expected one being

the IT teacher; then, with a smaller likelihood the principal, the janitor, secretary and so on. The same goes for the items, which in this example could be some technical parts, e.g., laptops, phones, hardware components and so forth. Creating a database like that is a tedious work, but once it is set, it gives suitable results. The only thing left for the user to do is to come up with a story and enter that into the dialog field.

5.6.2 Describing the Game

Now that the whole game is set and ready, the whole thing must be conveyed to the user so he or she can get a notion of the structure. To do this, the user first gets a graph displayed when a game is created automatically, like in Fig. 5.5. With this, the visual impaired users can learn visually how the game is constructed in one view. The blind have the speech assistant to tell them about the main aspects, like the size, the amount of branches and how many ways there are to reach the goal. Then the start node gets selected with the including data, that is the location, the person and the item of the event. With the *Enter* key the user jumps to the next node with the new information. When the player has the overall information, he or she can start thinking of a story that might fit the given suggestions. That is not completely necessary, however, since the user can edit any part of each event at anytime. When the editing starts, the player gets confronted with the information similar to when the node was selected in the graph. Any information can be skipped or turned off at any time.

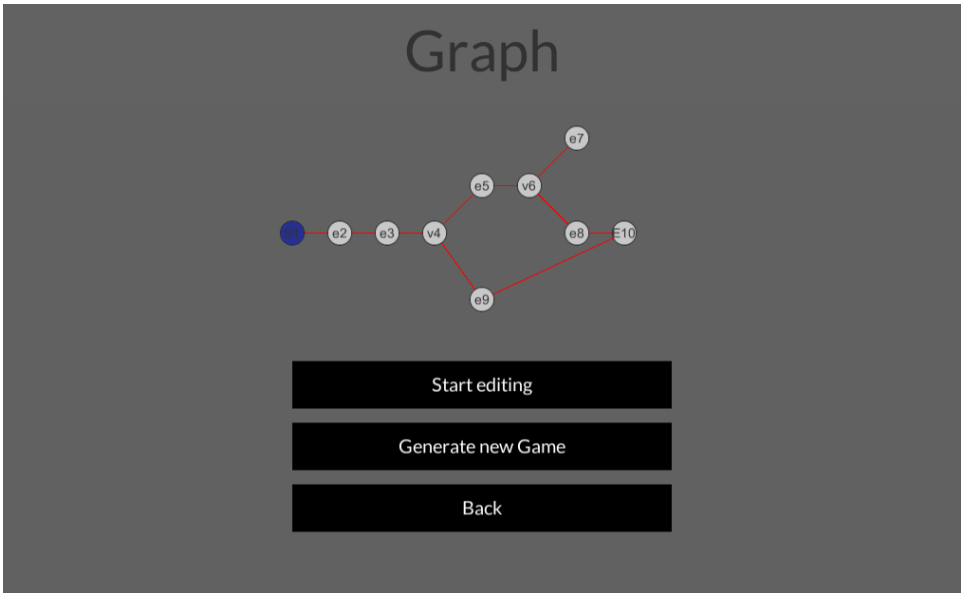


Figure 5.5: The graph window in the game editor with the buttons “Start editing”, “Generate new Game” and “Back”.

Chapter 6

Evaluation

The goal of the evaluation is to find out whether the users are willing to use the assistant with the automated creation of levels, and if it is beneficial to increase the amount of games that can be played later. The method used to find out this information was by conducting a test where the users can use the assistant and give feedback at the end. Additionally, they had to answer an online questionnaire. The evaluation system used a Likert scale, which measures the degree of agreement with predetermined statements.

6.1 Participants

Nine students (3 males and 6 females) from the BBI participated in the study voluntarily. Their age ranged from 15 to 23 years (M = 18, SD = 2.28). The school language of the participants is German; therefore, the whole survey was conducted in German. However, a translation of the instructions is also present in the appendix.

First, the participants were questioned about their usage method when using a PC, where 55.56% specified that they solely depend on screen reader and/or refreshable braille displays. Another 11.11% use the display visually and a screen reader together, resulting in two third of the participants needing alternative methods to receive information from a PC.

When they were asked about their user behavior, 88.89% reported that they use a PC daily for school. However, only 33.33% stated that they use a PC to play games up to 3 times a week, while the rest practically never play PC games.

6.2 Task

The task was to let the assistant of the BlindBits editor create a game for them. Then the participants had to listen to the screen reader that explains

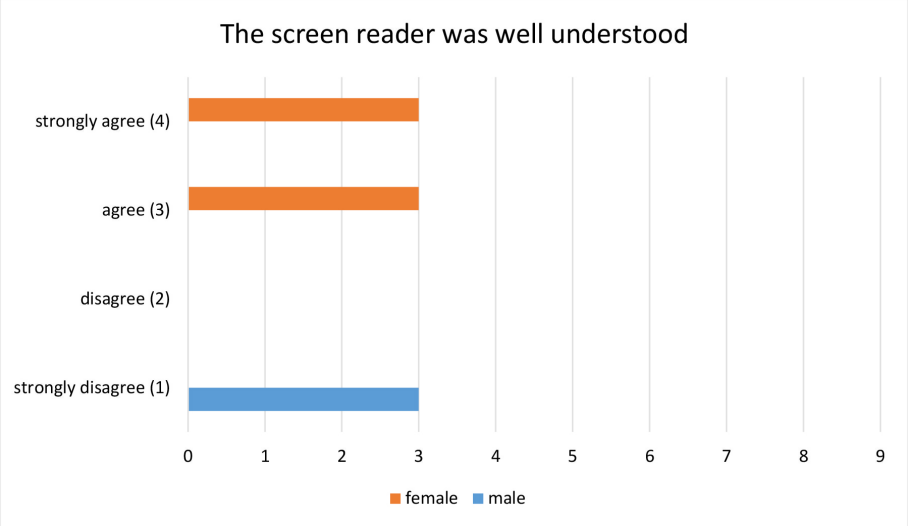


Figure 6.1: Evaluation of the comprehensibility of the screen reader.

the structure of the game with all its contents. When they had a notion of the whole setup, they could start editing the levels by filling in the dialog options to complete the game. After they finished, an online survey had to be completed.

6.3 Data Results

The assessment of these statements was made on a four-point scale consisting of the items “strongly disagree”, “disagree”, “agree” and “strongly agree”. The number of stages was chosen to be even, to avoid a neutral central position and thereby to derive clearer trends from the responses of the participants. Seven statements were given that had to be graded, and which will be discussed in the following subsections. At the end, the users had a chance to give some feedback about the whole project.

6.3.1 Comprehensibility of the Screen Reader

Since most users are dependent on alternatives to visual aids, it is important that the screen reader is well understood. The statement was: “The screen reader is understandable”. The results can be seen in Fig. 6.1.

6.3.2 Explanation of the Game Structure

It is important to know if the participants feel that the game structure is well explained. Otherwise it would be hard to nearly impossible to create

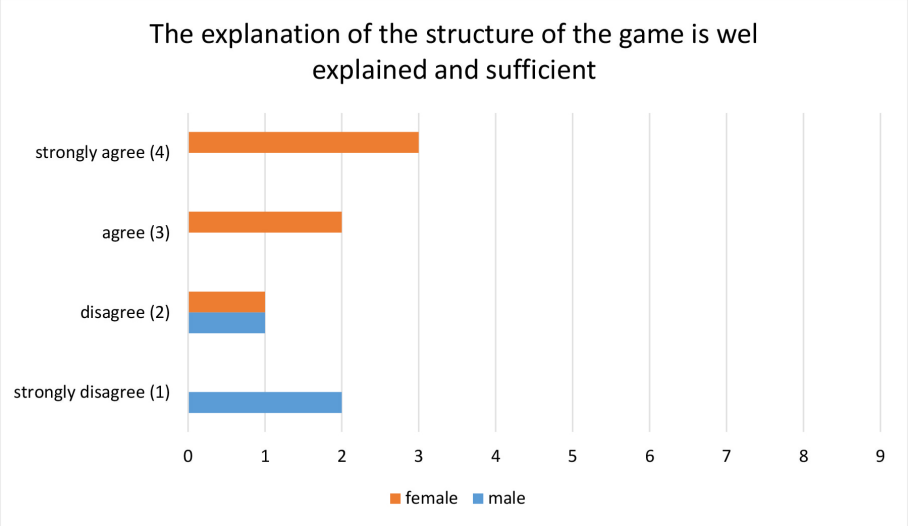


Figure 6.2: Evaluation of the explanation of the game structure.

sufficient games with the assistant, thus removing its purpose. The statement was: "The explanation of the structure of the game is well explained and sufficient". The results can be seen in Fig. 6.2.

6.3.3 Mental Representation of the Game

Besides having a good explanation of the game, the user should also have a mental representation of the game. The statement was: "I could imagine the structure of the game well". The results can be seen in Fig. 6.3.

6.3.4 Navigation

In order to get a good impression of the game, the user has to repeatedly go through the events. Therefore, a good navigation system is needed to reach that requirement. The statement was: "Navigating through the events is simple and understandable". The results can be seen in Fig. 6.4.

6.3.5 Overview of the Game

To determine if the users had any difficulties with the editor at any time presents any flaws in the UI design. The statement was: "I knew what to do at all times". The results can be seen in Fig. 6.5.

6.3.6 Usefulness of the Assistant

The whole purpose of the assistant is to push more games out for the Blind-Bits app, and hence why it is important for the users to see its usefulness

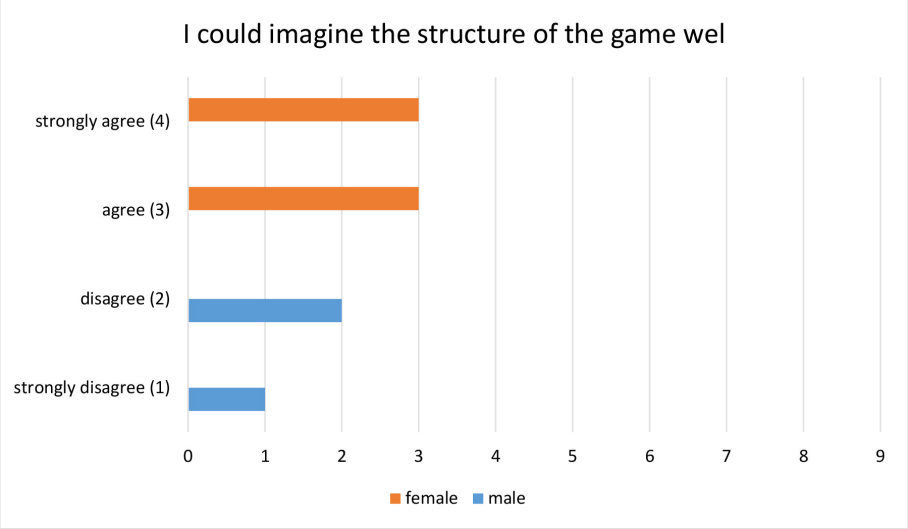


Figure 6.3: Evaluation of the mental representation of the game.

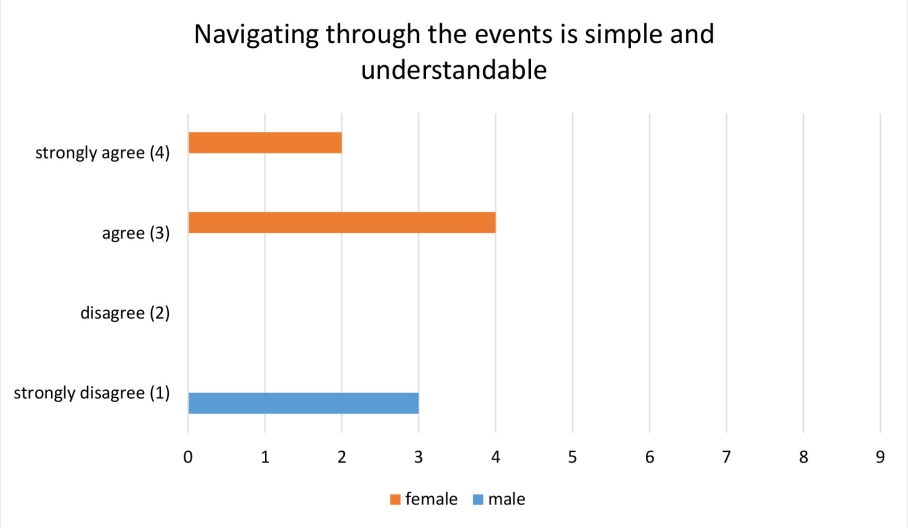


Figure 6.4: Evaluation of the navigation.

and to actually use it. The statement was: "I can imagine using the assistant to create games". The results can be seen in Fig. 6.6.

6.3.7 Speed of the Assistant

Even if the assistant works as desired, it is not adequate as a factor alone. It must also be faster than a manual creation of a game, otherwise it will

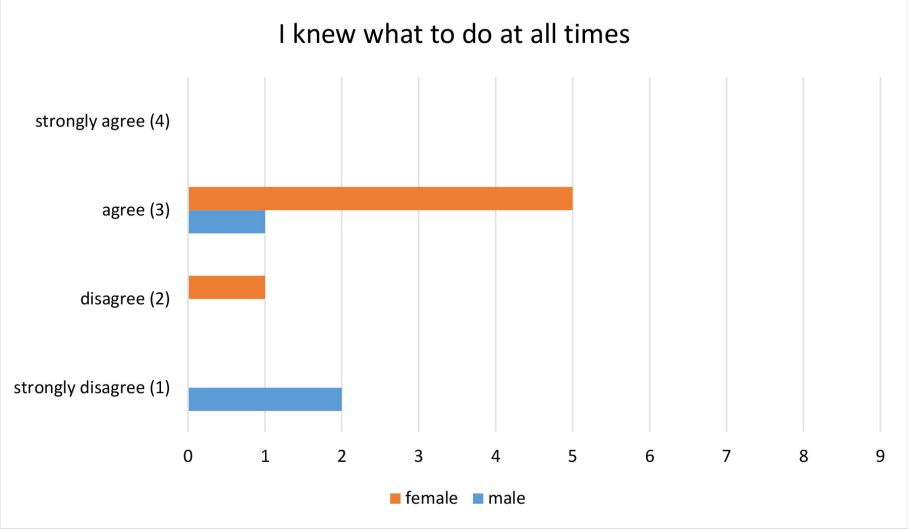


Figure 6.5: Evaluation of having an overview at all times.

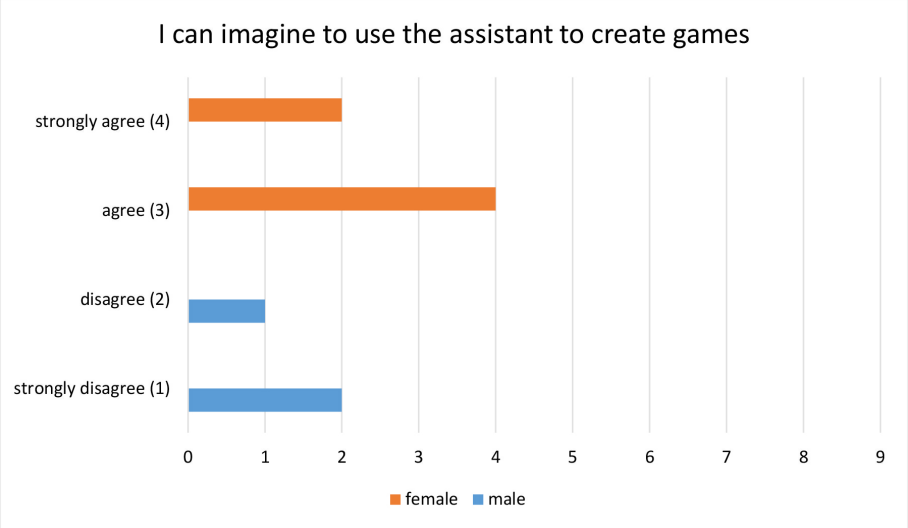


Figure 6.6: Evaluation of the usefulness of the assistant.

not be a motivation to use it. The statement was: "I can imagine that I can create games much faster with the assistant". The results can be seen in Fig. 6.7.

6.3.8 User Experience

At the end, the users had four open questions they could fill in voluntarily.

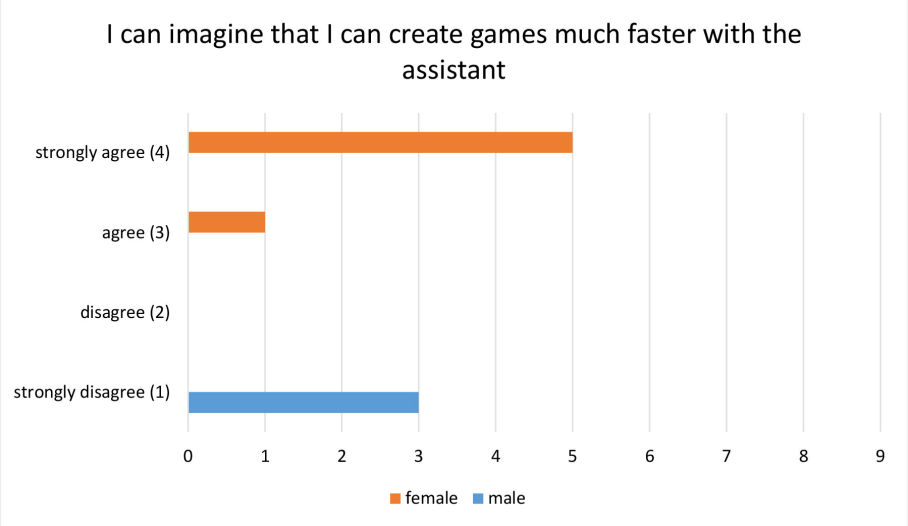


Figure 6.7: Evaluation of the speed of the assistant.

Suggestions for the improvement for the navigation

Seven participants gave their feedback on this matter. One suggestion was to be able to use the mouse to select an event in the graph and edit it from there. Another was to reverse the navigation with the arrow keys, while for others it was good and sufficient, which means an option in the settings to choose their navigation style could be helpful.

Suggestions for improvement to describe the game structure

Three people gave their sentiment. One was more a request than a suggestion, by commenting that it should be described better. A different one was that the structure was okay.

Suggestions for improvement for the automated game creation

Four users had suggestions for this. The first one was that the game should have more events. This was planned anyway, but for the sake of the time limit it was chosen that a game should have a maximum of ten events. Two proposals were about adding information to get the location of the next event when editing, the purpose being to be able to consider the place to guide the player properly in the dialog.

Any feedback

If the participants had any additional feedback or criticism that did not fit into the other questions, they could include it here. Three participants

had something to say. One was that the experience using the editor had significantly improved since using it for the last workshop. A second opinion was to have more possibilities in the game. Unfortunately, this is too vague and therefore could mean a lot of things. It is assumed that it was meant having more ways to edit an event. The third idea was about adding sound effects to events, e.g., hearing a water flow. This is likely to be implemented in the future.

Chapter 7

Conclusion

7.1 Summary

This work presented the BlindBits project with the editor and game, and with the addition of an assistant to accelerate the production of games that are playable for the blind. First, the purpose of O&M training was explained, with several examples of digital games that incorporated some training techniques to teach blind people skills in a playful way. Then the project was explained with its purpose, design method and game mechanics. Next, a short insight into PCG was made with some methods explained and examples in the game industry that use PCG algorithms.

7.2 Result

The editor has been proven in the workshops to be a tool that works well in creating games. However, it was a tedious endeavor for some, and thus making a tool that generates games was virtually essential. Therefore, the assistant was developed, and the results have shown that the approach was right and that students can actually gain from this method to create more games in a much faster fashion. The female participants mostly agreed or strongly agreed to the statements, except in two cases, where one girl disagreed respectively. However, the male participants disagreed or strongly disagreed to all the statements. It is assumed that the negative outcome from the male participants was a result of misunderstanding. The received feedback on the survey and during the sessions does not fit the data gathered through the Likert scales. Furthermore, during the survey some students asked whether option 1 or 4 was the best. This might be due to the Austrian grading system, where 1 is the highest grade.

7.3 Outlook

The next step is to implement the 3D world into the editor and evaluate it with the target group. Two different control systems have to be considered; one for the blind and another for the visually-impaired. The idea is that the visually-impaired students can move freely in a continuous manner, while the blind have to follow a rail-like approach to prevent them from hitting walls or undesired places. Additionally, we gathered more ideas for future extensions that should be addressed in later iterations. A required extension of the current design is related to familiarizing first-time users. A tutorial for the game editor is needed that allows users to get to know the available functionalities. The tutorial should involve step-by-step instructions for creating a small game where all the necessary elements are used at least once. For the 3D world, a specific path is prearranged where the sound cues are introduced one after another. To keep the motivation and replay value of the game(s) high, meaningful rewards should be added to the game concept. Thus, to make long-term achievements visible and allow competition with fellow pupils, the introduction of a scoreboard or reputation levels needs to be considered. Pupils could, for example, score points by completing games, depending on the difficulty of a game. Finally, another idea is to provide a third UI of the editor, by making it available on smartphones. Most visually-impaired pupils use their smartphones regularly and are familiar with making text input. With a mobile version, pupils could create games while navigating in the real school. Input would be handled by swiping and tapping. For TTS output, the native TTS engine would be used.

Appendix A

CD-ROM Content

A.1 Thesis

Pfad: /

Said_Areen_2016.pdf .	Master's thesis as PDF file.

A.2 Study Results

Pfad: /study results

questionnaire.pdf	The questionnaire template (German).
questionnaire_english.pdf	The questionnaire template (English).
questionnaire_results.pdf	Results from questionnaires completed by study participants (German).

A.3 Project

Pfad: /project

/executable	Contains the *.apk file and the *.exe file with its data needed to run the prototype.
/sourcecode	Contains the source code of the Unity projects.

A.4 Online Literature

Pfad: /literature

Sentient Sketchbook.pdf	A copy of the website for the Sentient Sketchbook.

The Verge - Minecraft.pdf	A copy of the website of the “Verge” article about the Minecraft sales.
The Word of Notch.pdf	A copy of the blog from Notch.
Unity Doc.pdf	A copy of the website from the Unity Doc.
Wired - No Mans Sky.pdf	A copy of the website of the “Wired” article about No Man’s Sky.

References

Literature

[1] David Adams. “Automatic Generation of Dungeons for Computer Games”. Bachelor’s thesis. University of Sheffield, Dept. of Computer Science, May 2002 (cit. on pp. 32, 33).

[2] Kevin Allain et al. “An audio game for training navigation skills of blind children”. In: *Proceedings of the 2nd VR Workshop on Sonic Interactions for Virtual Environments (SIVE)*. (Arles). IEEE, Mar. 2015, pp. 49–52 (cit. on p. 8).

[3] James F. Allen, Curry I. Guinn, and Eric Horvitz. “Mixed-initiative interaction”. *Intelligent Systems and their Applications, IEEE* 14.5 (Sept. 1999), pp. 14–23 (cit. on p. 16).

[4] Eric Butler et al. “A Mixed-initiative Tool for Designing Level Progressions in Games”. In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST ’13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 377–386 (cit. on p. 14).

[5] Noam Chomsky. “Three models for the description of language”. *IRE Transactions on Information Theory* 2.3 (1956), pp. 113–124 (cit. on p. 18).

[6] Erin C. Connors et al. “Development of an Audio-based Virtual Gaming Environment to Assist with Navigation Skills in the Blind”. *Journal of Visualized Experiments : JoVE* 73 (Mar. 2013) (cit. on p. 5).

[7] Joris Dormans. “Adventures in level design: generating missions and spaces for action adventure games”. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM. 2010, p. 1 (cit. on pp. 19, 20).

[8] Hartmut Ehrig, Michael Pfender, and Hans Jürgen Schneider. “Graph-grammars: An algebraic approach”. In: *1973. SWAT’08. IEEE Conference Record of 14th Annual Symposium on Switching and Automata Theory*. IEEE. 1973, pp. 167–180 (cit. on p. 32).

[9] David Eibensteiner. “Prozedurale Generierung von endlosen Landschaften mit softwarebasierten Agenten”. German. MA thesis. Hagenberg, Austria: Interactive Media; FH Oberösterreich, Fakultät für Informatik, Kommunikation und Medien, 2015 (cit. on p. 17).

[10] George Kelly and Hugh Mccabe. “A Survey of Procedural Techniques for City Generation”. *ITB Journal (Institute of Technology Blanchardstown)* (14 Dec. 2006). Ed. by Dr. Brian Nolan, pp. 87–130 (cit. on p. 14).

[11] Antonios Liapis, Gillian Smith, and Noor Shaker. “Mixed-initiative content creation”. In: *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Ed. by Noor Shaker, Julian Togelius, and Mark J. Nelson. Springer, 2015. Chap. 11, pp. 195–214 (cit. on pp. 16, 24).

[12] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. “Sentient Sketchbook: Computer-Aided Game Level Authoring”. In: *Proceedings of the 8th Conference on the Foundations of Digital Games*. Society for the Advancement of the Science of Digital Games, 2013, pp. 213–220 (cit. on pp. 16, 24).

[13] Todd Lubart. “How Can Computers Be Partners in the Creative Process: Classification and Commentary on the Special Issue”. *International Journal of Human-Computer Studies – Special issue: Computer Support for Creativity* 63.4-5 (Oct. 2005), pp. 365–369 (cit. on p. 16).

[14] Michael G Main and Grzegorz Rozenberg. “Edge-label controlled graph grammars”. *Journal of Computer and System Sciences* 40.2 (1990), pp. 188–228 (cit. on p. 33).

[15] Lotfi B. Merabet et al. “Teaching the Blind to Find Their Way by Playing Video Games”. *Public Library of Science ONE* 7.9 (Sept. 2012). Ed. by Hugo Theoret, pp. 1–5 (cit. on p. 4).

[16] J. Sánchez, M. Espinoza, and J. Garrido. “Videogaming for wayfinding skills in children who are blind”. In: *Proceedings of the 9th International Conference on Disability, Virtual Reality and Associated Technologies*. (Laval). Ed. by Paul Sharkey and Evelyne Klinger. Reading: The University of Reading, Sept. 2012, pp. 131–140 (cit. on p. 3).

[17] J. Sánchez and J. P. Rodríguez. “Videogame for improving orientation and mobility in blind children”. In: *Proceedings of the 8th International Conference on Disability, Virtual Reality and Associated Technologies*. (Viña del Mar / Valparaíso). Ed. by Paul Sharkey and Jaime Sánchez. Reading: The University of Reading, Sept. 2010, pp. 299–303 (cit. on p. 3).

[18] Jaime Sánchez and Miguel Elías. "Guidelines for Designing Mobility and Orientation Software for Blind Children". In: *Proceedings of the 11th IFIP TC 13 International Conference on Human-Computer Interaction*. Ed. by Cécilia Baranauskas et al. INTERACT 2007. Rio de Janeiro, Brazil: Springer-Verlag Berlin, Heidelberg, 2007, pp. 375–388 (cit. on p. 3).

[19] Jaime Sánchez and Matías Espinoza. "Audio Haptic Videogaming for Navigation Skills in Learners Who Are Blind". In: *Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*. (Dundee). ASSETS '11. New York, NY: ACM, Oct. 2011, pp. 227–228 (cit. on p. 4).

[20] Jaime Sánchez, Mauricio Sáenz, and Miguel Ripoll. "Usability of a Multimodal Videogame to Improve Navigation Skills for Blind Children". In: *Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility*. Assets '09. Pittsburgh, Pennsylvania, USA: ACM, 2009, pp. 35–42 (cit. on p. 3).

[21] Jaime Sánchez et al. "A Model to Develop Videogames for Orientation and Mobility". In: *12th International Conference on Computers Helping People with Special Needs (ICCHP), July14-16, 2010, Proceedings, Part II*. Ed. by Klaus Miesenberger et al. ICCHP. Vienna, Austria: Springer-Verlag Berlin, Heidelberg, 2010, pp. 296–303 (cit. on p. 10).

[22] Gillian Smith, Jim Whitehead, and Michael Mateas. "Tanagra: Reactive Planning and Constraint Solving for Mixed-Initiative Level Design". *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (Sept. 2011), pp. 201–215 (cit. on p. 24).

[23] Grace P. Soong, Jan E. Lovie-Kitchin, and Brian Brown. "Does mobility performance of visually impaired adults improve immediately after orientation and mobility training?" *Optometry and Vision Science* 78.9 (2001), pp. 657–666 (cit. on p. 3).

[24] George Stiny et al. "Introduction to shape and shape grammars". *Environment and Planning B* 7.3 (Nov. 1980), pp. 343–351 (cit. on p. 20).

[25] George Stiny and James Gips. "Shape Grammars and the Generative Specification of Painting and Sculpture". In: *International Federation for Information Processing Congress 71*. (Ljubljana). Ed. by C. V. Freiman. Vol. 2. Amsterdam, 1971, pp. 1460–1465 (cit. on p. 20).

[26] Julian Togelius, Noor Shaker, and Joris Dromans. "Grammars and L-systems with applications to vegetation and levels". In: *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016. Chap. 5, pp. 73–98 (cit. on pp. 19, 20).

[27] Julian Togelius et al. “Search-based Procedural Content Generation”. In: *Applications of Evolutionary Computation*. Vol. 6024. Lecture Notes in Computer Science. Springer, 2010, pp. 141–150 (cit. on p. 14).

[28] Julian Togelius et al. “What is Procedural Content Generation? Mario on the Borderline”. In: *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. PCGames ’11. Bordeaux, France: ACM, 2011, 3:1–3:6 (cit. on p. 14).

[29] Nikolaos Vlavianos, Stavros Vassos, and Takehiko Nagakura. “Towards a novel method for Architectural Design through µ-Concepts and Computational Intelligence”. In: *Proceedings of the 1st Workshop on Artificial Intelligence and Design*. (Ferrara). Ed. by Francesca Alessandra Lisi and Stefano Borgo. Sept. 2015, pp. 55–60 (cit. on p. 16).

[30] Georgios N. Yannakakis and Julian Togelius. “Experience-Driven Procedural Content Generation”. *IEEE Transactions on Affective Computing* 2.3 (July 2011), pp. 147–161 (cit. on p. 14).

Films and audio-visual media

[34] Firaxis. *Sid Meier’s Civilization IV*. Game [Windows, Mac OS]. Oct. 2005. URL: http://www.2kgames.com/civ4/home.htm (cit. on pp. 14, 15).

[35] Hello Games. *No Man’s Sky*. PlayStation 4, Microsoft Windows. Aug. 2016. URL: http://www.no-mans-sky.com/ (cit. on p. 22).

[31] Markus Persson. *Minecraft*. Game [Windows]. May 2009. URL: https://minecraft.net/ (cit. on pp. 14, 21).

[32] Bethesda Game Studios. *The Elder Scrolls: Oblivion*. Game [Windows, PlayStation 3, Xbox 360]. Mar. 2006. URL: http://www.elderscrolls.com/oblivion (cit. on p. 15).

[33] Valve. *Counter-Strike*. Game [Windows, Xbox, Mac OS]. Nov. 2000. URL: http://store.steampowered.com/app/10/ (cit. on p. 17).

Online sources

[36] Chris Higgins. *No Man’s Sky would take 5 billion years to explore*. English. Ed. by Wired. Aug. 2014. URL: http://www.wired.co.uk/article/no-mans-sky-planets (visited on 09/11/2016) (cit. on p. 23).

[37] Antonios Liapis. *Sentient Sketchbook: Computer-Assisted Game Level Authoring*. English. 2013. URL: http://www.sentientsketchbook.com/ (visited on 08/20/2016) (cit. on p. 24).

[38] Markus Persson. *Terrain generation, Part 1*. Mar. 2011. URL: http://notch.tumblr.com/post/3746989361/terrain-generation-part-1 (visited on 09/17/2016) (cit. on p. 21).

[39] Unity Technologies. *Building Plugins for Android*. Aug. 2016. URL: https://docs.unity3d.com/Manual/PluginsForAndroid.html (visited on 08/17/2016) (cit. on p. 30).

[40] Tom Warren. *Minecraft sales top 100 million*. English. Ed. by The Verge. June 2016. URL: http://www.theverge.com/2016/6/2/11838036/minecraft-sales-100-million (visited on 09/12/2016) (cit. on p. 21).