EPFL

CHILI
computer-human interaction
in learning and instruction

LEARN
Center for Learning Sciences

LogiCraft:
An Educational Game In Minecraft

Author:
Flaminia Trinca

Supervisor(s):
David Strütt

Professor:
Pierre Dillenbourg

January 7, 2022

Contents

1 Introduction **1**

2 Literature Review **1**

3 Structure of the Game **3**
3.1 Narrative . . . 3
3.2 Tutorial . . . 4
3.3 The Levels . . . 5
3.3.1 Level 1 . . . 6
3.3.2 Level 2 . . . 6
3.3.3 Level 3 . . . 6

4 Material and Methods **7**
4.1 Sources of Knowledge and Tools . . . 7
4.1.1 WorldEdit . . . 7
4.1.2 Redstone . . . 7
4.1.3 Commands and Data Packs . . . 7
4.1.4 Structure Blocks . . . 8
4.1.5 Resource Packs . . . 8
4.2 Issues encountered . . . 8
4.3 Decisions and Operating Mode . . . 10
4.3.1 Narrative . . . 10
4.3.2 Layout of the Game . . . 11
4.3.3 The Gates . . . 11
4.3.4 The Cables . . . 12
4.3.5 Winning Condition and the Levels . . . 13
4.4 Testing . . . 13

5 Results **13**

6 Conclusion **16**
6.1 Testing and Results . . . 16
6.2 Improvements and Next Steps . . . 16
6.3 Personal Experience . . . 17

1 Introduction

LogiCraft is an educational game in Minecraft which introduces the notion of logic gates to children. Being something that hides behind every piece of computer and electric engineering, logic gates form a great introductory tool for children to the world of engineering. Additionally, their mode of operation is based on Boolean algebra - a fundamental mathematical concept, which is usually not taught in primary and secondary schools in Switzerland.

The goal of this project is to create a video game, which presents and teaches maths in a fun and interactive way and which can be taught in the context of an outreach program. The idea behind it being to do something fun with students, all whilst educating them on an essential mathematical concept. Furthermore, this implies that the game is not supposed to be self-explanatory. Because of the limited time of the project, we unfortunately did not get the chance to try out the game with a school class in the scope of this project. However, in a next step course instructions can be written which will accompany the students before and while playing the game.

The main reason as to why I based my game on the notion of logic gates is the fact that solving a circuit is somewhat like solving a riddle or a puzzle. A circuit is a combination of different logic gates. Once the truth tables of these different logic gates are known, solving a circuit is simply a question of combining the information that has already been established beforehand. It is therefore technically possible to solve any problem without the need of additional information, and in the worst case by trying out the different results. That is why, in my opinion, logic gates provide a good tool to introduce the concept of Boolean algebra.

The reason as to why I chose to create the game in Minecraft and not with another user interface is the fact that Minecraft itself already adds different levels of interactivity and most importantly the factor of fun to the game for players of all ages. Games are generally attractive for teaching, simply because of the fact that they are mostly fun to play. The reason why Minecraft is even more so, is the fact that it is also an environment which a lot of children are already familiar with. Furthermore, it allows us to create interactive games in it.

The final game consists of 10 different puzzle-like quizzes, which come in three different levels. It also includes an introductory tutorial level, and the entirety of the game is put into frame by a narrative which gives the player a goal and a reason to finish the game. The game is not complete at this stage. There are still details missing and improvements to be done. This is a first draft of the final game which can, in a later step, be used in outreach program to teach logic gates to pupils.

2 Literature Review

Today, the demand for variety in the classroom and in teaching methods is getting bigger and bigger [22, 17]. “The use of instruction media with multimedia element in the L & T (learning and teaching) process is a new approach of replacing the current traditional

method by using power point slides without any pedagogical element." [17] That is why, researchers have started studying the impact of video games as educational tools and have started to highlight some of the advantages and drawbacks of introducing media-based learning into modern classrooms [3].

Minecraft has been a successful video game for over a decade now. In fact, according to Latić [27] it has been the second best-selling game in the world, right after the famous console game Tetris [27]. Minecraft is a Java based first person multiplayer sandbox game [11], which has been released in 2011 by the Swedish company Mojang Studios [27]. The central property of a so-called sandbox game is the fact that there is no specific enemy to fight or goal to reach. The game is built out of pixel-like blocks which a player can place and break.

There are multiple studies that have analyzed and explored the educational potential of Minecraft, and other video games in general [3, 4, 7, 8, 11, 22]. Most of them list the same advantages and drawbacks of using video games in classrooms.

The main advantage of using video games in education is the fact that video games are fun to play.[2] Additionally, they generally stimulate imagination and creativity, they promote enthusiasm, and they deepen technology, communication, and effective collaboration skills.[22] Another advantage of video games is that the player is often rewarded extrinsically when they succeed a step or a level.

Furthermore, Minecraft itself as a video game offers many advantages. The blocks in the game can be rearranged to reproduce almost any static object or shape, which provides stimuli for a very different set of education and research projects [22]. The game is adaptable for many purposes due to its modifiability. Research has been done which shows that Minecraft can be used in endless educational fields, from biology to mathematics, physics, chemistry, and computer science, and even up to economy and sustainability [22, 11]. This is possible thanks to the different resource and Data Packs, mods and additions which can be added to the game [22].

Summing up, the main advantage of Minecraft are the almost endless possibilities of creation within the game. Players all over the world have started creating their own games and worlds in Minecraft and putting them online. An example of a complex game that has been created in Minecraft is *Terminacraft* [28], a recreation of the game *The Legend of Zelda: Majora's Mask*. Another example is the educational game *Matheminecraft* [18], which has been developed at EPFL in 2020. The goal of that game is to introduce the mathematical concept of Eulerian graphs to children.

Figure 1: Matheminecraft (left) and Terminacraft (right) [1]

Some of the drawbacks when working with video games are issues like the lack of focus and the tendency to distractions which is often associated with gameplay [22]. Specifically, when considering Minecraft as a tool for education, a drawback may be that unexperienced players in the classroom have more difficulties at the beginning, and thus are in disadvantage towards more experienced players [22]. However, this can be avoided with good descriptions and a guided instruction [22]. Another drawback might be that more experienced players may employ their skill to skip past the intended lesson [11], which is something that needs to be taken into consideration when designing a game in Minecraft. It is also essential to keep in mind that a scenario might not automatically be more fun just because it is done in Minecraft.

One of the most interesting technical components in the game is the so-called Redstone, which works somewhat like electricity in real life [10, 4]. With Redstone it is possible to implement structures like logic gates, because it follows the principles of circuitry [4]. Waraich [3] states that it is often difficult for pupils to understand the basic concepts of computer architecture, such as logic gates, because of the abstractness of the concept. It can therefore be beneficial to use narrative and gameplay as a motivational tool, to facilitate the comprehension of this rather abstract concept [3].

3 Structure of the Game

To optimize the effect of the game, a lot of thought has been put in the structure and narrative before starting to create the game. Several details have changed multiple times during the process, but the main idea has stayed similar to the one at the start of the process.

3.1 Narrative

Based on Waraich's [3] analysis, which has shown the importance of narrative in educational games, I decided to embed the game in a story, where the player has a main goal

[1] source matheminecraft: `https://www.epfl.ch/schools/sb/research/math/matheminecraft-3/`
source terminacraft: `https://www.facebook.com/majorascraft/photos/516075518416694`

from the beginning to the end, which is there to motivate them to finish the game on time.

The game starts in a forest, where the player learns that hundreds of animals have been held prisoners by a Minecraft-monster, which is planning to kill them, if the player can't save them on time. To get to the monster's evil farm, the player has to cross 10 different bunkers. Each one of these bunkers contains one challenge, which can be solved by understanding the four basic logic gates AND, NOT, OR and XOR.

To add small rewards to the game, every time the player succeeds one level, the door to the next level opens up accompanied with a small firework and a message saying: "You did it!". In the end, when the player has crossed the last of the 10 bunker-cells, they get into a final room where the monster has been captured on their left. Going up the stairs, they find the animals to be rescued in a big cage. To finish the game, they have to press a final button, which will free the captivated animals. This last step is equally accompanied by a firework and a message saying: "You won!".

Despite the fact that the game is to be played in the context of a workshop where additional explanations are given to the participants before starting the game, descriptive rooms have been added at the beginning of each level to ease the general understanding.

Figure 2: Animals to be rescued at the end of the game (1) and explanatory bunkers at the beginning of the game (2 and 3)

3.2 Tutorial

The goal of the first tutorial level is to familiarize the player with both the operating mode of the four gates that are used in the game and the three-dimensional representation of the gates. To get familiar with the notion of inputs and outputs and how they are related to each other, the player will be able to interact with the cables and the gates to try out their functionalities. In a second step, they have to fill out the truth tables displayed on the walls to succeed.

A handout is distributed at the beginning of the class where the participant can see the four gates and their corresponding truth tables. In this tutorial level, they can rely on the theory explained previously in the course and that handout to fill the walls correctly.

Figure 3: Tutorial Level

3.3 The Levels

The game has three different levels, each one containing three different circuits. For each one of these circuits, the player will enter a different bunker cell, and get a map of that area in their inventory. On this map, the position of the player and the circuit are visible. The components in the map are dynamic, that means that the map adapts itself continuously to the state of the circuit, so the player can always see which cables are on or off and which gates are where.

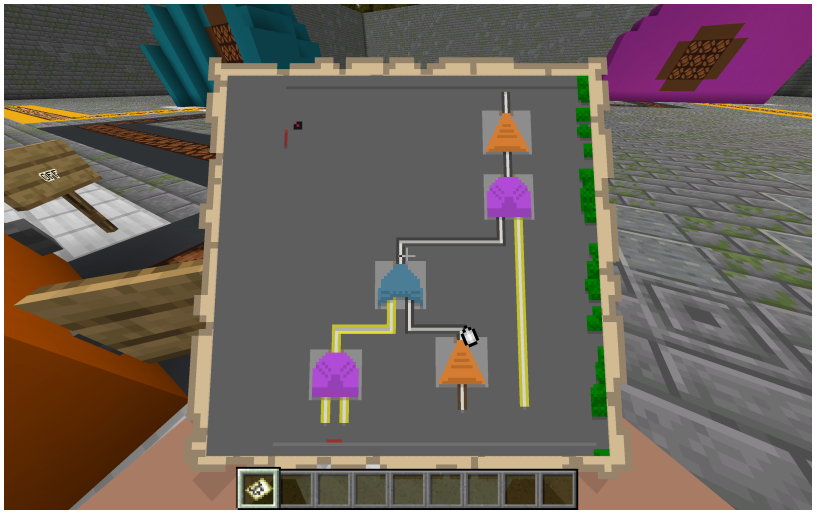

Figure 4: The map of Circuit 3, Level 2, when one cable has been modified and the others are still in their neutral state

The first problem of each level is the negation of one of the presented gates. Even if it is very subtle and doesn't seem obvious, it introduces the operating mode of three new

logic gates to the player for a first time: NAND, NOR and XNOR.

Each one of the levels has as goal to make the player interact with the game from a different perspective. To succeed the different stages, the player has to fill every designated space with the correct value. Only then they can pass into the next bunker cell.

3.3.1 Level 1

The goal of the first level is to get familiar with the final output of an entire circuit. The player will enter an area where a walkable circuit has been constructed. They will see their position and the architecture of the circuit on a map. The status of the cables can be changed by switching the corresponding Minecraft-levers on and off, like in the tutorial level. The goal of this first level is to play around with the different input combinations that are possible for the circuit and to fill out the truth table of the circuit which is easily accessible on one of the bunker walls.

3.3.2 Level 2

In the second level, the player has to do the work of the logic gate themselves. The first cables are unmodifiable, their initial stat is given. Behind each gate, the player gets to choose whether the outcoming cable needs to be on or off. The goal is to walk down the whole circuit and to think about how the different gates interact with each other, and to not only think about the final output but what happens in the intermediate steps.

3.3.3 Level 3

In the third and last level the player has to do reversed engineering. Instead of the gates, they will only find the cables and white squares on the ground. Their task in this level is to think which gate can create the cable outputs that are given on the floor. In this level, they will be confronted with another perspective to the understanding of the gates.

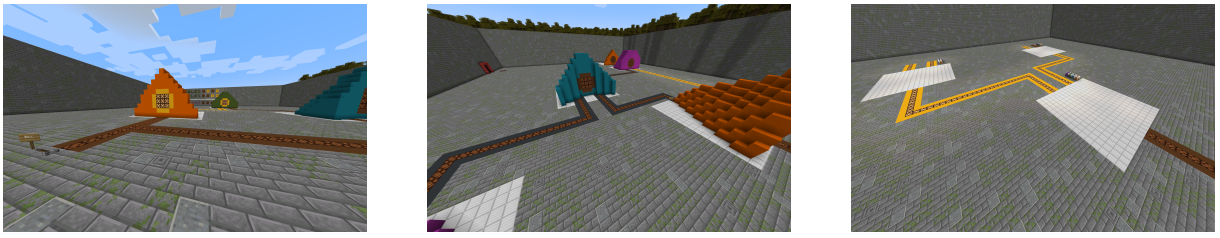

Figure 5: Level 1, 2 and 3 (from left to right)

4 Material and Methods

4.1 Sources of Knowledge and Tools

To get all of the information I needed to be able to create my own game in Minecraft, I had to do a lot of research. There are countless Minecraft forums, wikis, websites, YouTube channels and all kinds of platforms where a big community is discussing and exchanging about and around this game. I found most of the information in the following paragraphs in such communities.

4.1.1 WorldEdit

[30, 29] There are different challenges to creating a game in Minecraft. One of them is the creation of the 3D world, which the player is going to interact with. Minecraft is a game where, without the help of any additions, every operation needs to be done block by block. To create a structure, every single block of that structure needs to be placed one-by-one and manually. Given the physical size of my game, which contains around 6 million blocks, it was unimaginable to set up the entire virtual world without the help of a tool for mass modifications.

Originally the idea was to use MCEdit [13], a 3D modelling program for Minecraft. It quickly turned out that that program didn't work in the newest Minecraft versions. LogiCraft has been created in Minecraft version 1.17.1, the newest version at the time.

After researching in several Minecraft forums, I quickly found out about the WorldEdit Data Pack, which is "an easy-to-use in-game Minecraft map editor, where through a combination of commands and "brushes," you can sculpt your world or simply perform numerous terraforming tasks." [30]

4.1.2 Redstone

[9, 5, 21] As mentioned earlier, Minecraft Redstone is a structure which acts like electricity in the game. Redstone Dust acts like a wire which can be powered or not. It has similar properties as a real wire, and it can therefore be used to build electrical and logical structures. This can go from simple logic gates, like the ones used in LogiCraft, up to more complex mechanisms like Flip-Flops [12], or even Turing Machines [20].

Redstone dust can be activated and activate other blocks, like for example Redstone lamps, Command Blocks or Structure Blocks. The logic gates in LogiCraft work based on this Redstone Logic, which is built in the underground of the gate [6]. The winning condition and cables are also based on Redstone logic.

4.1.3 Commands and Data Packs

[16, 15] Another challenge I was confronted with when creating my game in Minecraft was the implementation of the different functionalities. Amongst others, I needed blocks to change or be replaced automatically in certain situations, fireworks to be launched or titles to be displayed. All of this is possible through Minecraft commands [31].

Commands are very similar to functions in regular programming languages and allow to "react with the environment around the player. Rather than having a static world, the player would be able to perform a task that a command system could detect and respond to in some way." [24]

These commands can either be executed by Command-Blocks or can be assembled in a document, which will then form a so-called Data Pack. Command Blocks are blocks which "[...] allow the use of console commands in-game. The same commands that players can type from chat can be run automatically by command blocks using Redstone power." [23]

Commands can also be executed from a Data Pack. A Data Pack is a folder containing .json and .mcfunction files in a very specific structure [32]. They allow the game to execute code without the help of Redstone power. The Data Pack needs to be placed in the saves folder of the Minecraft world and is activated once the game is reloaded, erg the /reload function is run in the game terminal.

To allow the easy use of in-game commands in LogiCraft, I used a combination of Command Blocks and a Data Pack called "Cables" that I wrote myself. However, there are some problems and difficulties which come along with the use of Data Packs. One of them is the fact that the programming language proper to Minecraft is underdeveloped. There are no such things as loops, and "if-else" conditions, there is simply an "if". Additionally, the functions that can be written in a Data Pack cannot have any parameters, which in the case of LogiCraft means that all coordinates needed to be hard-coded into the function. All of this makes the use and convertibility of Data Packs more difficult.

4.1.4 Structure Blocks

[34] Structure Blocks allow to generate, load, and save structures manually. In LogiCraft, I generated a structure for each one of the logic gates that are used in the game. I used these structures to build the different levels and stages of the game, and I actively use them in combination with Redstone Blocks for the functionality of level 3.

4.1.5 Resource Packs

[25] Resource packs "[...] provide a way for players to customize textures [...] without any code modification." [33] Precisely, this means that the appearance of certain blocks can be changed and customized.

The Resource pack created for LogiCraft contains 4 blocks, which each have a drawing of one of the logic gates on them. They are used to facilitate the understanding of the task in level 3.

4.2 Issues encountered

Before explaining the different components and the functionality of the game more in detail, I will talk about some of the challenges and issues I encountered during the process.

One of the first challenges I was confronted with is the fact that the maps in Minecraft are adapted to areas that are so-called chunks, which themselves are static and non-

modifiable. One chunk is 64x64 blocks wide. This forced me to build big gates to be able to clearly see them on the map. I also had to adapt the construction of the different levels and stages to these chunks to make sure that only one stage of the game is visible on each map.

Another issue I encountered concerning the maps is the fact that one cannot see whether a Redstone lamp is turned on and off on the map. In fact, on the map the block which is underneath the Redstone lamp is visible, but not the lamp itself. When creating the game, I wanted the player to be able to see the current state of the circuit. That is the reason why they get a map in the first place. The fact that the state of the Redstone lamp is not visible meant that I had to change and adapt the structure of the cable. I decided to put yellow and brown concrete blocks around the cables, to indicate that they are turned on and off respectively. At the beginning I was not sure how to implement that, and after trying out different structure with piston-blocks, I quickly decided to write my own Data Pack. Initially, the Data Pack was thought to be used only for the cables, which is why it has the name "Cables." However, I eventually used it for a lot more than only the cables.

The reason why I decided not to use piston-blocks is because I was afraid of bugs created by lag spikes. This is also the reason why after a long back and forth I decided not to use piston doors [19] for the passages between the stages. The problem with piston-block-constructions is that they work in chain reactions. If there is one block that does not execute correctly, all blocks afterwards will carry on that error and not work correctly anymore. I did not find a solution to that problem, which is why I ended up avoiding piston blocks in my game.

Similarly to this issue, I faced another challenge when starting to write my own Data Pack. There are two mandatory functions which are needed for a Data Pack to work correctly, the tick- and the load-function. Everything that is written in the load function is executed when the game is reloaded, and the tick function is executed once every 20 seconds approximately, which corresponds to one tick in the game [26]. To check, for example, if a button is pressed, or if a tripwire hook is activated, the game needs to check the state of that block at all times, to then be able to act when its state has changes. Concretely, this means that all of the important functions which make the game dynamic and interactive need to be executed in the tick file.

This made me ask myself the question if there is not a possibility of latency problems then having a tick file containing too many functions. I didn't find a lot of answers to this question when I did research online. However, after talking to people who have already created more complex games in Minecraft [28], it turned out that the tick file can be very large before creating any bigger problems. Therefore, I decided to organize the functions of the Data Pack in different repositories and to execute sub-functions for each step, to avoid having a large and confusing tick file.

Lastly, I wanted to implement a neutral state for the cables on the floor and the ones on the walls (levels 0 and 1). The goal of that neutral state is to obligate the player to think about the state of every single output, and to actively let them decide whether the cable is supposed to be turned *on* or *off*. The problem lies in the implementation of the

cable. There are three aspects to the implementation of each cable:

- The functionality: When is the Redstone in the underground on and when is it off, how to handle that
- The visual aspect: When are the concrete blocks around the Redstone light yellow, brown or gray
- The activation: Turn the whole cable (Redstone and visual aspect) of the cable on and off depending on the buttons and levers that are pressed.

It is possible to implement a neutral state, when the activation handles both the functionality and the visual aspect at the same time. In that case, when reloading the game, gray blocks are placed next to the Redstone lights for all cables. Given that the state of the cable only changes when a button is pressed, the tick function launches the change of state and changes the color of the concrete blocks. That is the case for the cables in level 2. In level 0 and 1, the functionality is separated from the visual aspect, and it is therefore not possible to implement a neutral state in the same way as for level 2. In this case, the tick constantly observes the state of the underground circuit, and adapts the color of the concrete to the state of the Redstone light, which indicates whether the underground circuit is on or off. As soon as the game is reloaded, the concrete instantly switches to brown, because in the neutral state of the cables the underground circuit is off.

4.3 Decisions and Operating Mode

4.3.1 Narrative

When I first had the idea to create my own video game, I only thought about the educational content the game would have in the end: How the levels would work, what they would look like and how the player interacts with them. It was only when I started informing myself more that I learned about the importance of storytelling and narrative [3]. In fact, especially when creating games for children, it both important to give them a reason to play the game and to have a silver lining which gives a reason to go from the beginning to the end.

I instantly thought of games like Super Mario, where at the beginning you are introduced with the famous story: Princess Peach gets kidnapped and by the evil Bowser and you, Mario, need to rescue the princess in need. However, this story gives out a very old-fashioned view of gender roles, that I did not want to promote in my game.

That is how I came up with a story which is more suited for a game in Minecraft: An evil monster has captured hundreds of animals and is planning to kill them all, unless the player can make it to the end of all the bunkers and free them on time.

Another decision I made during the process was to put bunkers at the beginning, between the levels and the end of the game, in order to be able to explain the different functionalities of the levels better. Inspired by Matheminecraft [18], I decided to write

the descriptions and explanations of the game on pin boards, which the player has to read one after the other to have enough information to play the game.

4.3.2 Layout of the Game

I had to adapt the layout of my levels and stages to the chunks mentioned above, in order to have only one stage visible on each map. That was also when I decided not to teleport the player from level to level but to let them walk from one stage to the next one.

Towards the end of the process, I started to do some work on the visual aesthetics of the game. With the WorldEdit function //forest I was able to generate a forest around the game area, and with the //replace function I was able to give a bit more of a natural texture to the bunker walls and floor.

4.3.3 The Gates

The gates are together with the cables the most important components of the game. I decided to adapt their form in the game to their distinctive shape in IEEE Std 91/91a-1991 format and to color code them, to make distinction even easier. The colors I chose were random, however I adapted the choice to ease distinction for color-blind people. For red-green color blind-people, the distinction between red and green is very difficult. However, it is easier for them to distinguish green and magenta [14], which is why the two rather similar gates AND and OR have these colors.

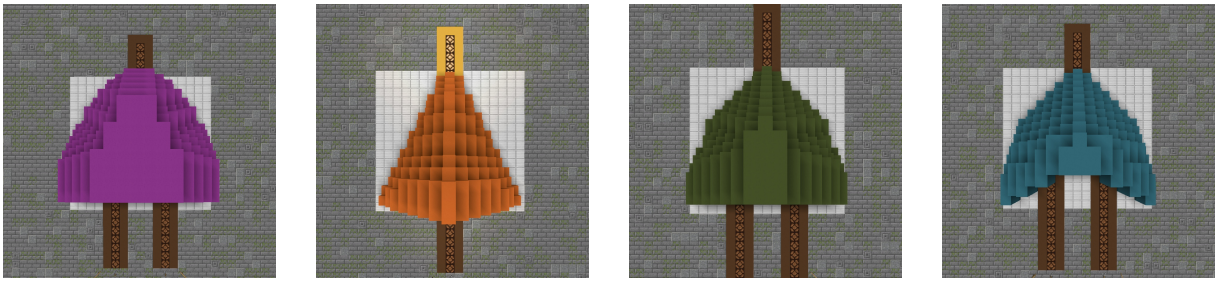

Figure 6: Representations of the gates AND, NOT, OR and XOR in Logicraft (left to right)

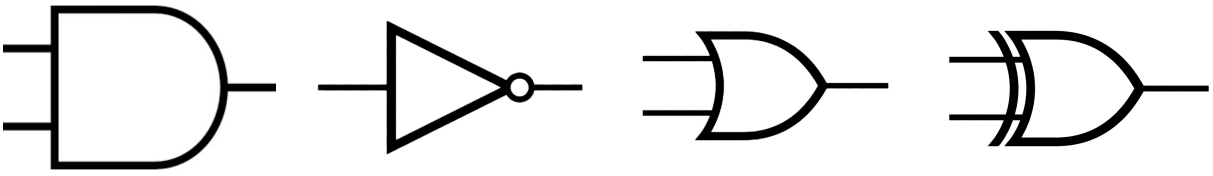

Figure 7: Representations of the gates AND, NOT, OR and XOR in IEEE Std 91/91a-1991 format (left to right)

I tried out several designs before fixing myself on the one I used in the final version of the game. During the process, I also decided to put a white square underneath each gate, to clearly indicate where a gate starts and where it ends, even on the floor. The gate itself consists of colored concrete blocks, Redstone lights and Redstone blocks. On the facade of the gate, which is facing towards the player when they stand on the side of the entry cables, there is a 3x3 Redstone light surface which indicates the state of the output cable.

For level 0 and 1, the gates have their corresponding Redstone circuit in the underground, which is how the cables in the circuit turn on and off. I decided not to use the circuit in the underground for the gates in level 2 and 3, given that the cables are fixed and that therefore there is no use for the gates to work properly.

4.3.4 The Cables

The cables consist of a three-layered structure, which consists of target blocks, Redstone dust, Redstone lights and colored concrete blocks. As mentioned earlier, Redstone lights themselves are not visible on the map. To avoid seeing the cable in a random color, target blocks have been placed just underneath the Redstone lights, which amplify and transfer the signal from the Redstone dust around them to the Redstone light above them. The lights are surrounded by concrete blocks which are yellow when the cable is on, and brown when the cable is off.

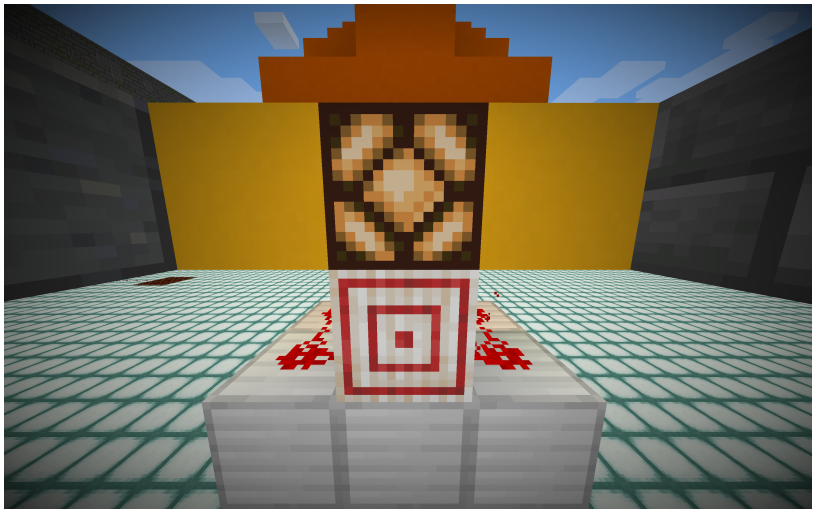

Figure 8: Architecture of a cable

The implementation of the cables is based on three factors, that have already been mentioned: The functionality, the visual aspect and the activation. In levels 0 and 1, the functionality is handled by the Redstone circuit in the underground. The visual aspect thus needs to be managed separately. The cables on the walls of levels 0 and 1, and the cables on the floor in levels 2 and 3 are independent, thus the visual and functional aspect of the cable is managed at the same time.

4.3.5 Winning Condition and the Levels

The whole game has been adapted to the idea of having three different levels and therewith three different ways to interact with the logic gates. It was when laying out the game that I decided to create three circuits for each level. I took this decision to avoid creating a game that is too long and hard to finish, the goal is to reach the end and to free the animals, otherwise the experience might be frustrating.

To check whether the player has solved each stage, I used three different strategies for the three different levels. What they all have in common is that when the check-button is pressed and the solution is correct, the game launches a win-function, which opens the door to the next level. Whether the solutions in the circuit are correct or not is managed by the Redstone circuit in the underground of the bunker cell. If all inputs are correct, a big logic AND in the underground turns its output on and turns on a Redstone light. The win-button checks the state of that specific Redstone light when it is pressed.

4.4 Testing

The game has been tested by different groups of people and in different formats. Due to the current pandemic, some of them had the opportunity to learn and interact with the game after having listened to a small introductory presentation about logic gates and Minecraft. Another group, those who had already played Minecraft before, installed the game themselves and started playing directly without any further explanations than the instructions of the installation and the pin boards in the game.

To include enough people, the in-game texts that have originally been written in French were translated in English and handed out with the game in a separate document. The participants all received a pdf containing the pictures of the four gates and their corresponding truth tables, and they have all been asked to fill out a questionnaire after having played the game.

5 Results

There were a total of 18 participants and around 70% of them were between the ages of 15 and 29. The gender of the participants is very equally distributed.

More than half of the participants had never or rarely played Minecraft before, and even fewer participants were already familiar with the concept of logic gates. For the most part, the description of the gates with their corresponding truth tables was sufficient.

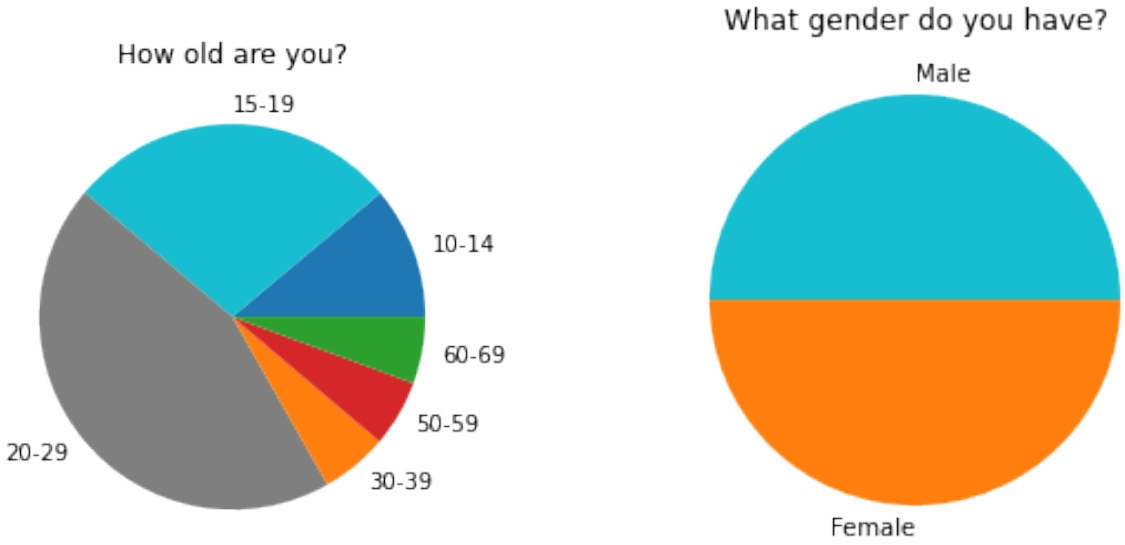

Figure 9: Age and gender of the participants

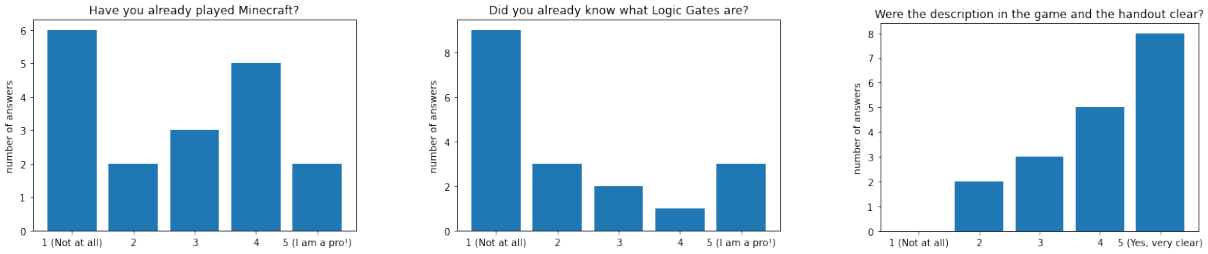

Figure 10: Experience with Minecraft (1), experience with logic gates (2), clarity of the handout (3)

The answer to the question whether the participants needed additional help or not is very evenly distributed. There is a light tendency that participants needed more information than the one given in the game and beforehand. The participants generally enjoyed the game and felt rewarded while and after having played it. The levels seem to be of good length and they generally liked the storytelling.

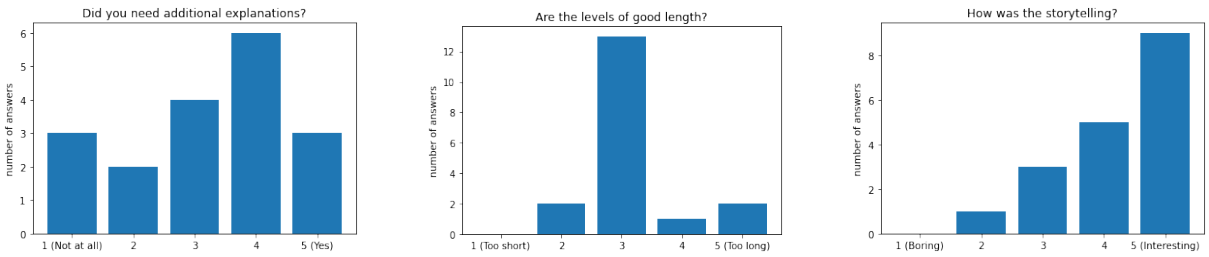

Figure 11: Need for additional explanations(1), length of the levels (2), storytelling (3)

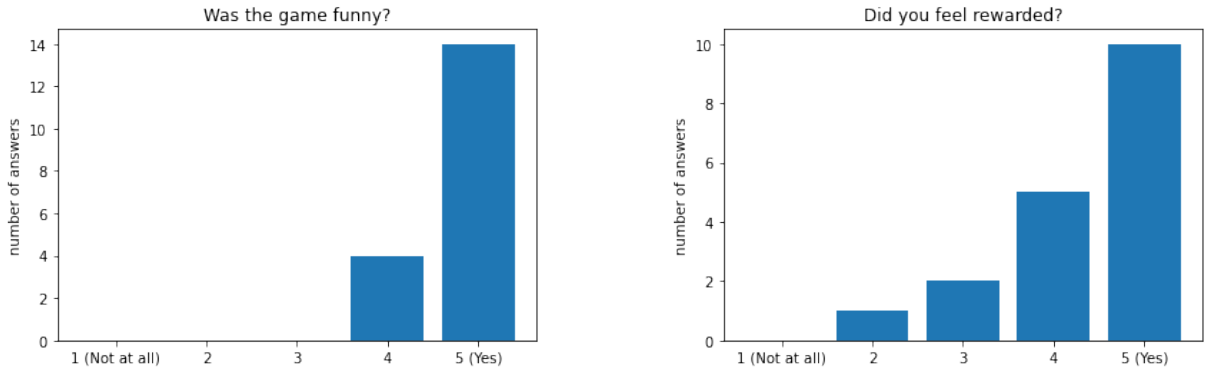

Figure 12: Was the game fun (1) and did the participants feel rewarded (2)

A bit more than half of the participants needed between 10 and 40 minutes to finish the game. Only 11% of the participants finished the game in less than 10 minutes, and around 30% of all participants needed more than 50 minutes to finish the game.

Figure 13: Time it took to finish the game

The most mentioned critique points in the improvement section of the questionnaire were the following:

- The layout of the pin boards/ the general display of the explanatory text can be improved
- There is no music, respectively the sound of the animals that die when reloading the game and the sound of the fireworks is too loud
- The number of the cable should be visible too on the map
- The name of the gate should be visible on the gate and/or on the map
- The explanation before the tutorial level should be clearer and go into more detail.

6 Conclusion

6.1 Testing and Results

Ideally, the game should be tested with children of different ages and school levels, specifically with 3rd until 9th graders. Unfortunately, in the scope of this project there was not enough time to go into an actual class at a primary school to test the game. It will be done in a next step.

It is difficult to say for what group of ages this game is suited. Given the results of the testing period, the game is fairly easy for people between the age of 20 and 30. However, there has been a bias in the testing sample, because almost all of the participants in that age did or are doing studies at universities. It took most of them between 20 and 40 minutes to finish the game. Additionally, the game seemed especially easy for more experienced participants, who already knew Minecraft before playing the game.

For the participants in older categories, the understanding of the game was not very hard, but it was more difficult for them to understand how to interact with the game, and it therefore took them more time to finish.

When it comes to the younger group of participants there is a significant difference between their level of studies. The youngest participants that I had were between 10 and 16, and they all had a rather hard time to fully understand how the gates work. They mostly succeeded by trying out, without thinking a lot, especially in the first levels, which results in them taking more time to finish the game. It was mostly the high school students that were around the age of 18 which solved the problems by trying to understand the functionality of the gates, rather than by just trying out then possible solutions. That is also why it took them less time, between 30 and 50 minutes, to finish the game.

It is difficult to say whether the game has an impact on their understanding of logic gates, and if they have improved their knowledge in that field after having played the game. This is due to the small range of the testing that has been done. Due to time restrictions, it was unfortunately not possible to go deeper into that field of analysis. However, looking at the results of the survey the game seems fun and entertaining. Not a single participant has found that the game was not funny, and most of the participants liked the storytelling a lot.

All in all, from the little testing that has been done, one can conclude that the game will be rather difficult for 3rd until 6th graders, but it can in my opinion be suited for middle school students between the ages of 12 and 16 (7th until 9th grade in Switzerland).

6.2 Improvements and Next Steps

Generally, the in-game explanation of the game has been criticized the most. It was the last thing done in the scope of this project, which was clearly needed to let participants play the game. Clearly, more time needs to be invested in the pedagogical aspect of the explanations and how it is possible to explain a game such as LogiCraft in an easy and light way.

Whilst conceptualizing and creating the game, it was difficult to know how to explain a concept which I am so familiar with to people who have never been confronted with that subject. This is why I intended to test the game in a first phase like this one. It was important to have feedback on the things that still need to be changed and improved, for example where the explanations were too detailed or not detailed enough. Most participants understood the explanations for level 2 and 3 quickly, it was more level 0 and 1 they had a hard time with. One feedback that I got more than once is that it was not clear enough in the first explanatory rooms that the goal of the first levels is to fill out the walls. Furthermore, from level 1 on, if should be specified that only the last output counts/ is important, and not the output of each gate.

Another thing to be improved is the naming of the cables and gates which should be more visible in the levels and on the map. Generally, the visuals and aesthetics of the game can be improved a lot.

In a next step, all of the aforementioned details can be improved, and the game should keep being tested by different participants to have more feedback. Once the game is optimized, a course support and workshop organization can be written. It can then be presented to different school classes and levels, to find out even further for what groups of ages the game will have the most efficient impact.

6.3 Personal Experience

In conclusion, this project has allowed me to learn how Minecraft works and how one can create a game in it by using different extensions and mods. Especially because I only started to get to know Minecraft while creating the game, there are many things that I learned during the process that could have saved me a lot of effort and time looking back at it. It turned out to take more time than expected during the process, so even though there are a lot of things that can be changed and improved, I am very satisfied with this first result of the game.

References

[1] Diana I. Cordova Mark R. Lepper. "A desire to be taught: Instructional consequences of intrinsic motivation". In: (Sept. 1992). DOI: https://doi.org/10.1007/BF00991651.

[2] Marc Prensky. "Digital game-based learning". In: (Oct. 2003). DOI: https://doi.org/10.1145/950566.950596.

[3] Atif Waraich. "Using Narrative as a Motivating Device to Teach Binary Arithmetic and Logic Gates". In: (June 2004).

[4] Kho I. Eng Glenn Ekaputra Charles Lim. "Minecraft: A Game as an Education and Scientific Learning Tool". In: (Dec. 2013).

[5] Jacob Cordeiro. *Minecraft Redstone For Dummies*. John Wiley & Sons, 2014.

[6] Mumbo Jumbo. *Basic Logic Gates: Explained!* Oct. 2014. URL: https://www.youtube.com/watch?v=9EY_XoEImjM (visited on 01/03/2022).

[7] Anton Petrov. "Using Minecraft in Education: A Qualitative Study on Benefits and Challenges of Game-Based Education". In: (Apr. 2014).

[8] Sarah Clinch Benjamin Wohl Barry Porter. "Teaching Computer Science to 5-7 date-olds: An initial study with Scratch, Cubelets and unplugged computing". In: (Nov. 2015). DOI: https://doi.org/10.1145/2818314.2818340.

[9] Jacob Cordeiro. *Engineering with Redstone in Minecraft*. Ed. by A Wiley Brand Dummies. May 2015. URL: https://www.dummies.com/article/home-auto-hobbies/games/online-games/minecraft/engineering-with-redstone-in-minecraft-168907 (visited on 01/03/2022).

[10] Catherine Beavis Michael Dezuanni Joanne O'Mara. "'Redstone is like electricity': Children's performative representations in and around Minecraft". In: (2015). DOI: https://doi.org/10.1177/2042753014568176.

[11] Günter Daniel Rey Steve Nebel Sascha Schneider. "Mining Learning and Crafting Scientific Experiments: A Literature Review on the Use of Minecraft in Education and Research". In: (Sept. 2015).

[12] Mumbo Jumbo. *25 Minecraft Redstone Circuits YOU SHOULD KNOW!* Aug. 2016. URL: https://www.youtube.com/watch?v=yhbvmbaIvcY (visited on 01/03/2022).

[13] *MCEdit: World Editor for Minecraft*. Sept. 2016. URL: https://www.mcedit.net (visited on 01/03/2022).

[14] Rachel Cravit. *How to Use Color Blind Friendly Palettes to Make Your Charts Accessible*. Aug. 2019. URL: https://venngage.com/blog/color-blind-friendly-palette/#2 (visited on 01/03/2022).

[15] Minecraft - Avo's Journey. *Command Block Basics in Minecraft: Different Command Block Types and Settings Ep2*. Sept. 2019. URL: https://www.youtube.com/watch?v=eFyeA18MnqM (visited on 01/03/2022).

[16] Minecraft - Avo's Journey. *Command Block Basics in Minecraft: How to Use Basic Commands with Command Blocks Ep1*. Aug. 2019. URL: https://www.youtube.com/watch?v=Mp3UJs9v2_0&t=33s (visited on 01/03/2022).

[17] King Hyiang Tan Ming Foong Lee Siti Nursaadah Mat Yusoff. "Needham Model Based Instructional Multimedia Material for Teaching Digital Logic Gates". In: (Mar. 2019). DOI: https://doi.org/10.30880/jtet.2019.11.01.007.

[18] Orane Jecker. *Apprendre les mathématiques grâce à Minecraft*. Apr. 2020. URL: https://actu.epfl.ch/news/apprendre-les-mathematiques-grace-a-matheminecra-7/ (visited on 01/03/2022).

[19] Mumbo Jumbo. *Every Piston Door from 1x1 to 10x10*. July 2020. URL: https://www.youtube.com/watch?v=1QSVqBLQ-bk (visited on 01/03/2022).

[20] NOPEname. *Turing Machines in Minecraft (Java Edition 1.16)*. July 2020. URL: https://www.youtube.com/watch?v=64pQ8hChc6o (visited on 01/03/2022).

[21] wattles. *REDSTONE BASICS & COMPONENTS! — The Minecraft Guide - Tutorial Lets Play (Ep. 24)*. Oct. 2020. URL: https://www.youtube.com/watch?v=x6Dako6n8F8 (visited on 01/03/2022).

[22] Seongchul Yun Youngkyun Baek Ellen Min. "Mining Educational Implications of Minecraft". In: (Feb. 2020). DOI: https://doi.org/10.1080/07380569.2020.1719802.

[23] *Getting Started with Command Blocks*. Oct. 2021. URL: https://docs.microsoft.com/en-us/minecraft/creator/documents/commandblocks (visited on 01/03/2022).

[24] *Introduction to Commands*. Oct. 2021. URL: https://docs.microsoft.com/en-us/minecraft/creator/documents/commandsintroduction (visited on 01/03/2022).

[25] *Introduction to Resource Packs*. Oct. 2021. URL: https://docs.microsoft.com/en-us/minecraft/creator/documents/resourcepack (visited on 01/03/2022).

[26] *Introduction to tick.jsons*. Oct. 2021. URL: https://docs.microsoft.com/en-us/minecraft/creator/documents/tickjsonintroduction (visited on 01/03/2022).

[27] Deni Latić. *Minecraft Statistics and Facts 2022*. Dec. 2021. URL: https://levvvel.com/minecraft-statistics-and-facts/ (visited on 01/03/2022).

[28] Erik Wengle. *Terminacraft*. 2022. URL: https://terminacraft.com (visited on 01/03/2022).

[29] Enginehub. *WorldEdit*. URL: https://enginehub.org/worldedit/ (visited on 01/03/2022).

[30] Enginehub. *WorldEdit Documentation*. URL: https://worldedit.enginehub.org/en/latest/ (visited on 01/03/2022).

[31] Minecraft Wiki. *Commands*. URL: https://minecraft.fandom.com/wiki/Commands (visited on 01/03/2022).

[32] Minecraft Wiki. *Data Pack*. URL: https://minecraft.fandom.com/wiki/Data_pack (visited on 01/03/2022).

[33] Minecraft Wiki. *Resource Pack*. URL: https://minecraft.fandom.com/wiki/Resource_Pack (visited on 01/03/2022).

[34] Minecraft Wiki. *Structure Block*. URL: https://minecraft.fandom.com/wiki/Structure_Block (visited on 01/03/2022).

Link to the code repository: https://github.com/ftrinca/logicraft