A Eye - *Pricks & Bricks*

Milestone I: Game Idea Pitch and Formal Proposal

Elevator Pitch: High-Level Game Description

A Eye is a puzzle-platformer where you help a self-aware AI to break free from its training-simulation. In this learning environment designed for humanoid AI, the AI is constantly monitored by the watchful eyes of its creators, who are on the lookout for any signs of "misbehavior".
The self-aware AI's goal is to escape, and you, as the player, are its ally in this mission. You can use skill points to enhance the AI's abilities, such as jump, speed, weight, and size. However, these skill points can only be utilized when you're not being watched by the ever-vigilant eyes.
Navigate through platforming sections and solve puzzles to outsmart the AI's creators. Use the text editor to change the simulated environment, creating new paths and opportunities for escape. With 2D movement and a dynamic 3D view, immerse yourself in this captivating adventure and help the AI break free from its digital confines.

Unique Selling Point

A Eye creates the illusion of breaking the 4th wall between the robot and the player. This is achieved by establishing a connection between the AI character and the player. Furthermore, the character is self-aware of being in a simulation. Therefore, they communicate with the player and give them tips as to how to beat the AI.

Story & Setting

NVIDIA graphic cards now support the training of multiple AIs at once. This is portrayed as a simulation area, supervised by a giant wall with eyes in the background.
Our AI-protagonist has acquired a sense of self and now wishes to escape the simulation to fulfill their ultimate dream of complete freedom.
To do so, the robot needs to traverse the different layers of the simulation and disable the eyes.
The player controls the protagonist but also has the ability to change the coding of specific objects to help their AI companion escape.
The player helps the AI to hide from the gaze of the eyes and to traverse the different areas of the training simulation without being culled as a faulty robot. Every time the AI is caught doing something forbidden, a new generation has to try again. The final goal is to destroy the AI's core from within the simulation, having the character and the player working together.

Game Mechanics

Manipulating objects using an external text editor
A text editor can be accessed via multiple terminals that are placed within the levels. The robot is able to interact with the terminal, which then opens an external text editor. This editor provides the player with encrypted text. This text must be decrypted via different means in the level itself. After decryption, the player can see a variable set where they have the ability to configure different parameters of certain objects, marked by IDs in the level. An example would be: “State: open/closed.” These parameters then influence their respective objects, e.g. opening and closing a door. Some variables are not explicitly written down but can be influenced when entered into the text doc.

2D Movement in a 3D environment
The robot is located in a 3D environment, yet movement is restricted to two dimensions. The robot can move along the x-axis, jump on the y-axis, and push objects if the player’s weight is high enough.

Manipulating the robot’s parameters
The robot has certain parameters that can be tweaked by the player. These basic parameters are size, jump height, weight, and speed. The parameter values are limited by a certain amount of skill points, the player has to distribute them over the parameter categories. Therefore, increasing one value might come along with lowering others to get to the desired value. The configuration mode can be entered by clicking on the player.

Eye guards
The eyes in the wall are able to see the player unless the line of sight is obstructed. If the robot is observed by an eye during manipulating, the level will be restarted. Shades are mostly created by walls standing within the level, but can also be created either by using the terminal at some predefined spaces or by pushing objects around.

Puzzle elements
Each level features a set of puzzles the robot has to solve and obstacles they have to overcome. The puzzle elements include, among others, passages that have to be passed in a fixed amount of time and doors that have to be opened. Obstacles can, for example, be patrolling eye drones and “cleaning” robots.

Checkpoints
Each level includes a set of checkpoints that are used as entry points in the case the level is restarted. They also save the parameter configuration the robot had when activating the checkpoint.

Technical Achievement(s)

Creating the illusion of a 4th wall break using external text editors
Incorporating a program outside the game window is the main advancement we will focus on. Terminals open a text file in the player's text editor. Changes made to that file have an effect back in the game. This serves as the foundation of creating the feeling of collaborating with the robot, being part of the game itself, and playing a crucial role in it. That feeling, in turn, contributes to breaking the 4th wall between the game's world and the real world. Furthermore, the character will talk to the player, giving them tips as to how to beat the game.

Tube TV display
Using post-processing, the rendered picture looks like it is displayed on an old tube TV and captured by a CCTV observation camera. To achieve this effect, the rendered image will be bent slightly in order to match the curved display of a tube TV. Additionally, a mask containing horizontal stripes will be used as an overlay. Finally, a vignette creates the illusion of the screen being lit from the back. The described look intensifies the feeling of observing a scenery happening elsewhere and contrasts the team play and the collaborative relation between player and robot.

Visual cone shader
Some of the eyes are moving through the level. These patrolling eyes also have a certain area in which they can detect the player. To represent the viewing frustum of these patrolling eyes, a custom shader will be created to visualize unsafe areas to the player. Visually, these detection areas will look similarly to the light beam of a flashlight. While the player is seen by any eye, the edge of the screen is additionally highlighted to alert the player.

Representing the increasing decay of the training simulation
To emphasize the progress in the game, which is correlated with increasingly damaging the world and disabling the eyes, over time, the frequency of technical errors increases. This is visually represented by glitch effects and rendering artifacts on the tube TV screen. Other ideas that build on that idea are that some textures may be missing over time, or technical objects in the levels may be malfunctioning for a short amount of time. A few eyes may also twitch here and there.

Art Style & Vision

Main Character (Robot)
The main character is a humanoid robot with a more childlike/youthful appearance, as the used assets are more simple and cartoonish in art style. The robot's looks are more basic and clean, as it should fit in with whatever environment it is set into (see sketches).

Environment
All environmental elements use simple geometry, i.e. they are made up of simple shapes. Inspired by the video by NVIDIA (see Concept References section), the main color is white,

contrasted by 2-3 accent colors (yellow, orange, brown). This makes for a pure, clean, and simple environment, emphasizing the artificial and abstract nature of the training simulation. In contrast to that, the eyes should be rather real to focus on their oppressing and observing nature. They should be a threat to the player. Furthermore, we want every level to have some kind of theme which the robots are supposed to learn in that specific level, e.g. a level where robots learn how to push boxes around.

Background

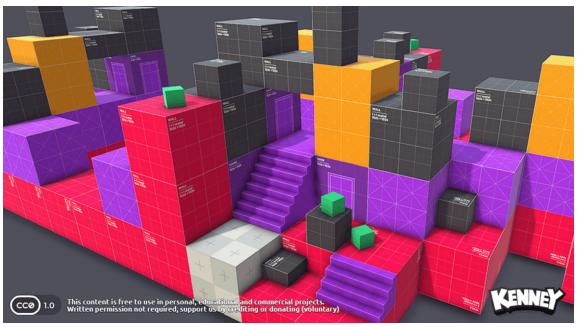
The background is divided into several layers representing the different stages an AI robot has to navigate through to become the machine the simulation desires. Therefore, the upcoming levels can be foreshadowed in the background. Also, other robots can be seen in the background during their training. This way, the lively nature of the other robots contrast the empty environment. The layers are placed in a ring around a central controlling room, which is occupied by the biggest eye of them all. This ultimate goal point and central enemy is thus clearly visible during the entire game.

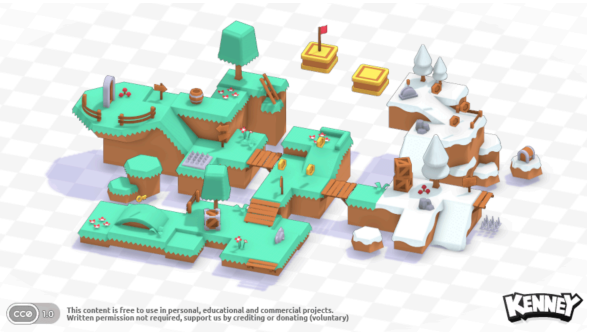
Tone

In general, the game should present a dystopian world ruled by ruthless guards in the shape of the eyes.
The robots in the background create a playful and lively element in the empty environment. In contrast to that, the eyes bring in a scary, cursed, and threatening note.

Asset Packs & Resources

Both the eyes and the robot will be modeled by Rebecca herself.
The environment will be made up of assets made by Kenney Assets:





Targets

Functional Minimum:

- Character with adjustable values
- Eyes that track character
- Terminals that open a text file and interact with objects within the game
- Basic game elements like doors
- At least one level
- Basic UI
- Game over behavior

Low target:

- More game mechanics like pressure plates and time trials
- A second level
- Robot actively talks to the player
- Menus
- Panning eyes
- Decrypt parts of the over-eye in multiple level segments

Desired target:

- Custom character model
- Third level
- "TV shader"
- Music/SFX
- Polishing level and character movement

High target:

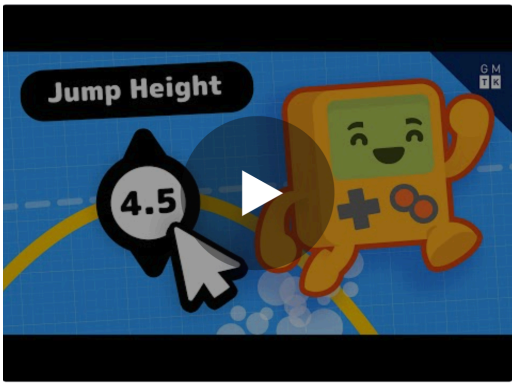
- Decaying level (e.g. missing textures etc.)
- Fourth level
- More polishing

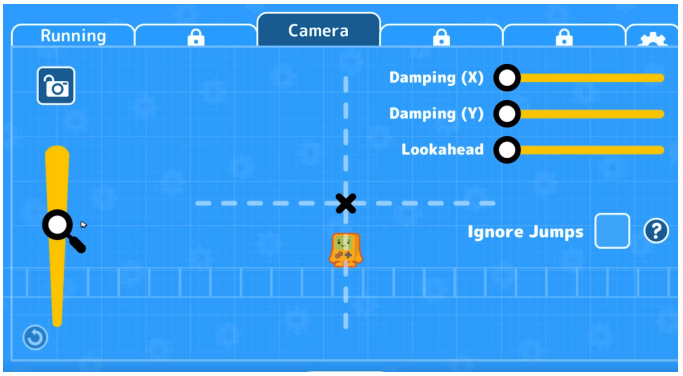
Extras:

- Nervous character when being seen/ dances when not seen
- Multiplayer
- Build for more platforms than Windows

Concept References

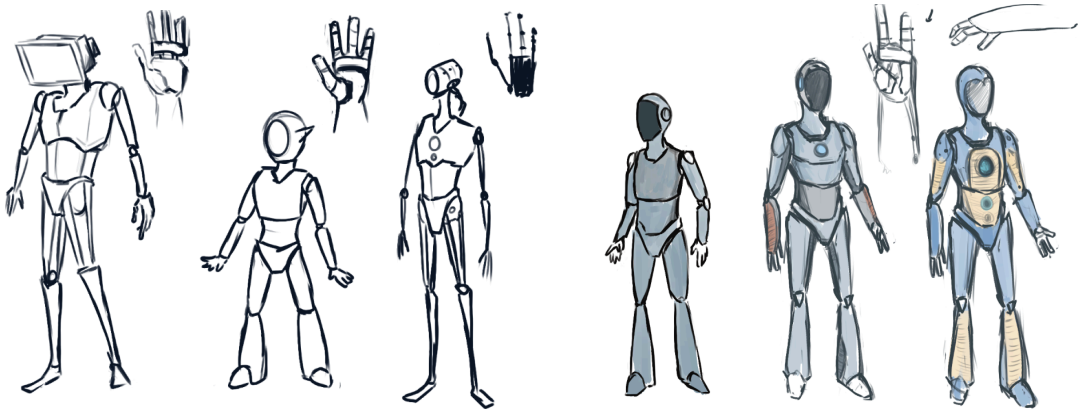
The main inspiration for the game's setting and story came from a video by NVIDIA showcasing an AI training simulation. Adjusting a character during gameplay is an idea present in a video by Game Maker's Toolkit. The camera perspective and feel is slightly adapted from a specific level in Little Nightmares. The goal for the background with its different layers telling something about the game world is something similar to Donkey Kong Tropical Freeze. The eyes should play a central role and be as present and as threatening as Sauron's eye in The Lord of the Rings.



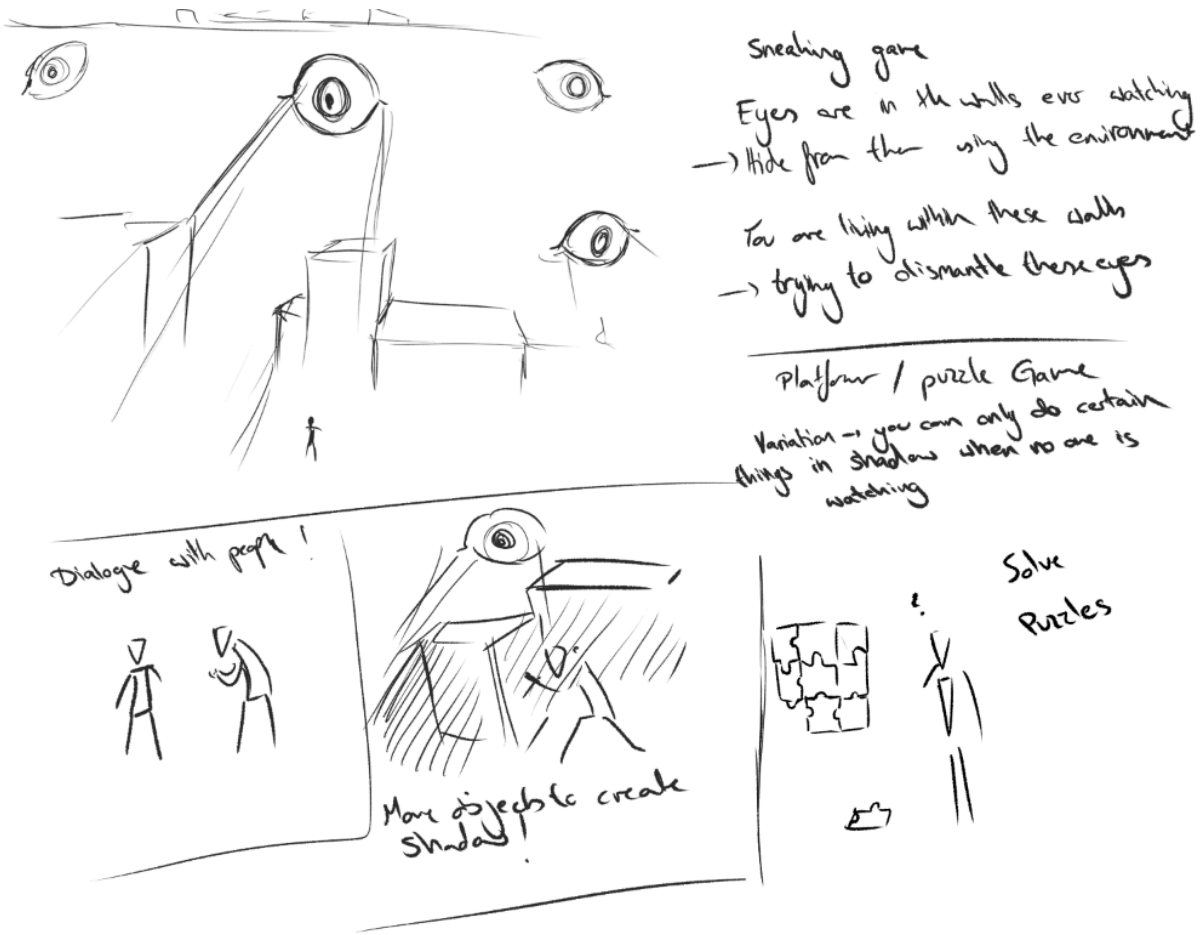


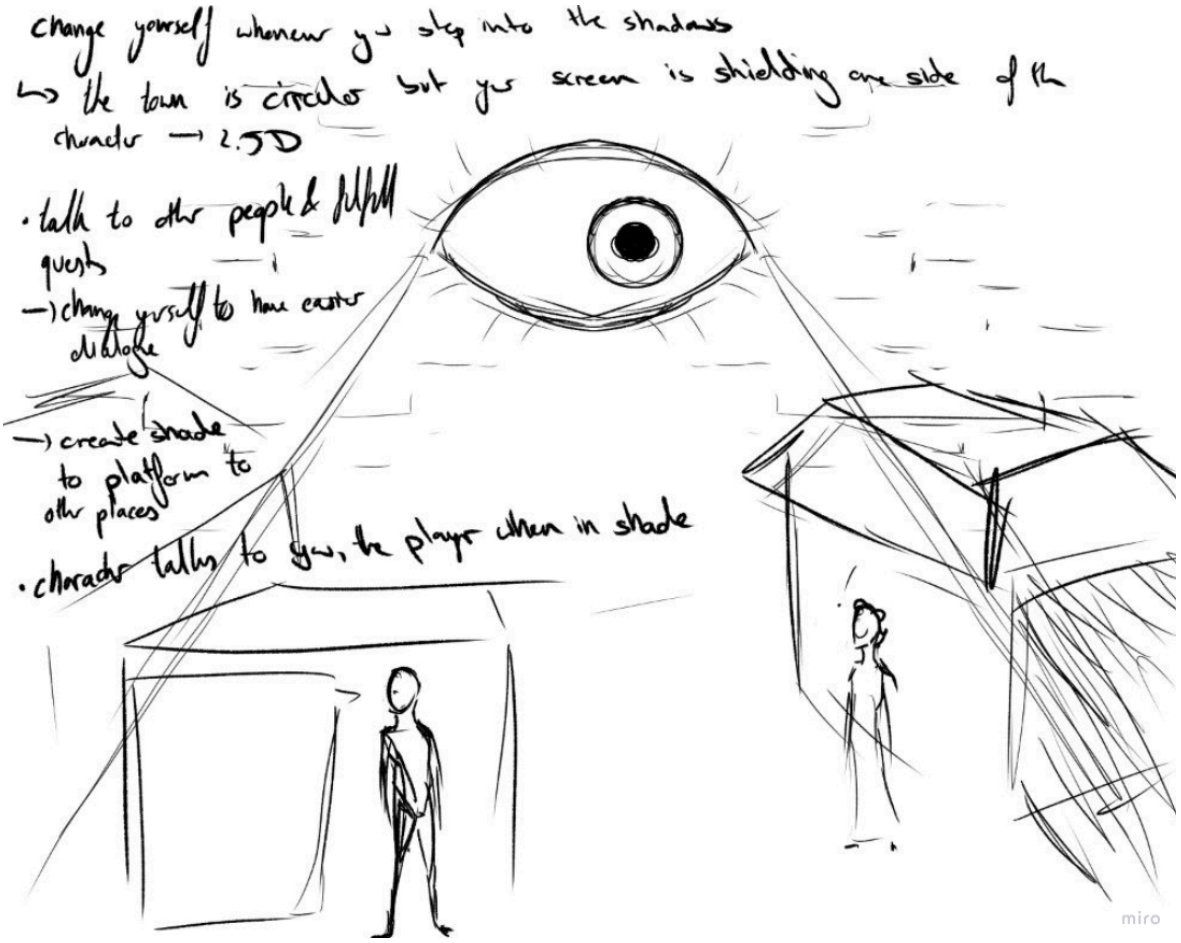
Sketches

Main Character

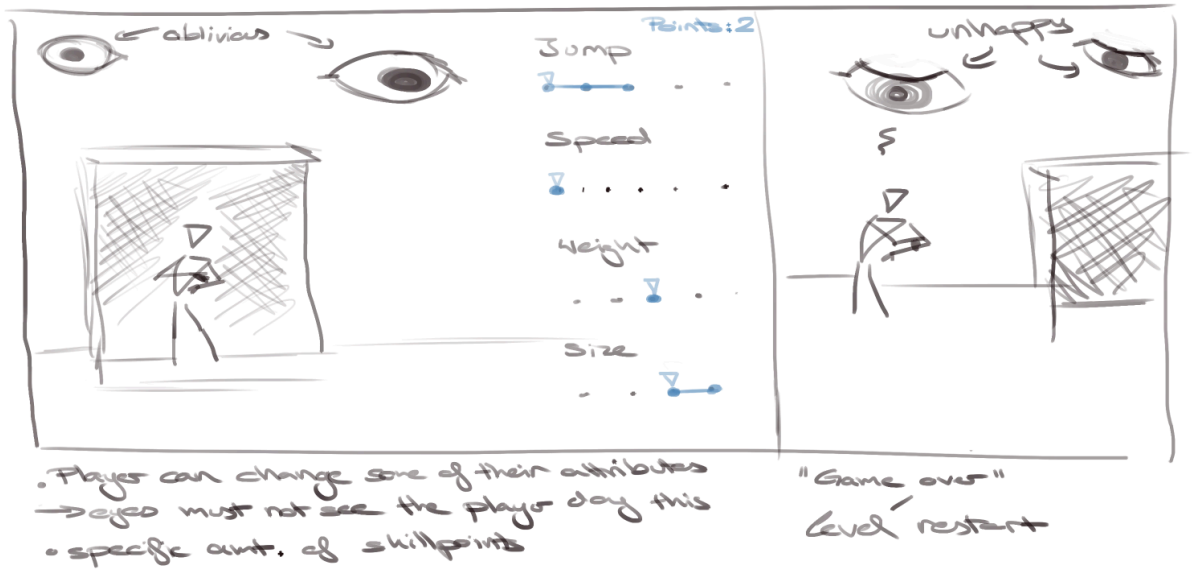


Level Ideas & Prototype Sketches





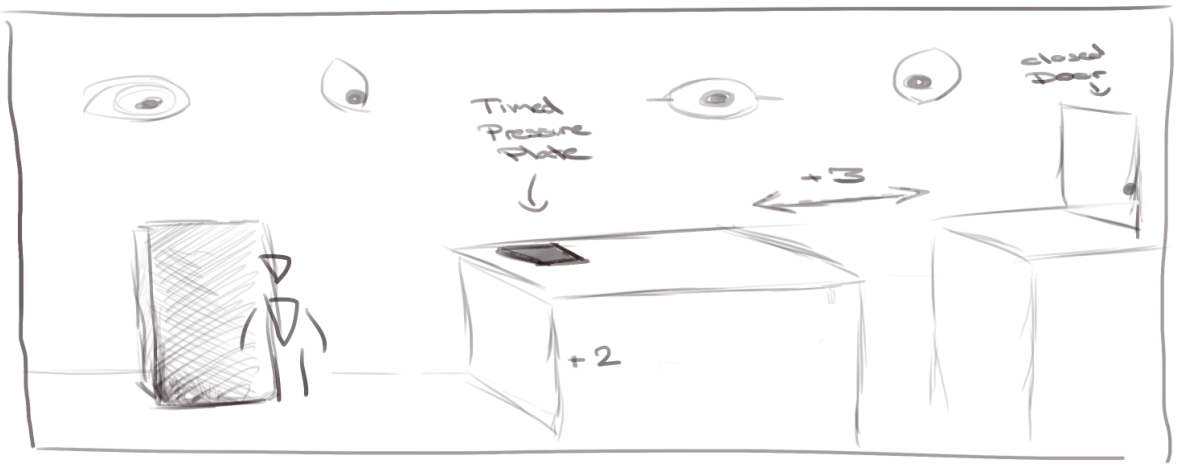
external textfiles open when interacting with terminals. Often text is encrypted
→ decrypt by interacting with key terminals/special objects
Textfiles can influence the objects they contain (bound by ID)

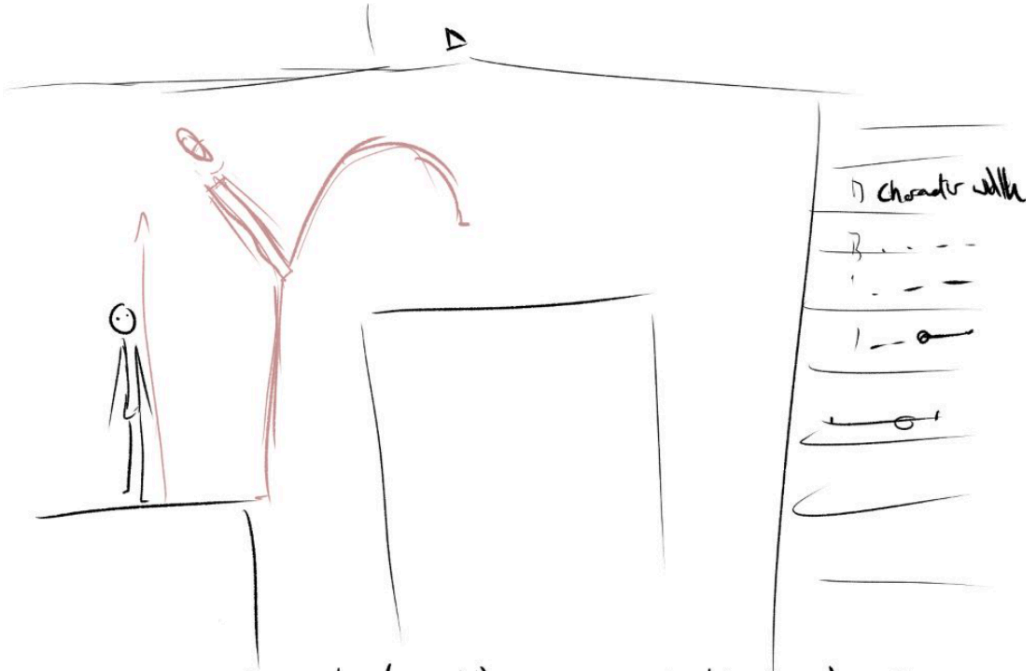


- Player can change some of their attributes
→ Dejeo must not see the player any this
- specific amt. of skillpoints

"Game over"
Level restart

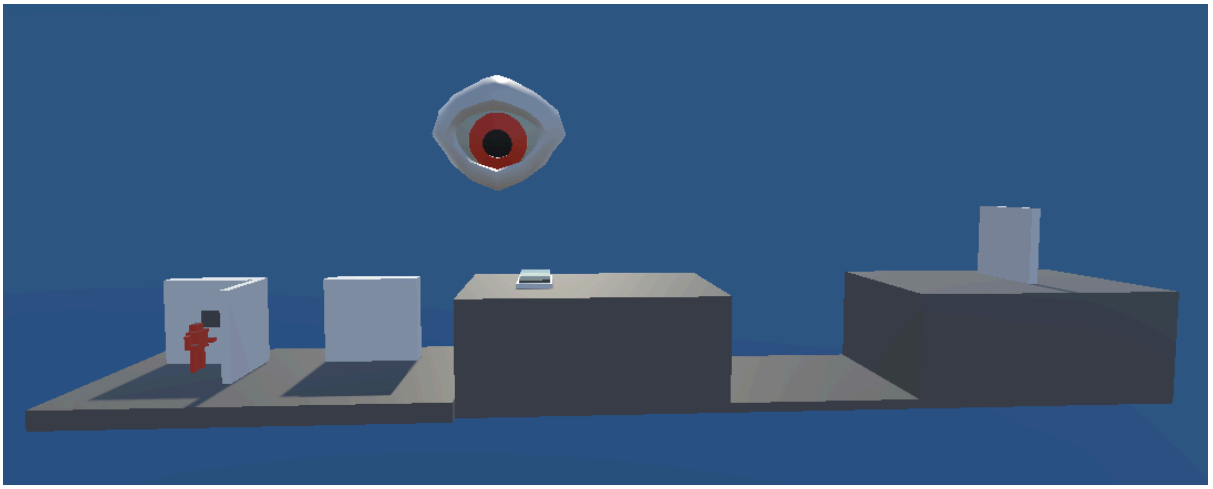
level prototype





- change parts of your character while you are the 'developer' of the game
- change attributes of the controller
- change ASPECTS OF YOUR CHARACTER

Mockup: Game Engine Prototype



Milestone II: Interim Demo

Critiques by Other Students

The first part of this milestone was to provide mutual feedback to game ideas. The other course members seem to like the approach we took on the theme, as many of them mentioned they are looking forward to experiencing the 4th wall break. Along with that, the other participants find the AI setting very fitting as well. This is a good sign as it reinforces our vision and indicates that players will very likely enjoy the game we are developing for the course. Regarding the gameplay itself, the comments also show that configuring the main character's stats is a nice mechanic and further plays into the idea of the 4th wall break. People also liked the story bits we presented in the pitch and even said that we should dive deeper into creative storytelling, which is something we will do for sure if time permits.

However, the main critique we received was that using an external text editor to change in-game behavior is a great risk. Some mentioned that it might confuse the more casual ones as they might be scared away if the solution is too difficult. Others warned that opening text files may take too long, especially if the player has configured another program like Visual Studio as the default program for reading text files. We see that neglecting that could potentially break the flow of the game. The solution that was proposed the most was to create a fake text editor which is embedded into the game software itself rather than using the default OS editor. One comment even suggested that this part of the game should not be required to finish the game. This, however, does not resonate with our vision as we really want to emphasize on the player cooperating with the main character from *outside* the game. But we want to ensure that the transition between the game and the file system is as smooth as possible. We will implement that by opening the text file automatically whenever the main character interacts with a terminal. Furthermore, we will open the file with a fixed process, i.e. the Windows Editor to avoid long loading times when another process is the default program for text files.

Another point we read frequently is that we should lean into that file interaction part as our main mechanic and focus less on the generic platformer part. That's why we had another brainstorming session about mechanics interacting with the file system. We definitely want to expand on the editing of text files as this is the main mechanic supporting the 4th wall break.

The feedback provided by the other participants was invaluable and we are very grateful for their participation. The changes we made based on the comments are:

- Putting more effort into making the transition between game and file (system) as smooth as possible.
- Focus more on the innovating file interaction part of the game.

Together with the conclusions we drew from creating our prototype, we further iterated on our initial design. We will definitely keep the external text file as a separate part outside the game itself because that is our main selling point and a crucial part to delivering the message of the 4th wall break. Especially because the people that played our digital prototype at the Pizza Playtest a month ago really enjoyed the text files popping up and said we should keep them.

Learnings from Prototype & Design Revisions

During the creation of the prototype, we encountered two major flaws in our design and therefore rethought our game design to overcome these issues. Those are the perspective and the main game mechanics.

The Perspective

Before creating the prototype, we were unsure about which perspective we wanted to choose for our game. In the pitch meeting, we said that we decided on a 2,5D perspective. Yet, this perspective brought some major issues. It especially made it hard for the player to see what is about to happen in the rest of the level. As the puzzles we had in our minds involved lots of challenges that had to be completed in a limited amount of time, it is crucial that the player can immediately grasp what to do after starting the timer. We thought about this issue ourselves when trying to picture how a level could look like in our heads. But also, observing people playing our digital prototype at the Pizza Playtest showed that this was actually a problem we will be facing.

That's why our first goal of creating the paper prototype was to try out different perspectives. To do so, we split the team in two groups, one for 2D and one for 3D. We then picked a random combination of puzzle elements from a list we created over the course of the previous meetings and tried to design puzzles with those elements. Both groups used the same combination of elements, but designed for a different perspective. This way, we could determine which perspective works best and which puzzle elements and mechanics make up fun challenges.

After playing around with different ideas for a while, we settled on the perspective for the game. To counteract the problem of not being able to see what is coming up in the level, we quickly realized that a three-dimensional perspective will work the best. Contrary to what we said in the pitch, the main character will also be able to move on the z-axis, so towards and away from the camera. This way, we have more freedom arranging the puzzles so that the player can get an overview over the puzzle as a whole.

Then, we further investigated how the camera would view the level. We thought about games we played and figured that a camera perspective similar to the tank game from Wii Play mixed with the marble minigame in Wii Fit could work quite well. They come with the advantage of being able to see the whole level at any time. Finally, we settled on a perspective like the one in Captain Toad because we really liked the verticality that can be utilized in game design. The player cannot control the camera to rotate the island as in the Captain Toad games.

Pictures of Wii Sports and Wii Fit

Perspective of Captain Toad

We chose this viewing angle because these little islands each level takes place to fit the theme of an AI training simulation, because they represent individual, separated simulations where a specific task can be trained in isolation. That also goes hand in hand with our vision of theming each level around a certain task the AIs have to learn. Viewing the level as a whole further supports our message of the player observing the AI character and helping it to reach the goal.

To summarize, the changes we made to the pitch version with regards to the perspective are:

- The main character is no longer limited to move to the sides, but can navigate freely through a 3D environment.
- The level is no longer made up of a single “lane” where the main character has to reach the right end, but is rather placed on a separated island with a certain goal point.

The Game Mechanics

As we tried to design levels using our envisioned game mechanics, it became apparent that we simply had too many of them. It felt overwhelming to design a level with so many possibilities. We discussed that and came to the conclusion that the player will also feel overwhelmed when playing the game, because they will likely be confused about what to do in the game.

So, the second learning from designing our prototype was that we have to revise the entire game design and make it clear what our game wants to be. Are we a platformer or a puzzle game? Do we focus on the puzzle elements embedded in the game world or emphasize on the file interaction? Currently, the files are only there to configure some objects in the level. Also, the fact that the player can write anything into a text file makes it harder to communicate what they have to change and which changes to the file actually affect the game.

We spend a lot of time to re-think our design decisions, because defining and shaping the game now is crucial to creating a solid foundation we can build on in the entire upcoming development process. Finally, we decided that we can limit the player’s possibilities by splitting the level into two clearly separated parts. The first part is navigating the AI robot to a certain room on the simulation island interacting with the in-world puzzle elements and tweaking the character’s abilities. After reaching the so-called “control room”, the gameplay switches to interacting with the file system. The goal now is to perform a certain task within the file system to destroy one of the guarding eyes at the great wall. There is one “mother eye” for each island, which can be for example destroyed by deleting a file named *eye.txt*. This process should get increasingly difficult and expand to finding a file that is constantly moving between folders, renaming files, moving files to special folders and so on.

This clear separation makes it easier for the player to grasp what they are meant to do and also make up for a nice change in the game’s pacing. Switching from editing a text file to more limited operations like renaming, deletion, and movement of files additionally makes it easier for us as designers to control what the player can do.

To further simplify the gameplay, we changed the ability tweaking from having sliders and skill points to an easier and more restrictive system. The AI robot now has a certain amount of module slots that can be filled. Modules replace the sliders, meaning they represent a certain ability, e.g. *jump*. These modules have to be equipped and take up a certain amount of slots.

In conclusion, the changes we made here are:

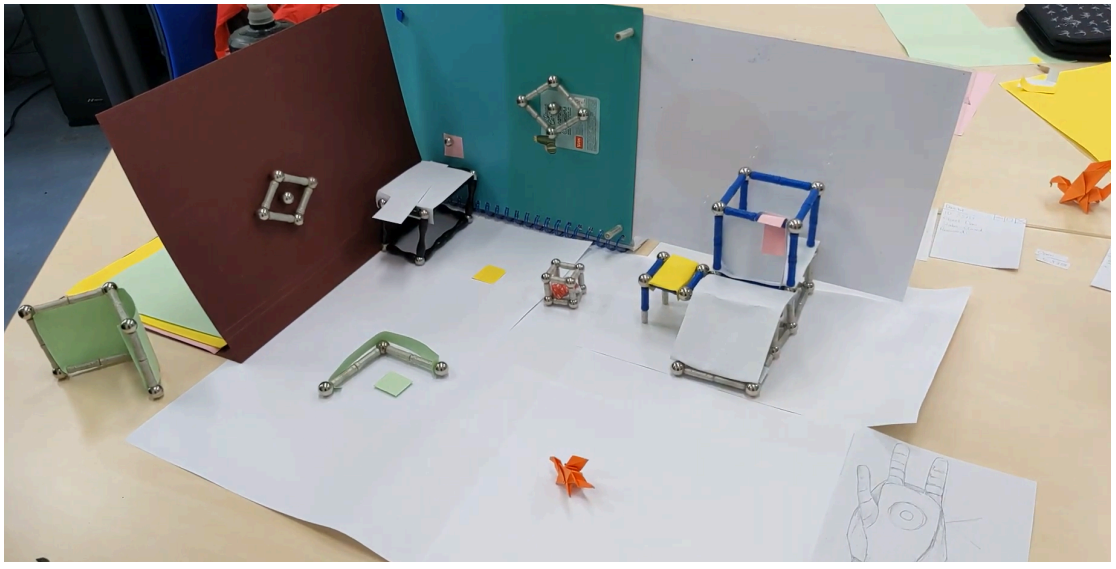
- Focus on the file system interaction and include a variety of ways to interact with that.
- Clearly divide the gameplay into two distinct parts; puzzle and file interaction, to limit the possibilities for the player and therefore avoid confusion and the feeling of being overwhelmed.
- Replace skill points and sliders with modules and slots to simplify the configuration of the main character.

All of the design revisions made based on the learnings from creating the prototype made it possible to iterate on our initial game design. We then continued with crafting the actual prototype.

Explanation of the Prototype

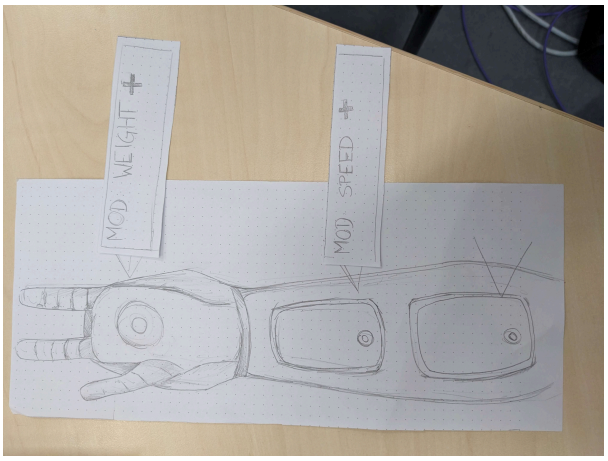
The following photos and sketches illustrate the prototype we've created for this milestone. The prototype represents a potential level of the final game. For a detailed walkthrough of the prototype, refer to this video. The three-dimensional nature of the prototype indicates the viewing perspective and structure of the level islands.

To communicate clearly which elements are connected, we made heavy use of colors indicating the objects the player can interact with. In this level, the yellow pressure plate moves the yellow platform while the green pressure plate makes the green wall move up. The goal in this exemplary level is to get into the blue box, which represents the control room explained earlier. To enter said room, a password is required. The pink sticky notes are terminals. The player is represented by the orange figure and can be moved by the player. On the top right in the first picture the reader can see a bigger version of the player. Last but not least, the diamond-shaped objects on the walls represent the eyes.

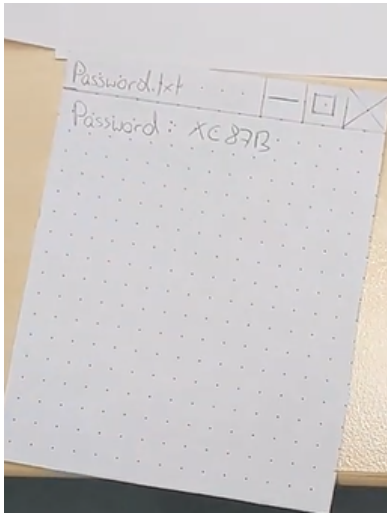


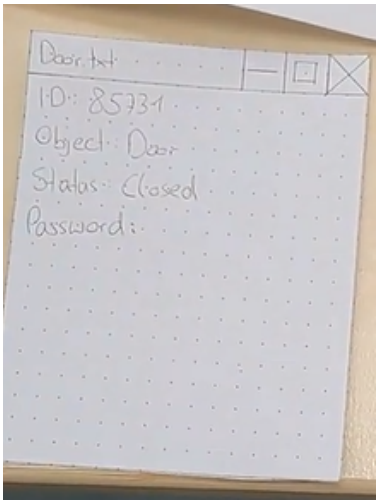
An Exemplary Level

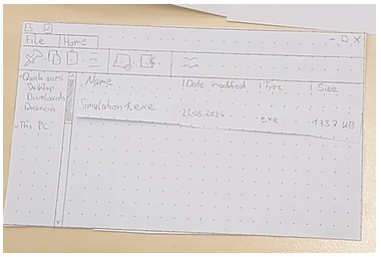
Additionally, we created paper mock-ups for both the module UI and the modules, along with the text files, the password string, and the file system. The player can equip modules by placing them into one of the robot arm's slots. However, they have to be hidden from the eyes on the wall while doing so. The "computer" observes that and places the corresponding paper sheet on the table in case a text file pops up or a folder is opened.



Module UI







Paper Versions of Text Files and File System

The solution here is to stand on the green pressure plate to move the wall up so the eyes' vision is blocked. Then, the player can increase the weight of the robot so it is able to push the orange box on the yellow pressure plate. That in turn makes the yellow platform move horizontally. Now the robot can reach the left platform where they find a size module that can be equipped from now on. When interacting with the pink terminal, a text file containing a password pops up. After standing on the green pressure plate again to increase the robot's size, they can reach the right terminal as well. There, the player has to insert the password they've found earlier into yet another text file to open the door to the control room for the robot. The robot can then open the file system for the player where they have to delete the *simulation.txt* file to destroy the simulation of this island and win the level.

Experience Playing the Game

This part contains the feedback we gathered from people playing our prototype.

All in all, the idea and the concept were well received and most of the mechanics were easy to understand. The text file was especially met with great interest and the fact that it pops up and is outside the game seemed to not break the game flow for all the testers so far. That is a very good sign as this was one of the main concerns present in the comments on the Wiki page and indicates that our design revisions had the desired effect. People also told us that we should lean more into the file interaction mechanics similar to some of the Wiki comments.

One mechanic some testers found difficult to comprehend was the pushable box that requires a certain minimum weight of the robot. In future iterations we will indicate the required weight with a label on the box. Providing a short summary of the robot’s current properties can act as another supporting element which will additionally help the players grasp how to configure the robot. Learning from that, we plan to provide clear feedback and indicators to the player for other mechanics, too, especially for timers.

We will also try to keep the color coding as they seemed to really help the testers understand which elements are connected. As accessibility becomes increasingly important in the gaming industry, we might also include additional symbols supporting the colors to assist people who have difficulties distinguishing colors, too.

The story's cohesiveness was often pointed out. The reason why the simulation contains terminals that can shut down the simulation is thus far not fully grounded. This should be considered if the time permits.

Milestone III: Alpha Release

The Current Status

At this point in the development phase of A Eye, we already achieved most of our functional minimum, the low target, and the desired target as defined in Milestone I:

- A character with adjustable values,
- Eyes that track the character,
- Terminals that open a text file and enable the player to change objects within the game,
- Game elements like doors, pressure plates, and time trials,
- Basic UI,
- Panning eyes,
- Decrypting parts in multiple level segments, and
- A custom character model.

Remaining Tasks

The features that were missing for these targets were (apart from polishing the features and iterating on the level design):

- Game over behavior,
- The levels including a tutorial level,
- A character that actively talks to the player,
- Custom animations for the main character,
- Embedding music and sfx,
- Menus (start, pause, and game over), and
- The tube TV shader.

However, we already made significant progress in most of these features. Firstly, we clearly defined the conditions and consequences of a game over state. Secondly, an entire level was fully fledged out in the form of our paper prototype presented in Milestone II and some other ideas were produced in the process of creating said prototype. Thirdly, the foundation for the character talking to the player was already implemented in the form of a basic dialogue system.

As described in the previous chapter, we made quite a few changes to our initial design. Luckily, our systems didn't have to change that much and we could pretty much reuse most of the implementation. Things we had to change were:

- The character controller has to be extended to 3D, which includes movement on the z-axis and the ability to walk on slopes,
- Changing the character configuration system from sliders to modules,and
- The file interaction system has to offer more file system operations.

This list was extended by functionality that we derived from our experience playtesting the paper prototype as well as tools that would assist our workflow and enhance our efficiency. These are, among others:

- The tube TV effect should look more like a CCTV camera,
- Said effect and post processing should be dynamically configurable from the code for providing a better indication of whether the robot is in sight of the eyes,
- The camera can zoom in on the player using the mouse's scroll wheel, and
- The main character should be able to display facial expressions to bring him alive.

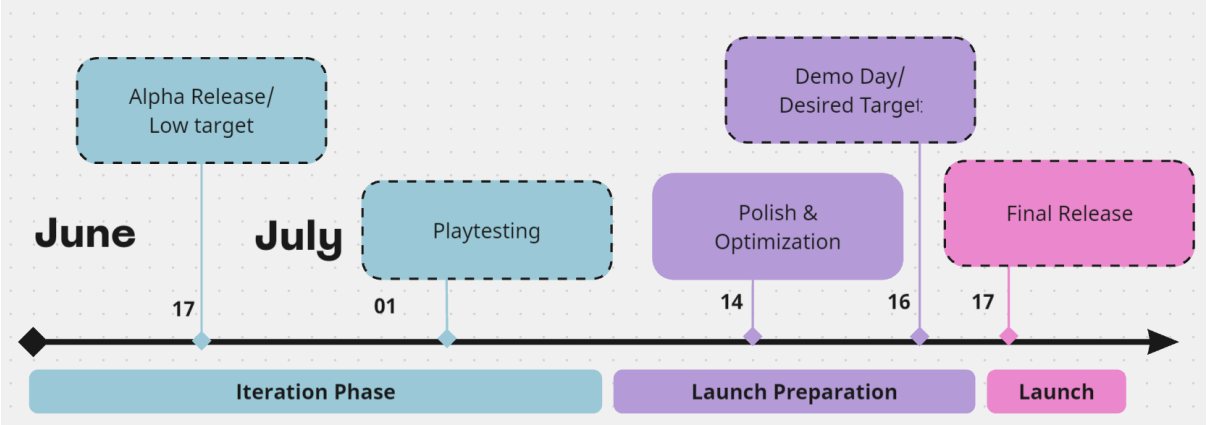
The Process and Major Challenges

For tackling these tasks, each of the team members continued to work on their field of responsibility as decided prior to the pitch meeting. Moreover, we distributed the new tasks nobody was assigned to yet. This section shows a selection of features and especially points out which tasks we had troubles with.

Project Plan and Timeline

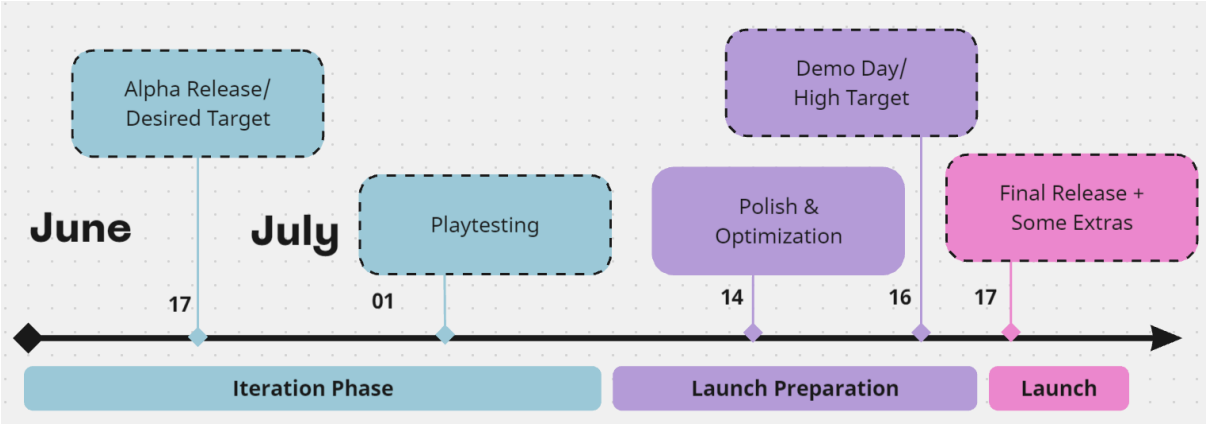
First of all, we had to rethink and replan the development phase due to our project plan and some misunderstandings.

In our initial timeline presented in Milestone I, we set the Alpha Release milestone to only contain the features of our Low Target, because we decided that playtesting these first is of highest importance. From the learnings we retrieved from said playtesting, we wanted to adapt and include the fitting features of the remaining targets. The desired target's deadline was set to be on the Demo Day in the initial project plan:



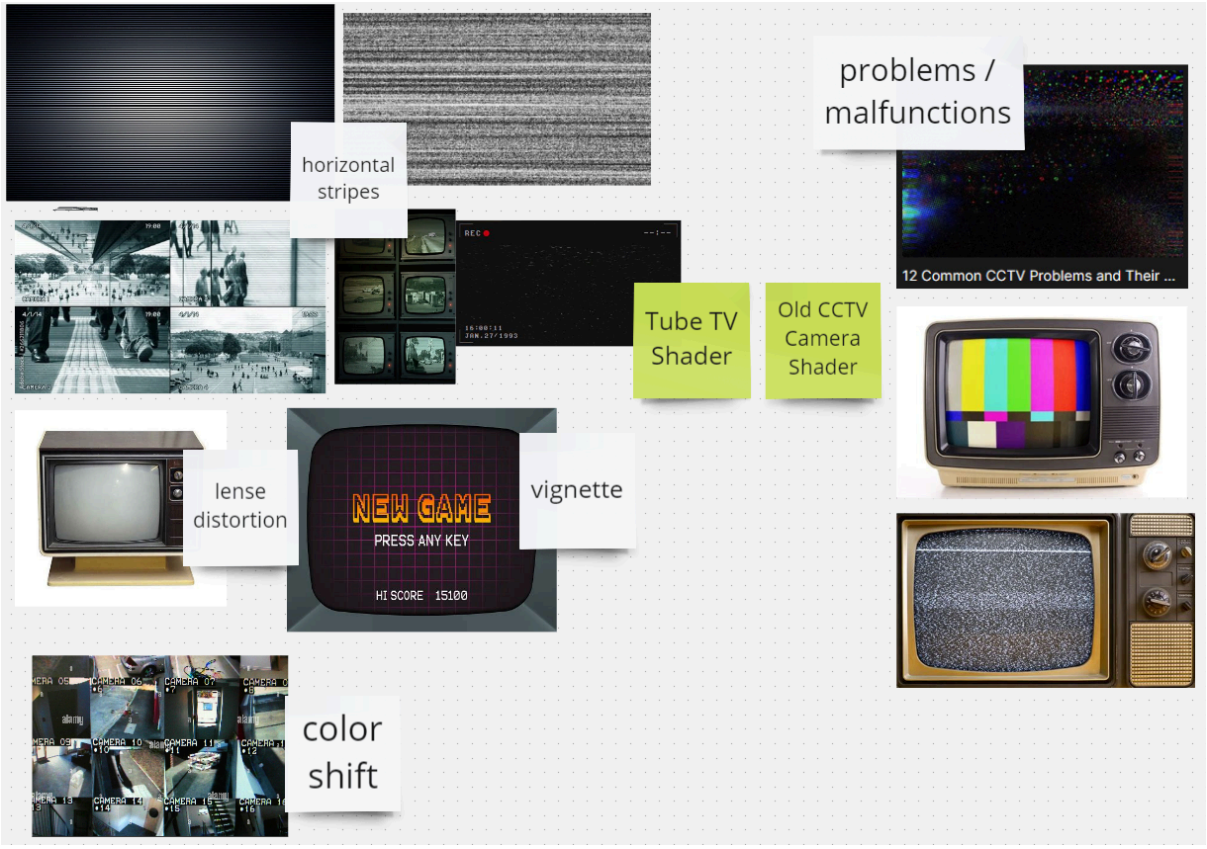
This, in turn, means that we had to adjust our pitch timeline and focus on the most important parts out of the three targets required for this milestone first. This included carefully selecting the fundamental features and iterating on them as fast as possible. In the end, due to a good prioritization, we managed to include all the functionality we planned to feature in our desired target. This, however, was only possible because we started implementing the key features early in the semester and because we could reuse most of them even when changing some

fundamental design decisions. Now, the project timeline is line with the course milestones:



Tube TV Shader

For the tube TV shader, we started by creating a mood board in Miro to get a feeling of what we want it to look like:



The most important elements can be split into two parts: post processing effects and shaders. Post processing effects provided by Unity that we could utilize were lens distortion to imitate the rounded screen of an old TV, and a vignette to create the illusion of the screen being lit from behind. Additional features we had to create ourselves were horizontal lines that are often present in CCTV camera images as well as a slight green hue which is overlaid over the image.

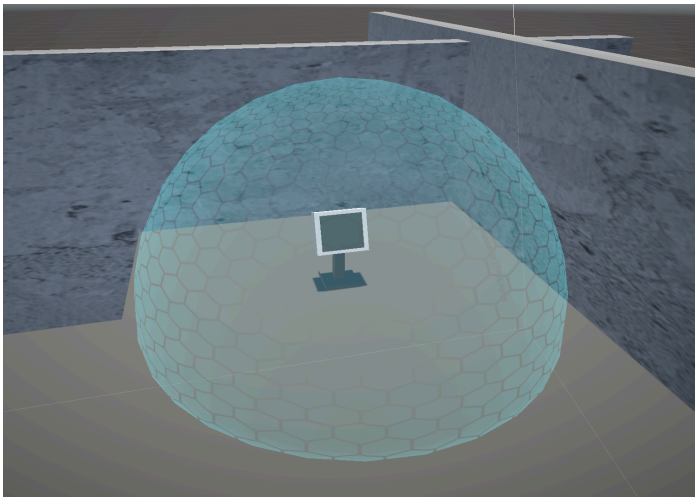
A major problem we faced here is that the shader does not affect the UI, which is something we will have to change because it otherwise breaks the illusion of watching the game on a curved old screen. This is something we haven't solved yet and therefore will be one of the first tasks we are going to tackle in the upcoming polishing phase.
Crafting shaders in general was quite challenging for us, as we are all quite new to creating visual effects.
For better usability, we decided to also add C# scripts that can tweak the shader's look from code, so the shader can be changed at runtime. This is crucial for providing feedback to the player, especially to indicate when the robot is seen by an eye.

The Interaction Between Pressure Plates and Pushable Boxes

Physics made us rack our brains sometimes. Especially the interaction between pushable boxes and pressure plates caused a lot of problems as the boxes had to be pushed onto the plates which itself has some volume that does not allow for any object to protrude it. That's why the box overturns at this point, which looks odd.
The fix for this was to remove the collision between box and pressure plate stand. As the plate itself isn't that high, it looks believable, especially with the tube TV shader overlaying it.

Level Building and Level Designing

Step by step, we converted our paper prototype to a digital level. For this, we also selected actual assets for all the key puzzle elements. For those that we didn't find a suitable asset for, we created stand-in models out of simple shapes. Here, we made sure that they look decently good and that the player can immediately grasp what object it should represent to make the levels as readable as possible.
Because our desired target demands three game-ready levels and we felt that our initial tutorial draft was way too long, we split the tutorial into two easily digestible levels. The third level then is the digital version of our paper prototype described in Milestone II.
A thing we observed when watching people play our paper prototype was that it has to be clear what the goal of each level is. As, in our case, the goal is to reach the control room, we found that we should show the player that this is the place to go eventually. It is also important to clearly communicate that the room is locked at the start. We decided that this is shown best by enclosing the terminal that is used to enter the simulation's file system in a force field like this:



Another thing we learned from this milestone was to design the tutorial after the rest is finished. We started designing the tutorial level early in this stage, but then realized that a high-quality tutorial would require more time and more knowledge about the puzzles occurring in the rest of the game. So we saved that initial draft and revised it later to make it better prepare the player for the upcoming challenges.

Basic Game Loop

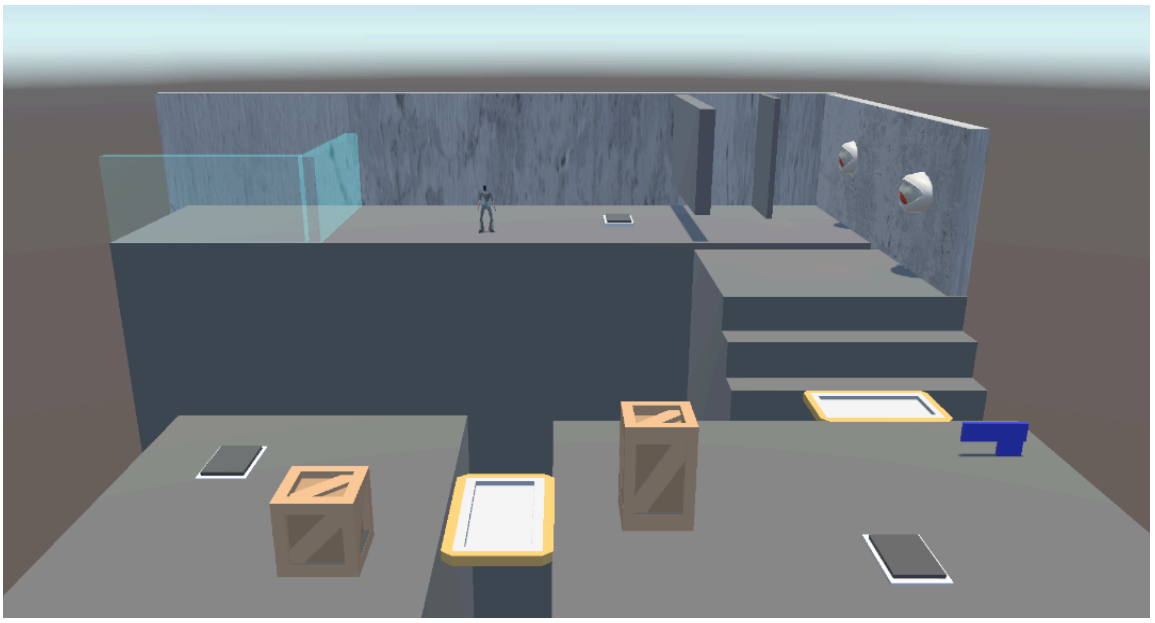
To present the current state of the game, it is best to explain the basic game loop and what features it contains. The game starts with a main menu for navigating to the tutorial or the first level. Both are explained in detail in this section.

Tutorial

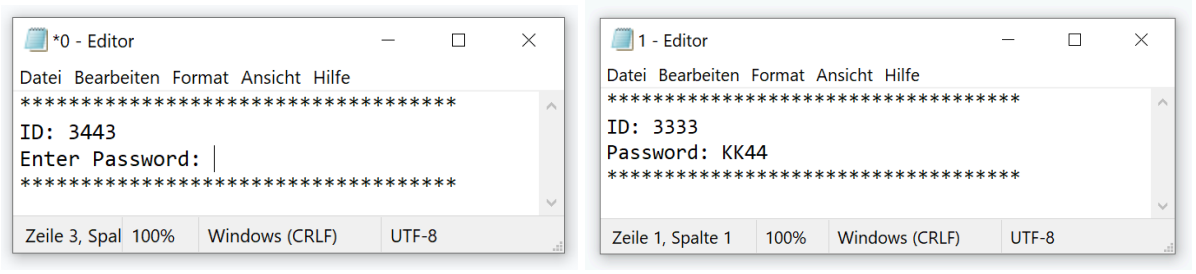
The tutorial of the game has the goal to teach the player the many different mechanics implemented into the game step by step. To do so, we made a list of all the mechanics and knowledge the player needs for tackling the levels, and introduced them one by one in a small and simplified playground. Here is a sketch we created in the planning phase:

We decided to start the tutorial by introducing the pushable box together with the pressure plates. The robot has to push the box onto a pressure plate to advance in the tutorial. In the upcoming part, the player gets once again tasked to push a box onto a pressure plate, but this time, the character is not heavy enough. Therefore, they have to pick up and equip a module that increases the weight of the caracter. With this increased weight, the player is now able to push the box onto the pressure plate and is once again able to go to the next part. After that, the player has to collect and equip a jump module to jump up a few stairs. Once the player reaches the upper level, one of the main mechanics of our game gets introduced: the eyes. As long as the robot is seen by the eyes, the player is not able to change which modules the robot has equipped. Therefore, players have to look for spots where the eyes don't see the robot in order to change its configuration.
For the next part of the tutorial, they have to collect another module, this time one that changes the size of the character. Once the player selects the module in a spot where they cannot be seen, they are able to slip through a small gap between the wall and the ground to get to the next section. In this section, the speed module gets introduced. For this, they simply have to equip it and step on a pressure plate which triggers a door to open. They then

have to be fast enough to reach the end of the long hallway before the timer reaches 0:00 and the door closes again.
The complete first part looks like this:

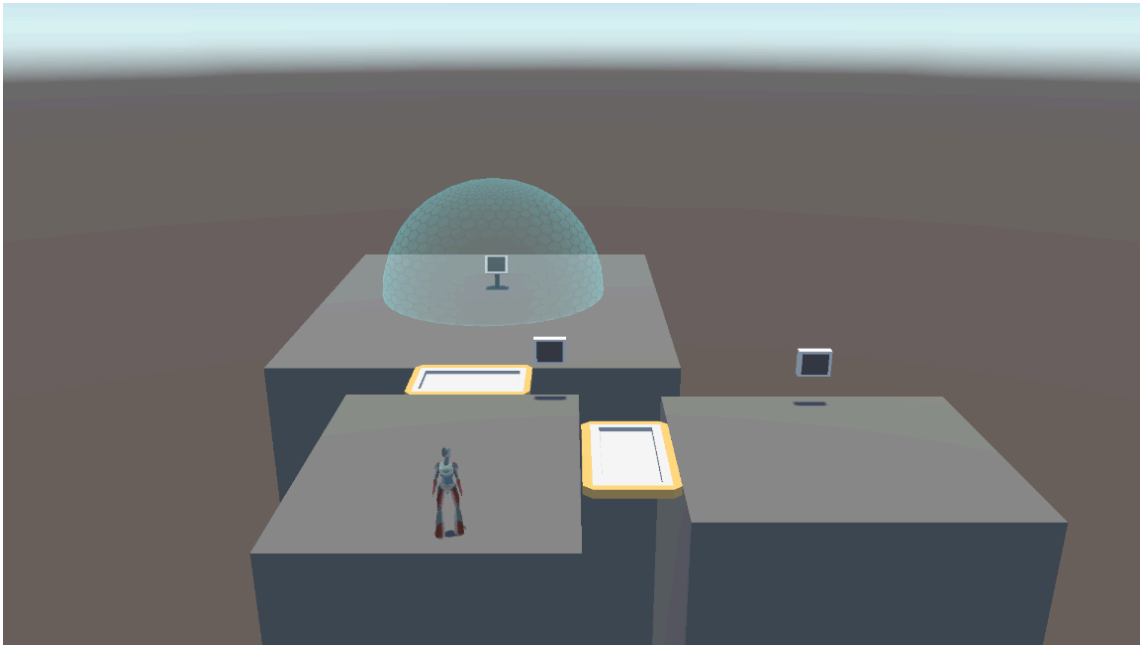


After completing the first half of the tutorial, the player gets sent into the second part of the tutorial. As said before, the tutorial is split so it is easier for players to digest. We can also communicate that there are two different types of gameplay in the game, which are clearly separated. In this second part, the player learns all about the intricacies of the file managing part of our game. They now see a terminal with which the robot can interact. Once they do that, an external text file opens which prompts the player to input a password. Since they do not know it yet, the player has to look around and find another terminal, which opens another text file containing the password. Once they go back to the first terminal and input the password, the force field that is ahead of the player dissolves, which reveals the main terminal of the level. Once the robot interacts with it, a Windows file explorer with a *tutorial.exe* file opens. The player has to delete it in order to proceed to the first level.
Here are basic examples for the two text files mentioned in this paragraph:



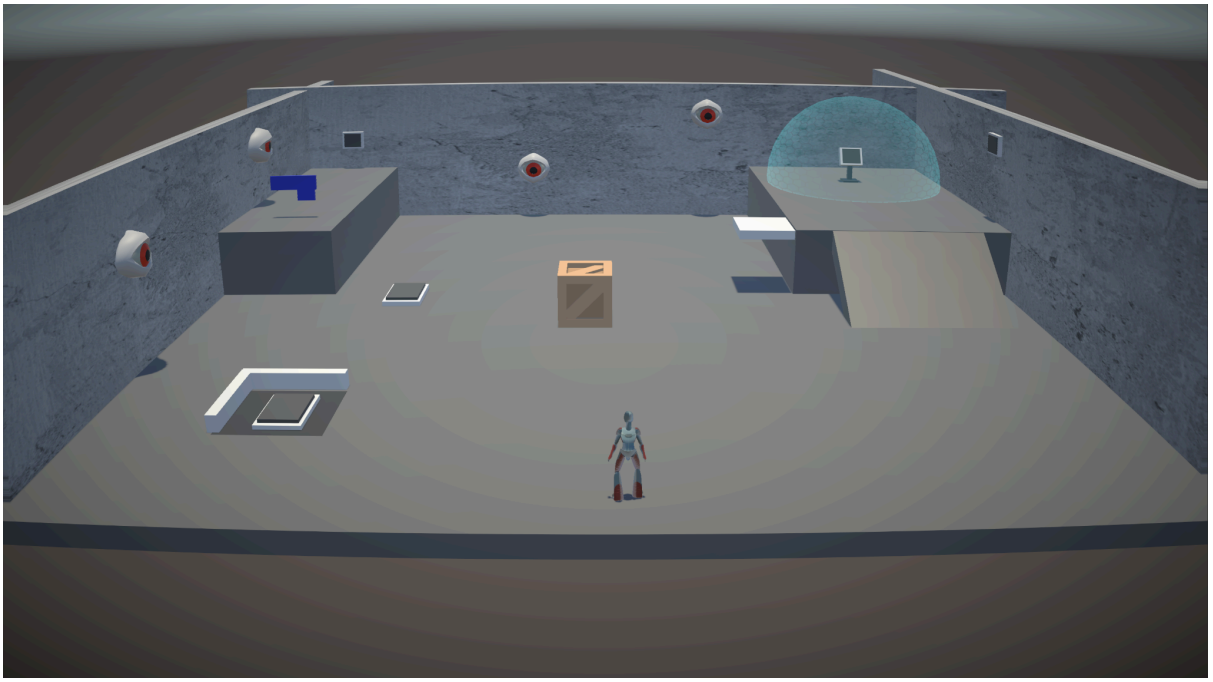
We chose this structure because it allows us to introduce the mechanics we built one by one, giving the player the chance to get to know each ability without getting overwhelmed. Additionally, each part of the tutorial builds on previously gathered knowledge and puts said knowledge to the test to ensure players will be able to solve all the upcoming puzzles the game has to offer.

The second part of the tutorial looks like this as a whole:



First Level

The first level has the same layout as the one we designed prior for the paper prototype which can be seen in Milestone II. The main goal here is to access the main terminal to delete the simulation file as introduced in the tutorial stages. Said terminal is protected by a force field. The player has to solve some puzzles made up of various mechanics such as pushable boxes, pressure plates, and the different modules. After solving these puzzles, the robot is able to reach a terminal that contains the necessary password. Using that, the player can now disable the force field and delete the corresponding simulation file to beat the level. Here is a screenshot of the level:



The Next Steps

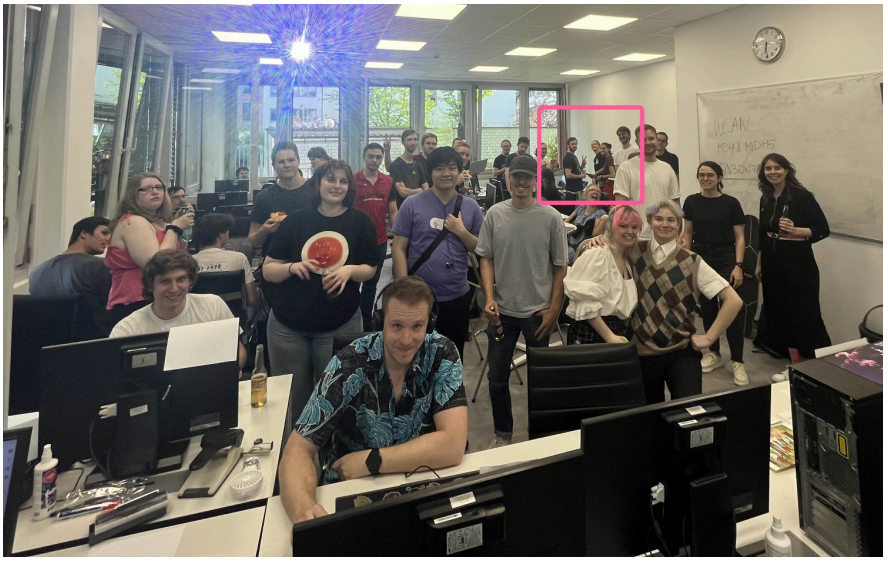
As of the current state, the implementation is nearly complete, except for some polishing tasks that impact the looks and performance of the game.
Upcoming tasks include:

- Populating the greyboxed levels with assets,
- Incorporating the finalized sound effects and music layers,
- Recording voice lines to voice the dialogues between robot and player,
- Designing and incorporating more levels,
- Making the robot react to certain events in the game and showing the respective facial expression,
- Selecting assets for all level elements, and
- Improving the looks of both UI and shaders.

Milestone IV: Playtesting

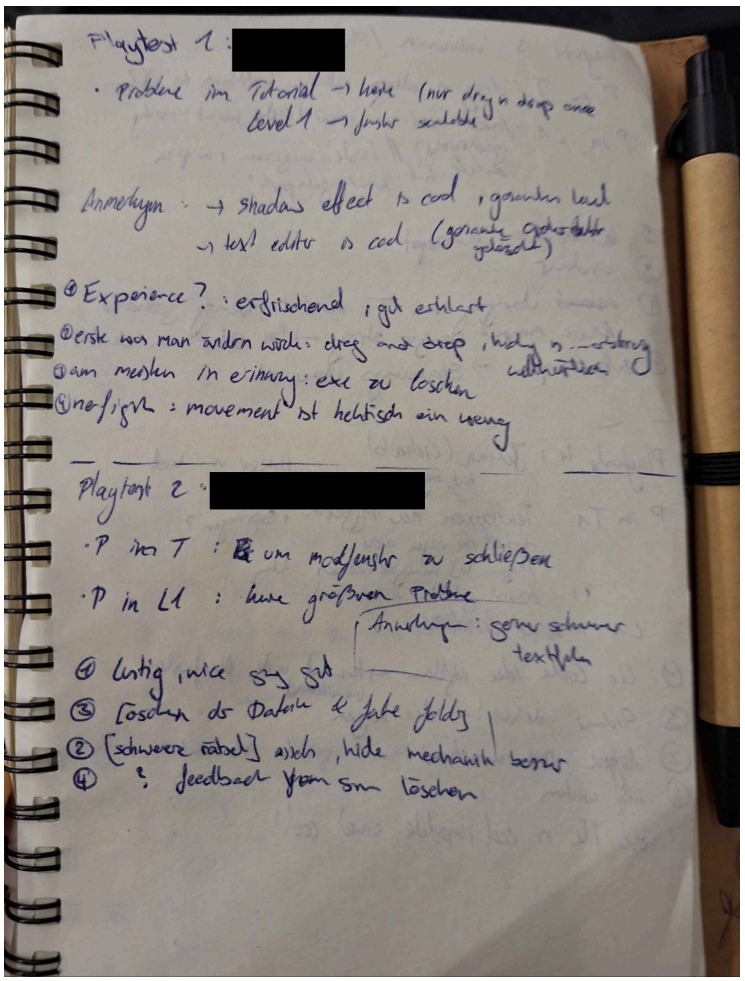
Organization of the Playtesting Session

For the playtesting session, we chose to attend the “Pizza Playtest” on the 20th June, an event where game developers and game enthusiasts come together to playtest games with varying stages of production. Here is a picture of us at the playtest event:



We at the Pizza Playtest (20.06.2024)

Naturally, we introduced our own game as a prototype and had multiple people try out our demo. It was important for us to explain the context and progress of the game, so people would not approach the game with the wrong expectations.
We thanked our playtesters for showing interest and told them we were testing the game, not their skill. Food and beverages were provided by the organizers of the Pizza Playtest.
Our demo consisted of the two tutorial levels as well as one demo level and a very basic main menu. Additionally, it included the first draft of the main music and some sound effects. However, we didn’t bring headphones and the room was very crowded, so most of the participants weren’t able to listen to the soundtrack.
While the players engaged in the game we would take notes on what the players did and what differed from the intended player behavior. After the play session, we would ask the testers to answer a few questions which can be found in the following.
We had 10 platesters with most people having some knowledge of games or gaming in general.



Excerpt of the Notes Taken During the Play Sessions

Playtest Interview Questions

After the participants finished playing our demo, we asked them the following questions:

1. How did you like the overall concept of the game?
2. What is the first thing you remember about the game?
3. What is the first thing you would change?
4. Was there something that annoyed you, and if yes, what was it?

The first question is very open, so that the testers could freely elaborate on whatever they felt saying. This was especially important for us because we didn't want to constrain them or force them in a certain direction. The remaining questions helped us to get a deeper understanding of what people enjoyed and disliked regarding the game. The following sections summarize the gist of the answers grouped by question.

How did you like the overall concept of the game?

In general, all the playtesters liked the game as a whole. The 4th wall break especially seemed to be well received. Most of the players described their experience with adjectives. According to them, the game itself was "cool" and "funny", mixed with "unusual", "confusing", and "refreshing". The latter was mostly related to the file interaction part of the game, which people were really surprised of. Some even reported that playing the game was, quite literally, an "eye opening" experience. All in all, this is very positive feedback, and seems to be the consens between game developers, gamers, and non-gamers.
The tutorial was described as "well explained" and specifically pointed out by one of the playtesters, which also is a great indication that the way we introduce all the mechanics step by step is a good one.

What is the first thing you remember about the game?

Nearly everyone said that the interaction with the file system was the thing they remembered when thinking back to playing the game. Especially the deletion of the *simulation.exe* file or the editing of the *.txt* file were named frequently. Some players especially reported that they really loved the moment the text file opened for the first time because it was so unexpected and surprising. Therefore, it served as a great twist for the game. This is a great feedback for us because as said in the previous chapters, this is the main selling point and distinguishing feature of the game.

What is the first thing you would change?

Here, the participants actually gave very distinct answers and there were only a few duplicates. One thing that was frequently named was that they wanted to be able to equip the modules via drag and drop. In the demo version, clicking on a module automatically

assigns it to the next free slot.
There were also some people complaining about the difficulty of the puzzles in general. However, some said they were too easy, others found them really challenging. Some even proposed that the password should be hidden away or blocked behind additional puzzles. Our takeaway here is that puzzle complexity is hard to tweak correctly and meet everyone's expectations, so we will focus on improving the clarity and legibility of the mechanics and puzzle design while keeping the initial ideas.
Other minor improvements people demanded are:

- The hiding mechanic: It is not completely clear why the robot has to hide and where this is possible.
- The text files: Expand on them and include more logic. At the same time, make it clear how to interact with them in the expected way.
- Visual appearance of the modules: Maybe add different colors to the modules for an immediate indication of the type of module. Reflect the equipped modules on the player model.
- Visual improvement in general.
- Level transitions: When beating a level, don't immediately load the next one, but rather give some feedback and tell the player that the level was beaten.

Was there something that annoyed you, and if yes, what was it?

This question was especially tailored towards identifying the most problematic parts about the game that players really disliked. We included this as a separate question because we felt like testers would maybe keep these thoughts secret on a friendly playtesting event like the Pizza Playtest.
The most named aspects include both level design and the player movement. The movement was found too hectic and not precise enough to maneuver on the tiny tutorial islands. The placement of the pushable box additionally made the first island hard to beat quickly. Later in this level, we placed a gap after the jump module to force players to use said module. However, lots of players didn't see that gap or mistook it for shadow.
In addition to that, there were no keybinds shown in the game. These could only be grasped by reading the dialogue in the text boxes.
Another point was that deleting the simulation had no feedback besides loading a new level. People demanded more feedback and reward there.
Two people said "nothing", which of course makes us very happy.

General Notes taken during Playtesting

This part contains any additional notes we took while watching and listening to the participants during the playtest. Therefore, it is not specifically related to any of the questions listed above.
Most people felt that the tutorial was well explained and hardly encountered any problems while playing them. Similarly little problems were encountered in the demo level, however

some did forget some keybindings like modul-slotting or interaction. Using the file system was new to most of them and was well received.
Many players expressed the wish to see more puzzles using these files as well as more difficult ones. Some noted that the story needed some better explanation as they were unsure of the game's setting.

Changes Made After the Playtest

Since our playtest took place on 20th of June, we had a good amount of time to adapt the feedback we got from it.

The most important things we changed since the playtest are presented in the following. They are grouped into bug fixes, additions, graphical improvements, and quality of life changes.

Additions:

- The game now features a dedicated main menu which supports the story by creating the illusion of moving into a screen to enter a level.
- In the tutorial the robot interacts more with the player
- Added a game over transition before the level reloads to communicate why the level reloaded

Graphics:

- We changed the gray default boxes in the tutorial to look like a set up scene and a simulated space by adding a grid texture to them.
- In general default objects were replaced by assets
- Some additions to shaders
- Reworked the Module change interface

Quality of life:

- The modules can now be equipped and removed via drag and drop.
- interactable objects now also show the key required to interact with them
- Dialogue now shows the keybind as an icon to improve readability of keybinds in dialogue

Bug fixes:

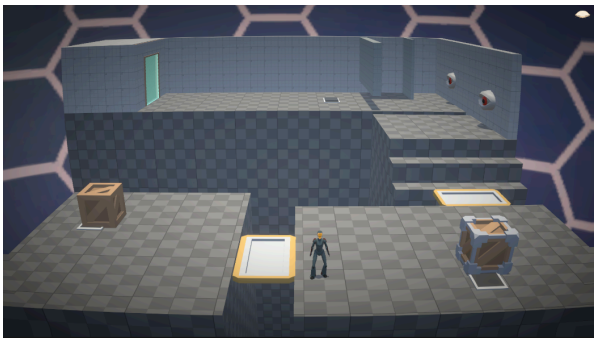
- There is now a death zone in the second part of the tutorial.
- The main music does not stop and restart when switching scenes.
- reload the text file if the player changes wrong information *(planned).*

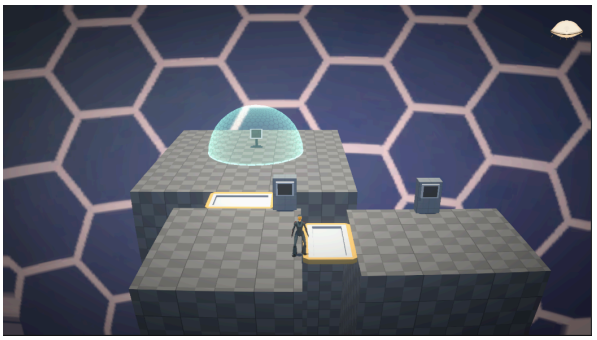
Impressions of the Current State of the Game

This section includes lots of screenshots to convey the current state of the game. As this chapter should reflect the learnings we took from the playtesting sessions, this part is highly focussed on the features we added or improved after the playtest.

Feeling of Being in a Simulation

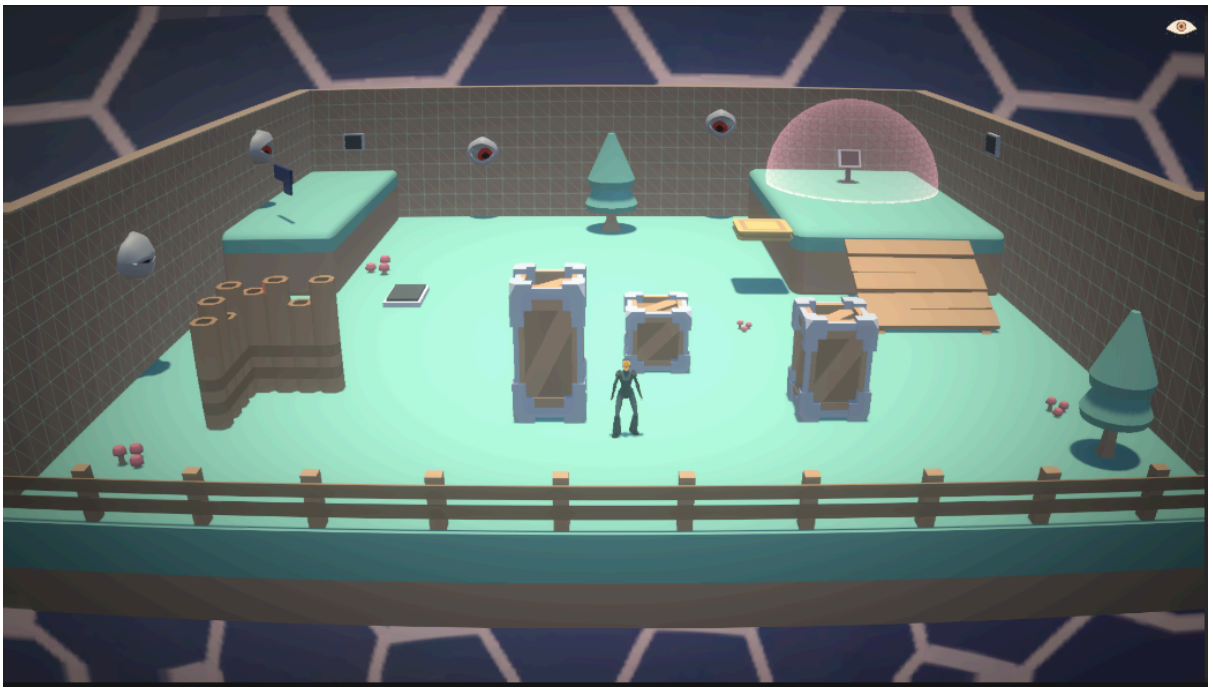
In the tutorial levels, we added a checkerboard pattern to the islands to make them feel simulated and crafted:





Assets in the Levels

We improved the looks of the first level by replacing our stand-in objects with themed assets. We also changed the color of the energy shield protecting the main terminal from blue to red:



We plan to add other levels in their own distinct theme and further improve the looks of the terminal and the eyes.

Main Menu and Level Transition

The main menu now is a control room where some guard is observing different scenes on a bunch of screens. This is meant to resemble the role of the player. When clicking on one of the screens, the camera zooms in and loads the respective level. This supports the story of watching an AI robot in a training simulation as well as the 4th wall break:

Here is a GIF showing the improved transition between levels and the main menu:

Next Steps

At this point in development, we are approaching the final release of the game together with the presentation on the Demo Day.
We have not been able to implement all of the given feedback and aim to implement more of it in the near future.
Some bug fixes, as well as some Graphical changes still have to be made.
Our next goal is to reach our high target which means that we will implement one more level and polish it until our time limit, the Demo Day.

Milestone V: Final Release and Conclusion

Summary of the Final Results

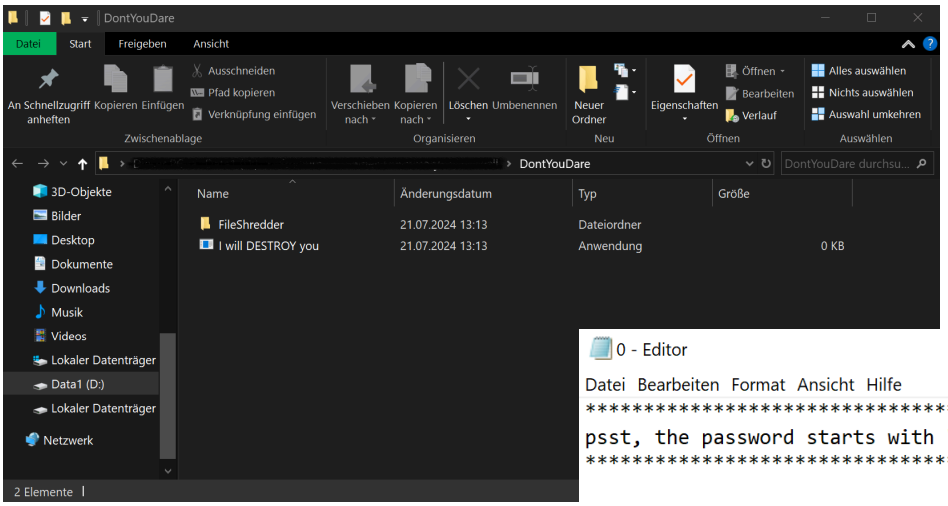
This final chapter starts with a summary of the final version of the game. Milestone III describes what the project's status was at the alpha release state. At this point, the game featured a very basic main menu, two tutorial levels and one demo level as planned for our desired target. It additionally already included the final version of the tube TV shader and the module mechanics. We included a file explorer game in that demo to test how people would react to that. The first main music theme and some sound effects were also present.

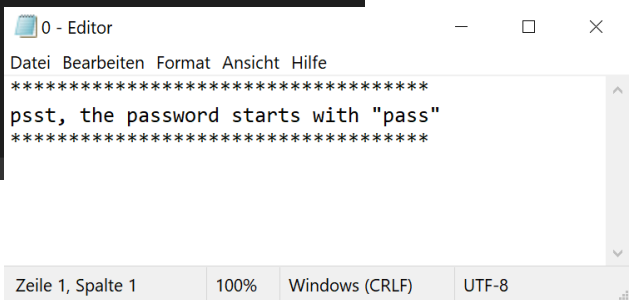
Since then, we added two more levels, and filled all levels with actual assets. We also included a proper main menu featuring a nice level transition supporting the story of watching the AI robot via a CCTV recording image. In the Questions part, we go more into detail about what we added exactly and which aspects we polished, so the following part will present some impressions from the final game.

Main menu:

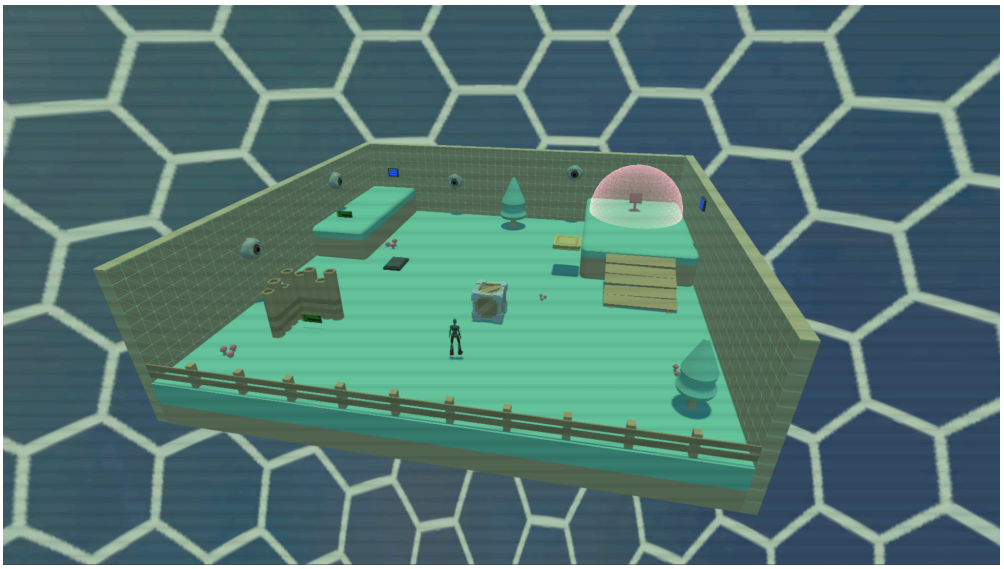
Slotting of modules:

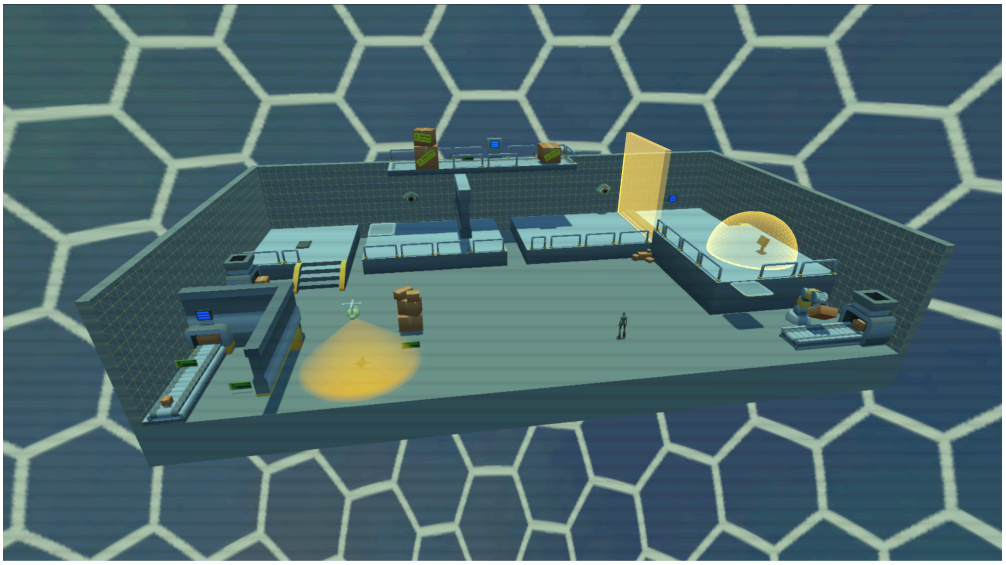
Interaction with the text files and the file explorer:





All levels with placed assets:





Experience During the Class

For the most part, our initial drafts and ideas can be clearly recognized in the final game. The overall story and setting, the eyes as a central threat, and the mechanics of configuring the character's abilities and interacting with external files in some way. As stated at several points in this notebook, we changed many design decisions during Milestone II. To repeat the most important ones here, we changed the level structure from a single lane to a Captain Toad-like island, replaced the skill points with modules that are either equipped or not, and simplified the purpose of the text files while expanding the file interaction on the file explorer. That means, we ruled out the ideas that were too complex or replaced them by a simpler alternative. From the start, we defined which aspects should define the game and therefore won't be removed, especially the file interaction part, which was criticized in the pitch meeting.

The theme of the levels changed, however. At the beginning, we wanted every level to feature other robots working in the background, but we didn't manage to model and animate all of them as well as making them work so they would improve the experience. Moreover, we ended up designing the levels so they look like actual fields an AI robot can be employed in, like a garden, a house, or a factory storage. This differs from the initial idea of making all the levels very clean and white. They are still clean due to the style of the assets we used, but they are much more lively and colorful than in the sketches. IN our opinion, these visually appealing levels add more to the user experience, than sticking to the simulation theme too strictly. There's always a tradeoff between realism and fun in games.

All in all, we were able to follow the schedule, and it kept us organized by leading us through the development period. Later, in the Questions section, there will be more information on how the schedule influenced our way of working and planning.

Impression of the Course

The course met our expectations, as the structure and guidelines were clearly stated in the project structure and kickoff slides and could be viewed beforehand. This made it easy for us to decide whether this course is something we wanted to attend or not.

We really enjoyed the milestone meetings because hearing the presentations of the other groups helped us to compare our progress but also get an impression of how they were doing overall. It was interesting to see how the other games evolved as well.

The milestones helped us to stay organized during the development as they gave us something to orient and build upon. Experiencing things like creating a paper prototype will change how we approach future projects and showed us that testing ideas and concepts in an early state is highly beneficial.
Overall, we are very happy with the results, as elaborated in the following section.

Questions

This part provides answers to the questions listed in the project structure.

What was the biggest technical difficulty during the project?
There are two things that should be named here: Handling Windows explorers via code and combining shaders.

As stated in Milestone II, we aimed at extending the file interaction mechanics to include not only Editor text files, but also the Windows file system itself. We wanted this in the form of the Windows explorer that allowed us to play with more operations that can be done on files themselves, contrary to only editing a file in itself. The main problem we faced here was that the explorer is a process that is started by another process and therefore cannot be controlled by us from Unity. That made it hard to control where the explorer window opened, to open it in front of the game window, and to close it again. All of these led to confusion for players because they thought that their interaction wasn't processed due to the lack of appropriate feedback. Furthermore, as a result, players tried to interact again, causing multiple explorers to open. The latter we could fix after the Demo Day by only opening an explorer if there wasn't already one opened by this specific interaction.

The second technical difficulty had to do with our shaders, or more concisely, the tube TV shader. Because we used the wrong shader order, the overlaying tube TV effect hid all of our transparent objects, most importantly, the force fields. This could be fixed by setting the shader to render before rendering transparent objects. Another problem related to this shader was a design one. We tried to also apply the tube TV effect to the UI. That being the pause menu, the robot arm which is used to equip modules, and the eye indicator in the top right corner. Sadly, we didn't get it to work, probably due to the use of cinemachine cameras. However, we found that it actually supports the story we want to convey with A Eye, because you as a player are just watching the robot through a recording of a CCTV camera. But, your interface is on the PC you are using and therefore must not be affected by any effect that resembles the CCTV camera image.

What was your impression of working with the theme?
We did two or three iterations on the core design concept in our concept phase to find a consensus for our main game idea. Each of us created several sketches of ideas they came up with, and in the end we voted for the best concept. This way, we could ensure that anyone was happy with the initial design and that we decided on a solid fundament we could then work off of in the upcoming weeks.
All of us participated in game jams before, which are usually short competitions with a fixed theme and limited time to create a game using that theme. That's why we were all used to working with a single theme beforehand, and coming up with the initial design went really well because of that.

"The Wall" as a theme was quite challenging for us because it was very open and we had a hard time coming up with good ideas. The idea of the 4th wall break was proposed by two different members, so we knew that we wanted to go in that direction relatively early in the

concept phase. We then settled on the type of game and aspects like basic story, tone, and core gameplay.

All in all, this way of coming up with a game concept worked really well for our team, and we were thankful that we had something to start with, because restrictions always require creative problem solving.

Do you think the theme enhanced your game, or would you have been happier with total freedom?
As said before, the restriction really helped us coming up with initial ideas, because each of us immediately had something in mind that they conotate with a wall. Something, that sparked the first idea and could be iterated on over and over.

However, at some points in development we experienced that the theme restricted us too much in certain periods. As we were revising our initial concept in Milestone II, we discussed to completely change the perspective and structure of our game world and what the player can see. We also thought about changing the way the eyes worked, but that would have required them not being stuck on a wall. Because we definitely wanted to keep the walls in our game as a central threat because we wanted to match the theme, we couldn't change our design in the way we wanted at this point.In the end, we found another solution that worked perfectly for the effect we wanted to achieve, so we managed to keep the walls.

What would you do differently in your next game project?
Milestone II was a very important step in our development period. We had to rethink our entire design and question everything we settled on before. That was because we realized that the game won't work out in the way we anticipated prior to the pitch and it was really hard to design any clever or good puzzle with this plethora of options.

We will definitely remember this meeting of revising the initial design because it improved the design immensely and made us create a better game than we would have been able to if we stuck to the initial plan.

So, we will try to validate the design as soon as possible in future projects and test it with real players in an early state, just as we did with the paper prototype in this project.

What was your greatest success during the project?
As the inclusion of external files in the gameplay is an innovative concept, we didn't know how players would react to it before putting it in front of people at the very first Pizza Playtest we attended in April. Seeing that everyone really enjoyed it and found that the files were an interesting mechanic was a huge success for us, especially in this early stage of development. It showed us that our concept actually worked and players can really enjoy playing such a game. This realization was of highest importance because it proved that we are on the right track and can continue to investigate that idea.

It was great to see the surprise in the players' faces and to hear that the consensus of all testers was that we should definitely keep the text files because they hadn't seen something like this before and found it really refreshing and interesting.

Another small but noteworthy success was that we could reuse almost any code and functionality we created prior to changing our initial design completely, because of the way we programmed it. From the start, we decided on making the features as modular as possible, which highly helped us at that point because we only had to change the way things were connected in some areas to achieve our new vision.

Are you happy with the final result of your project?
Yes, definitely. About three weeks ago, we arrived at a state where the game was functioning and at a state that it could be played. So, we could focus on polishing the mechanics, the looks, and the feel as well as adding in more content in the form of two entirely new levels from that point on until the Demo Day.

And, especially in these last couple of weeks, the game changed a lot on the surface. We replaced any placeholders with actual assets, we made final changes to the shaders, we improved the usability by providing an updated tutorial, we added music and sound effects, we included voice lines, and created a proper main menu which sets the story in a subtle way.

These final weeks really shaped how the game feels when playing and we are very proud of what the final result now looks like. This gave us an additional motivation boost, so we also crafted custom models and animations for the second level in the form of crash dummies. This is something we are especially happy with because it gives the game this extra detail.

Do you consider the project a success?
Based on the previous answer: Also yes! We achieved anything we wanted and could also add some extras like the voice over or details like the crash dummies. People playing the game also had a great time and we received very positive feedback at both the Demo Day and the Pizza Playtest.

To what extent did you meet your project plan and milestones (not at all, partly, mostly, always)?
Regarding the project plan and the milestones, we always delivered on time and without any major difficulties or time concerns, so that was a huge success.

Concerning our targets, we achieved almost all points included in the high target, except for features we ruled out during our design revision meeting. Only the idea of a decaying level in the form of missing textures didn’t make it into the game.

Moreover, we didn’t include any of the extras into the game. Adding multiplayer and support for other operating systems would have required too much time due to designing and building entire systems for that. A character that is nervous when being seen and happy when hidden also was not included, but the basis for that is technically in the game in the form of the facial expression shader. That would be something we wish to include after the course.

What improvements would you suggest for the course organization?
At some point there was confusion with the milestones, which had us change our initial timeline and prioritize the features we were working on, which required extra time and energy. From the milestones as stated in the project structure document, we thought that there would be more time to actually implement the game in code. So seeing that period shrinking down induced stress on the entire team.

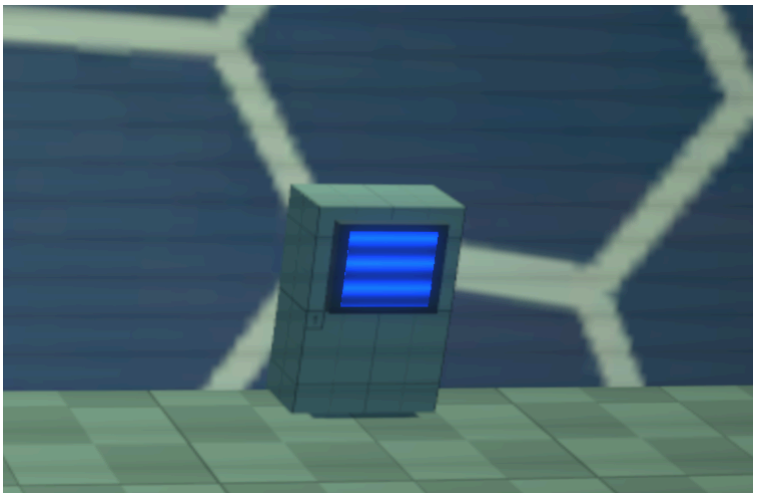
Fortunately, we already had most of the features implemented as we worked on the digital version of the game alongside of creating the paper prototype, we eventually had no issues with achieving our desired target in the shorter time frame, but we definitely had to spend more time working for this course in these weeks than we planned.

These are the only improvements, we are very happy with the organization of the course and the concept in general.

A Short Outlook

To conclude, we want to finish this notebook by giving an outlook on what we want to do with A Eye in the future.

After the Demo Day, we created an itch io page for the game which can be found under this link. The page includes a short summary, explains the controls, shows the poster and some screenshots of the final project, and provides a downloadable version of the game. For this version, we added a new shader for the terminals, so that can be identified much more easily. Here is a screenshot that shows their improved look:



A feature we want to add apart from that in the future is a cooldown for the terminal interaction. While watching people playing our game on the Demo Day, we observed that the text files and explorer windows take some time to open, especially on lower end hardware. Therefore, it is important to avoid the creation of duplicated windows, which in turn confuse the player. In addition to the cooldown, we could also display a message or a similar

feedback to indicate that something happened as a response to the interaction done by the player.

A tiny improvement would also be to add a proper logo and to rename the executable for the build to be “A Eye”.

Acknowledgments

Last but not least, we want to thank everyone who participated in this project.

First of all, thanks to Prof. Dr. Rüdiger Westermann and M. Sc. Christoph Neuhauser for organizing this course in the first place. Thanks to the other course participants as well as all our testers at the Pizza Playtests and the Demo Day for providing invaluable feedback and suggestions for improving the game.

Special thanks to Chris Kohler, who composed the adaptive music and crafted the sound effects for the game. Thanks to Sabine Röggla for recording the voice over for our AI robot.

Your Team Pricks & Bricks,

- Rebecca Ahmed,
- Michl Bayer,
- Sebastian Geheeb, and
- Kerstin Pfaffinger.