Interim Report

DupliCat

14.12.2022

Brush n' Rush

Clemens Fromm

Georg Eickelpasch

Kim Simon

Klejdi Sinollari

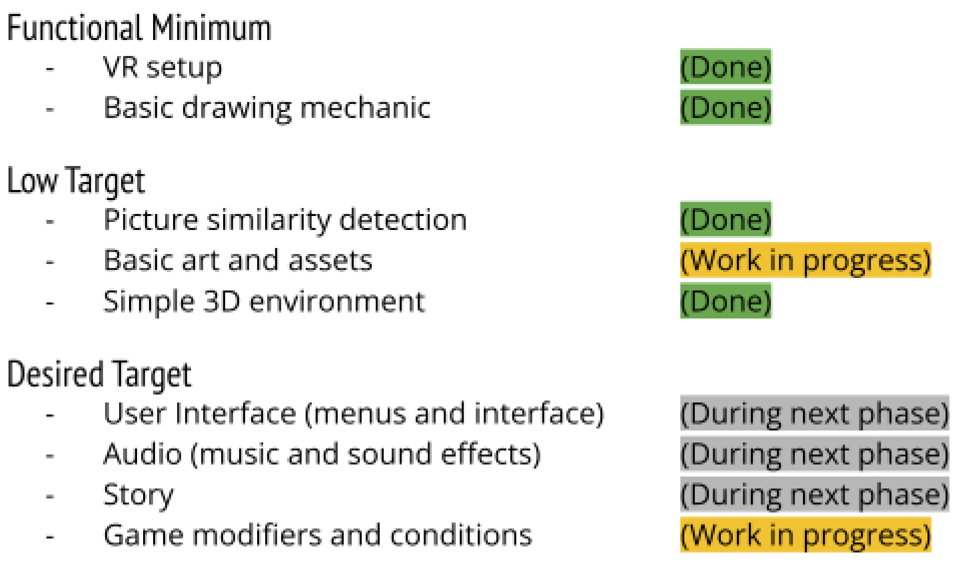
Leonard Keil

Overview & Team Cooperation

Now that we finally started coding, we need a little bit more management tasks than before to efficiently use our time. Therefore, our regular meetings were now focused mainly on work allocation rather than idea brainstorming or design questions. We managed to distribute our tasks in a way that everybody can work on their tasks simultaneously. Klejdi and Georg worked on the picture similarity algorithm, Leo took care of the VR setup and drawing mechanics, Clemens did the canvas, some assets and prepared some guard AI and Kim did the level design and started looking into the mechanics, setting up the flashlight**.**

Task Progress

We are overall on track for our planned schedule. We have fulfilled the majority of the tasks set out in our layered tasks overview up to the desired target. The functional minimum was made up of a VR setup and a basic drawing mechanic. Our low target included the picture similarity detection, a simple 3D environment and basic art and assets. All of these tasks have been addressed and, except for additional assets that are required at a later point, also fulfilled.
Our current state is within the desired target. Our game modifiers and conditions are in various states of progress, either already in production or in their early stages. This is one of the few parts that needed some minor adjustments in our schedule, requiring some additional time during the development phase towards the alpha release. UI, audio and story will be addressed further along the line as well but that will be handled as planned.

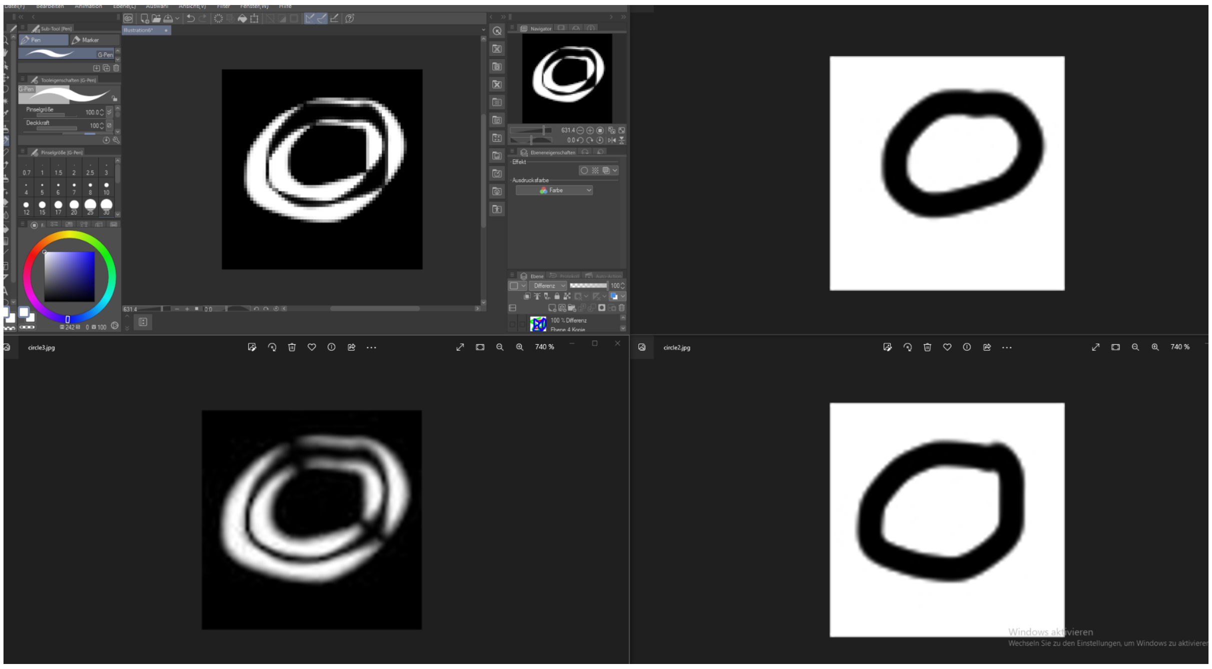


Functional Minimum	
- VR setup	(Done)
- Basic drawing mechanic	(Done)
Low Target	
- Picture similarity detection	(Done)
- Basic art and assets	(Work in progress)
- Simple 3D environment	(Done)
Desired Target	
- User Interface (menus and interface)	(During next phase)
- Audio (music and sound effects)	(During next phase)
- Story	(During next phase)
- Game modifiers and conditions	(Work in progress)

Similarity Algorithm

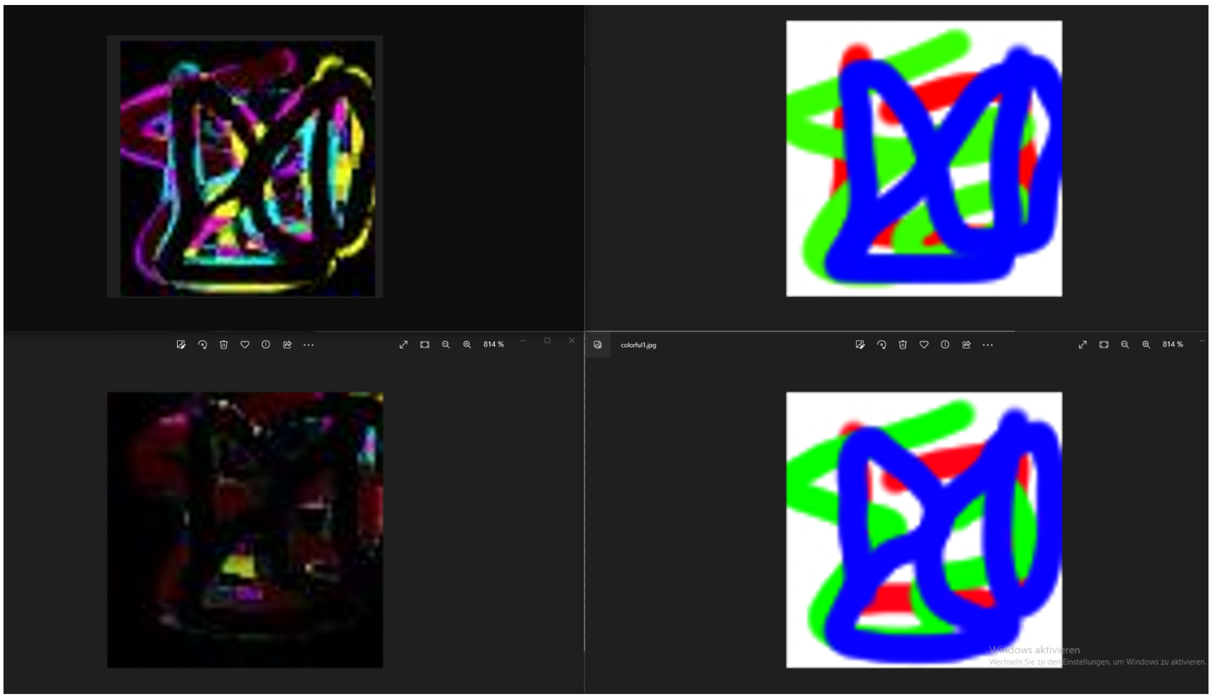
Our first idea was to simply write an algorithm that compares an area of the duplicate with the respective area of the original. We implemented this by setting up a filter (we tried various sizes from 3x3 to 13x13) that averages each pixel in the area and compares the two number values. Then we calculate the difference of the average area value for each pixel.

Top left: true difference
Bottom left: our algorithms averaged pixel difference
Top right: original
Bottom right: duplicate



We realised quickly that this solution was not what we had in mind. The averaging did not lead to reduced mistakes but instead just blurred the true difference. This makes sense, once we thought more about it and we came up with a few different approaches. One promising approach is to find the most similar pixel in an area and calculate the difference of the most similar pixel in its proximity. By increasing the area of proximity we would give the player more leeway and the results looked much more like what we had in mind.

Top left: old result (blurred difference)
Bottom left: new result (comparison with most similar in proximity)
Top right: original
Bottom right: duplicate



As you can see, the areas that are colourful in the new result are the areas that are most wrong in the duplicate. We also calculate a score out of the mistakes, to make it easy to rank how right or wrong a duplicate is. We also had some problems with file formatting and loading different files into unity, but we eventually solved all of those conversion issues. We also added a buffer around the textures, so that our filter would not reduce the output size of the picture. We also have some further ideas to use different approaches for difference calculating which we will compare and evaluate in future iterations. We also tried different versions of graphical output. Next to the difference in colour we also created an algorithm that provides an overlay over the original picture that shows in red which areas are wrong. For that we use different alpha values (transparency) to show slightly wrong areas and areas where there is a high difference in value.

VR Setup

As part of the initial work on the game prototype we set up a Virtual Reality environment within Unity.
There are multiple possible solutions and frameworks out there that can be used, offering varying degrees of compatibility with different Head Mounted Displays and their unique features.
Due to our usage of the Valve Index as development hardware, we had to use the SteamVR Middleware in any case. This also locked our choice of the VR backend down to just OpenVR, as it's the only one fully compatible with SteamVR.

In addition to this barebones setup we elected to install the VR Interaction Framework which adds a variety of base features for VR applications. We chose to do this as it allows us to focus on the unique mechanics of our own game without having to spend valuable time perfecting basic VR mechanics, which, if done wrong, would greatly distract from the game.
The setup was completed and a modified VR rig was used as the base for our player controller. It allows for basic movement using teleport or thumbstick inputs.
Players can also pick up certain objects with their virtual hands.

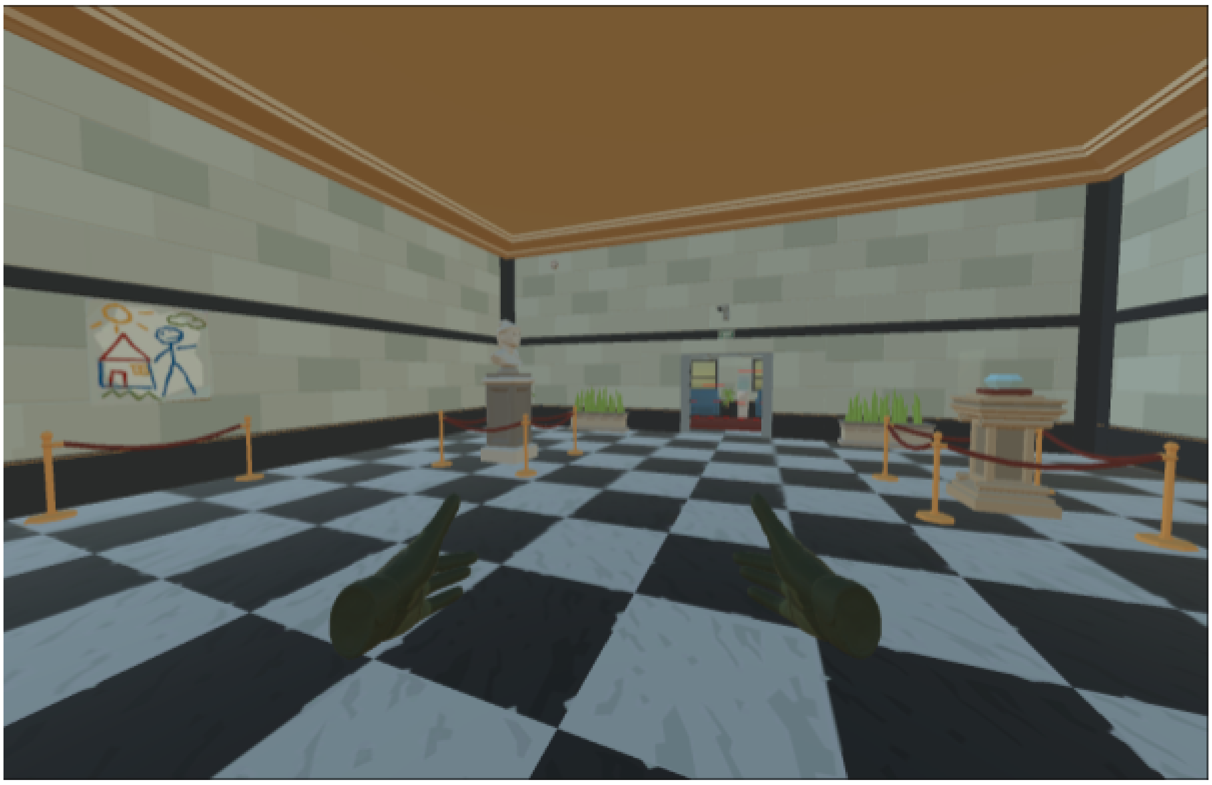
VR Mechanics

One additional mechanic we implemented is a flashlight that can be picked up by the player and pointed at the world. It uses dynamic lighting and can be toggled on and off using the triggers when handheld.
An additional mode was partially implemented where the torch would gradually lose energy, resulting in a gameplay mechanic where the player has to turn it off to recharge in between.

We also implemented the first iteration of the drawing mechanics.
A pen item can be picked up by the player and used on certain objects which are on a specified render layer.
The drawing is done using Unity’s line renderer feature on a diegetically placed UI canvas. Every new stroke is its own line, which we want to use in the future to implement an undo button to fix mistakes.
Currently this method of vector based drawing is not yet compatible with our similarity algorithms, however we already discussed possible solutions for rasterization of the image.
One such way could be the use of a secondary camera in the scene which captures the image using an orthogonal viewcone to save the image to a render texture, which is the compared to the actual image using our algorithm.

Visuals

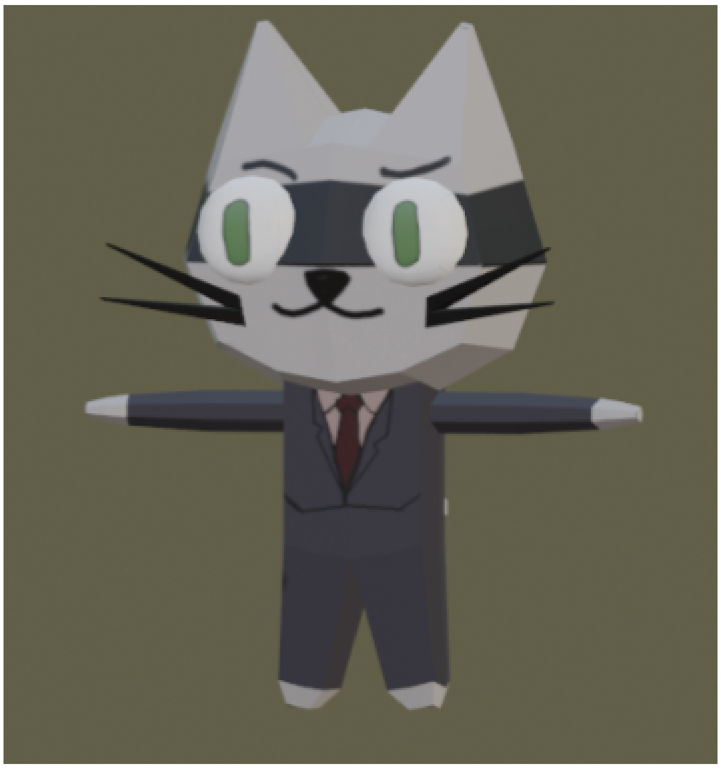
Our initial low target included the task of adding basic art and assets for our game. We have opted to use a combination of both self-made and third-party assets. Third party assets were primarily used in creating the environment and level of the Miauseum, the main stage of our game. We decided to go with third-party assets as the bulk of our assets for an easier and more streamlined process to save time and resources due to our limited skills in 3D modelling and art.



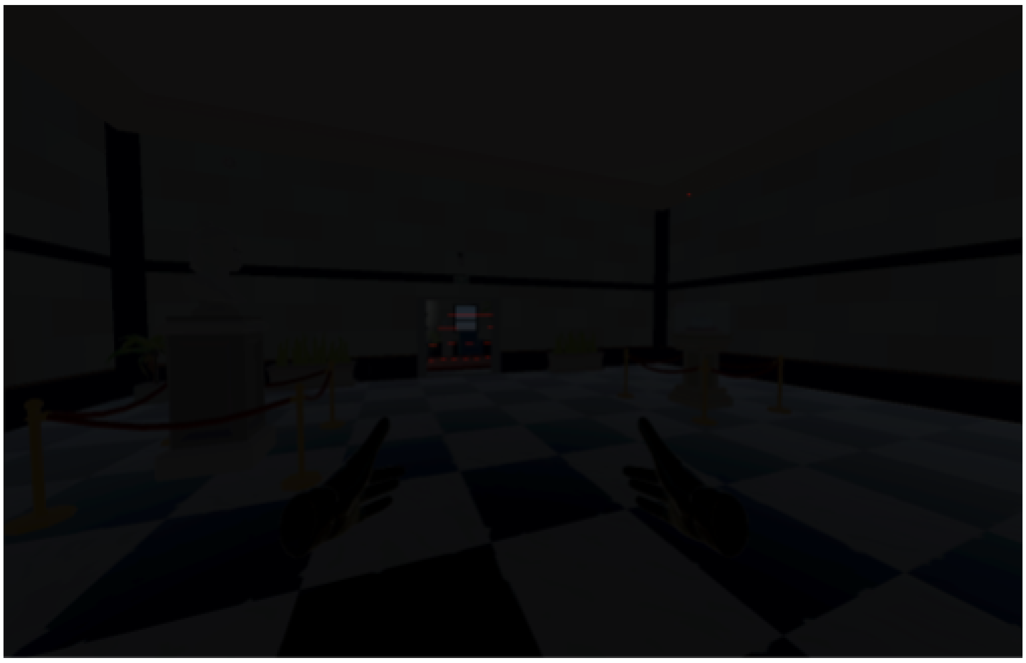
The Miauseum's main exhibition area

Despite using these third-party assets, we still wanted to create some models and art on our own. The intention behind that decision was both a general learning process as well as the freedom that comes with custom assets. These assets are primarily relevant to more specific game mechanics or to parts of the game that relate to the story. This for example includes the protagonist of our game, guards and the easel and canvas used in the drawing mechanic.

Easel & Canvas


Duplicat

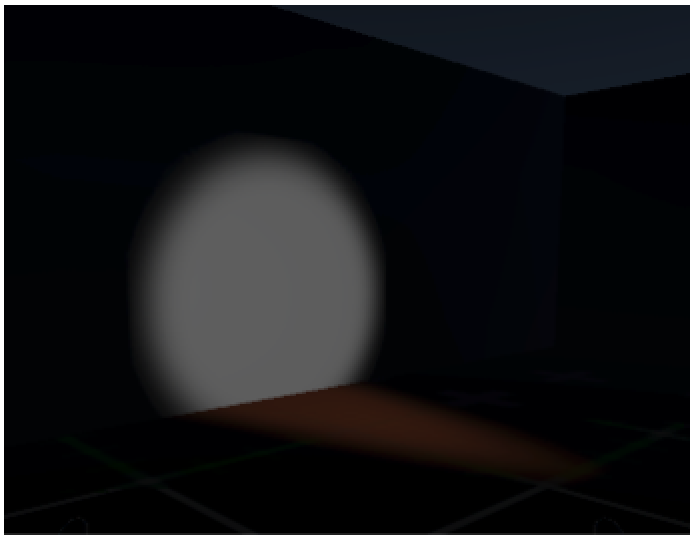
Some of our time was also spent on experimenting with different lighting settings. One of the main contributors to the overall atmosphere of our game was a dark environment and the use of a flashlight to navigate around. We haven't fully decided on a specific level of darkness to put our players into. This will primarily be tested during playtesting and adjusted accordingly.


The Miauseum in the dark

Game Modifiers and Conditions

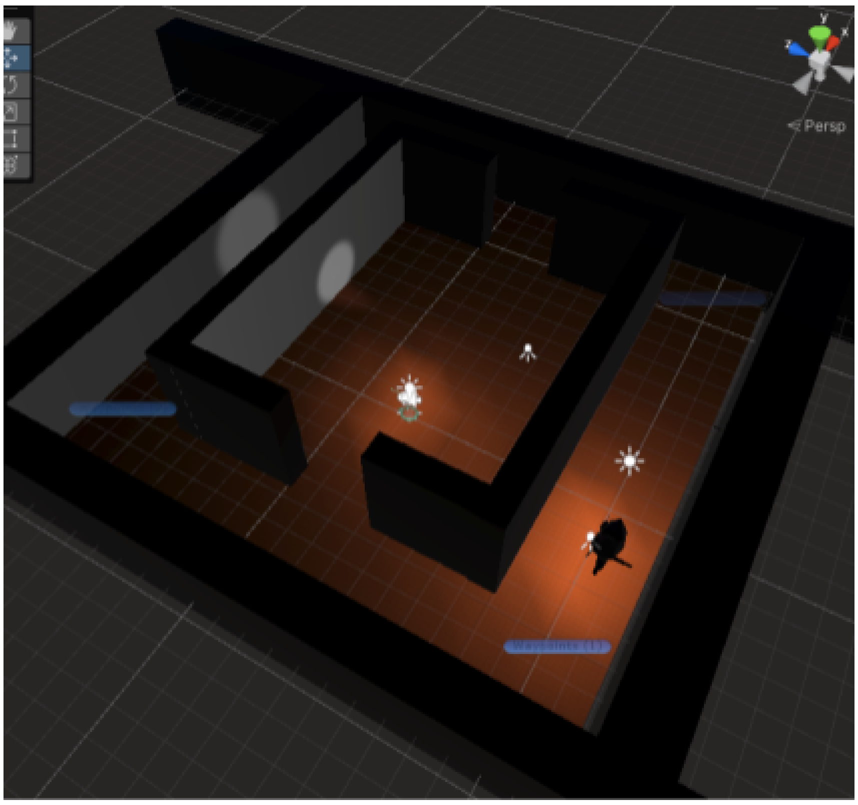
We focused on two primary modifiers and conditions during our first development phase that we considered to be the most vital and important. The flashlight and the guards. Both mechanics are in their early stages and either need more work or additional refinements to be fully polished.

The flashlight sheds a significant amount of light in a dark room but we decided to keep the cone relatively narrow so that the player wouldn't be able to see the entire room at once. Slight adjustments need to be made to the flashlight based on our preferred lighting settings and visibility options for our players.



Flashlight cone in the dark

Our guards are currently in their basic state. They move along waypoints in a room to simulate patrols during the night. We plan for them to periodically check the main room as well as react to certain things that the player might do. They are equipped with a flashlight to give the player a visual indicator of their overall presence as well as their field of view.



Guard Prototype