

Project Report

Part 4 - Alpha Release



STATIC

Computer Games Laboratory
Winter Term 2021/2022

Team Well-Meaning Scorpion

Matthias Ellerbeck
Musfira Naqvi
Mert Ülker
Doğa Yüksel

Overview

We achieved our functional minimum, our low target, our desired target, and one goal from extras while adding some small elements to our design. Some revisions have been made on the high target tasks while we also designed and implemented additional elements, which were not initially listed on our layered task breakdown. With the completion of these tasks, the principal design and coding of our game is complete and the game is ready for the following playtesting sessions.

Status of our original targets

Functional Minimum

Shooter mechanics

The shooter mechanics are done. The player can walk, run, aim, shoot, reload, and interact with items. Only one weapon is currently available, which features muzzle flash, bullet travel, and bullet impact effects for different materials. Furthermore, the shooter mechanics have recoil integrated which further enhances the player's experience.

Base enemies

The base enemies have new and improved animations integrated. The animation clips are prepared using Blender and later on imported into Unity along with the corresponding rigged models. Although the player cannot observe the enemy animations due to the core mechanic of the game, the animations are still of value since the player can see the enemies frozen in poses in between the keyframes of the animation clips (e.g., enemy frozen mid-attack after player turns around to look at it).

Physics module interacting with enemy simulation

In our last report we listed three options for determining the visibility of an enemy. Previously, we used (2): a visibility check based on view angles and raycasts. While this method was fast and simple to implement it also resulted in a high number of false negatives, i.e., visible enemies were sometimes labeled as not visible.

For the alpha release we implemented option (3), which is based on rendering. We start by assigning a unique 32 bit bitmask to every enemy in the scene. Only one bit is set in this bitmask. We then use a custom renderer feature to render all enemies into a separate, single integer render texture while still having the main camera's depth buffer bound. We use a custom shader for this which outputs only the specific enemy's bitmask. The result is a texture which contains the enemies bitmask in every texel in which the enemy is visible in the screen target. We then copy this texture to a compute buffer and reduce it with bitwise or operations in multiple compute shader dispatches to a single integer. In this integer, each

visible enemy's bit is set and all others are not. We then transfer this integer to the CPU's memory and check the set bits.

This implementation results in no errors; but compared to (2) it is much slower. Since we do not have problems with performance and freezing enemies are the core feature of our game we deem the milliseconds spent in this feature as justified. The implementation is also limited to 32 enemies.



The images show the rendered scene as seen by the player on the left, and the mask on the right. (Both are edited for clarity.)

One hand-crafted level

The level did not change since the last report. We used it to test changes and additions.

Low Target

Spatial audio cues

Our audio cues warn the player both when an enemy starts to follow them and when an enemy attacks. We use Unity's built-in spatial audio for these cues.

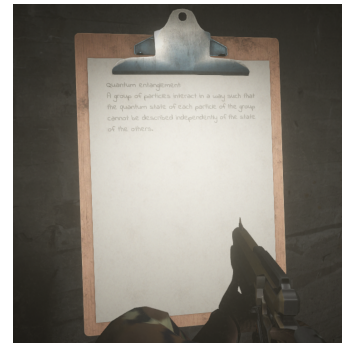
Multiple hand-crafted levels

As we initially decided, we have created multiple levels manually for this iteration of the game. The levels were modeled in Blender by constructing initial building blocks and utilizing them in combination in order to construct a complete level layout. These building blocks can also potentially be utilized for the procedural level generation in the future, which we listed in our extra target for the project.

Narrative Design in 'Static'

While developing the game proposal of Static, the narrative elements were designed simultaneously. It was important to insert story components to the environment as it would add to the player's knowledge about the origin of the enemies. These elements are aimed to explain how they were created and why they exist from the perspective of a scientist by using theories of quantum physics.

Currently the story elements can be found as clipboards with which the player can interact with. These clipboards are scattered throughout the level and can be collected by the user. The found pages will be marked and can be revisited through the game's main menu.



The game's story is shown as two scenarios:

1. As clipboard items placed on random locations in levels
2. Whenever the player dies

Not only does the player get to know the motivation for the experiment by collecting and reading the clipboards—but they also receive additional information about the facility. These are displayed in the game-over screen.

Desired Target

Environmental hazards

We created an area based environmental hazard in two styles which damages the player when they are inside the hazard's area. This is not the only hazard we envisioned, and we may be able to add more for the final release.

Custom shaders for improved aesthetics (e.g. dither)



We use custom shaders for the enemies. Instead of rendering a solid surface with continuous shading based on lighting, we compute the lighting intensity and then clip any fragments which are darker than a random value generated per fragment per frame. Surviving fragments output pure white. This results in a dithered gradient between transparent, i.e., black, and nearly opaque, i.e., bright as seen in the above image. During the computation we map the light intensity range of $[0, \text{infinity})$ to $[0, 1)$ to avoid solid white areas.

High target

More enemy variety

Like the base enemy, the explosive enemy is also fully implemented. It features a custom model, animations, and effects. We will most likely not add further enemy types.

More weapon types

We will most likely not add more weapon types.

Extras

Different environments/styles for varying levels

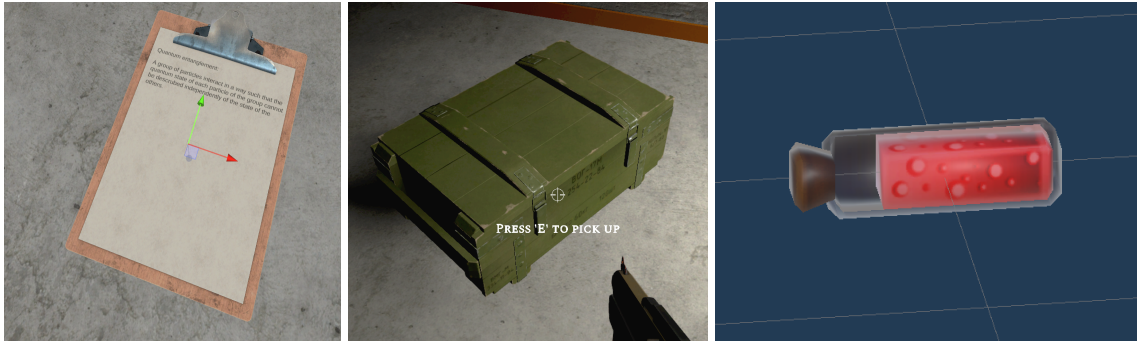
We added additional materials which can change the architectural style of a level, e.g., painted walls instead of naked concrete. This adds some variety to our levels.

Procedurally generated levels

We will most likely not implement this.

New additions and changes

Items and gadgets



Currently the game provides 3 types of interactable items:

1. Clipboards: Displays the narrative elements towards the player. When interacting with it, the clipboard moves to the player so that they can read its content.
2. Ammunition-Box: Provides the player with some additional bullets.
3. Health-Potion: Heals the player to a certain degree. We may change the visualization of this to first-aid-kits at a later date.

Player HUD



HUD is an important part of every game where the player should be able to see all the relevant information at a glance. For our action/horror shooter game the most relevant information is how close the player character is to dying and their ammunition count, as they would be defenseless if they run out. As with other aspects of our visual design we went for a stylish yet minimal approach as we did not want to clutter the screen too much and keep the player immersed.

The HUD also includes text pop-ups when the player is close to an interactable object to remind them of the controls.

Lastly, we have the aiming reticle visible to aid the player's aim when they are not using aim down sight.