CS135 Final Project Report
by
Cote Feldsine, Kent Phan, and Alejandro Sherman

Introduction

What is the project about?
Our project is a collection of ideas as imagined by each member of the team. The unifying aspect to the project as a whole is the theme of puzzle solving/maze navigating in a 3d space. Along with this, many ideas such as the existence of enemies, spot lights, traps, multiple routes, etc, were brainstormed and put on paper.

Why is the project interesting?
What makes our game interesting lies in that, other than the original theme and ideas we all started with, each team member was allowed near complete control over the section or "zone" we were designing. As a team of three, figuring out a method of collaboration was the first step in the creation of the game.

As such each team member was able to come up with and implement their own vision for their zone. In this way, each team member could add to their zone, without needing to wait on each other member to do so. Any idea that fit with the agreed upon theme was fair game. This also allowed for each team member to include certain aspects of the design we had considered before, and incorporate that addition into their zone.

The result of this workflow is a game that gives a player a distinct experience depending on whose zone they are playing in, while retaining the core conceit of a puzzle/escape game. Additionally, the difference of gameplay provided by each zone allows for a level of variety that would be hard to achieve if all group members were working on one section.

How does VR make the experience more interesting/immersive/challenging/etc.?
Similar to the lab exercises we have done in the course up to this point, under remote working circumstances, our game is not a VR experience.

That is not to say that our game is completely unrelated to the principles to what goes into making a VR game however. Our game is a first person experience, and as such each zone was designed with this view of the player in mind. This means that while the game was not made for VR hardware, the first person perspective still allowed us to play with puzzles/mazes that shift the player's view for added complexity.

Related Works
As our game features three distinct zones, each one can be compared to differing games.

For example, the structure of zone 1 can be compared with escape room games, gauntlet games, and puzzle games.

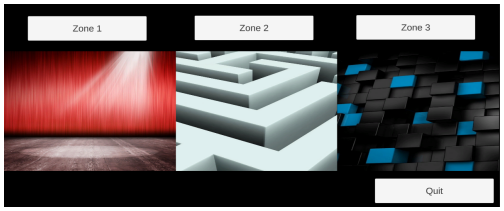
In this respect the inclusions in zone 1 compare such that, while zone 1 may not be the most fleshed out in terms of these games, zone 1 is versatile in that it demonstrates these multiple aspects in it's one design. In this way, zone 1 acts as a technical demonstration of multiple concepts, within a game that is already a collection of differting game philosophies. Additionally, zone 1 makes use of moving and non-moving blocks to add strategy to each area.

In contrast, a zone such as zone 2, can be compared with maze-like, and action-adventure games. In comparison to such games, zone 2 utilizes random generation, and progression with key acquisition and enemies to encounter, to create its own feel of play.

And finally, zone 3, as a designed maze (not randomly generated) it can be compared with games with designed structures and objectives to reach. In comparison to such games, zone 3 incorporates numerous objectives and has the player return to the starting area once the objectives are complete to finish the zone.

Design
As mentioned before, all parts of the project were designed to be played with a keyboard and mouse, similar to how previous labs were designed. In particular, each zone features character control via the first person controller prefab. However, the way in which each team member implemented player controls and interfaces, was unique to them. This being the case, the design philosophy used in each zone will be detailed separately.

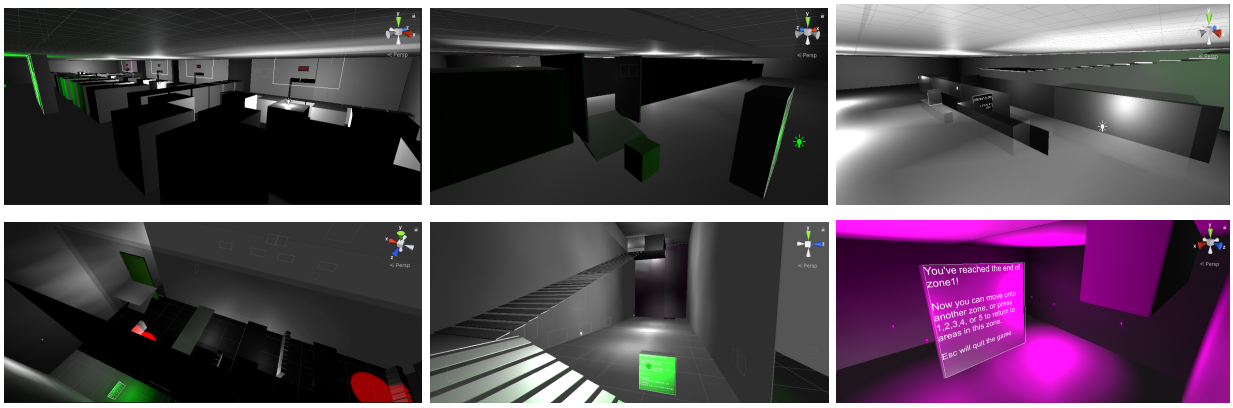
Main Menu:



To begin with the game begins with a simple main menu. The main menu itself works as an introduction to the player. While minimalistic, the background and name choice was chosen to give the player a hint of what's in store. Following the main menu, the level or "zone" selector is shown. Here the player can choose which zone they'd like to play this run. Again while simplistic, the purpose is allow the player to select the zone they wish to play, and load that zone

up. Additionally, the images presented for each zone is again intended to give the player an idea of what might await them in that zone.

Zone1: (Images are for areas 1-6 in order)



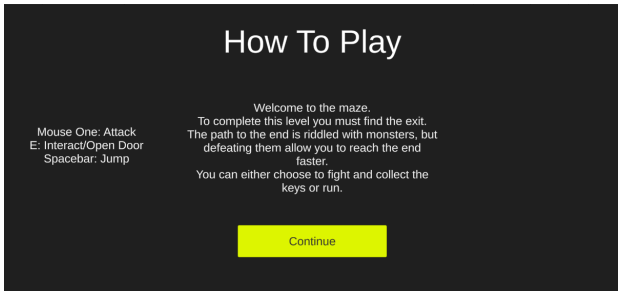
The main considerations for the design of zone 1 laid in the creation of “areas” that each gave the player a certain obstacle to overcome and move on to the next checkpoint. The general structure of zone 1 follows the player starting at an instruction wall that describes the objective of that area, the player traversing through that area, and arriving at the next checkpoint. The instruction wall also detailing the universal buttons that can be pressed in zone 1. Buttons including the default arrow keys to move, shift to run, and spacebar to jump, esc to quit, along with zone specific actions, such as “c” to return to the last checkpoint, and “r” to reset moveable block position and numbers “1”, “2”, “3”, “4”, and “5” acting as a hotkey to quickly move to that corresponding area in the zone. These additions were included if during play, the player were to get stuck, or blocks end up in a position that won’t allow further progress.

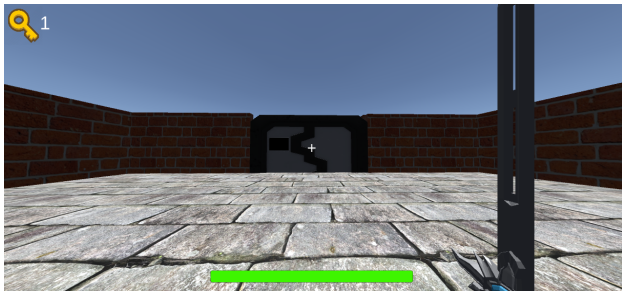
Each area in zone 1 was designed to bring a different mechanic to light.
Area 1 consists of a room with four exits, and is populated with many blocks in the way. The player must learn of the mechanics of moving a block when directly in contact with them, in other words, the pushing mechanic. For difficulty, this area is also populated with blocks that cannot be pushed, and a trap light system that alternates from white-> red that, when the player is caught, they return to the previous checkpoint.
Area 2 retains the light trap of area 1, but this time features a single pushable block and one narrow pathway to traverse. Area 2 was designed to have the player interact meaningfully with just one block rather than a roomful, as they must bring the one block with them to clear the wall at the end of area 2.
Area 3 lacks any traps to give the player a break from that mechanic, and features a line of transparent blocks that need to be aligned to create an invisible staircase to the next area.
Area 4 features the last light trap, but this time there are two spotlights that follow a rectangular rotation, and like before, if the player is caught in the light, they are sent to the last checkpoint. The player can hide underneath canopies to avoid the spotlights as they traverse past walls of movable blocks until they ultimately reach a tall bridge to knock over and cross to reach area 5.

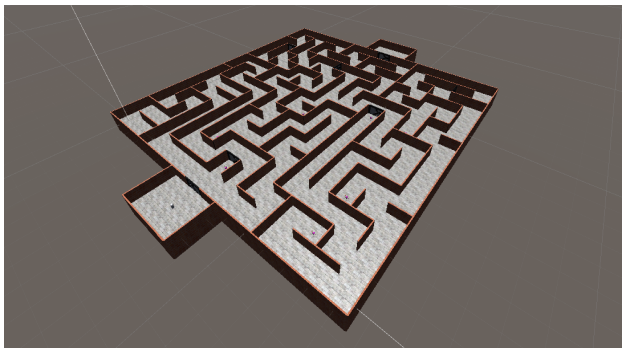
Area 5 lacks any real obstacles, and simply allows the player to choose a staircase to the next area, or an elevator. Both ways converge at a room with a drop, that leads to *Area 6* which is a victory room, and the end of zone 1.

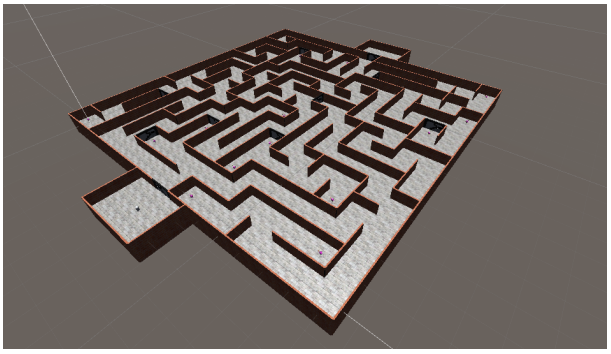
Zone 1 works best when the player is able to use both the movement keys and the mouse at the same time, as movement and FOV are used throughout many areas of the game. Otherwise, some elements of randomness are present as the moved blocks don't always fall the correct way (such as the area 4 drop bridge), however this is what the "r" reset block button is for.
As for VR practices, I feel that zone 1 conforms comfortably to the practices, as text is always placed on an object in the world, rather than on the player's view.

Zone2:



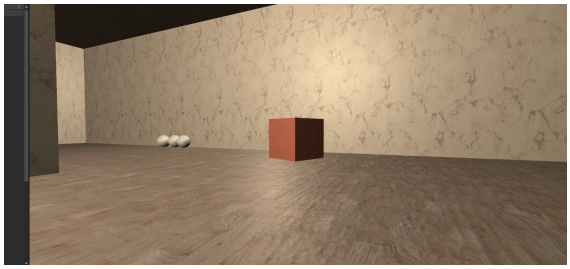
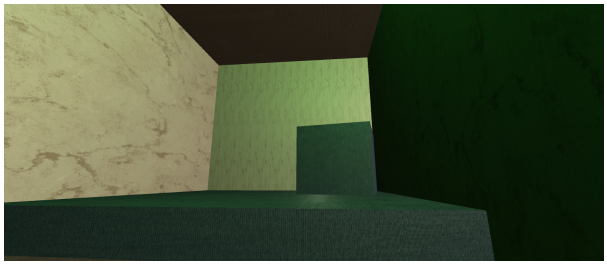
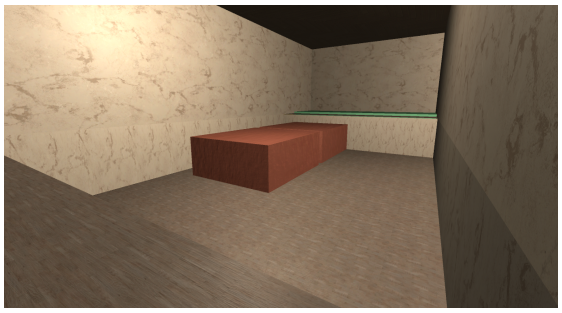
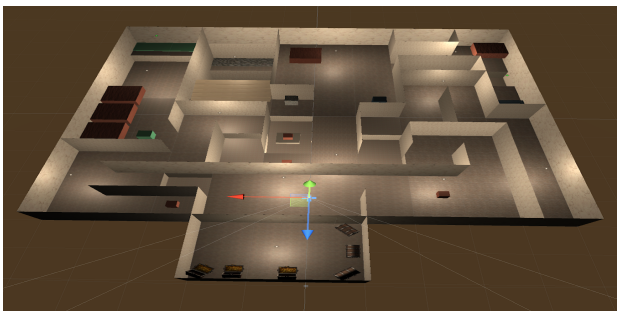






Zone 2 was heavily influenced by Kent's love for action adventure video games. It fit the theme of a puzzle as the map was a randomly generated maze that the player had to navigate through. While navigating, the player also encounters many COVID-19 looking enemies that drop keys upon being slain. The player can then use those keys to open doors, which are shortcuts around the maze, however the end of the maze can be reached by little to no enemies. Since we did not have VR headsets, we went with a mouse and keyboard interface, but tried to make it VR adaptable by making all our levels first person. For this specific level, a larger maze with a large amount of mob spawns would cause a small amount of frame drops as the bodies do not disappear upon death, but most computers should be able to handle it. As a group we choose a puzzle design because it was the first thing that came to mind, but for our individual levels, Kent made many design decisions because he was influenced by the games he plays. He was interested in implementing the features of his favorite games into his own level.

Zone3:

Zone 3 was designed as a 3-part experience - and thus is divided into 3 areas. The user must bring each block to its corresponding platform (indicated by matching colors) in order to clear this zone and get access to the victory room at the end. When a block has successfully landed its corresponding platform, a green light indicates that this area has been cleared. In order to move blocks, the user is given projectiles which collide with blocks and push them around. In each area, there are some orange blocks that the user must use to navigate through the maze and push their colored blocks with them.

The first area is the easiest - the user must push big orange blocks to fill in the pits and create bridges to walk over, and to push the blue block over to get to its platform. The only way to really mess this area up is pushing the big orange blocks too fast - then they will speed over the pit and land on the other side. These big orange blocks have one of their translation degrees of freedom restricted so they slide on a track. This makes moving them into the pits much easier for the user.

The second and third area pose similar challenges, with different twists to keep the zone interesting. Once these are cleared, the user may return to the spawn room (using a shortcut provided if it is found) and will see a door has opened to their victory room, with treasure chests.

I used the mouse/keyboard interface because I don't have an oculus and that's how unity allows me to test my projects. The controls are normal and expected for a first-person mouse and keyboard game, and they are displayed by a tutorial message when the user enters the zone.

Zone 3 doesn't have a best/worst in terms of user experience - it is very intuitive so most people could pick it up easily. Obviously those more interested in puzzle games would appreciate it more.

One design decision I made was that pushing blocks would be disorienting for a VR user, as they would have to get very close to them, and wouldn't be able to see on the other side. That's why I used projectiles, which appear further away.

Implementation

Implementations of ideas in the game, too varies based on the zone that each team member designed.

Zone1:

The bulk of zone 1 was made with standard assets. These being, planes, blocks (lots and lots of blocks), canvases, text, triggers, lights (spot and point), fps controller prefab (and it's scripts), simple color materials and of course C# scripts, the scripts being the way zone 1 allows for many of its features. All scripts that were created for zone 1, utilize tutorials from the unity website/stack overflow answers.

Scripts were used for:

The main push block functionality seen in almost every zone, where any object that has a rigid body, and is not kinematic will move when the player pushes (touches) it.

The light trap used in *Area 1* and *Area 2*, where a row of four white lights will have one alternate being red in sequence, and if the player touches it, send them to the last checkpoint.

The spotlight search traps used in *Area 4*, where the two red spotlights move in a given path of points and if the player comes into contact directly with the light, they move to the previous checkpoint.

A similar script as the last, used to have an elevator from *Area 5* to *Area 6*, where the elevator platform follows the vertical path at a set speed.

The reset block and reset position functionality that allowed the player to reset the block position, player position, and current area with a button press.

The updating of the current saved checkpoint when the player comes into contact with the next checkpoint area.

Zone2:

The implementation of Zone 2 consisted of a procedurally generated maze that incorporates RPG elements. The maze generation algorithms were from the unity asset store, however the instantiation of the walls, floors, mobs, and other game objects were edited and rewritten to fit the goal of the game. The mobs, weapons, and doors were taken from the asset store as well, but animations for the doors and weapons were created by Kent.

The main scripts for this zone were the ray cast for the opening doors and attacking enemies, enemy controllers, and player actions. The ray cast scripts helped open the doors so that the player could take shortcuts throughout the maze. The scripts also allowed players to engage in combat with the mobs to obtain keys to open the shortcut doors. Scripts to manage the player and enemy were composed of many smaller scripts, but they played an important role in combat and overall interacting with the environment. There was also sound managing scripts that allowed global sound playing. A major script that created the Navmesh for the AI of the mobs were runtime navigation mesh scripts provided by Unity. This is because currently Unity does

not have APIs for runtime navigation mesh building, especially for maps that include procedural generation.

The most challenging part of implementing the maze was that the asset itself contained a bug that would place two walls in the same position as each cell had a front, back, left and right. This allowed for the overlap of walls if adjacent cells had a wall present between them. I was able to find a workaround for this by heavily modifying the maze spawner script to not spawn walls if there was already going to be a wall in an adjacent cell. This was the cause of the Z-fighting that occurred throughout the maze that I was confused about for a while.

Zone3:
The majority of zone 3 was made of standard assets (planes, cubes, spheres, point lights, FPSController , etc.), but other assets were added as well. For example, an escalator was added to area 2, and several treasure chests were added to the victory room.
The scripts used accomplished several different things"

- A Projectile script was attached the the FPSController, allowing the user to shoot on mouse1
    - This script worked simply by cloning a sphere from outside the map in front of the user, giving it forward velocity, and timing its life to 1 second.
- A fps_controller script was attached to the FPSController, giving the user the ability to reset blocks, reset to spawn, and to the game
    - Reset blocks was done by taking the locations of all blocks when the scene is loaded, and when the button is pressed setting them back to their initial positions
- A trigger script was attached the the three trigger zones above each colored platform, allowing them to check if their corresponding block has collided with them, and to turn on the green light
- A victory room script was attached to a segment of wall in the spawn room so that when all lights are activated, this wall is removed revealing the victory room behind.

Lessons Learned
Of course, when designing the project, issues arose for zones, and general UI elements.
For example, at the very beginning of the project design the first feature that needed to be implemented was the ability to move an object when the player pushed on them. In this case, pushing blocks was to be a core mechanic of zone 1. The solution to implementing this feature came from the unity guides website at:
https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnControllerColliderHit.html
Another example was when the moving searchlights and moving elevator were also designed for zone 1. In order to figure out how to get the objects (i.e lights and elevator pad) to move on a certain path, the answer too was available on the unity forums.

All this to say that when an obstacle was hit, starting with the unity documentation was a safe bet as expanding understanding of the unity tools was a great help for that particular issue, and issues going forward (such as the searchlight script being useful for the later elevator script).

In terms of shared workload, as mentioned before each team member was responsible for designing their own zone. The root idea being that, after our initial brainstorming session, each team member would then design their own area of the game based on their own vision. Then in the end, the game would function as a collection of game sections, that each reflect a differing design philosophy. This structure allowed each team member to design their own section without the need to worry about merge conflicts in the unity collaboration as each member mostly focused on their own scene.

Additionally, a team member suggested that we use a kanban-style trello board to note what aspects we were each working on as we did them. This setup also allowed for collaboration on aspects outside of the zones, such as the main menu and level selector, as one team member created the functionality of those panels, and another gave them a design other than the generic template.

All things considered, I feel that this setup, along with communication via discord was a good recipe for collaboration on a unity project and if a similar project were to be done in the future, this would be a workflow worth repeating.

In terms of feedback during the demo, many people commented on the randomly generated aspects of the project as seen in zone 2. We learned the difficulties of handling randomly generated aspects such as maze layout and enemy placements, but also the benefits of a working random implementation.