CS324 Coursework

Mohammed Rafi 1915409

January 2022

WORD COUNT: 931

Game Description

I created a game where the user plays as a mouse trying to navigate a maze and get cheese which is randomly scattered across the maze, getting said cheese increases the players score by 1, however, scattered across the maze are also mousetraps which reset the players position and decrements their score by 1, the must also watch out as the maze is suspended the air and any wrong moves could lead to the player falling, also resetting their position and decrements their score.

Acknowledgments

Code

The main game has been adapted from the pointerlock controls example provided by three.js laying the foundation when it comes to player movement and some basic physics and collision detection[1].

The maze generation code has been adapted from JavaScript: Random Maze Generator by Chirp Internet which laid the foundation for the games maze generation[2]

Textures

Our ground texture[3] was created by the user katsukagi from the website 3dtextures.me `https://3dtextures.me/` which provide free PBR textures, The wooden plank texture[4] was provided by the website `https://ambientcg.com/` which also provide free textures. The sky HDRI texture[5] provided by `https://www.hdri-hub.com/`

Sounds

The sound we used, "8 bit death sound"[6] and "Coins 1"[7] were provided by users MentosLat and ProjectsU012 on the website `https://freesound.org/` which provide free sounds to use.

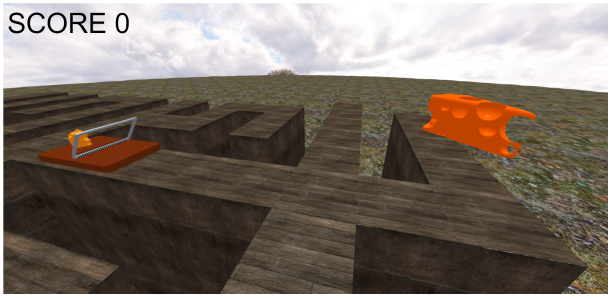
Essential Requirements

Interactivity

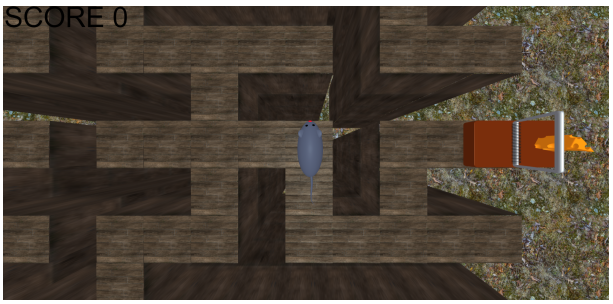
The player can interact with the game with the mouse and keyboard, they can use the mouse when in first person view to move the camera around using three.js pointer lock controls[8] and keyboard interactivity is possible using a regular JavaScript event listener which triggers variable changes for movement and other functions such as switching view and pausing the game.

Camera

The game offers two different perspectives, first person and top down, we switch these with a simple Boolean variable `change_cam`. Once the variable is changed we switch perspectives, removing pointer lock controls while in top down and showing our mouse model, I also increased the FOV to allow the user to see more of the maze while they are in a top-down view



(a) First Person 1



(b) Top Down 2

Figure 1: 2 different views ingame

Menu

The game has an initial menu which provides the player information with a brief description of the game and how to play it with the key bindings.

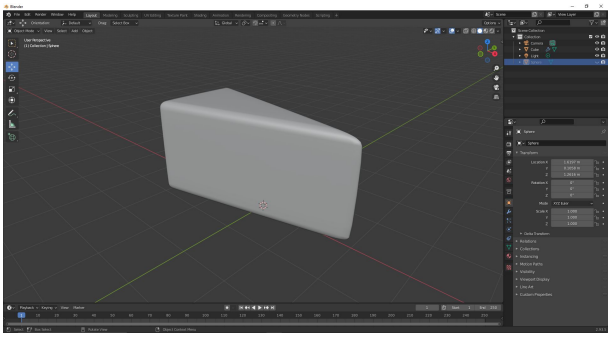
Levels

The game has a vast amount of levels as the mazes are generated randomly, we can estimate the amount of mazes to be created using the number of nX6 binary arrays with all 1s connected and a path of 1s from upper left corner to lower right corner as an example with a 12x6 array having 1101801030283339240 possible combinations[9][10] and as we have a 12x12 maze the number of mazes will be much larger.

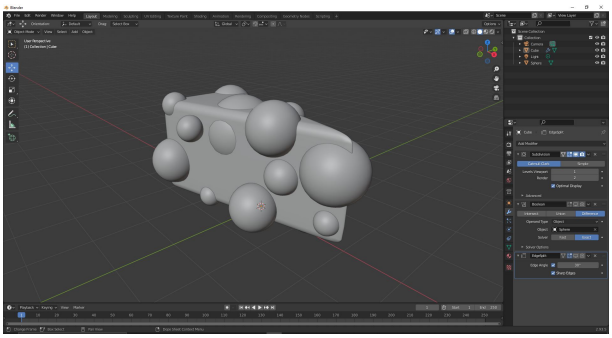
The cheese and traps are placed randomly across the level and once the player collects the cheese or falls into the trap they move to another random location in the maze.

Model

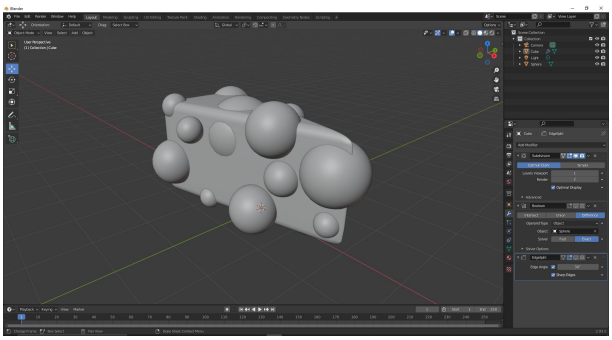
For the game I created three models, a cheese, mouse and mousetrap model.



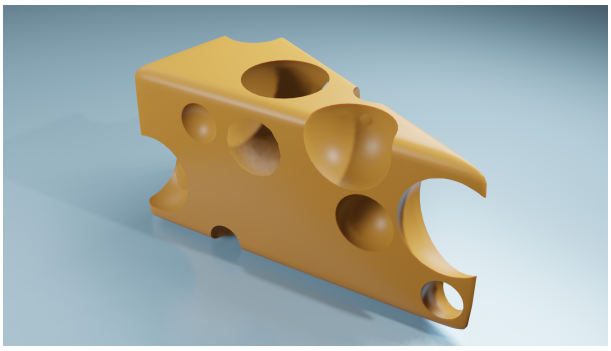
(a)



(b)

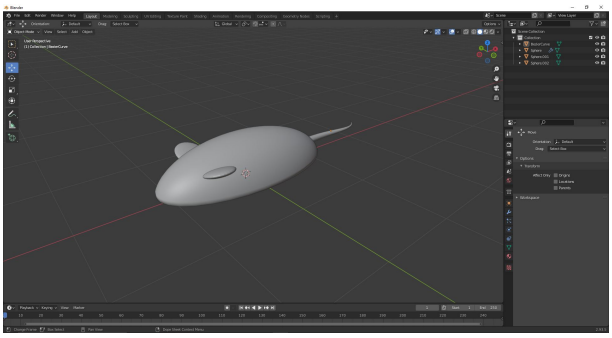


(c)

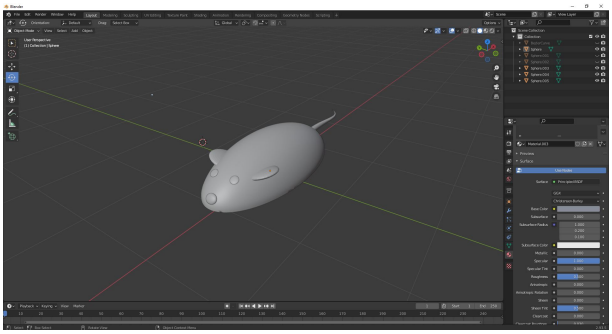


(d)

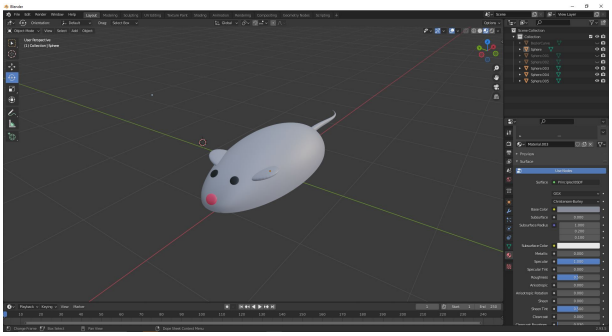
Figure 2: Creation of cheese model in blender.



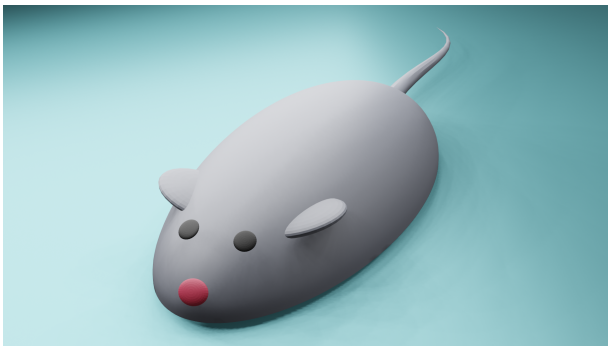
(a)



(b)



(c)



(d)

Figure 3: Creation of mouse model in blender.

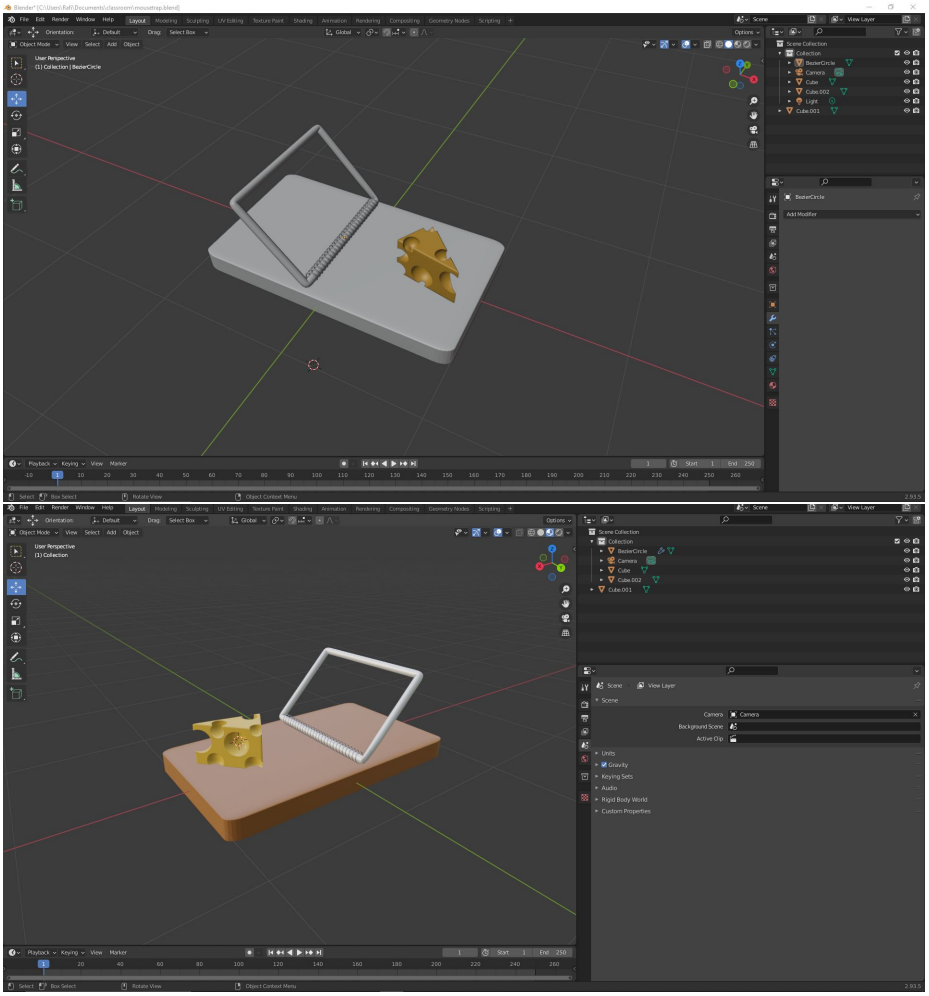


Figure 4: Creation of mousetrap model in blender.

The cheese model was created by using a Boolean modifier in conjunction with multiple spheres to achieve the holes in the cheese.

The mouse trap used a circle bezier curve and a screw modifier to create the spring.

The models were loaded into our game using the `GLTFLoader`[11]

Lighting

The game has two main lights a hemisphere light and a directional light which could be initiated with `THREE.DirectionalLight`[12] and `THREE.HemisphereLight`[13]

Textures

We have introduced multiple textures into the game, one for the ground, maze and sky. These textures were loaded into three.js using the `THREE.TextureLoader`[14] and as we wanted are textures to interact with the scenes lights we used `THREE.MeshPhongMaterial`[15] which allowed us to use multiple maps such as a normal, displacement and roughness map which gave the textures more realism.

Desirable Requirements

Heads-Up Display

We have a simple HUD showing the players score using HTML which is displayed above the three.js render using an absolute CSS position, and with JavaScript we can update the score variables in

real time by changing the divs JavaScript `innerHTML` value each frame.

Once the player has a score of 10 we send out a JavaScript alert to notify the player they have won the game, if they press "OK" the page refreshes and generates a new maze for them to play again.

Sounds

Sounds are played when the player performs actions such as collecting cheese or falling of the maze or going into mouse traps.

We load the audio files using the `THREE.AudioLoader`[16] and are played using the `THREE.AudioListener`[17].

Collision Detection

We have two forms of collision detection in our game, a raycaster for collisions with the level and hitboxes for our cheese and trap models.

We create are hitboxes / bounding boxes using a `THREE.Box3`[18] object, they have a useful method `setFromObject` which automatically creates a bounding box for external models such as our cheese and mousetrap models. To check if we collided we use the min and maximum x and z coordinates from the bounding box and check if we are within the values.

We create the raycaster using a `THREE.Raycaster`[19] which checks our list of objects, in our case the cube meshes which create the maze for intersections using the `intersectObjects` method for intersections with objects, if we are intersecting the player is allowed to jump on the objects.

Physics

The game has simple physics by calculating the players direction based on the velocity each frame, the player y velocity is calculated by multiplying the players mass by our gravity constant(-9.8) and change in time.

Environment Map

I loaded my HDRI enviroment map using the `THREE.RGBELoader`[20] and set our scene environment to the HDRI to provide more realism in the lighting.

Animation

We have simple animations in our game by rotating the cheese every frame to make it more noticeable for the player and also rotate the mouse model based on the the players movement.

References

[1] three.js. *misc$_c$ontrols$_p$ointerlockexample*. `https://threejs.org/examples/?q=contr#misc_controls_pointerlock`. [Online; accessed 22-Jan-2022].

[2] Chirp Internet. *JavaScript: Random Maze Generator*. `https://www.the-art-of-web.com/javascript/maze-generator/`. [Online; accessed 22-Jan-2022].

[3] Katsukagi. *ground forest 003 : 3D TEXTURES*. `https://3dtextures.me/2020/03/20/ground-forest-003/`. [Online; accessed 22-Jan-2022].

[4] ambientCG. *ground forest 003 : ambientCG*. `https://ambientcg.com/view?id=Planks012`. [Online; accessed 22-Jan-2022].

[5] HDRI Hub. *HDR Sky Cloudy (free): HDRI Hub*. `https://www.hdri-hub.com/hdrishop/freesamples/freehdri/item/76-hdr-sky-cloudy`. [Online; accessed 22-Jan-2022].

[6] MentosLat. *8 bit Death sound: Freesound*. `https://freesound.org/people/MentosLat/sounds/417486/`. [Online; accessed 22-Jan-2022].

[7] MentosLat. *8-bit Video Game Sounds Coins 1: Freesound*. `https://freesound.org/people/ProjectsU012/sounds/341695/`. [Online; accessed 22-Jan-2022].

[8] three.js. *PointerLockControls: three.js docs*. https://threejs.org/docs/#examples/en/controls/PointerLockControls. [Online; accessed 22-Jan-2022].

[9] Vepir. *How many valid mazes of some size are there?* https://math.stackexchange.com/questions/2679666/how-many-valid-mazes-of-some-size-are-there. [Online; accessed 22-Jan-2022].

[10] R. H. Hardin. *A163006: Number of nX6 binary arrays with all 1s connected and a path of 1s from upper left corner to lower right corner*. https://oeis.org/A163006. [Online; accessed 22-Jan-2022].

[11] three.js. *GLTFLoader: three.js docs*. https://threejs.org/docs/#examples/en/loaders/GLTFLoader. [Online; accessed 22-Jan-2022].

[12] three.js. *DirectionalLight: three.js docs*. https://threejs.org/docs/#api/en/lights/DirectionalLight. [Online; accessed 22-Jan-2022].

[13] three.js. *HemisphereLight: three.js docs*. https://threejs.org/docs/#api/en/lights/HemisphereLight. [Online; accessed 22-Jan-2022].

[14] three.js. *TextureLoader: three.js docs*. https://threejs.org/docs/#api/en/loaders/TextureLoader. [Online; accessed 22-Jan-2022].

[15] three.js. *MeshPhongMaterial: three.js docs*. https://threejs.org/docs/#api/en/materials/MeshPhongMaterial. [Online; accessed 22-Jan-2022].

[16] three.js. *AudioLoader: three.js docs*. https://threejs.org/docs/#api/en/loaders/AudioLoader. [Online; accessed 22-Jan-2022].

[17] three.js. *AudioListener: three.js docs*. https://threejs.org/docs/#api/en/audio/AudioListener. [Online; accessed 22-Jan-2022].

[18] three.js. *Box3: three.js docs*. https://threejs.org/docs/#api/en/math/Box3. [Online; accessed 22-Jan-2022].

[19] three.js. *Raycaster: three.js docs*. https://threejs.org/docs/#api/en/core/Raycaster. [Online; accessed 22-Jan-2022].

[20] three.js. *DataTextureLoader: three.js docs*. https://threejs.org/docs/#api/en/loaders/DataTextureLoader. [Online; accessed 22-Jan-2022].