LunarLand

- William Graham, 101228346
- Mason McLeod, 101229241
- Jacob Bonner, 101234823

Section 1. Introduction

Our game takes place in the titular theme park of LunarLand. The park is barren and dark, a far cry from normal amusement parks. The park recently had to be evacuated due to a depressurization incident, with various items left behind that the company wants to retrieve - the central objective of the player. The goal of the game is for the player to explore the park in search of three items: the Soda Chip, the Health Inspection Papers, and the Lunar Lint. The player can explore different branches of LunarLand, from the dome with sparse and underwhelming attractions, to the employee's sector with poor employee amenities and a security office, and to the panopticon where office work for LunarLand is done. By walking around the park, players can come across audio recordings of employees of the park before its evacuation. Listening to these logs might give the player clues as to where some of the items have been left, and how to access them.

Section 2. Walkthrough

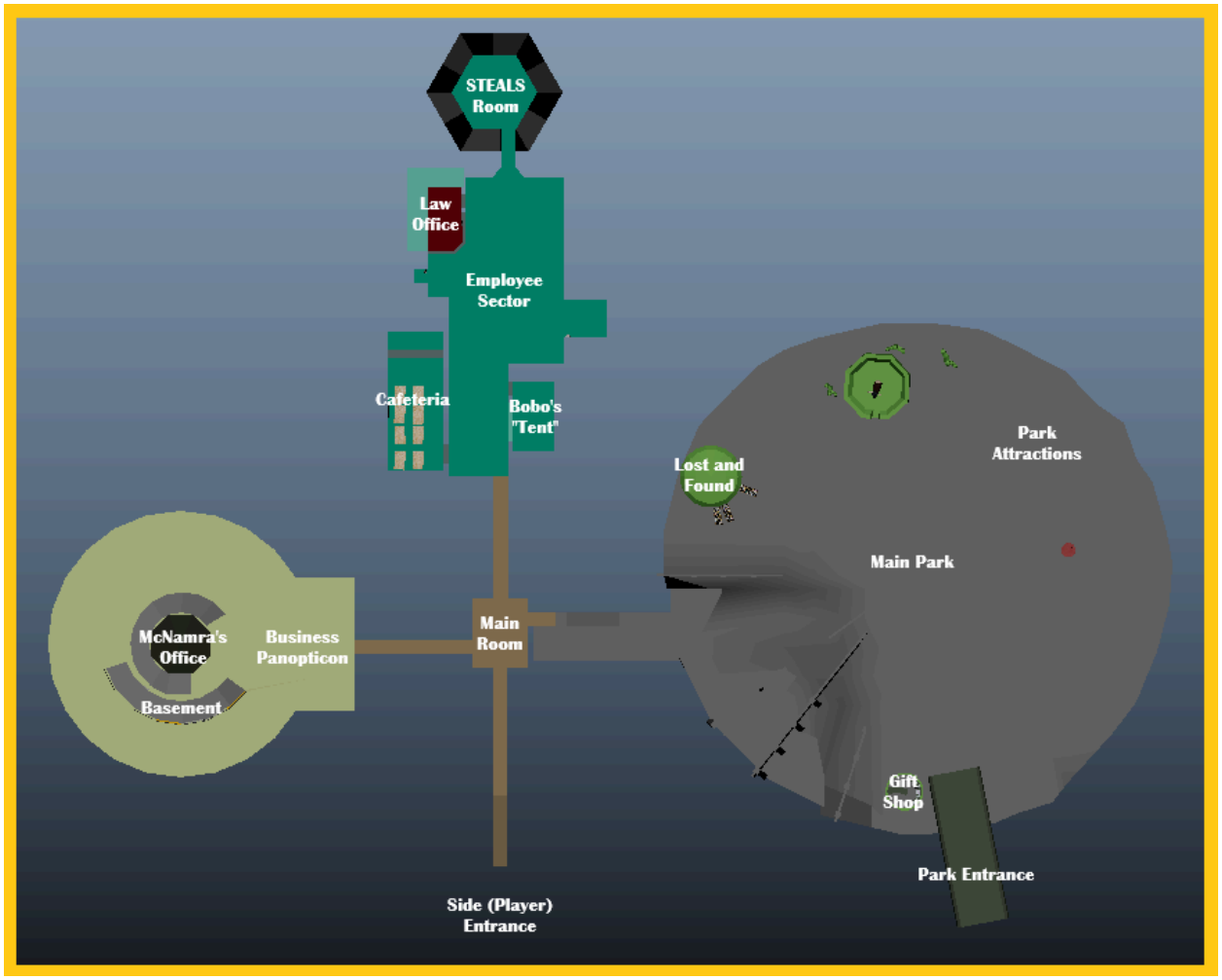
2.1 Controls

The player starts on the surface of the moon, just outside the park. Susie from LunarLand HR will tell you the controls needed to play the game. The controls are as follows:

- W: Move Forward
- A: Move Left
- S: Move Right
- D: Move Backwards
- Left Click: Interact
- Esc: Pause/Unpause

For the best experience, play in full screen.

2.2 The Golden Path



The game starts with you on the cold lonely surface of the moon. Susie (your lovable HR rep) will tell the player to enter the building, where the first room will contain three computer terminals telling the player more about the items they were sent to retrieve. Each terminal is next to the path that you will take to the area where the item can be found, you can tackle these in any order but we recommend doing it in this order. Business Panopticon, Main Park, then the Employee Sector. She also gives you an important hint on how to play the game: as you approach certain objects, they will glow blue, signifying that they can be interacted with in some manner by a left click on the mouse.

After interacting with its terminal, take the path to your left and enter the accounting panopticon! This is where the office workers of Lunarland once toiled away crunching numbers.

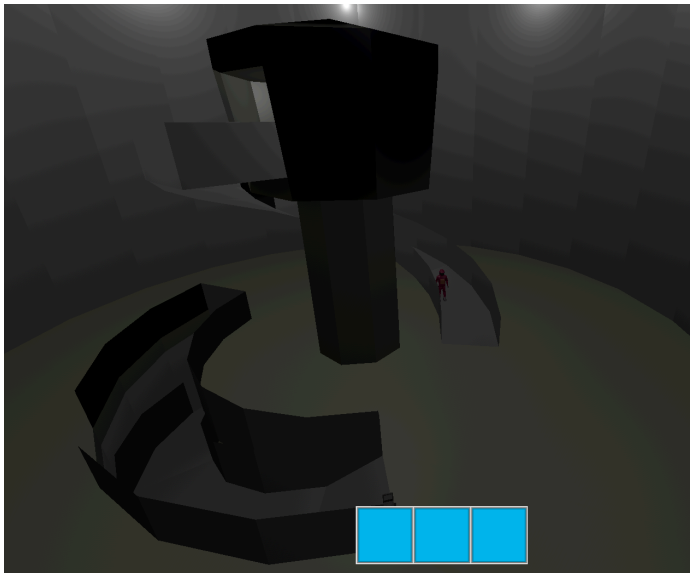
After retrieving the Soda Chip, you return to the hub area and, after interacting with its terminal, make your way to the right this time, descending into the ruins of the amusement park proper. Here lies the sad remains of a once mediocre amusement park.

After finding the health inspection papers, you return to the hub area. After interacting with its terminal, you take the path straight down to the LunarLands employee ‘lounge’. At the behest of the terminal you make a beeline for the security office, and begin the tale of theft that keeps you from your prize.

After obtaining the Lunar Lint, you now have all three of the items you were sent here for, relax and enjoy our end credits scene.

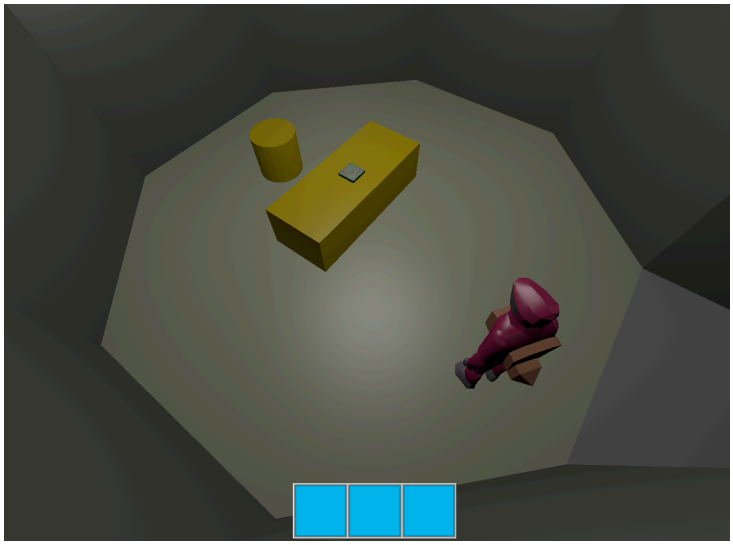
Player entrance:

Accounting panopticon main floor:

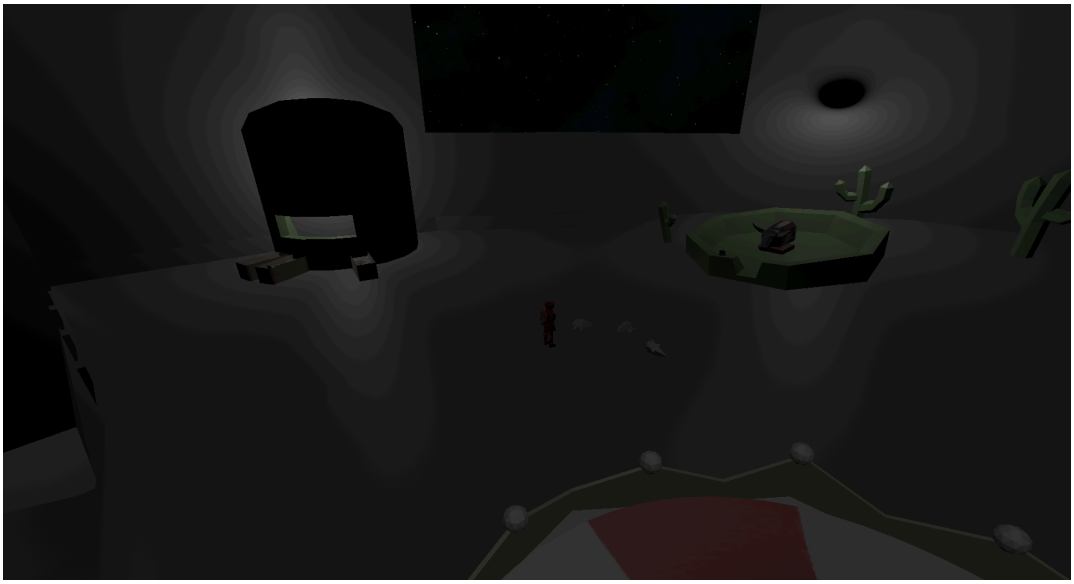


Accounting panopticon basement:

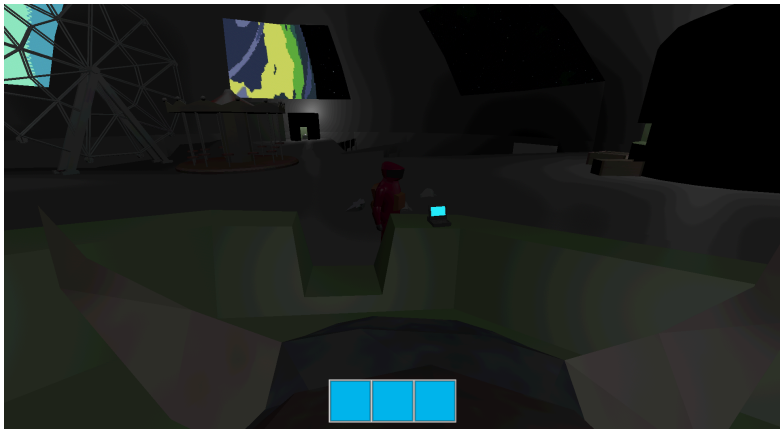
Accounting panopticon bosses office:



main park area:



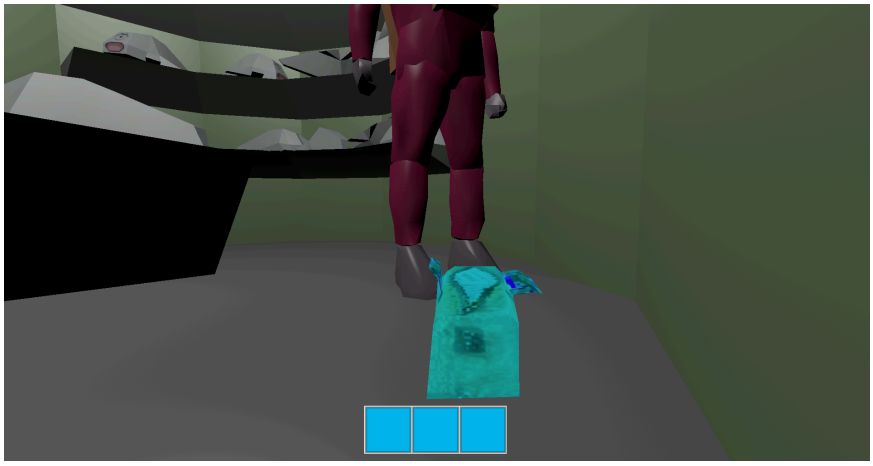
Game view from bull pen area:



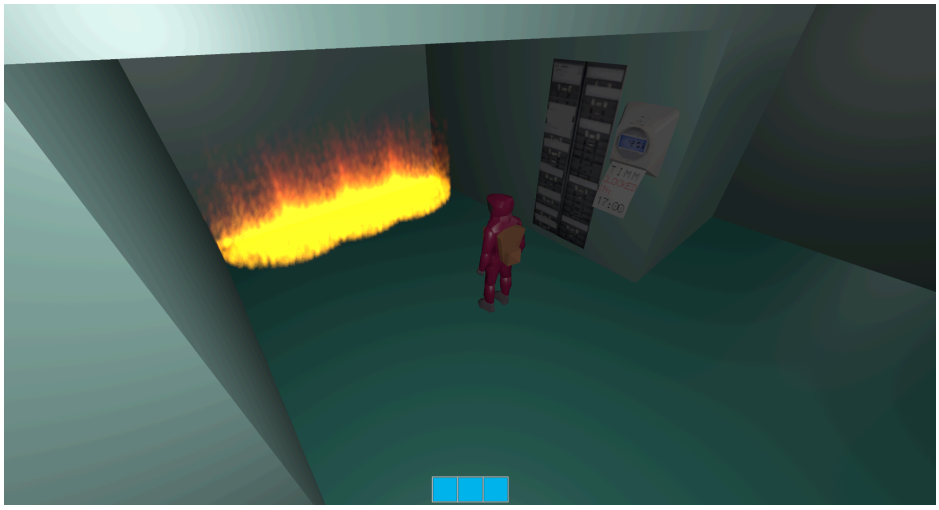
Game view from lost and found booth:

Game view from the gift shop:

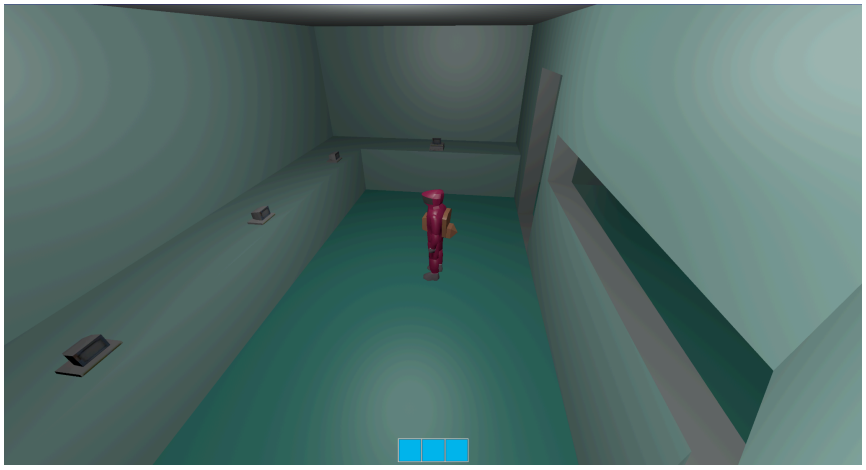
Game view from the gift shop, behind the counter.



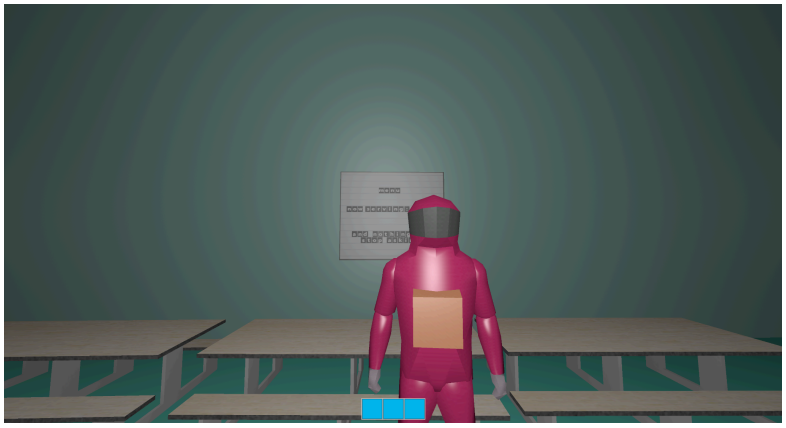
A fire in the pool area!



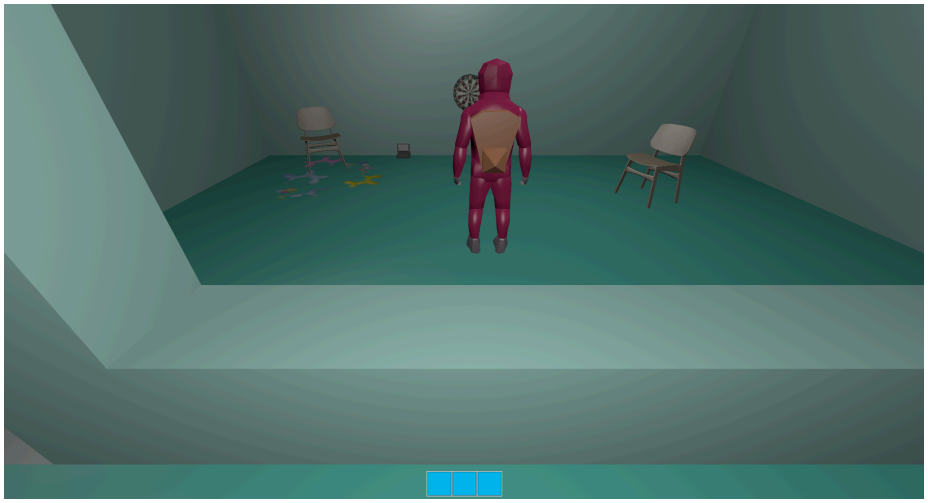
In the law office of Joe Law:



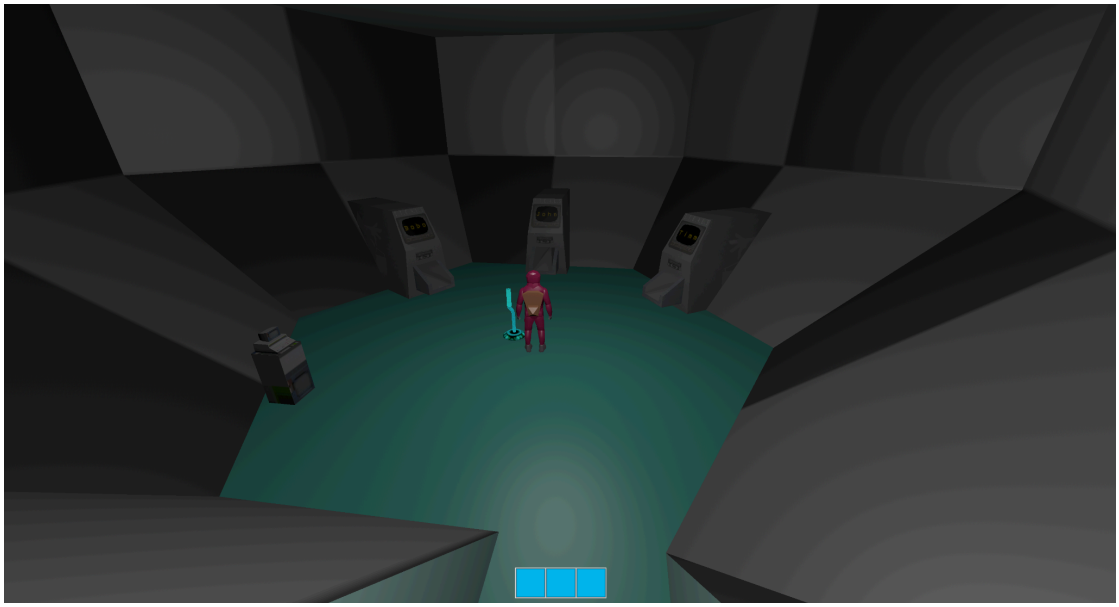
I wonder what's for dinner today in the cafeteria?:



Clowning Around in Bobos tent:



Cranking the generator and finding the culprit in the STEALs unit:



2.3 Spoilers!

The three audio logs in the first room when you enter the building give context as to what it is you are trying to retrieve. The audio logs are placed beside the hallway to the area corresponding to the item the log indicates. The hallway corresponding to the soda computer chip leads to the panopticon where most of the office work for the park is done. The hallway corresponding to the health papers leads to a domed area that contains the park's attractions. The hallway corresponding to the lunar lint leads to the employee sector of the park.

After you take a gander of your surroundings in the accounting panopticon you find a log on the staircase and listen to it. The log is from Busy Izzy who tells you to find Tyler, who is working on the Soda Chip in the back room of the basement. Sure enough, when you travel downstairs, you can find a log by Tyler that tells you that management still needs to approve it. Traveling further into the basement, you will find one more audio log from Mr McNamra, who is yelling at Tyler for not gouging the prices high enough, not approving of the chip. He also tells you that the chip is on his desk. With nowhere else in the basement to travel, there is only one way left to go - up! After traveling up the panopticon to the center tower, you will find Mr McNamra's office. Sure enough, just as he said, the chip is on his desk and ready for you to take!

After taking in the unsightly remains of Lunar Land, the player is free to roam around the domed amusement park section of LunarLand. By doing so, you can find three audio logs, each detailing an encounter between the health inspector and an unknown employee from three different points of view. From these logs, there is certain important information you can glean as to the whereabouts of the health papers. Hootenanny Harry's log at the mechanical bull tells you that the employee that took the health inspection papers was carrying a load of stuffed aliens, a few of which have been left behind and are on the ground for you to observe. Sebastian Stanza's log at the poetry stage tells you that the thief of the papers headed towards the park's entrance. Penny Pincher's log at the lost and found tells you that the thief stowed the papers in an off beige bag. With all this knowledge in mind, if you head towards the park entrance in the dome, you will find the gift shop. If you enter the gift shop, you will see the park mascot as well as other plushies that resemble the stuffed aliens left behind at the scene of the theft, indicating that the papers might be around here. Further examination of the gift shop reveals an off beige bag hidden behind the gift shop counter. Interacting with the bag will give you the health inspection papers.

After entering the law office, listen to the 4 audio logs inside, you will get full context for the theft of the Lunar Lint, the three suspects, and their alibis. The prime suspect is currently Timmy. After some exploring and you enter Bobos 'tent', you find environmental details like his dart board and some balloon animals on the floor. This supports his claim that he was playing darts, and tying balloon animals during the time of the theft. An audio log reveals Bobo's suspicions of Timmy here as well. Next you go to the cafeteria and listen to an audio log of Timmy and Johnny eating something… slurpy. The part of the cafeteria menu board that says what the menu was has suspiciously been removed. Next you go to the employee entrance to the pool, now on fire, however you can just make out a timecard that confirms Timmy was indeed at his second job during the time of the theft! Next you might spot a trashcan in the corner of the main room, and upon further investigation you see the removed letters from the menu in the cafeteria. They spell *pasta*, wait didn't Johnny say he was eating steak!? You also might recall that he lied about seeing Timmy right before the robbery, but he couldn't have because we know he was at his job. It's Johnny! You make your way north to the STEALs room and select

Johnny's locker, out comes the Lunar Lint, and with it the truth. However, if you chose the wrong locker, the entire system will power down until you use the hand crank at the center of the STEALS room enough time to power it back up. The number of cranks required multiples 10 fold with each incorrect guess. So… don't guess wrong!

Section 3. Overall Assessment

We were able to meet all of the technical requirements, some in more meaningful ways than others, but we stuck to the letter of all of them, and the spirit of most of them. We maintained adherence to the aesthetic requirements. There are a number of ways we went beyond the minimum, such as a third person multi-camera system as the main point of view, a system of interactable objects in game that will do different things when the player interacts with them, and an inventory system to track what items the player has found. Most of these add to the aesthetics of our game, attempting to create a world that was truly believable with the resources we had. If we had more time, we would likely spend it populating the environment further with more items you'd find around a moon base turned theme park like LunarLand.

Section 4. Technical Requirements

0. Using Godot with GDExtension to create the game; readable code with no serious bugs, suitable class hierarchy, good documentation

We used GD Extension for everything asides from the game interface and staging, we have found no game breaking bugs or major exploits. One minor bug was found, in which to avoid the player getting stuck on geometry in the floor of the map, the player may randomly appear to "hop". Although this is intentional to avoid them getting stuck, we did not find a way to make it appear more seamless before submission. We made good use of inheritance, especially with our Interactable family of objects (Interactable.{h/cpp}, AudioInteractable.{h/cpp}, etc). We were generous with our comments.

1. Large textured heightfield terrain with collision detection

The player begins the game on the lunar surface, which is a heightfield terrain. The player can freely walk around the terrain, with collision detection keeping the player on the ground in tandem with gravity.

Associated Files: Player.{h/cpp}, HeightMapTerrain.{h/cpp}, terrain.gdshader

2. Procedurally generated heightfield used in the game (generation can be done offline, but show the code)

The height map as described in requirement 1 was procedurally generated using the code below. The code constructs a noise image that is saved for future use as the basis of the height field. Since generating this image takes some time, the generation is done offline, and as said before, the image is saved once generated.

Associated Files: HeightMapTerrain.{h/cpp}

```
// This is an excerpt from code in HeightMapTerrain.cpp
if (ResourceLoader::get_singleton()->load(vformat("%s%s%s%s",
"res://Terrain/", texture_names[type], "_heightmap", ".png"),
"Image") == NULL) {

        // Generating noise based on pixels
        FastNoiseLite noise_gen;
        srand(time(NULL));
        noise_gen.set_seed(rand() % 100);
        noise_gen.set_noise_type(FastNoiseLite::TYPE_PERLIN);
        noise_gen.set_frequency(0.003);
        PackedFloat64Array pixels;
        for (int i = 0; i < image_height; i++) {
            for (int j = 0; j < image_width; j++) {

                // Determining noise per pixel
                float noise = noise_gen.get_noise_2d(i, j);
                pixels.append(noise);
            }
        }

        // Creating a new image and buffer for the noise image
        Image new_noise_image;
        new_noise_image.create(image_width, image_height, false,
Image::FORMAT_RGB8);
        PackedByteArray noise_image_buf;
        noise_image_buf.resize(image_width * image_height * 4); //
Multiplied by 4 to accommodate for all values in RGB8
        noise_image_buf = new_noise_image.get_data();

        // Copying collected noise to the buffer
        for (int i = 0; i < image_height; i++) {
            for (int j = 0; j < image_width; j++) {

                // Converting value to grayscale
                int val = int((pixels[i * image_width + j] + 1.0) *
```

```
127.5);

                // Appending three times for r g and b
                noise_image_buf.append(val);
                noise_image_buf.append(val);
                noise_image_buf.append(val);
            }
        }

        // Setting the image data
        new_noise_image.set_data(image_width, image_height, false,
Image::FORMAT_RGB8, noise_image_buf);
        new_noise_image.save_png(vformat("%s%s%s%s",
"res://Terrain/", texture_names[type], "_heightmap", ".png"));
    }
```

3. Game environment populated by textured, illuminated objects

Every object that visually appears in the game, namely the player, interactables, environment objects, the building, and the heightmap all incorporate textures by default through their code from textures stored in the Texture folder. Additionally, each of the aforementioned items supports lighting from multiple sources. Both the texture and lighting are applied by shaders written specifically for each item.

Associated Files: custom_scene_3501.{h/cpp}, Interactable.{h/cpp}, EnvObject.{h/cpp}, BuildingObj, HeightMapTerrain.{h/cpp}, Player.{h/cpp}, interactable.gdshader, envobj.gdshader, buildingobj.gdshader, terrain.gdshader, player.gdshader

4. At least one use of a screen-space special effect, with your own custom shader providing the effect

Given that we have chosen to use multiple third person cameras to display the game to the player, and the game setting necessitates the abundance of technology, it seemed fitting to use a static glitching screen space effect. At random points no sooner than ten seconds apart, for intervals of one second, a static effect will appear on the screen. At random, this static view will be slightly distorted creating a “glitching” effect for these cameras.

Associated Files: custom_scene_3501.{h/cpp}, static.gdshader

5. At least two distinct particle systems, created with your own custom code and shaders

The first particle system is a burning fire. The park in our game is meant to be dysfunctional, so the fire was used as a visual gag, strategically placed in front of the door to the pool, indicating that an area that is filled with water has been set alight.

The second particle system is meant to be a repulsor lift (technology that would help something hover in the air). It is attached to the park mascot that is found in the gift shop. The mascot hovers up and down in the air, and the particle system "powers up" periodically making it appear to output more thrust at certain times than others.

Each of these two systems has their own shaders that help create their distinct effects.

Associated Files: custom_scene_3501.{h/cpp}, particle_system_3501.{h/cpp}, Mascot.{h/cpp}, lonefire_ps.gdshader, lonefire_ss.gdshader, hover_ps.gdshader, hover_ss.gdshader, helpers.gdshaderinc

6. At least one hierarchical object with independently moving parts (e.g., a robot arm reaching, or branching plants swaying in the wind)

Our hierarchical system is the mascot for the amusement park. They are situated in the gift shop, and bob up and down in the air using floating technology visualized by the particle system (see requirement 5). The right arm waves continuously.

Associated Files: Mascot.{h/cpp}, MascotPart.{h/cpp}, MascotArm.{h/cpp}, MascotBody.{h/cpp}, MascotHead.{h/cpp}, particle_system_3501.{h/cpp}, EnvObject.{h/cpp}, envobj.gdshader, lonefire_ps.gdshader, lonefire_ss.gdshader, helpers.gdshaderinc

7. Player-centric camera with player controls linked to current orientation.

Our camera system is third person, with distinct cameras in each area of the park that switch as the player exits the area of one camera and enters the area of another. Although there are areas in which the camera is static and does not move, there are plenty of cameras throughout the park that either keep the player at the center of the frame (track them), or pan to keep the player close to the center of the screen when traveling across long distances. In all, the goal with every camera placement is to adequately display where the player is in any given area, as well as to display the player's surrounding environment in relation to them.

As for controls linked to orientation, each camera supports the same kind of movement in relation to the character. Moving forward will always move the player away from the camera relative to where it is looking, and moving backwards will always move the player towards the camera relative to where it is looking. Similarly, moving right or left will move the player in those respective directions relative to where the camera is looking. This is to say that regardless of the way the camera is facing the controls and the direction the player moves is uniform and predictable such that changing cameras is not jarring on how the player moves.

Associated Files: CameraTrigger.{h/cpp}, PlayerCamera.{h/cpp}, Player.{h/cpp}

8. A skybox

A skybox was implemented with a starfield representing the space environment in which the player finds themselves. The earth can also be seen on one side of the skybox. This can be viewed either from the starting area or from the windows of the domed park area. The skybox is displayed via a shader.

Associated Files: SkyBox.{h/cpp}, sky.gdshader

9. Basic game interface and staging functionality (probably provided by Godot): opening screen, pausing, game over screen

An opening, pausing, and game over screen were all provided. The pausing menu was done through GDExtension out of necessity of it being in the same scene as our gameplay, while the opening and game over screen were all provided with Godot. Although they were not required, a "How to Play" and "Credits" page were also included.

Associated Files: start_menu.tscn, how_to_play.tscn, credits.tscn, end.tscn, PauseScreen.{h/cpp}

Section 5. Beyond the Minimum

There are a number of systems we wrote that exceed the minimum requirements. Some major features beyond the minimum that we make use of are the interactable systems, our multi-3rd person camera system, inventory implementation, and indoor environment in tandem with environmental storytelling.

Our camera system

Lunar Land makes use of a 3rd person multi camera setup like the old metal gear solid or resident evil games. A mixture of static, panning, and tracking camera are switched between depending on where the player walks. Not only does this add to the overall eerie atmosphere, but it allows us to set up camera angles. These camera angles are sometimes used for aesthetic reasons, and sometimes used to tell the story, to highlight a particular item or area in the room. In some cases, moving very close to a hidden item will reward the player for their exploration with a shot that highlights the item, allowing them to acquire it. The camera system allows us to curate how we show off our stunning Indoor environment. Great care was taken to make sure the system still met the player centric camera criteria while offering something unique to the gameplay.

Also we made it so that if you keep holding down the direction you were moving before the camera switched, you will still be headed in that direction when the camera switches, that and a few other quality of life features make the camera system really feel good to use.

Indoor environment

Roughly 95% of our game takes place inside of the enclosed LunarLand environment. The building has varying elevations, windows that allow you to see outside, and other factors we had to plan around when designing the space. We had to think not just about how we wanted to design each room to facilitate gameplay, but also what might be natural for a space like this to have, and factor that into the design. The spaces have set decoration, wall fixtures, attractions, recreational equipment, desks, etc.

Inventory system

The core narrative drive of Lunar Land is that you were sent here to retrieve various artifacts of great import to the company. You need to add all three of these items to your inventory to finish the game, you do this with our interactable objects. As you add items to your inventory, the hotbar at the bottom of the screen will fill up with the items you have found.

Interactable Objects & Triggerable Effects

There are various interactable objects scattered throughout Lunar Land, audio logs, items to collect, a power generator, and mechanized lockers. Lunar Land is first and foremost a puzzle exploration game, where you navigate and interact with the environment to find the items you want. Interacting with audio logs to learn relevant lore that concerns the item you are hunting, picking up the items when you find them, are all important parts of the experience, and were worthy of the time investment. We also wanted to set the employee lounge part of the game apart from the other 2, instead of hunting an item, you already know roughly where it is, and rather need to determine who stole it so you know whose locker to spend your precious electricity on to open it. Here you are exploring to find the truth. It adds diversity to the gameplay, and requires you to further engage with the story, as making the wrong accusation leads to more cranks required to power the generator to make your next guess.

You will know you can interact with all of these items when parts or the entirety of the object glow blue, demonstrating that the respective item can be interacted with. In a cluttered environment, it was necessary to have a method to display that you can interact with a particular object, and making it glow helps it stand out in the environment.

A significant amount of time was spent planning and implementing the interactables system. Considerations as to their design included what kind of behaviour interactable items needed to have, how to distinguish an interactable in the environment, and what kinds of items we would need to be interactable. In terms of implementation, a large chunk of time went into concerns such as making sure lockout interactables were synced with each other as well as a counter interactable that would unlock them, item interactables working in tandem with the inventory, and audio sourcing and design for all interactables in general.

Narrative & Environmental Storytelling

This is the big one, all of these systems were made to facilitate environmental storytelling. Listening to audio logs describing an action or detail, like someone playing a round of darts, and then going to their bedroom and finding a dartboard with darts still in it, makes you feel like you are exploring a lived in world. Each level requires you to engage with the storytelling present in both the environment, and what you hear on the audio logs, to solve their respective puzzles.

Take, for instance, finding the Lunar Lint (SPOILERS) by establishing Johnny's guilt. You can either hear on an audio log that Johnny said Timmy was running somewhere at 5:00pm and then find Timmy's timecard confirming that he was already clocked into his second job at 5:00pm (highlighted by an up close camera angle), and figure out that Johnny was lying to cast suspicion off of himself. Or you can listen to an audio log of Johnny eating something slurpy in the cafeteria, remember he said he was eating steak, and then find out that he tried to destroy the menu in the cafeteria, and figure out he's guilty that way. Or you can find other evidence that exonerates Bobo and Timmy, and find out he's guilty via process of elimination. You can only reach these conclusions by interacting with both the audio logs and the environment, by engaging with the storytelling to solve the puzzles.

Section 6. Post-Mortem

Initially our project was much more ambitious. Various tools and powers, 3 different puzzles that each required their own custom subsystem, a small game economy, it was a lot to implement. We eventually decided that this would not be a good idea to go ahead and build, so we went back to the drawing board. When we replaced the game to address the scope problem we decided we had to go all in on one or two core features. We decided to turn one of the puzzles, a “murder mystery” type game into the entire game. From that came the interactables system, and environmental storytelling, all other design decisions and “beyond the minimum” components were born from that. We were able to get everything that this second draft of the project outlined. If we had more time we would definitely just work on details, improving the models, populating the game space with more objects and detail, adding actual doors to the pool and employee dormitory sections, enriching the world we have right now without adding onto it too much. Additionally our “last pass” of the wall and floor textures got corrupted and lost less then a day out from the deadline, so we hope you enjoy the finest monochrome color pallet LunarLand has to offer.

Each member of the team has their own “incredibly biased” aspect of this game that we are most proud of. For Mason it's the world and the models he spent forever retexturing and tinkering with. For William it is the 25 cameras and 63 triggers he placed, and the system that made them and the player movement possible. For Jacob it's the interactable system he spent forever on, without which the entire game would not be possible.

A major piece of advice, though it was said in class, would be to start early and work often. There were some things in the development of the game that were left later than they should, and became large time sinks all at once in large part because they were not done beforehand. Another piece of advice would be to have a plan. Initially, as previously stated, we had many ideas for the game but no plan on how to implement them. This proved problematic when we got to talking about making the game, and we realized how infeasible our vision was. Readjusting our ideas and creating a plan as to all the aspects of our game and what would be needed to implement them proved to be invaluable through the rest of the design process.

We think the key to our success was constant communication, every week (and as we got closer to the due date, every day) we would talk about what was next for the project, and what each person would be working on. We constantly brainstormed, gave feedback on each other's work, tested each other's code, and just generally had a great time together.

Section 7. Credits

Jacob:

- Interactables (including Audio, Item, Counter, and Lockout Interactable subclasses and shader)
- EnvObject (environment objects class and shader)
- BuildingObj (building class and shader)
- HeightMapTerrain (height map class, shader, cage class that keeps the player in a certain area on the terrain)
- Inventory
- Mascot (the mascot class, the MascotPart class that builds the mascot and associated subclasses)
- Particle systems (the fire and the repulsor lift effects)
- Skybox
- Implementation of lighting and textures (both in C++ and shaders, across all classes)
- Voice line outsourcing/processing
- Inventory sprites (hotbar, soda chip, health papers, lint)
- Placing interactables, environment objects, building, heightmap, particle systems, hierarchical object

Mason:

- Assisted in programming for the player and camera systems.
- Model geometry and textures

William:

- Cameras and camera triggers (everything in CameraTrigger/{h/cpp} and PlayerCamera.{h/cpp})

- Player movement and most of the Player class (Player.{h/cpp}
- Placing the cameras, camera triggers, interactables, light positions, and items.
- Re_parent(), and some other small utility functions.

Voice Cast:

- Susie from HR → Caitlin Wardle
- Joe Law → Brennen Ray
- Bobo the Clown → William Graham
- Johnny the Hungry → Luke Beausoleil
- Timmy Two-Jobs → Jacob Bonner
- Hootenanny Harry → RJ Fromm
- Sebastian Stanza → Cameron Carter
- Penny Pincher → Ella Richmyre
- Busy Izzy → Aurora Tracy
- Tyler → Luke Beausoleil
- Mr Mcnamra → Benjamin Whitten

Other Credits:

- Moon Surface Texture → Freepik (https://www.freepik.com/free-photo/black-white-details-moon-texture-concept_29662110.htm#fromView=keyword&page=1&position=2&uuid=817eafe0-6585-4bc4-ba7c-eac145809aef&new_detail=true)
- Sand Texture → Freepik (https://www.freepik.com/free-photo/yellow-textured-wall-background-texture_1198462.htm#fromView=keyword&page=1&position=14&uuid=b3a5d0e7-395c-41ca-b51f-492bbf48e2d7&new_detail=true)
- Starfield Texture → Space Spheremaps (https://space-spheremaps.itch.io/space-spheremaps)
- Earth and Moon Textures → Deep-Fold: (https://deep-fold.itch.io/pixel-planet-generator)
- Thaleah Fat Font: (https://tinyworlds.itch.io/free-pixel-font-thaleah)
- Ferris Wheel Model: Perkoules (https://www.cgtrader.com/free-3d-models/architectural/other/ferris-wheel-771b6f9d-816e-4cdc-bc15-6ec8456254f2)
- Carousel Model/Texture: agardiandesinger (https://www.cgtrader.com/free-3d-models/furniture/outdoor-furniture/carosuel-for-funfair)
- Chair Model/Texture: ponomarev1983 (https://www.cgtrader.com/free-3d-models/interior/kitchen/chair-e6b64a95-188b-4fc2-a1b3-7bab1673a593)