Max Philip (836472), Jonas Olausson (751462)

COMP30019 Project 2 Report

Brief explanation of the game

The game is titled “Astro-Exterminator”, a 3rd person controlled shooter game. The player takes the form of an astronaut that travels to different planets and performs a pest extermination type service upon the request of the inhabitants of the respective planet. The game’s main objective is to eliminate enemies taking the form of alien brains (called BRALIENS) suspended in coloured force fields. The BRAILIENS are able to damage the player by making contact with them, and the colour of the force field indicates the life status of the enemy (intuitively green for full health and red for low health).

The player navigates the surface of a spherical planetary game area, which differs in terrain across the three current levels, and attempts to find and eliminate all of them by shooting them with their projectile weapon. A mission/level is completed once all the enemies have been eliminated without the player dying. Furthermore, completing a level has the potential to set a new high score (shortest time to clear the planet), if the level is cleared faster than the previous high score. This gives two levels of progression to the game. First, working towards being able to fully clear each level at least once, and second, setting new high scores and further challenging even the most experienced players.

The player earns money by eliminating enemies, which can be used in the equipment store to upgrade their equipment. Players are able to upgrade their weapons (to new weapons with higher damage output), maximum health, jet fuel capacity and sprint speed. The final piece of equipment the player has access to is their jetpack, which can be used to fly upwards perpendicular to the planet surface. The jetpack is limited by its jet fuel capacity, which refills when not in use.

How to use the game

Controls consist of:

- **MOUSE** for looking around in 3rd person view. Restricted to left and right rotation of the character.
- **WASD/arrow keys** for movement on the planet surface.
- **LEFT CLICK** for firing their projectile weapon. This is the player’s means to eliminate enemies and comes in three varieties (in ascending order of damage), the **rock, grenade,** and **pulse bomb**.
- **SPACEBAR** for using the jetpack. It is used by holding the spacebar down, which propels the player upwards while the jetpack still has fuel.
- **RIGHT CLICK** for flipping the player in mid-air. This only works while the player is off the ground, and so is usually combined with the jetpack. This is a purely cosmetic function that simply games the game a more polished, seamless feel.
- **LEFT SHIFT** for sprinting. Increases the player’s move speed on the ground and has no limit to its use.
- **ESCAPE BUTTON** for pausing the game while on a mission.

The game user interface makes use of menus through which the user can navigate, allowing them to select their character's weapon, sprint speed, jetpack capacity and maximum health. Upgrades can be purchased on the "Equipment" menu with in-game currency earned through playing the game.

The game proceeds to the "Mission Select" menu, where the available levels (and short descriptions) are displayed and selected by being left-clicked. The level difficulties increase from left to right. The **tutorial** has no damaging enemies and is simply for practice. **Dune** is a desert planet, with 12 harmful enemies roaming around. **Terra** is a grassy, tree-filled planet, with 20 harmful enemies roaming around.

The in-game UI consists of current health, jet fuel and money in the top left. Remaining enemies on the planet and time elapsed are displayed in the top right. Pausing the game stops the timer and displays the controls for reference. From the pause screen, the player can resume, return to the main menu, and quit to desktop.

If the player dies before eliminating all the enemies, they must return to the main menu from the death screen. However, players keep the money earned on failed missions, and so can still upgrade their gear before trying again.

If the player succeeds, the success UI is displayed. If the completion time is a new record, the high score time on the "Mission Select" screen will be updated for the given level.

How the objects and entities were modelled

From the beginning, we wanted a "lowpoly" aesthetic to the game. A variety of lowpoly assets from the Unity Asset Store were used to model the planets (listed in references) and were customised to suit the theme of each planet. The spherical surface of each level was an imported asset made in Blender (by a third party), in order to achieve a much smoother planet surface – which fixed the issue of the resolution of the default Unity sphere game object not scaling well. Several different lowpoly asset packs were used in conjunction to achieve more variety in the level design, ranging from various trees, rocky outcrops, flora, and many miscellaneous additional objects.

Furthermore, the rocky ground surfaces were done by repurposing mountainous models, through manipulating their scale and rotation to give the appearance of unique terrain throughout the planets.

The player astronaut model was also a free asset from the Unity Asset Store, which came rigged. This allowed us to manually link animations to player input, and achieve relatively seamless custom animation transitions between walking, running, jetpack flight, and mid-air flipping.

There is a day night cycle on each planet, which was done by applying orbit scripts to two directional light sources.

Projectiles include a rock, a grenade and a pulse bomb. The particle effect for rock collision was taken from the particle effects lab files, and both the effect for the grenade and the pulse bomb were taken from the Unity Particle Pack 5.x. All the particle effects had their attributes and properties quite heavily modified for basic functionality (such as not looping once fired), as well as to suit the game style. The other particle system used is the smoke particle system for the jetpack.

The enemies were modelled by placing a brain model and a point light within the spherical game object with our force field shader applied to it. The brain models are separate to the force field, which enabled us to have them wander aimlessly when not chasing the player, but when engaged the brain rotates around the z-axis and points directly at the player.

The planets also have water areas, which was done by placing a flat water plane (with our Gerstner shader) surrounded by rough terrain.

Audio was all sourced from royalty-free online repositories, and was played using AudioSources within "SoundControllers" and the sound-emitting objects themselves.

How the graphics pipeline and camera motion were handled

Part of the reasoning behind choosing a 'low poly' artistic style to all our components in the world, was to reduce the load on our computers, as well as a stylistic choice we thought suited our game the best. The fewer polygons on each of our models heavily reduced the load on the GPU and led to a smoother playing experience. Furthermore, the mesh colliders on many of the models (save a few exceptions), were set to be convex. This had the simultaneous benefit of reducing the necessary resolution of the collider (due to the convex collider reducing the total number points defining the collision area) and negating the occurrence of tunnelling in these cases. As is the case in default Unity projects, back-face culling was employed as a step in the graphical pipeline to reduce the rendering load.

Our camera mechanics was defined as a fixed 3rd person perspective with respect to the astronaut player character. The camera is positioned above and faces down at a slight angle. The camera is fixed in the pitch axis (in terms of direct player manipulation) but is allowed to move in the yaw axis with respect to the player. As a result, the player cannot look up or down but only left and right. An important note is that, after conducting our evaluation, we made a slight change to the camera rotation up and down. As mentioned previously, the player is not able to directly control this rotation, however, the yaw is tied directly to the player's distance from the planet surface. The higher the player is (through using the jetpack), the further down the camera is angled. This allows the player to get a wider perspective of the map mid-jump and proved to be a better fit to our game. It eased the fixed nature of the camera perspective and makes the camera less one-dimensional.

Custom shader descriptions

1. Enemy "force field" shader (ForceFieldShader.shader**)**

This shader was the custom shader we employed that was **not** explored in labs. The effect is essentially that of a spherical force field that surrounds the enemy brains. The colouration

of the force fields is indicative of the enemy's current health. We found this to be more intuitive and aesthetically pleasing than having a health bar. This shader is a surface shader, and so effectively alters the look of the material applied to the force field. In fact, the material is dynamically set based on the enemy current health, and then the shader uses the surface function to take the 3D model data and output the desired rendering properties. The view direction part of the input struct is used to give the transparent centre appearance of the sphere (making the brain visible to the user). The "edge" property gives the fade in colour around the edge of the sphere – with the colour specified by the material the shader is applied to.

2. Gerstner wave shader (WaveShader.shader**)**

This water shader gives the effect of clear waves. It works similar in principle to the water plane used in Project 1, however several substantial changes were made. In place of the simple sine wave used previously, the wave effect was done in a Gerstner wave style. This effectively combines multiple sine and cosine into the wave calculation, which gives a more natural and interesting look. Further changes were made to parameters such as texture tint to better suit the environment. This shader was the one we included that **was** explored in labs. It simply displaces the vertices on the plane according to the aforementioned Gerstner

Querying and observational methods

Questionnaire: One querying technique to evaluate (5 participants)

For this evaluation technique, we asked participants to play through each level of our game with none of our intervention. Once finished with the game, participants were asked to fill out this questionnaire. We handed them the computer and let them type away their responses to the three questions below. We recorded 5 residential college students from the ages of 19-21 all either slightly familiar with video games or being very familiar with video games.

Question	Participant's Response (A)	(B)	(C)	(D)	(E)
Which aspect of the game did you like the least?	"Felt the fixed camera perspective was difficult to get used to at times"	"I disliked the enemies, didn't really feel like spheres fit"	"I couldn't really see my bullets shoot when I was in the dark parts of the world, also I couldn't see the enemies well in the dark either"	"Didn't like the jumping mechanic"	"I found it easy to move around"
Which aspect of the game did you like the best?	"The aesthetic was really cool, I liked the character model, it all fit together nicely"	"Being on a tiny planet and seeing your bullets orbit the planet was a great touch"	"Loved the low poly environment you've built! The mars planet looks very well done"	"Could see the projectile going cross the screen. aesthetically satisfying environment"	"I was unable to jump sometimes"
What do you think could be improved?	"Adding in a sense of progression would be nice, it would get boring just shooting at spheres the whole time"	"make the enemies look more threatening"	"Make things glow or something, hard to see when theres no light. Also I believe you can make the controls more obvious to the player"	"Have instructions more clear for the tutorial"	"Make it so you can jump"

Think Aloud: One observational method to evaluate (5 participants)

For this evaluation method, we gave the player our game and gave them a debrief about what they were about to do. We let them play our game except we gave them zero directions and allowed them to see how much they could figure out from just playing the game. We told the participants to verbalise their thoughts out loud as much as possible. We recorded their verbal responses during their gameplay. We recorded 5 residential college students from the ages of 19-21 all either slightly familiar with video games or being very familiar with video games.

Participant	Recorded Verbal Responses During Gameplay
A	"How much money do I have at the start?" "Can I sprint?" "you should make tutorial more of a tutorial, I still have no idea how to play" (while in the dark part of the planet where bullets are hard to see) "I can't shoot anymore, think I've run out of bullets"
B	(During tutorial level)"What do I do next?"
C	(when trying to jump, they didn't you had to hold spacebar instead of press)"I'm not jumping?"
D	"How do I go faster?" "how do I jump', 'do I hold space bar"
E	"Where are the enemies?" "its too hard" "how do I run" (after pressing space bar quickly)"jumping doesn't do anything"

Description of the participants (how many, demographics):

We sampled in total 10 students of the University of Melbourne. All were of ages 19-21 with either a slight familiarity with video games or were very familiar with video games

Description of the methodology (which techniques did you use, what did you have participants do, how did you record the data):

For the think aloud method, we took audio recordings of the gameplay, which we went through afterwards. This allowed us to take in the most feedback possible.

For the questionnaire, we simply used a pencil and paper as the data recording method.

Changes made based on evaluation

We found that all the users we worked with enjoyed the low poly aspect of our game and its general aesthetics, furthermore they found the main premise of the game engaging and worth their time.

We attempted to take the following constructive criticisms from users and turn them into improvements in our game.

Camera Motion

Multiple users reported having felt at unease with the camera's fixed nature. We attempted to make the camera above the player at a greater distance and with its movement independent from the player character. This meant having the camera follow the player around in its direction of movement but from an almost birds eye view. This proved to be difficult to do since our camera had to be fixed to the sphere, this proved to be less useful in conveying player movements then we had previously thought. **We then tried to incorporate a changing field of view when the player performed a jumping movement.** This allowed the player to get a larger perspective of the map when on a higher elevation from a jump and proved to be a better fit to our game and eased the fixed nature of the camera perspective.

Unclear Tutorial

Player's mostly felt confused about the controls of the game. This was an obvious oversight as there were no explicit guidelines on how to play the game. Even though there was a button to go to the control manual page, users couldn't really see it since it was blue and the background of our skybox was black. **We then decided to add text explaining where to go when you confused about the controls.**

Another issue during the tutorial level, was that several participants felt a lack of guidance and kept asking to themselves, 'what do I do now'. The cubes moving up and down were supposed to be target practice for the user and this was not immediately clear to most participants. **We ended up providing directions as an overlay during the tutorial level as well to guide the user to understand the game better.**

Jump Mechanics

A common issue amongst participants was the lack of a clear jumping mechanic. Users had thought the command 'jump' entailed a simply press of the space bar button. This led to them confused when they were not able to jump, as our game required them to hold the space bar. We felt that it was a good idea to make it clear that the jump function was actually a jetpack. **This led to us adding in a particle system with sound effects during the action of using the jetpack.** This immediately gives the user with visual and audio feedback that enforces the action of using the jetpack. Furthermore, through testing the game with some participants, we were able to identify bugs where users were not able to jump at certain locations and times. This allowed us to identify the issue and correct it.

A More Pronounced Sense of Progression

A participant pointed out in the questionnaire that they did not feel a strong sense of progression throughout the game. **We agreed completely and decided to add an equipment section where the user could purchase upgraded abilities with money - points awarded to the character when eliminating enemies on planets. In addition, we added a best time feature to each level - displaying the shortest completion time of the level as a high score. Lastly, a participant was confused about the mechanics of the equipment store and how many things they could buy. We took this feedback and made the amount of money you currently have larger and bolder, so users could not miss it.**

Indicators (additional change)

In our first iteration, we were split on a design decision – whether we should include markers that indicated the location of nearby enemies. During the evaluation process, we had five of our testers run through a game level twice with the indicators on, and twice with the indicators off. After the testing we asked our users closed questions in the interview with regards to their preference, and we were surprised to find that four of these five testers preferred having no indicators. They gave reasons such as a more satisfying level of difficulty and accomplishment when killing enemies without the aid of visual cues, and that the atmosphere of the game felt more natural when identifying threats by the sound of the enemies alone. Given this feedback, we made the decision to remove enemy indicators from the final game.

External Resources

Videos used as reference:

Menu tutorial: https://www.youtube.com/watch?v=zc8ac_qUXQY

Some logic for type writer effect: https://www.youtube.com/watch?v=1qbjmb_1hV4

Links used as reference:

References for surface shading (for "force field" shader):
https://www.alanzucconi.com/2015/06/17/surface-shaders-in-unity3d/

http://www.shaderslab.com/demo-43---misc.-rim-effect.html

Reference to understand spherical gravity: https://github.com/SebLague/Spherical-Gravity

Official Unity tutorial for transparent shaders:
https://unity3d.com/learn/tutorials/topics/graphics/making-transparent-shader

Projectile lab for COMP30019.

Audio files sourced from:

https://www.dl-sounds.com/

https://freesound.org/

(Some) Assets used from the Unity Asset Store:

LowPoly Environment Pack	Korveen
Low Poly Survival Essentials	Broken Vector
Free Low Poly Pack	Broken Vector
Low Poly Pack	Andrey Graphics
Low Poly Game Kit	JayAnAm
Real Stars Skybox	Geoff Dallimore
Brain Meal	X-FACTORY

Group Contributions

Max Philip (836472)

- Level design (Dune & Terra)
- Evaluation (Think Aloud)
- Sourcing and applying effect audio (including weapon sounds, button click sounds, misc.)
- Gravity system
- Store and equipment system
- Projectile systems
- Enemy pathing and roaming AI
- Force Field shader (both)
- Gerstner shader (both)
- Statics and player values (health, max fuel, weapon etc.)
- Mission Select UI
- In-game heads up display
- In-game pause screen
- Jet pack functionality
- Projectile particles and particle adjustment
- Victory screen
- Player animation linking & logic

Jonas Olausson (751462)

- Level design (tutorial)
- Evaluation (Questionnaire)
- Sourcing and applying music audio
- Gravity system
- Enemy design
- Implementing pond effects
- Force field shader (both)
- Gerstner shader (both)
- Main Menu, type writer effect
- Controls text
- Jet pack particles & adjustment
- Sprinting
- Mission Failed screen