COMP3501A - Final Project Report

Group Members

Navin Haider | 101081934
Pablo Rascazzi | 101110139
Ben Williams | 101105663

Git Repository

https://github.com/benwilliams140/comp3501-project

Clone Instructions

Found in the repository README.md file.

Project Structure

The headers and source files are split up into the `include` and `src` directories; these are then further split into `Control`, `Objects`, and `Renderer`.

The CMake structure is recursive in nature - each of the `src` subfolders have their own `CMakeLists.txt` file containing each of the .cpp files and any additional subfolders.

All of the resources such as meshes, textures, and shaders are in the `res` directory.

We also use the ImGUI library to generate and render all of the GUI elements such as the HUD and the menus.

Class Tree

Code that we wrote.

Code from demos that we edited.

Code that Pablo wrote for other projects.

```
.
├── include
│   ├── Control
│   │   ├── GUI
│   │   │   ├── Menus
│   │   │   │   ├── game_over.h
│   │   │   │   ├── hud.h
│   │   │   │   ├── main_menu.h
│   │   │   │   ├── pause_menu.h
│   │   │   │   ├── text_window.h
│   │   │   │   ├── upgrades.h
│   │   │   │   └── you_win.h
│   │   │   └── menu.h
│   │   ├── game.h
│   │   ├── input.h
│   │   ├── mathematics.h
│   │   ├── path_config.h
│   │   ├── resource_manager.h
│   │   ├── resource.h
│   │   ├── scene_graph.h
│   │   ├── scene_node.h
│   │   └── time.h
│   ├── Objects
│   │   ├── Hazards
│   │   │   ├── acid_pool.h
│   │   │   ├── geyser.h
│   │   │   ├── hazard.h
│   │   │   └── mud_pool.h
│   │   ├── Hovertank
│   │   │   ├── Abilities
│   │   │   │   ├── ability.h
│   │   │   │   ├── energy_cannon.h
│   │   │   │   ├── EnergyEmitter.h
│   │   │   │   └── machine_gun.h
│   │   │   ├── hovertank_track.h
│   │   │   ├── hovertank_turret.h
│   │   │   ├── hovertank.h
│   │   │   └── scanner.h
│   │   ├── Projectiles
│   │   │   ├── Enemy_linear_projectile.h
│   │   │   ├── EnergyBlast.h
│   │   │   ├── linear_projectile.h
│   │   │   ├── parabolic_projectile.h
│   │   │   └── projectile.h
│   │   ├── Artifact.h
│   │   ├── CarePackage.h
│   │   ├── ChaserEnemy.h
│   │   ├── Enemy.h
│   │   ├── EnergySource.h
│   │   ├── EnvironmentObject.h
│   │   ├── particle.h
│   │   ├── Player.h
│   │   ├── ShooterEnemy.h
│   │   └── terrain.h
│   └── Renderer
│       ├── camera.h
│       ├── ebo.h
│   └── vao.h
├── res
│   └── ...
├── src
│   ├── Control
│   │   ├── GUI
│   │   │   ├── Menus
│   │   │   │   ├── CMakeLists.txt
│   │   │   │   ├── game_over.cpp
│   │   │   │   ├── hud.cpp
│   │   │   │   ├── main_menu.cpp
│   │   │   │   ├── pause_menu.cpp
│   │   │   │   ├── text_window.cpp
│   │   │   │   ├── upgrades.cpp
│   │   │   │   └── you_win.cpp
│   │   │   ├── CMakeLists.txt
│   │   │   └── menu.cpp
│   │   ├── CMakeLists.txt
│   │   ├── game.cpp
│   │   ├── input.cpp
│   │   ├── resource_manager.cpp
│   │   ├── resource.cpp
│   │   ├── scene_graph.cpp
│   │   ├── scene_node.cpp
│   │   └── time.cpp
│   ├── Objects
│   │   ├── Hazards
│   │   │   ├── acid_pool.cpp
│   │   │   ├── CMakeLists.txt
│   │   │   ├── geyser.cpp
│   │   │   ├── hazard.cpp
│   │   │   └── mud_pool.cpp
│   │   ├── Hovertank
│   │   │   ├── Abilities
│   │   │   │   ├── ability.cpp
│   │   │   │   ├── CMakeLists.txt
│   │   │   │   ├── energy_cannon.cpp
│   │   │   │   ├── EnergyEmitter.cpp
│   │   │   │   └── machine_gun.cpp
│   │   │   ├── CMakeLists.txt
│   │   │   ├── hovertank_track.cpp
│   │   │   ├── hovertank_turret.cpp
│   │   │   ├── hovertank.cpp
│   │   │   └── scanner.cpp
│   │   ├── Projectiles
│   │   │   ├── CMakeLists.txt
│   │   │   ├── Enemy_linear_projectile.cpp
│   │   │   ├── EnergyBlast.cpp
│   │   │   ├── linear_projectile.cpp
│   │   │   ├── parabolic_projectile.cpp
│   │   │   └── projectile.cpp
│   │   ├── Artifact.cpp
│   │   ├── CarePackage.cpp
│   │   ├── ChaserEnemy.cpp
│   │   ├── CMakeLists.txt
│   │   ├── Enemy.cpp
│   │   ├── EnergySource.cpp
│   │   ├── EnvironmentObject.cpp
│   │   ├── particle.cpp
│   │   ├── Player.cpp
│   │   ├── ShooterEnemy.cpp
│   │   └── terrain.cpp
│   ├── Renderer
│   │   ├── camera.cpp
│   │   ├── CMakeLists.txt
│   │   ├── ebo.cpp
│   │   └── vao.cpp
│   ├── CMakeLists.txt
│   └── main.cpp
├── CMakeLists.txt
├── project_report.md
└── README.md
```

Assets / Credits

- **Fonts:**

Asset Name	Author	Link
Freedom.ttf	HXDes	https://www.fontspace.com/freedom-font-f14832
FabulaValhalla.ttf	Adam Alexander T. Croke	https://www.1001freefonts.com/fabula-valhalla.font

- **Meshes:**

Asset Name	Author
plant.mesh	Pablo Rascazzi
pool.mesh	Ben Williams
rock1.mesh, rock2.mesh, rock3.mesh	Pablo Rascazzi
hovertank_Chassis.mesh	Pablo Rascazzi
hovertank_Scanner.mesh	Pablo Rascazzi
hovertank_Scanner_Cone.mesh	Pablo Rascazzi
hovertank_Track_Front.mesh	Pablo Rascazzi
hovertank_Track_Rear.mesh	Pablo Rascazzi
hovertank_Turret.mesh	Pablo Rascazzi
hovertank_Machine_Gun.mesh	Pablo Rascazzi

Asset Name	Author
artifact.mesh	Pablo Rascazzi
enemy.mesh	Pablo Rascazzi
parachute.mesh	Pablo Rascazzi
rubber_bullet.mesh	Pablo Rascazzi
spike_ball.mesh	Pablo Rascazzi

- **Textures:**

Asset Name	Author	Link
acid.png	Horus	https://www.filterforge.com/filters/2152.html
alien_plant.png	Frostguard	https://www.filterforge.com/filters/8489.html
dirt.png	Sneath	https://www.filterforge.com/filters/11775.html
geyser.png	Jakob Owens	https://unsplash.com/photos/nKFPNayRbFE
mud.png	G.A.G	https://www.filterforge.com/filters/6278.html
rocky.png	David Mould (Texture Demo)	
snow.png	Ghost IV	https://www.filterforge.com/filters/992.html
electric.png	Rian	https://www.filterforge.com/filters/4740.html
energy_blast.png	Francisco Jacobo	https://www.filterforge.com/filters/14211.html
rubber.png	ScreenscapeS Edited by: Pablo Rascazzi	https://www.filterforge.com/filters/13303.html
chaser_enemy.png	Bobbiethejean	https://www.filterforge.com/filters/9039.html
crate.png	Mike Blackney	https://www.filterforge.com/filters/750.html
hovertank_texture.png	Katsukagi Georges "TRaK" Grondin Fupi Edited by: Pablo Rascazzi	https://3dtextures.me/2019/04/30/metal-plate-012/ https://3dtextures.me/2019/06/11/metal-plate-020/ https://opengameart.org/content/wall-cover-3 https://opengameart.org/content/old-ventilation-cover-dark https://opengameart.org/content/metal-panel-dark-brown-alt https://opengameart.org/content/wall-cover-4

Asset Name	Author	Link
		https://opengameart.org/content/warning-stripes-trim-trim-medium-border-light https://opengameart.org/content/shiny-window-pane
parachute.png	minglepingle (Cloth) Novaesky (Steel Cable) Edited by: Pablo Rascazzi	https://www.filterforge.com/filters/12704.html https://www.filterforge.com/filters/15744.html
scanning.png	capnsparrow Edited by: Pablo Rascazzi	https://www.filterforge.com/filters/4106.html
shooter_enemy.png	Bobbiethejean	https://www.filterforge.com/filters/9039.html
sparkle_particle.png	Ben Williams	
water_particle.png	David Mould (Fire Demo)	
circuits.png	Vladimir Golovin	https://www.filterforge.com/filters/215.html

Asset Name	Author
button.png	Navin Haider
energy_bar_background.png	Navin Haider
energy_blast.png	Navin Haider
game_over.png	Navin Haider
health_bar_background.png	Navin Haider
injured_screen_effect.png	Ben Williams
linear.png	Navin Haider
mission_accomplished.png	Navin Haider
parabolic.png	Navin Haider
title_screen.png	Navin Haider

Fulfillment of Requirements

Game Description

The setting of the game takes place on an alien planet, where you are an archeologist tasked with finding artifacts hidden across the world. Within the game itself we brief the player of their mission in a text box at the beginning of the game. The text is actually the paragraph right below which you can look at for reference.

"Hello fellow archaeologist, You've been tasked by Mould Incorporated with finding and scanning all the ancient artifacts on planet L113. Beware of the native alien lifeforms as they are protecting these artifacts and will attack when approached. Your employer specifically stated not to kill any of these alien lifeforms, so you must only defend yourself with non-lethal weapons. For every artifacts scanned, your employer will pay you with credits which you can use to buy upgrades for your hovertank. These upgrades will be air dropped to you, so watch the sky for falling care-packages. Good luck!"

Technical Requirements

0. Written in C++ using OpenGL to render; readable code with no serious bugs, suitable class hierarchy, good documentation.

- We wrote the project in C++ using OpenGL with documentation throughout.

1. Large textured heightfield terrain with collision detection.

- We generated a terrain heightfield using three layers of Perlin noise.
- The terrain has two types of collision detection, one which does checks for collision between a line and a triangle, and another that interpolates the height of all 4 corners of a terrain quad.

2. Game environment populated by textured, illuminated objects.

- All objects in the world are textured, some objects have moving textures.
- All objects in the world are illuminated using Blinn-Phong lighting and a directional lighting.
- The world is populated by three variations of instanced rocks along with a plant.
- There are three types of environmental hazards throughout the world, a mud pool, an acid pool, and a geyser.

3. At least one use of a screen-space special effect.

- When the player is injured, the screen flashes red on the corners.
- There is a chance for the player to be stunned when hit with an enemy projectile - this adds a multi-layered screen shaking effect that reduces over time.

4. At least two distinct particle systems.

- All of our particle systems use a single point which is instanced and then expanded to quads in the geometry shader.
- Each geyser has a particle system that simulates water being shot up into the air.

- Each artifact has a "sparkling" effect before it has been scanned, this informs the player that it needs to be scanned.

5. At least two hierarchical objects with independently moving parts.

- The hovertank is a hierarchical object that has a base as the root, and a turret and four hover tracks as children of the base. The the tank's turret has a scanner and 2 guns as children.
- The care package has a parachute as a child which rotates individually from the crate.

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

- Our game's camera is in third-person with the tank as its target.
- The camera rotates around the tank on the y-axis using the mouse movement.

8. Multiple stages to game.

- The game has a main and a pause menu, along with an upgrade screen (that pauses the game), and mission accomplished/game over screens.

Gameplay Requirements

0. Hovertank navigating alien planet.

- The game is set on an alien planet and has a hovertank as the main player.

1. A thought-out setting with a variety of objects, including scenery and interactable objects both friendly and unfriendly. Give the player a role in the setting and a reason to take on the game's tasks.

- The world is populated with a variety of rocks, plants, and environmental hazards that can be collided with.
- Each of the environmental hazards has a different effect on the player; the geyser shoots them up into the air, mud pool slows them down, and acid pool slowly damages them.
- There world has interactable objects such as artifacts, which needs to be found and scanned, and care packages which fall from the sky when you buy a weapon upgrade for your tank.
- Two types of aliens inhabit the world, one that shoots projectiles and one that chases the player and deals melee damage.
- The player has a reason to be on the planet, he is an archaeologist that was contracted to find all the artifacts on an alien planet.

2. Tasks for the player to do.

- The player must find and scan all artifacts the artifacts while actively evading aliens and hazards.

3. At least three distinct devices the player can use.

- The scanner: used to scan artifacts and open care packages.
- The machine gun/energy cannon: different types of projectiles that stun the enemy when hit, one that is affected by gravity and one that isn't.
- Energy blast: a shield like blast that stuns enemies in it's radius.

4. Some kind of upgrade path fostering a sense of progression.

- When the player scans an artifact, they are awarded a set number of points, which can then be used to purchase upgrades in the upgrade menu.
- The player is able to upgrade their health, speed, and energy (each has two levels), along with the effectiveness of each of the hazards (The player can choose which upgrade path to start with).
- The player can also purchase different abilities: the machine gun, energy cannon, and energy blast; these arrive in care packages that the player must find and pick up.

5. Visual feedback about the player's status and actions.

- There is a HUD showing the player's health, energy level, weapon inventory, and number of artifacts collected.
- There are on-screen effects which appear when the player is injured.
- Artifacts have sparkling particles when they have not been scanned yet.
- When scanning, a scanning animation happens which lets the player know that he is scanning.
- Geysers have a particle system that rise and fall which informs the player when the geyser's effect is activated or deactivated.

6. Challenges that reward planning as well as reflexes.

- The player must collect a variety of artifacts, each one is in a harder location than the previous (if the player is following the natural path).
- The artifacts are protected by many aliens, so the player has to defend himself and plan paths to the next artifacts.
- There are environmental hazards which the player must avoid; the acid hazard deals damage to the player, while the mud slows them down (allowing enemies to catch-up).

Beyond the Minimum

- The terrain has two types of collision detection; the first is more advanced, using collision detection between a line segment and a triangle (plane).
- We added an in-depth GUI to our game using the ImGui library; it has many different menus and a HUD with custom elements (with textures) and text.
- Gravity affects almost all non-static objects in our game: the hovertank, care package, parabolic projectile (shot from the energy cannon), and all enemies.
- The hovertank has a physics-based movement system by applying forces in different directions, allowing us to push the hovertank into the air with the geyser hazard - this gives a more realistic movement.
- Our game uses a variety of custom-made meshes ad textures that help immerse the player into the game.
- Our game has two simple AI for the enemies, one that chases the player dealing melee damage and the other detects and shoots projectiles at the player from a distance.

Postmortem

1. Successes

- Creating a game mechanic, and having it work properly. (collisions, projectiles, gravity, cameras movement, upgrade system etc.)
- Create well designed meshes and assets for the game
- Creating a wide variety of interactable/dynamic objects in the world (artifacts, enemies, care package, tank, hazards)
- Well organized class structure
- Game is decently optimized - we aren't constantly sending unneeded data to the GPU. Some data is sent once on initialization.

2. Hardships

- Calculating the terrain normals correctly, and have them be as accurate as possible
- Working with other's premade code. Difficult to understand their intentions
- Trying to work on project with time constraints, had to cut a few good ideas to save on time
- Finding more ways to improve game performance

3. Contributions

- The work was spread out evenly. Meetings were held each week to assign work, and everyone always got their work done on time. Would often help one another with work when running into problems. Communication of the group was clear and concise.

4. Advice to future students

- Hold weekly meetings for group, and keep track of each others progress.
- Have a clear, detailed schedule.
- Have an easy fast way to communicate with others. (we used discord)
- Highly recommended to use git hub to share code if possible, VS has built in functions to work with git (at least the latest versions do)

5. If we had more time

- Implementing Shadows + normal mapping
- Focusing more on optimizing game performance
- Adding more enemy types and smarter AI
- Adding a more in-depth energy system (eg. mining energy crystals)
- Adding sound effects
- Balancing game object variables (eg. movement, health, projectiles, etc.)