Comp 3501 Final Project Report

Group members:

- Avery Ebeling
- Usman Saleim
- Dane Borgford
- Parsa Tafazoli

S1. Introduction. What is the main idea of the game? Describe the setting, the player's goals, and any gameplay you implemented.

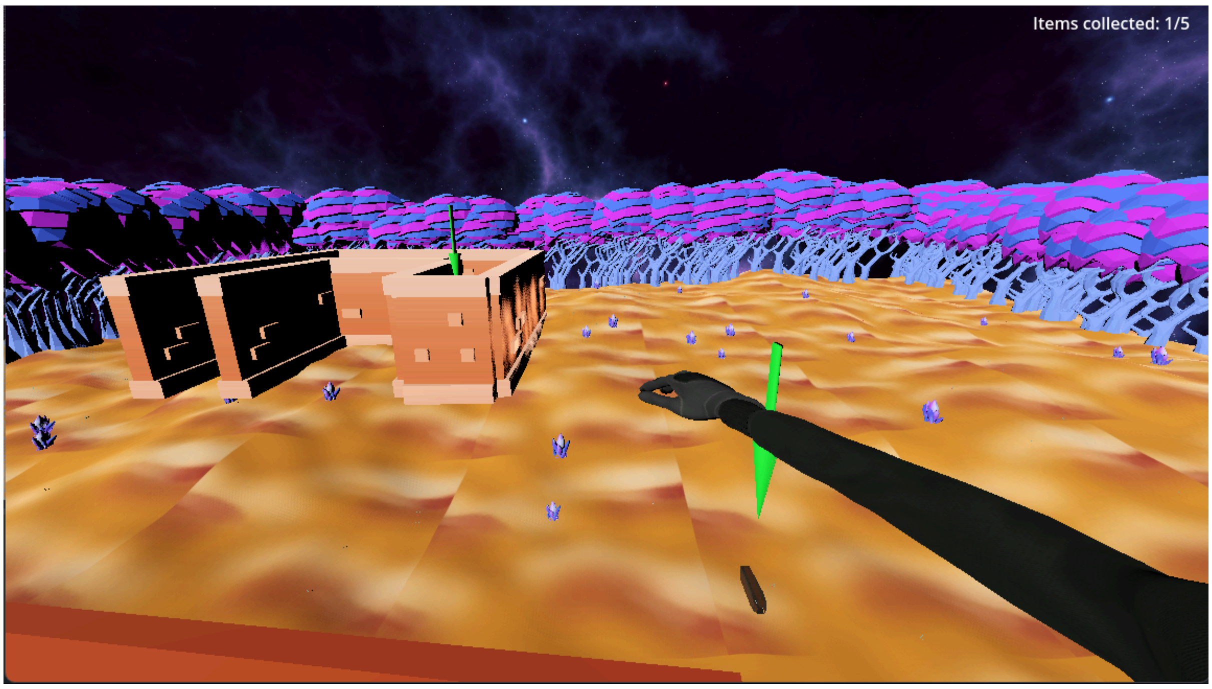
Answer: The main idea of Desolate Space is that you're a space droid stranded on an unknown planet. Your task is to navigate the world and locate the missing components of your spaceship which have scattered across the landscape.

S2. Walkthrough. Explain the controls and describe a semi-completionist approach to finishing the game. Provide solutions to all puzzles (put these in a special spoiler subsection). Include screenshots of interesting phenomena, actions, or vantages; try to show the most impressive aspects of your game. Include a map. List all actions, interactions, enemies, treasures, and other noteworthy elements.

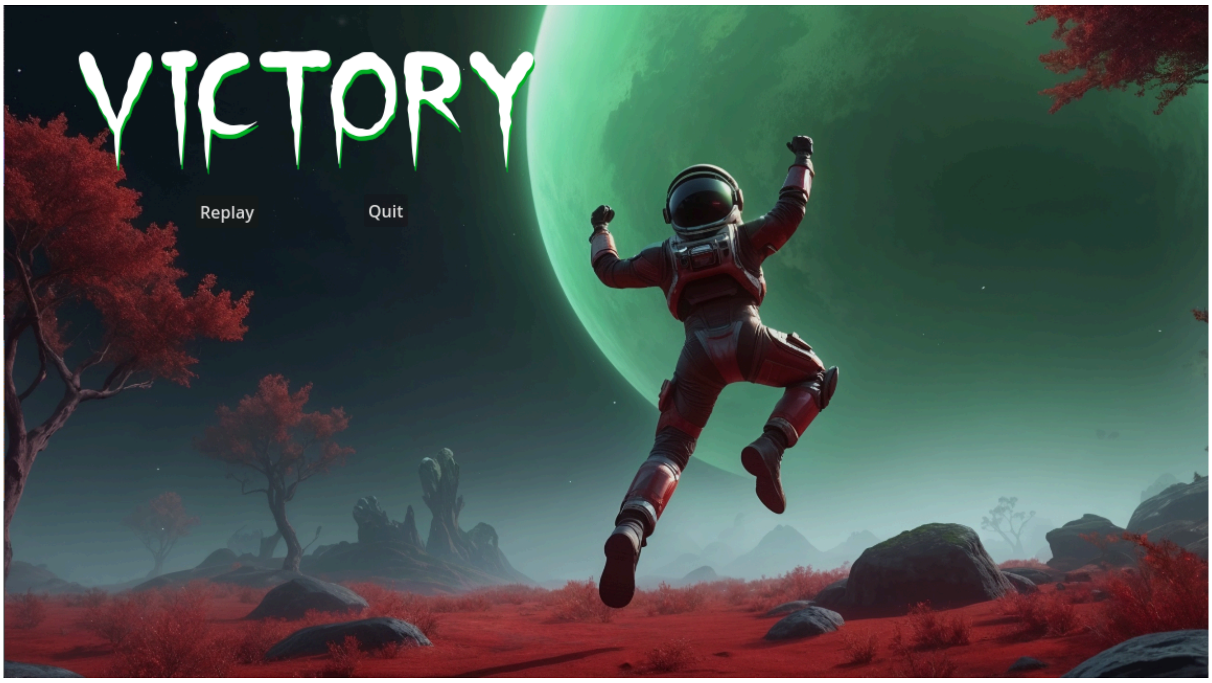
Answer: The game's controls are relatively simplistic:

- The player controls the camera with their mouse
- W,a,s,d controls forward, left, backward, and right movement
- Pressing the spacebar will cause the player to jump
- Holding shift while moving will speed up the player, allowing them to gain more momentum to make farther jumps
- Pressing p will pause the game
- Pressing the escape button will release the players mouse while in the game, with pressing it a second time relocking in the mouse

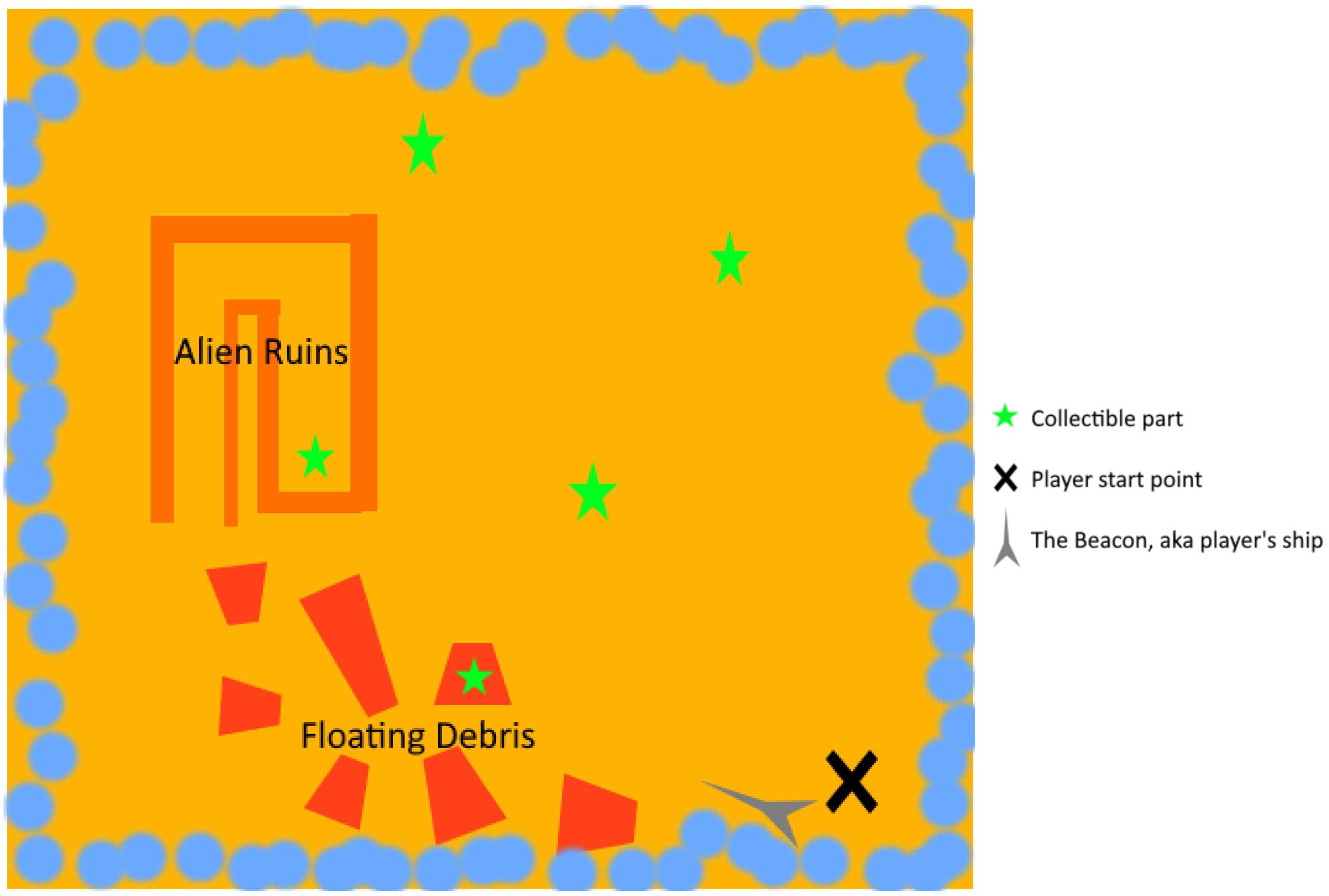
The goal of the game is for the player to collect all 5 of their missing ship parts, which are marked with arrows floating above them on the map. To collect them the player must simply walk through them. While most of the objects are randomly placed, 2 of the objects will always be found in static locations. One in the ruins of an alien building, and another at the top of a collection of floating debris caused by the lower gravity. The player must sprint jump to each platform to reach the top. From there, they have a vantage point where they can see the whole map:



Once all items have been collected, the game ends with a victory screen, where the player can choose to either close the game, or replay it. If they choose to replay, the game will reset with new positions for the ship parts, to make each playthrough a little different.


VICTORY
Replay
Quit

Level map:


Alien Ruins
Floating Debris
Collectible part
Player start point
The Beacon, aka player's ship

S3. Overall assessment. Provide a summary of your game's level of completion. Did you finish all required elements? What are the main aspects in which you went beyond the minimum? (Don't give a lot of detail here -- you will provide a complete discussion in S5.) List any bugs and incomplete portions of the game here.

Answer: In terms of completion, we believe we have met all of the technical requirements and gone beyond the minimum in some areas. The game is playable from start to finish with a good presentation. Some bugs that may be encountered include:

- Collectible objects can sometimes spawn inside the walls of the alien building, making them unobtainable.
- The ui is not scalable, i.e. the ui is in a different position when the player is in fullscreen.

S4. Technical requirements. Go through the list of technical requirements and say how your project meets each one (or does not, in the unhappy event that you did not manage to include one of these elements). Distinguish between requirements met by your GDExtension code (point to the file in which the functionality can be found) and those only met by relying on existing functionality within Godot.

Requirements report:

1. Large textured heightfield terrain with collision detection.

Completed by: Dane

Notes: Using the pseudo code provided during the lectures and what I learned in the mesh_vertex demo I created the terrain class which was a mesh that is given to terrain_instance class to create a CSGMesh3D. In the initializer collision detection is given to the terrain using the built-in set_collision_use() function, and a sand texture is applied to a StandardMaterial3D which is then applied to the mesh. The Terrain class was also given a height map (sourced from TA Sam's terrain demo) which saved the y-value terrain was at for each x,z vector.

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

Completed by: Dane

Notes: I added a TerrainInstance* object to custom_scene_3501 to either spawn in a new terrain instance or dynamically cast to an existing TerrainInstance in the scene. From there that object’s height map is used to determine environmental objects' spawning position.

3. Game environment populated by textured, illuminated objects.

Completed by: Usman Saleim, Dane

Notes: Game environment is populated using entirely commands in custom_scene_3501.cpp using the terrain height map to properly place objects on the ground. None of the objects in the environment are added using the godot editor. Some objects are spawned in specific places, and others in random places that change each playthrough. Collision is also implemented using C++ code. Lighting is done using Godot’s built-in DirectionalLight3D objects.

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

Completed by: Dane, Usman Saleim

Notes: Screen space shader that appears near the boundary of the map created by Usman, system to activate the effect when at boundary implemented by Dane. The screen-effect instance is a MeshInstance created in custom_scene_3501 which is given the “glitchy” shader and added to the scene as a hidden node under the main camera called “Vision”. The function playerOOB() is called in _process to check if the player is in the zone too close to the edge of the map. If they are, “Vision” is unhidden and the effect is shown. If not, the effect remains hidden.

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

Completed by: Usman Saleim,

Notes: Thruster particle system seen from ship at spawn implemented by Usman, used to create smoke and flame effect.

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

Completed by: Usman Saleim

Notes: Hierarchical object in the form of moving worms (originally grass) all around the environment. Implementation can be seen in all files with names starting with “grass”.

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

Completed by: Avery

Notes: I used basic velocity, pitch, and, yaw calculations, coming from the players input from the mouse and AWSD keys to rotate the camera and move and slide along surfaces, as well as a jump button with constant applied gravity, all code can be found in the **quatcamera.cpp and .h** files

7. A skybox.

Completed by: Parsa

Notes: I used a space nebula generation software to implement the skybox for the game. I feel like the bright colors and dystopic nebula add to the feel and theme of the game and create an uneasy aesthetic.

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

Completed by: Parsa, Dane

Notes: created the main menu, pausing and game over screen using AI stable diffusion image generation and various fonts and effects to create the game interface aesthetics. Wrote scene scripts for start, main, and victory screens in Godot that take the input of buttons to do things such as start the game, replay the game, or close the game.

S5. Beyond the minimum. Describe in what respects your project exceeds the minimum requirements. What are you especially happy with? What aspects of the game did you spend an inordinate amount of effort on? Identify specific subsystems you wrote (e.g., an inventory system) to support your gameplay. Try to keep your list short and focus on the major differences between your game and others. (Probably everyone thought about what colors to use -- do not put "created an expressive color palette" on your list, for example.) For each list item, describe why it qualifies as "beyond the minimum" and say why you thought it was an important element to spend time on.

Answer: In terms of going “beyond the minimum”, our team was quite happy with:

- The level design. The parkour section of our game involved making sure the movement system was dynamic enough to both allow the player to complete the section, while also giving enough of a challenge. Getting the player to learn how sprint jumping allows them to reach higher and go farther through the environment’s architecture is something we are very pleased with.

- The replayability. Crafting the interface to allow the player to play through the game repeatedly and with variety took some time to get right, however it was totally worth it in the end.
- The music and sound effects. It creates an atmosphere of otherworldliness which also helps with player immersion.
- Player immersion. By giving the player a robotic arm, and having their view glitch like a corrupted screen upon going too far outside the bounds of the map, we immerse the player in who they are playing. I.e. a space robot trying to get their ship back together. Having a system that checks if the player is out of bounds and modifies their view accordingly took a long time to correctly establish, however the final product speaks for itself.

S6. Post-mortem. Discuss your project relative to your hopes and initial proposal. What more would you like to do, if you had more time? What ideas did you have that turned out not to be feasible? What ideas worked extremely well? Was there any content you cut that you wish you could have finished? Which aspect of the project are you most proud of? Close the section by providing some advice that you wish you could have heard at the start of the term, or that you think other groups (next year's class, say) might benefit from hearing.

Answer: In the end, we were all quite happy with hope the project turned out. If we had more time, we would’ve liked to add more puzzles and complicated adversities that the player would have to overcome. We had an initial idea to create alien’s which the player would have to fight/avoid or an oxygen system which limited the player’s time and added a sense of urgency. The overall aesthetic and feel for the game worked out extremely well, the uneasy sense of exploring an unknown world ended up coming together quite nicely. We didn’t have to cut out any part of the project, however there were a few polish and puzzle elements we didn’t have time to complete. We’re certainly most proud of the overall feel and aesthetic of the game. The advice for future groups would be to leave plenty of time for polish, as completing the core game mechanics with more time left allows for cleaner polish and bug fixes which really add to the quality of the game.

S7. Credits. Say who worked on what; not much detail is needed, just 1-2 sentences per group member saying what their main contribution was. Also use this section to give details about any third-party libraries, code, assets, tools, or other resources you used in your game.

Parsa:

My main focus on this project were the aesthetics and artistic designs for the game. I implemented most of the UI elements including the main menu, pause, and game over screens. I also implemented the nebula skybox which completes the outer space feel of the game. I also rigged and animated the player arm which adds some juice to the player feel of the game.

Avery:

The first thing I implemented was the camera and movement, it uses the players input to calculate the pitch, yaw, and velocity, and uses built in functions to apply it to the player. The player may also sprint with left SHIFT to increase their speed, as well as press ESC to activate their mouse. Secondly I implemented the code to handle a scalable number of particle systems, as well as implemented my own particle system. The fireflies fly around the map and can be easily customized by their uniforms. Lastly I implemented all of the audio design and functionality. There is music in the main menu, as well as in the game itself, I have also added footstep/jumping audio when the player moves/jumps

Usman:

One of the things I did was populate the game environment. I added trees, crystals, worms (with hierarchical motion), a parkour section with platforms, as well as a broken building with walls to guide the player through. I also created the arrows that point towards each collectible to guide players towards them and implemented their turning into check marks upon collecting the corresponding collectible. While populating the environment, I also created custom 3d models for the platforms and ship, and converted every 3d model used in filling the terrain into a single mesh file to be instantiated via only the code. Furthermore, I drew the textures used for the walls, platforms, trees, sand on the terrain, and the ship. Besides that, I also designed the particle effect of the thruster on the ship (visible from spawn), as well as the screen space effect that activates when the player approaches the edges of the map. I also created the hierarchical moving worms (as mentioned earlier), that spawn in random positions across the map.

Dane:

I wrote the code for creating and adding the procedurally generated collision-detecting terrain and its heightmap to the main scene, the collectible objects, their collision detection and the system that counts how many have been collected and deletes the items as they are gotten, the win condition in c++ and replayability feature using Godot’s built-in scene script feature, rewrote the ui to match the # of objects the player had collected, all of the out of bounds restrictions including implementing the screen-shader effect (effect itself by Usman), fixed issues with the performance and placing of the hierarchical object (worms/grass), and helped with using the heightmap to populate the environment.

Asset credits:

1. floor texture: "Designed by valadzionak_volha / Freepik"
2. 3D models: "https://www.kenney.nl/assets/graveyard-kit"
3. Arm model: https://sketchfab.com/3d-models/hand-and-arm-for-first-person-perspective-5f1beb235d0c409eb4f1ea0059ee8a91#download
4. Skybox: https://sourceforge.net/projects/spacescape/
5. Ship parts: https://www.fab.com/listings/f3c6b13c-e8b1-4f0b-bb92-325bffc91e3a
6. Trees: https://www.fab.com/listings/053c6fcb-e8cf-403d-b596-b30635a574db
7. Crystals: https://www.fab.com/listings/f1c0d0e0-d64f-4236-af36-b3ab9d74875f
8. Arrows: https://www.fab.com/listings/63f23e15-c63c-47f8-8cd3-95179557c3cc
9. Checkmark: https://www.fab.com/listings/1dac4862-a690-4fb6-900b-81f358b5f81c
10. Audio:zapsplat.com