Shards of Roh

COMP 4501

Nick Martino 100934881

Alex Shulman 100963354

Christina Abramson 100943361

Contents

Controls and Objectives

Reference page that has all the controls and goals together in one area.

Camera:

- Move position: move the mouse to the edge of the screen or use the WASD keys.
- Rotate: CTRL + Right click and drag.
- Zoom: mouse scroll wheel

Mouse:

- Left click and drag to select units
- Right click to give selected units' new target [destination, attack, build, etc.]
- Left select item from Action Panel
- Right click to activate selected action (from above, action will then deselect)

Key Controls:

- Space: Skips cut scenes
- Escape: Exit application

Hot Keys:

- 1: Triggers the first action in the action panel*
- 2: Triggers the second action in the action panel*
- 3: Triggers the third action in the action panel*
- 4: Triggers the forth action in the action panel*

  * If the action doesn't exist in the action panel the key it does nothing

Cheats:

- 0: Adds 1000 of each resource (not including dynamite)
- 9: Player gains control of wild rhinos on the map
- 8: Changes the texture of rhinos on the map to a texture with glowing blue patterns
- 7: Changes rhinos on the map into dragons

Win/Lose Conditions: Short version

Elven Campaign Map: Get elf commander to the area labelled 'safety'. If the elven commander dies, you lose the map.

Dwarven Campaign Map: Defeat all the invaders. If your town centre dies, you lose.

Orc Campaign Map: Destroy the Tomb of Kings (no lose condition other than giving up and quitting). In the rare case all your units are killed, and you run out of resources, use the cheat that gives you extra resources to keep going.

Overview

Shards of Roh (SOR) is a Real Time Strategy game where you play as one of three races, the spawns of the Great Dragon Roh. In SOR, you face off against other races, building structures and units to fight. You play through the story of the Great Devastation, uncovering the lore as you experience all three campaign maps.

Content

Races

There are 3 races. Each race has 5 units and 4 buildings.

Elves and Undead – The firstborn of Roh, the Elves are wielders of an ancient strength. Low in population and physical prowess, they must outsmart their foes with use of powerful magic.

The Elves and Undead are made unique by how the undead units they create are hostile to everyone, including themselves. The Elves must use special abilities from their Elven or Lich units to gain control of the undead they create, and can use those abilities to gain control of other undead on the map.

Dwarves – The second shard of Roh, the Dwarves are masters of construction. They use their craft to construct mobile buildings of war. Like the Elves, the Dwarves remaining in this world are few in number, but their craftiness and ingenuity allow them to forge a path of stone through their foes.

The dwarven race is made unique by having access to a fourth resource, Explosives. Dwaves can spend resources at a Dwarven Town Centre to construct Explosives, and spend the explosives to fuel the special abilities of the Golem and Cannon units. The Golem temporarily gets a boost to attack speed and movement speed, and can stack this boost multiple times, while the Cannon launches an AOE attack at the targeted location.

Orcs – During the Second Great War, Roh created the Orcs to fight the Elves and Dwarves. Strong and numerous, the Orcs rely on raw strength and speed to conquer their enemies.

The defining mechanic of the Orcs is mobility. Orcs have powerful wolf units that move much faster than anything the Elves or Dwarves control, and have the only combat-focused flying unit in the game with the Dragon. They must leverage this mobility to defeat their opponents.

Campaign

The campaign is 3 maps, each of which involves playing as a separate race with a victory condition, preset units and buildings. The maps are designed to highlight the strengths and weaknesses of the race they are focused on, while also showing their part in the lore.

Elven Campaign Map: Play as an Elven Commander during the First Great War, struggling across hostile ground to get to safety. If the elven commander dies, you lose the map.

Dwarven Campaign Map: Experience internal struggles of the Dwarven tribes as your settlement is under attack by a larger force. Defend yourself and defeat all of the invaders. If your town centre dies, you lose.

Orc Campaign Map: Play as an orcish invasion force during the Second Great War, summoned by Roh to defeat the children who turned on him. Build a force and fight through the defenses of the Dwarves and Elves to destroy the Tomb of Kings.

Units

In addition to basic functionality (moving, attacking), certain units have additional abilities. These include but are not limited to **building and gathering, area attacks**, **flight**, **range**, **buffs,** etc.

Resources

There are three core resources available in the game; **food**, **wood**, **crystals**. These are present in finite quantities in the world and can be collected by specialized villager units. Wood and crystals are gathered from pre-set nodes, while food is harvested after killing wild rhinos. Dwarves have **explosives** as an additional resource.

Menu

Our menu is designed as a “Map Menu”, where you click on the races to select them, then click on the start game cube in the middle once you have one selected. You can click onto another race to swap which you have selected. They are shown as intractable by lifting into the air when you mouse over them.

Music, Cut Scenes, and Voice Acting

Each campaign map has its own voice acted cut scene, giving you a full view of the map while explaining the lore surrounding it. The menu also has a cut scene, without voice acting. The menu and each map has its own music selected to match the themes of that map. Cut scenes can be skipped with the space button.

Architecture

Container Model (Adaption of the MVC pattern)

There is a model used three times in the code, which will be termed the 'Container Model'. Two classes are involved in the Container Model: the Container and the Object. The Container is a MonoBehaviour and deals with all aspects related to input/output, including display, location, and controls. The Container has an instance of the Object in it. The Object has all the data and actions but is completely unaware of how it is being used. This abstracts the data and allows us to freely change the visuals without impacting the backend. It uses a call model (where the Container periodically checks back in with the Object to see if anything has changed) to keep the Container in sync with the Object.

The three locations this is used in the code are: Players, Buildings, and Units.

Factories

There are two factory classes used to construct things in a simple way: BuildingFactory and UnitFactory. Each of these have static functions that create new Buildings or Units (respectively), and properly initialize them for play.

Overview

There are three scenes used in the project. The first of these is the menu scene. When the menu scene is loaded up, it plays a short intro cutscene and then gives the user options to select a race to play as. Once selected, the start cube becomes active, and you can play the map associated with that race.

The programming architecture behind this is simple, each race has a RaceSelection script attached, and this manages the visuals (rises up on mouse over, camera move to when selected, etc), as well as the user input. Once a race is selected, it sends the appropriate data (race or map) to the MapManager and floats the camera to the correct position. When the start cube is selected, the MapManager sends the critical information to a MapLoader class with static variables so that it can keep the data across scenes, then sends the user to the second scene; map.

The map scene starts by reading the MapLoader, and building the terrain dynamically based on what was selected from the menu. It then creates the PlayerContainer (which makes the player), using the race (also in MapLoader). The race contains what units and buildings can be constructed by a player of that race.

As the terrain generates, it creates all the pre-set units and buildings. For each of these, a container is attached to the GameObject, and a Unit/Building of the correct type is created inside these containers.

Each unit and building type is its own class, with a wide range of adjustable variables. Once created, these variables can be adjusted on a per-unit basis. In addition, each unit and building know who it's owner is.

InputHandlers allow the user to interface with the program, and contextual clicks on BoxColliders allow handling of individual objects.

The UI is built using the built in Unity UI framework, and overlays on top and bottom of the screen. Several buttons are in the bottom right (they change function based on what you are clicked on; a villager and you get the ability to build, a unit with magic/special abilities/evolutions/upgrades and those show up there. A detail tooltip shows up on mouseover of various objects at the bottom of the screen.

Pathfinding is done by placing a 2D grid of nodes down over the whole world, then running A* over it. When a building is added, or a unit moves, the list of nodes automatically updates to reflect the new locations and paths change to support this.

There is also basic AI, units find nearby objects to interact with by drawing a circle around them, then running a heuristic algorithm to determine priority (ex. A villager has an enemy and a damaged friendly building in range, it will go to repair the building over fighting the enemy, while a non-villager would make the opposite decision).

Critical Classes

Certain classes require more a specific section to fully understand the architecture.

Player

Player knows what its name is, whether it is a user (instead of a computer), what race it is, what resources it has, and what it's current target is (building, unit, or ability).

PlayerContainer

PlayerContainer handles updating the visuals whenever the players' resources change, or a new building/unit/ability is selected or targeted.

Building

Buildings have a large list of variables that control different gameplay aspects. A building can have a name, max/cur health, what units it can produce, how much it costs to make the building, and it's defensive stats. The building can get hit (and calculate the damage it takes), and generate a unit it is creating.

BuildingContainer

This handles the input for a building; selecting it on right click if it is owned by the player, attacking it on right click if it is not, repairing if it is owned by the player and right clicked, etc. It also functions as the entry point to the Building object it represents for other objects (such as enemies trying to hit it).

Barracks/Grave/Etc

The subclasses of building set the variables stored in building.

Unit/UnitContainer/Solider/Etc

These function similarly to the Building variants, but they have different variables and responses to selection. While their structure is the same, the content of the classes is based on what a unit has and needs to do, as opposed to a building.

Grid/Node

The grid and node classes are responsible for pathfinding, generating the NodeGrid and finding units a path through it. They also determine whether buildings can be placed at various locations based on the grid.

Behaviours/Abilities

Behaviours and Abilities are classes that Units contain a list of, and can be added or removed to modify the unit during play. Behaviours designate AI behaviour (more in requirements), and abilities are special actions that various units are able to do, such as spawning a wild skeleton, or toggling flight, or launching an AOE attack.

Interactions

Units

Units interact the most with other objects. Villager units can collect resources, build buildings, and attack enemy units and buildings. Some more complex interactions are determined by the types of units and what side they are one.

There are three sides: the player's side, the enemy's side, and the wild (or neutral) side. Wild units, will not attack other wild units, nor will they attack buildings (of any kind). They will, however, attack enemy or player units. Player units can attack wild/neutral or enemy units and buildings, neutral units attack player/enemy units and not buildings, and enemies will attack player buildings and player/neutral units.

Different units have different attack ranges or might have special abilities. Melee units can only attack close range; this means that ground level melee units cannot hit flying units, and melee flying units cannot hit ground units because they are simply out of range. Ranged units obviously aren't limited in the same way, so a flying ranged unit can attack a ground melee unit without fear of being hit back, and a ground ranged unit can attack a flying melee without worrying about being hit back either.

- Example: The dragons (Orc race) are melee units who can fly. When they are flying they can only attack other flying units and can only be attacked by flying units or ranged units. When they are on the ground, they can only attack other ground units, and can only be attacked by other ground units or ranged units. Dragons cannot hit buildings while flying.
- Example: Elf wizards (Elf race) are ranged flying units. They can attack anyone regardless of if they are flying or not but can only be hit by flying or ranged units.
- Example: Cannons (Dwarf race) are ranged ground units. They can attack anyone regardless of if they are flying or not but can only be hit by ground or ranged units.

Elf commanders, elf wizards, and liches also have a special interaction with wild undead units, in that they can take control of the undead units (change the undead unit from a wild unit to a player unit). Liches also have the special ability to spawn more skeletons.

Buildings

Buildings are static for the most part. Villager type units can interact with them by attacking or building them, but the buildings do not directly interact with anything. They do however draw the attention of enemies on the dwarven campaign map, making the enemy units leave their path to the town center, which buys the player some time. They are mostly used to spawn units or to act as obstacles to units' movements and paths.

Resources

Resources, for the most part, need to be harvested by villager type units. The exception is Dynamite, which can be bought with other resources through the town center building. Crystals and Trees can be harvested directly, so only villagers can interact with them and they do not interact with anything (they are static). Rhinos, on the other hand, are dynamic and will wander or attack nearby units (that aren't neutral). They interact with other units by attacking them, and any kind of unit can attack them back. When Rhinos die, they leave behind a corpse which behaves the same way as Crystals and Trees.

Requirements

Implemented Requirements

Prototype 1

When entering any of the campaign maps, you can see we have:

- 3D rendering based on an isometric view, using appropriate illumination, and incorporating shadows
- 3D models for the terrain and units, with appropriate textures and material definitions assigned to them
- A few instances of each type of unit already created when the game starts

Collision Detection – All units, buildings, and resources have box colliders which prevent them from falling through the terrain. It also stops units from being able to walk through other units, buildings, and resources. The only exception is that the player's units can walk through each other. We decided to allow this exception to improve gameplay, because otherwise, if a unit is too far away from its target, it would be unable to perform its action (attacking, building, gathering, etc.)

Actions and Interactions Between Units – ***Refer to Interactions.

Camera Movement – The camera can be moved (translated) by moving the mouse to the edge of the screen or using the WASD keys. You can rotate the camera by holding the CTRL key and moving the mouse with the right button pressed. The camera can then be zoomed using the scroll wheel.

Prototype 2

Animation – After setting or determining a destination (via AI), units will travel using linear interpolation instead of teleporting.

Character Animation – Each unit, except the Cannon, is animated with at least 3 animation states (attack, walk, idle). Some, such as the Golem and Dragon, have more complex animations, with the Dragon reaching 6 (idle, idleFly, walk, land, attack1, attack2). All attacks are correctly synced to attack animations - when you visually see the unit strike will be the same time that the unit deals damage to it's target.

Physics and Steering Behaviours – We have seek and wander behaviours when units are within a certain range of targets (enemy units/buildings and resources). When units are selected, one of them is assigned as a leader. The other units follow the leader and position themselves around the leader using a formation algorithm so that they do not collide with each other (this algorithm doesn't trigger when the units are directed to attack or build something; the reasoning is the similar to why we turned collision detection off between the player's units).

Final Submission

Pathfinding – Units move around the map based on an A* Pathfinding algorithm from a node grid that we constructed. If you attempt an impossible path, the unit will instead stay still. Areas that are too steep, underwater, or blocked by a building or resource node are marked as not walkable tiles. When a building or resource node is destroyed, that area gets marked as walkable again.

Semi-Automatic Actions and AI – Our game implemented the sense-think-act structure through the Behaviour class. Behaviours represent the think-act section of the structure, and can be dynamically removed or applied to units based on stimuli. Sensing is based on the unit's sight radius, and its ability to respond to things such as being attacked. The Behaviour will then use that information, and information that it is tracking such as whether the unit is moving, to determine if it will enact. For instance, the Behaviour "IdleAttack" will, if the unit is not moving or attacking, scan a region around equal to its sight radius for an enemy to fight. If it sees one, it will direct the unit to move to and attack it. The behaviour "PatrolAttack" is similar, except it is designed to supersede a unit's movement - a unit with patrol attack that is currently moving will divert that movement to attack something it sees. Autonomous unit actions are developed the same way, where villagers will autogather or autobuild. For example, when a unit is instructed to gather resources, it gains the IdleGather behaviour, which senses nearby resources to continue gathering, prioritizing ones of the same type as the original instruction. Giving that unit any other command removes IdleGather, so the unit will once again listen to your commands. IdleBuild works the exact same way, except sensing unbuilt buildings instead of resources.

Meaningful Play – Each campaign map has its own win and lose conditions. * Refer to Content, Campaign.

Feedback to Player

Main UI:

- **Resource Bar**: At the top of the screen, the player can see the resources they have collected, so they can check if they have enough to purchase new buildings or units.
- **Action Panel**: In the bottom right of the screen there is the Action Panel. The Action Panel has displays different buttons depending on what the player is interacting with: if the player clicked a built building it will display the units the building can spawn and if the player has units selected it will display the abilities that the leader unit of the group can use. If the leader is a village, it will show the buildings that can be built. Clicking a button and then clicking on the map will cause the action to trigger.
- **Tool Tip**: There is also a Tool Tip along the bottom of the screen which is triggered when the player hovers their mouse over a unit, building, resource, or the Action Panel. When hovering over a unit, it will show the unit's name (their type), their current health, the unit's attack type, and a small description for the unit. When hovering over a building, it will show the building's name (their type), their current health, and a small description of what the building does. Finally, when hovering over the different buttons of the action panel, if the buttons are displaying a unit's abilities, it has a small description of what the abilities do. If the buttons are displaying buildings to be built, it will display the building's usual tooltip, but also includes the building's cost. Similarly, if the buttons are displaying units to be spawned, it will display the unit's usual tooltip, but also includes the unit's cost.

Health bars: All objects have health bars. User controlled objects have green health bars, enemy controlled objects have red health bars, and neutral (or "wild") units have white health bars. **When using the elves' ability to gain control of the undead, the user can tell they successfully gained control when the undead's health bar changes from white to green.**

Mana bars: Some units have abilities that require mana, these units have a blue bar under their health bar. This bar is only visible for player controlled units; enemy units that have mana have their mana bar hidden from the player.

End Screen: There is a separate scene which hold the end screen. This clearly displays to the player if they have won or lost the game and also allows the player to return to the main menu.

Additional Features – *****Refer to Additional Notes, Additional Features.

Missing Requirements

Prototype 2

Animation – Ease-in and ease-out was removed because we believe it interfered with gameplay.

Physics and Steering Behaviours – Using forces and flocking was not included in our final project because it didn't work well with the movement system and formation algorithm we already had implemented. It also made the animations behave strangely.

Additional Notes

Additional Features

- **3D Main Menu**: Interactable menu showing off the 3 races.
- **Cut Scenes**: Fully voiced, with dynamic camera motions and music.
- **Campaigns**: Three full campaign maps with different win and lose conditions.
- **Extra Units**: We included many more units than were required for the project.
- **End Screen and Play Again**: Which returns the player back to the main menu so they can start a new game without needed to exit and open the game again. (Values reset properly upon starting a new game).
- **Cheats:** Added for entertainment value or for testing purposes.

Difficulties

Version Control, File Size, and Metadata: One of our big issues was file size. We have so many models that our external assets folder became too large to upload to BitBucket. We ended up resorting to passing the Resources and External Resources folders around on a USB stick. Resource sharing outside of version control would then sometimes lead to issues with metadata. If the metadata wasn't also transferred, Unity would sometimes think that it was a new/different asset and create a new metadata for it, but because the new metadata is different from the one Unity expects to be linked up in the scene, it would say the resource it was looking for would be missing. This made it difficult to stay in sync with our scenes sometimes. We also often lost our layer data, which lead to a lot of issues with our old main menu because our sprites would be out of order and look weird or be hidden behind other sprites.

References

Models

Unity Asset Store

- 3DForge
  - Blacksmith
- Alex McDonnell
  - Construction Site Pack
- AndreiCG
  - Church 3D
- Beatheart Creative Studio
  - Round Tower
- Craft Studios
  - Humans – Watchtower
  - Orc – Watchtower
- Dungeon Mason
  - Mini Legion Rock Golem Handpainted
  - Mini Legion Grunt Handpainted
  - Mini Legion Lich Handpainted
- Ferolit
  - Hand Painted Forge
- Hannes Delbeke
  - 16th Century Cannon
- HONETi
  - 3 Free Characters
- LowlyPoly
  - Stylized Crystal
- Maksim Bugrimov
  - Fantasy Rhino
- Mariam Sarahvili
  - Post Box
- Polygon Blacksmith
  - Toon RTS Units – Demo
  - Toon RTS Units – Orcs Demo
  - Toon Soldiers – WW2 Edition
- PolyNext
  - Necromancer
- Yulya
  - Lowpoly tombstone

Music

https://opengameart.org/content/epic-fantasy-music