GAME DESIGN REPORT
APPENDICES

Hermann Ingi Ragnarsson
Jón Böðvarsson
Örn Orri Ólafsson
APPENDIX A - ASSETS & SOUNDS

3D Assets

Following are the main assets we used in the game. Most of them are acquired from the Unity Asset Store either for free or bought. A couple of them are made by our selves. The caption under each image will clearly state whether it was acquired from Unity Asset Store or made by our selves. All 2D assets were created by ourselves (minimap icons, heads up display assets, powerup icons, etc) so they will not be accounted for here.

Maze Element Ice Golem, acquired from the Unity Asset Store:

Slime level 1, acquired from the Unity Asset Store:
Battle Dragon, acquired from the Unity Asset Store.
https://www.assetstore.unity3d.com/en/#!/content/63644

Low Poly Troll, acquired from the Unity Asset Store.
https://www.assetstore.unity3d.com/en/#!/content/56744
The wonders were created by assembling assets that were acquired from the Unity Asset Store. The blue wonder was created from assets that belong to the Nature Extended asset pack (https://www.assetstore.unity3d.com/en/#/content/66146) and the red throne was assembled from assets belonging to the Low Poly Rock Pack (https://www.assetstore.unity3d.com/en/#/content/70164).

We created the cannons from scratch in Blender except for the rocks under the red cannon, which were acquired from the asset pack Low Poly Rocks the Unity Asset Store (https://www.assetstore.unity3d.com/en/#/content/70164).
We created the powerups from scratch in Blender.
Decorative weapons acquired from the Unity Asset Store pack Low Poly Weapons:
https://www.assetstore.unity3d.com/en/#/content/71680

Low Poly Cliff Pack from the Unity Asset Store and used unaltered:
https://www.assetstore.unity3d.com/en/#/content/67289
These are the assets we used from the Low Poly Rocks Pack(https://www.assetstore.unity3d.com/en/#!/content/70164) acquired from the Unity Asset Store except for those used for the red wonder asset.

These are all the assets we used from the Low Poly: Free Pack acquired from the Unity Asset Store: https://www.assetstore.unity3d.com/en/#!/content/58821
These are all the assets we used from the Nature Pack Extended (https://www.assetstore.unity3d.com/en/#!/content/66146) acquired from the Unity Asset Store except for those used in the blue wonder asset.

**Sounds**

These are the sounds that we used in the game and from where they were acquired:

**Footsteps on crunchy gravel:**  
https://www.freesound.org/people/botha9johann/sounds/326056/

**foley_footsteps_desert_boots_sand.wav:**  
https://www.freesound.org/people/pan14/sounds/388289/

**FootstepsGravel.wav:**  
https://www.freesound.org/people/mikaelfernstrom/sounds/68687/

**Fire Ball Release:**  
https://www.freesound.org/people/CTCollab/sounds/223611/

**Cavernous Drone:**  
https://www.freesound.org/people/gluelsobar/sounds/195137/

**aWindchimes113014LincolnNe24Degrees75Hi.mp3:**  
https://www.freesound.org/people/bmccoy2/sounds/256152/

**Ambience, Wind Chimes.wav:**  
https://www.freesound.org/people/InspectorJ/sounds/353501/

**Orch. BGM**  
https://www.assetstore.unity3d.com/en/#!/content/52167
APPENDIX B - CODE

3rd Party Scripts/Code

We used a few extensions in our project most notably we bought an implementation of a fog of war system by BrendanL.K that we wrote 4 scripts to interact with. We also used the Network Lobby functionality, Unity Test Tools for Unit Tests and Text Mesh Pro all provided by Unity Technologies.

Fog Of War https://www.assetstore.unity3d.com/en/#!/content/51344
Network Lobby: https://www.assetstore.unity3d.com/en/#!/content/51344
TextMesh Pro: https://www.assetstore.unity3d.com/en/#!/content/84126
Unity Test Tools: https://www.assetstore.unity3d.com/en/#!/content/13802

Our Own Scripts/Code

Following is a list of all the scripts and a breif description of the role that each script plays in the game. The list is order like the folder hierarchy in the project. The names of some of the in-game objects have changed since we started on the project, to preserve the integrity of the game and our code we have kept the old names intact. Inhabitants are referred to as creeps, cannons are referred to as towers and wonders are refered to as thrones in the script names.

COMMON

- **Billboard:** Ensures that the health bar on top of the characters faces the camera at all times.
- **Constants:** Constants that are used in more than one script are stored here for easy access.
- **CustomFogofWar:** A custom script we wrote that interacts with the Fog of War system in a way that suits us.
- **Projectile:** An abstract class from which other projectile classes inherit.
- **DummyProjectile:** A projectile that contains no function other than visual representation on the client. The actual
- **Events:** Keeps track of the numerous events occuring in the code.
- **Health:** an abstract class from which the other health classes derive.
- **HideeffectsInFogofWar:** Hides particle effects once they enter fog of war.
- **HideHealthBarInFogOfWar:** Hides the healthbar once it enters fog of war. Needs a separate script because Canvas Renderer that displays the health bar does not inherit from Renderer.
- **Stats:** A convenient way to set and retrieve stats on characters and structures.
- **SuperArray:** An abstract datatype used to keep track of numerous elements used in the game.
- **Team Setup**: Added to each gameobject that belongs to either team (titan, cannon and wonder).
- **Timer**: A class that stores the game time. Extracted functionality to allow for unit tests.

**Creeps**
- **CreepCamp**: Manages a inhabitants camp, has designated spawn points for the inhabitants and is responsible for spawning the creeps on the coordinateds designated by the spawn points.
- **CreepHealth**: Keeps track of the inhabitant’s health and lets the GameManager script know if a inhabitants has been killed.

**GUI**
- **HUDManager**: Implements all functionality of the heads up display.
- **SelectionIndicatorManager**: Controls the selection indicator (white circle around the selected gameobject).

**Interactions**
- **Right Click Action**: Manages the event when the right mouse button is clicked. Determines which action is taken depending on whether the Target Manager returns an object or a location in the game world.
- **Interactive**: Adding this script to a gameObject makes it interactive, that is it can be selected and and right clicked on.
- **PowerUpAction**: Added to the titan gameobject to allow him to use powerups.
- **Highlight**: Implements the functionality of selecting or deselecting interactive gameobjects.

**Managers**
- **CameraManager**: Makes the camera follow an object on the map and enables the camera to orbit objects.
- **CreepCampManager**: Manages the creep camps on the map, is responsible for making every creep camp spawn creeps at the same time.
- **CreepManager**: An AI controller for a single creep instance.
- **DefaultCameraCenterManager**: Enables camera movement controlled with the mouse and is the default object that the camera manager makes the camera follow.
- **GameManager**: Stores general information about the game, shows the countdown screen and prepares the game when the scene is loading. It also keeps the score for the titans’ kills and deaths, hands of information to the XPManager and ends the game when one player has won.
- **MinmapManager**: Sets the size of the minimap and determines whether a mouse click occurred within the minimap.
- **Mouse Manager**: Manages the left click interactions from the user and settings for the mouse.
• **TargetManager:** When the user clicks either of the mouse buttons on the game world this script is
returns the location of the map or the object that was clicked.

• **TimeManager:** Keeps track of the passing of time, stores the time in the Timer.

• **TowerManager:** Controller that makes the cannon attack the opposing titan if he walks into its
range.

• **TutorialManager:** Implements the functionality in the tutorial scene.

• **XPManager:** Implements the experience points system and updates levels and experience gained by
calling TitanXP.

**MENU**

• **AddCanvas:** Spawns the in-game menu into the game.

• **EnableMenuOnEsc:** Brings up the ingame menu when the escape key is pressed.

• **LoadSceneOnClick:** Convenient way to load a scene a button is pressed.

• **QuitOnClick:** Enables the player to exit the game from the main menu.

• **ReturnToLobby:** Returns the player to the lobby scene from the game scene.

**POWERUPS**

• **InvisibilityPowerUp:** The invisibility powerup.

• **PickUpOnTrigger:** Enables the titan to pick up powerups.

• **PowerUp:** Abstract class that all powerup classes inherit from.

• **PowerUpManager:** Manages the powerup system.

• **PowerUpSpawn:** Spawns the powerups.

• **SpeedBoostPower:** The speed boost powerup.

• **ThroneHealingEffect:** Implements the healing buff and visual effect that occurs when a titan is near
his wonder.

**STRUCTURES**

• **ThroneHealth:** Keeps track of the wonder’s health and instructs the GameManager to end the game
indicating which player won.

• **TowerHealth:** Keeps track of the cannons’ health and destroys them when their health goes below
zero.

**TEST**

Unit tests are located in this folder
TITANS

- **Titan Setup:** synchronizes certain values across the network such as their appearance and player number.
- **TitanHealth:** Manages the titans health and implements the damage system.
- **TitanLobby Hook:** Moves information about players from the multiplayer lobby and into the gamescene.
- **TitanInfo:** The GameManager stores information about each titan with an instance of this class.
- **TitanXP:** A script that keeps track of the XP and level of each titan.