BATTLE LANE

by
Steven Mare MRXSTE008
Thethela Faltien FLTTHE004

1. Requirements Analysis and Use Case Diagrams

- ## Functional Requirements

  Battle Lane is a top down, 3D, Battle Arena game in the style of popular MOBAs such as *League of Legends* and *DotA* but simplified to be 1v1.
  Each player controls a ‘Hero’ who is more powerful and tougher than the AI controlled ‘Minions’. The object for each player is to destroy the enemy's tower with the aid of periodically spawning minions, while preventing the enemy Hero and Minions from destroying their tower.
  When Hero dies (reach 0 life), gold bars are dropped and the hero is spawned near his/her tower after a short delay.
  When a hero kills enemy minions or the enemy hero they drop gold which the hero can collect. This gold is used to purchase more powerful minions to fight for the player. The player can spawn these powerful minions near his/her location.
  Power ups spawn periodically across the map which heroes can collect for temporary advantages, such as a speed boost.

- ## Non-functional Requirements

  The game environment is built using Unity in-built systems.
  The collision system that players use to pick up gold, power ups and interact with game objects in the game environment is built using Unity collision system.
  The Unity layer system is used for separating certain game interactions. Like only allowing minion to attack enemy minion and the enemy hero and making sure only players can pick-up powerups. Also the Unity layers system is used to make certain game objects invisible to certain cameras and used to prevent collision of game objects in different layers.
  The minions use Unity’s in built AI pathfinding system, Navmesh navigation. It allows the minions to move to a target efficiently and avoid collision with terrain features.

  A split screen is used in the game to differentiate the views of the two players with each screen taking up half of the native 1600x900 resolution. Each camera follows each player’s hero, showing a top down view of the hero, limiting the player’s view of the world.

  The game require two player that share a single keyboard thus player input is limited to a set of finite keys.

  The Unity game engine environment only supports one audio listener at a time and so all sound will be heard globally as hero centric sound is not possible with two heroes present.

- ## Usability

  Each player should be aware of various elements of the game at a time and so these elements should be clearly visible and understandable to the viewer.
  Firstly, a player should know their hero's location in the game world in relation to the terrain, towers, minions and the other hero. This is communicated through the use of a constantly updated minimap.
  A player needs to be constantly aware of their tower's current status as losing the tower ends in a loss for the player. The tower's current health is shown through a health bar on the player's Heads-Up-Display(HUD). The tower's sounds are also louder than many other game sounds so a player is made aware that the tower is attacking enemies.
  The minimap also shows the location of powerups on the map which the player can use to gain a strategic advantage.

  A player must also be aware of their own hero's health status and that of minions and the enemy hero. This is depicted through the use of health bars that appear above the head of each unit (heroes, minions and towers)..

  Each player has access to multiple abilities for their hero, namely a basic attack, special ability, and purchasing minions. Each of these is depicted on the HUD with an icon and tooltip of the corresponding key to activate it. In the case of purchasing minions, it also depicts the gold cost of purchasing. When a player uses an ability or makes a purchase, a cooldown overlay will show the time required before the function can be used again.

- ## Use case descriptions

- Starting Game

  Actor: Player/s

  The player begins on the main menu. The player selects to start a game. The system generates the map with obstacles and towers, then generates the heroes for each player at their corresponding starting position. AI minions are spawned and play begins.

  Game ends in victory for a player when the opposing tower is destroyed.

- Hero Moving

  Actor: Player/s

  Player/s are currently in game. By pressing the move forward key (default 'W') the player's hero character will move in the direction they are facing. If they press the rotate left button (default 'A') or rotate right button (default 'D), the hero will rotate in the desired direction, changing their orientation. The player can move forward and rotate at the same time.

- Hero Attacking and Spell Casting

  Actor: Player/s

  Player/s are currently in game. The player presses the attack button (default ‘Q’) to do a damage dealing effect on any enemy unit directly in front of the hero in the direction they are facing. The player can also press the spell button (default ‘E’) to do an area-of-effect ability (such as damage or heal) at the position where the hero currently is.

- Power Up Collection

  Actor: Player/s

  When the player moves their hero character over a spawned power up on the map they are prompted with a sound and visual effect that they have collected something. In the case of temporary boosts, such a speed or invulnerability, an ‘aura’ surrounds the hero so they know they are under the benefits of the corresponding powerup;

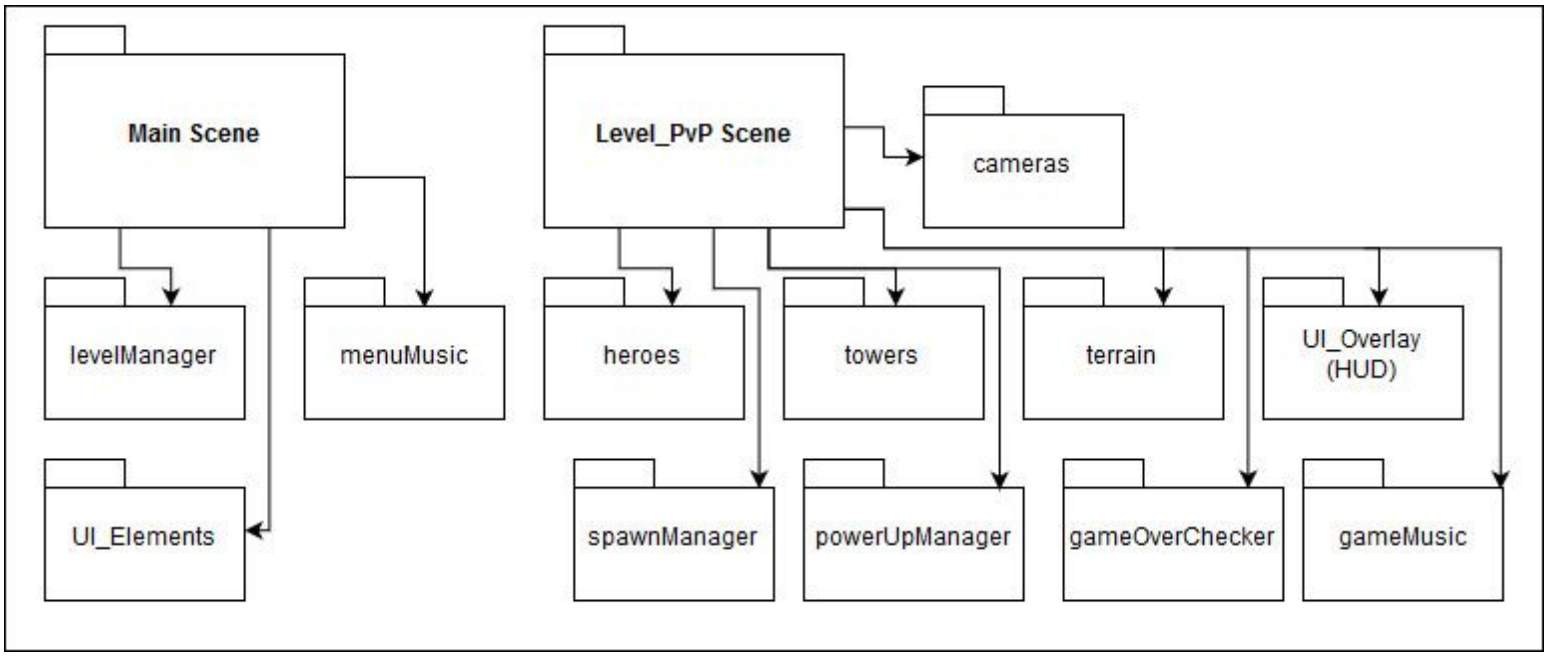
- Minion Purchases

  Actor: Player/s

  When a player has collected enough gold they can purchase one of several minions. To do this they hit the corresponding shortcut to the minion they wish to purchase. The corresponding amount of gold is deducted from their total and the minion spawns at their hero’s current location with a sound and visual effect.

2. High level design

The game application comprises two main Unity scenes which contain multiple modules. Some of these modules are made of multiple components (objects).



- **Main Scene**

- levelManager
  This component is responsible for starting up the Level_PvP Scene when prompted to start a game or to quit the application if the user/s requests it.

- menuMusic
  Responsible for playing the menu music during this scene.

- UI Elements
  Contains multiple UI components including background images, buttons, and info sheets.

- **Level_PvP Scene**

- cameras
  This scene contains various cameras that render different components of the scene. The two hero cameras follow the corresponding hero and render their view of the game world to one half of the screen each. They also render health bars for all units orientated to face that specific camera.

There is also a minimap camera that gives an orthogonal, top-down view of the map outline and indicators for units and powerups on the map.
A UI Overlay camera renders the HUD elements and a GameOver camera renders the final victory screen at end of game and clickable overlay to return to the main menu.

- heroes
  This consists of two game objects, one for each hero (the Barbarian Mage and Witch Doctor). These objects are controlled by the players and interact with the game world in multiple ways (moving, attacking, collecting, etc.)

- towers
  There are two game objects which contain models and AI scripts for the two opposing towers. The gameOverChecker module will monitor these components to see if an end game state has been reached.

- terrain
  The terrain module contains various components including a Nav-Mesh component, which minions use to navigate the world. It also includes a collection of 3D terrain assets that create the aesthetic of the game and a collection of 3D primitive blocks that act as colliders for the terrain world and outline for the minimap.

- UI_Overlay
  This module consists of two canvas components; one for the minimap and one for the HUD. These UI elements display useful information to the players. The minimap component displays all unit and powerup locations while the HUD displays tower health, gold amount and cooldowns of abilities and purchables.

- spawnManager
  This component is responsible for generating minions and assigning them to the correct teams, as well as designating what team and tower the minions should consider enemies.

- powerUpManager
  This component is responsible for randomly spawning powerups at set locations across the map. It receives information from collected and destroyed powerups to know what points are free to spawn a new powerup.

- gameOverChecker
  The gameOverChecker component constantly monitors the health state of both towers in the scene to know if a game over state has been reached which happens when one, or both towers health reaches zero.

- gameMusic

This module plays the selected track as background game music. It can be instructed to change tunes to a victory music loop when the game over state is reached.

3. Implementation

Each module, component or object makes uses of some data structures, UI elements and/or classes. Some objects have classes with the same name as the object.

- **levelManager**

- Classes:
  levelManager

- **menuMusic**

- Classes:
  Audio Source

- **UI_Elements**

- UI:
  Images of Heros
  Start button
  Quit button
  Info button
  Info sheets (help pages)

- **heroes**

- Data structures:
  Arrays
  States
  InputKeys
  Colliders

- UI
  Healthbar
  Minimap Indicator
  Direction Indicator

- Classes
  playerControl.
  Animation
  Health Manager

Game Object Identity
Audio Source
Capsule Collider
Rigid Body

- **spawnManager**

- Data structures:
  Arrays

- Classes:
  spawnManager

- **powerUpManager**

- Data structures:
  Arrays

- Classes:
  powerUpManager

- **towers**

- Data structures:
  Arrays
  States

- Classes:
  Tower AI
  healthManager
  gameObjectIdentity
  Capsule Collider

- **terrain**

- Data structures:
  NavMesh

- Classes
  Box Collider

- **gameOverChecker**

- Data structures:
  Arrays
  States

- UI:
  OK button

- Classes:
  gameOverCheck

- **camera**

- Data structures:
  Arrays

- Classes:
  followLook
  Audio listener
  Camera

- **UI_Overlay(HUD)**

- UI:
  Minion Images
  Tower Images
  Tower health indicator
  GoldBar amount indicator
  Skill indicator

- Classes:
  Mask

4. Program Validation & Verification

The testing of the game and game components happened on an ongoing, scheduled basis with regards to which mechanics and components were completed. This coincided with a weekly basis of tests.

- **Testing Week 1:**
  Testing Hero Movement
  Once hero models were introduced they were coded to respond to user input, moving and attacking when instructed. This also included making sure models were animated to coincide with what actions they were taking.

- **Testing Week 2:**
  Testing Minion Movement
  The minion models were introduced to the game but, unlike the hero models, they follow AI instructions. To make sure they traversed the map correctly, a Nav-Mesh was implemented that the minions would walk upon to avoid obstacles. This Nav-mesh/Minion interaction had to be tested to make sure it performed correctly. Then, walking animations were added to the minions as they moved.

- **Testing Week 3:**
  Test Hero Attacks vs Heroes and Hero Deaths
  Heroes were tested on the ability to damage the opposing hero and this damage/change in health to be reflected in a health bar. When a hero reached zero health, they needed to ‘die’, playing the corresponding animation and respawning after a delay.

- **Testing Week 4:**
  Test Hero Attacks vs Minions
  Next, minions were given a health component and so heroes had to be tested to see if they could accurately damage minions and this damage had to be reflected properly.

  Test Minion Attacks vs Minions and Heroes
  Minion attack and target selection was implemented and had to be tested. Minions had to accurately select both enemy minions and the enemy hero, move to the target, and attack and deal damage.

  Minion Deaths
  When minions reach zero health they need to die, playing the appropriate animation and being destroyed after a delay.

- **Testing Week 5:**
  Tower Attacks

The towers for both teams had target selection and attack damage implemented which had to be tested. At the same time, attacks vs towers from both heroes and minions was implemented and tested.

Test Golem Fear Attack
The golem minion had a special attack implemented that, after a certain duration, it would fear nearby enemy minions, making them run away from the golem. This interaction had to be tested thoroughly.

Test Dragon Breath
The dragon minion also had a special attack implemented were it would set fire to nearby enemies who would then begin to move erratically and take damage while on fire. This also required thorough testing.

- **Testing Week 6:**
  Minions Drop Gold and Heroes Collect It
  When minions die they had to drop gold which could be collected by the heroes. Minions had to drop gold for the enemy team which only the enemy hero could collect. The enemy hero would not be able to collect this gold if they were already at max gold. This entire interaction had to be tested for each type of minion and variation of teams.

- **Testing Week 7:**
  Powerups Spawn and Are Collected
  Four different types of powerups were created that would randomly spawn across the map. These powerups would then be collected by heroes which would give them a temporary boost depending on which powerup was collected. All these interactions had to be tested.

- **Testing Week 8:**
  Heroes Drop Gold
  Apart from minions, heroes also had to drop gold on death. This had to be implemented and tested.

  Gold purchases
  Using the gold acquired, players could purchase certain minions for a specific gold price but could not purchase two of the same type directly after each other. This interaction had to be tested.

  Particles and Sounds
  Particle effects and sounds were added to many different elements to provide players with feedback to certain actions. Effects and sounds were added to such things as attacks, abilities and power ups.

- **Testing Week 9-10:**
  Full playthroughs/playtest
  The game, now almost complete, underwent play testing to locate bugs, errors and imbalances. Many tweaks and iterations were done as part of this process.

5. Discussion

The initial intention of this game project was to make a 2 player MOBA game similar to single lane modes of other popular mobile and online MOBA games.
To achieve this intention certain requirements had to be met:

- Player controlled heroes with various abilities
- Navigable map with impassable terrain and obstacles
- Towers that must be destroyed to end the game
- AI controlled minions that spawn periodically and move across the map
- Sounds and effect to inform the players of certain events

Some additional goals were included to enhance the game experience:

- Co-op mode
- Powerups
- Gold and purchasables
- Enhanced minion AI
- Network playability

With regards to scope for the project all major goals and most additional goals were reached with the exception of co-op mode and network playability.
Network playability, although attempted, was a difficult and time consuming task to implement given the network and machines at hand while co-op mode was abandoned as there was deemed not enough time to implement a new gameplay mode.

Our personal impression of our game is that we achieved the project we set out to create to a good standard with a well balanced gameplay that seems to offer an enjoyable experience to those that have had an opportunity to play it so far.

6. User Manual

BATTLE LANE

by
Steven Mare MRXSTE008
Thethela Faltien FLTTHE004

A powerful Barbarian and mystical Witch Doctor fight it out to see who is fit to rule the land.

Objective: Choose one of the two heroes and destroy the enemy tower to win.
Be careful! If your tower is destroyed, you lose!

HEROES:
Play with one of the following two heroes to beat the other.

BARBARIAN MAGE:
Has basic attack and powerful slam.

Controls:
W - Move Forward
S - Move Backward
A - Rotate Left
D - Rotate Right
Q - Basic Attack
E - Slam Attack
1 - Summon Grunt
2 - Summon Golem
3 - Summon Dragon

WITCH DOCTOR:
Has basic attack and area of effect heal.

Controls:
Up Arrow - Move Forward
Down Arrow - Move Backward
Left Arrow - Rotate Left
Right Arrow - Rotate Right
/ - Basic Attack
Enter - Heal Effect
[- Summon Grunt
] - Summon Golem
\ - Summon Dragon

MINIONS:
AI controled units that assit the heroes in defeating the enemy.

Footman:
Simple melee unit.
Spawns automatically.

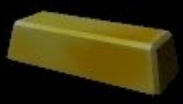
Lich:
Weak ranged unit.
Spawns automatically

Grunt:
More powerful melee unit.

Golem:
Tougher melee unit.
Can fear nearby enemy minions.

Dragon:
Squishy melee unit.
Can burn nearby enemy minions.

POWERUPS:
Collect these to give the hero a temporary boost.



Gold:
Used to purchase Grunts, Golems and Dragons.
Dropped by heroes and minions when they die.

Health Pack:
Restores some lost health.

Strength Boost:
Increase the effectivness of physical attacks.

Speed Boost:
Speed up the hero.

Invulnerability Potion:
Make your hero immune to damage.

Git Repository Link:

https://github.com/HAL22/Capstone-Game