**Guardian of the Outlands**

**Final Report**

**Deviations from the project plan**

The only deviation from the plan was that I didn’t modify the behaviour of enemies in the ‘No Target’ state. This state is entered when an enemy has no barrel to target, usually because other enemies are already trying to steal all of the available barrels (e.g. if there are 5 enemies but only 2 barrels remaining). The enemy will roam around the screen on a random path until a barrel becomes available.

The plan was to change from this aimless wandering to a Pac-Man-style player chasing, where enemies would try to move to a point in front of, behind, or next to the player, and upon getting close, try to crash into and destroy them. Moving to a point in front/behind/next to the player would have the effect of different enemies appearing to work together to cut the player off rather than heading straight for them.

The idea for this change came about during early testing where it became apparent that you could get a high score by simply sitting close to the barrels and just rotate your tank whilst constantly firing in their general direction. Any enemy that made it to the barrels was quickly destroyed without the player needing to drive their tank at all during the game.

However, this deviation was intentional as although it’s still possible to score without moving, power-ups (particularly the score multiplier) act as an incentive to move around as they make it much more lucrative to be constantly on the move. This approach of rewarding the player for moving rather than punishing them for not moving, combined with the risk that a wandering enemy can still accidentally crash into the player anyway, made this change seem unnecessary.

**Design Patterns**

The Factory design pattern is used to create pick-ups (hazards and power-ups), vehicles (player and enemies), and barrels. These are implemented in the PickUpFactory, VehicleFactory, and BarrelFactory scripts, respectively (located in ‘Scripts -> Gameplay -> Factories’). These factories are used by the GameManager script (located in ‘Scripts -> Gameplay’) to instantiate new instances of each type of prefab by passing it a string indicating the type of object it wants. The GameManager script contains methods to create a pick-up, vehicle, or barrel, which are called by the Update function if it detects that a new object needs to be created (e.g. if a wave of enemies has been wiped out, spawn a wave of 5 new enemies using the Vehicle Factory).

The Object Pooling design pattern was used to implement the player shooting bullets. Rather than instantiating and destroying instances of the ‘PlayerBullet’ prefab repeatedly and frequently, a pool of player bullet objects is created at the start of the game. The PlayerBulletPool script creates the object pool and contains a method used by the PlayerShooting script to ‘fire’ a bullet (make an inactive bullet active again at the new position it’s being fired from). The PlayerBulletPool and PlayerShooting scripts are located in ‘Scripts -> Gameplay -> Player’. A bullet is returned to the pool if it collides with an enemy (EnemyDeath script located in ‘Scripts -> Gameplay -> Enemy’) or exits the screen (DestroyArea script located in ‘Scripts -> Gameplay -> ScreenEdgeColliders’).

**Game Aspects**

**Collision Detection**

* When a player bullet hits an enemy it reduces the enemy’s health by 1. If the enemy’s health reaches 0, the enemy is destroyed. Located in the EnemyDeath script in ‘Scripts -> Gameplay -> Enemy’
* When an enemy crashes into the player, the player is destroyed (if they are not currently invincible). Located in the PlayerDeath script in ‘Scripts -> Gameplay -> Player’
* When the player drives over a power-up to collect it, its effect is applied to the player. Located in the PowerUp script in ‘Scripts -> Gameplay -> PickUps’
* When the player or an enemy drives over a hazard, its effect is applied to the player/enemy. Located in the Hazard script in ‘Scripts -> Gameplay -> PickUps’
* If the player drives off the screen, they ‘exit’ a collider that covers the whole game screen area. The game takes the position that the player left the collider at and transforms it to move the player to the opposite side of the screen so that they re-enter the collider
* There is a second collider that is slightly larger than the game screen area. If a player bullet exits this ‘destroy area’ collider (DestroyArea script located in ‘Scripts -> Gameplay -> ScreenEdgeColliders’), it’s returned to the player bullet object pool (PlayerBulletPool script located in ‘Scripts -> Gameplay -> Player’)

**Artificial Intelligence**

The EnemyMovement script located in ‘Scripts -> Gameplay -> Enemy’ contains the AI that controls an enemy’s state and behaviour. Enemies have three possible states:

* No Target – roam randomly around the screen until a barrel becomes untargeted or is dropped by an enemy after it was destroyed by the player
* Move To Barrel – a free barrel has been targeted so move towards it and collide with it to pick it up
* Escape – move to a randomly generated exit point somewhere on the edge of the screen to steal the barrel being carried

**Testing Strategy and Results**

I tested the game as it was being developed by running and play-testing it as new functionality was added. I carried out white box testing in this way to find errors and exceptions in the code, and test each code path. I also ran black box testing by just playing the game, using the menus etc without having a specific script or method in mind to test so that I could identify layout and playability issues.

I also sent builds to friends throughout the development process to play-test and give feedback. This was very useful as it’s sometimes easy to become blind to flaws in your own work when you spend lots of time on it. For example, I had increased the movement speed of enemies to what I thought was a challenging yet achievable level, but when a friend played it for the first time, it was too difficult to hit enemies. I concluded that by testing the game I had improved my skill level and so had tweaked the gameplay to my own skill which was too challenging and off-putting for a beginner.

Similarly, things that I had added that made sense to me turned out to be unclear to someone else. For example, I had feedback that the instructions for what effects the power-ups and hazards have were unclear as I had only provided the name and an example of what they look like in the instructions menu. When they played the game, they were confused when the oil slick made their tank spin to a random direction as the information I’d provided hadn’t specified that this would happen, and they thought it was a bug.

**What I would do differently**

I would change how the GUI during gameplay (timer, scores, waves) is shown as it’s currently displayed on top of the game area, which means that it sometimes occludes the gameplay. The most frustrating example of this is if a barrel is dropped underneath a GUI element and it appears that you have a certain number of barrels left when you in fact have an extra hidden one. Similarly, pick-ups are sometimes completely hidden whereas enemies disappear, but only temporarily as they are always moving. I would fix this by creating a toolbar area at the top of the screen solely for the GUI to separate it away from the gameplay.

I gave vehicles a rotation speed to allow larger enemies to turn more slowly than small, nimble enemies, but this has caused some problems where enemies orbit their movement target indefinitely. I partially overcame this in the ‘No Target’ roaming state by getting a new random movement target every 5 seconds but the problem persists (if infrequently) in the ‘Move To Barrel’ and ‘Escape’ states. I did experiment with steering behaviours of autonomous agents such as the ‘arrival’ behaviour which reduces the agent’s speed as it approaches its target, but this did not completely fix the issue. If I did the project again, I would use steering behaviours that have solved this problem by applying vector forces to the agent rather than simply translating it around the screen as I have done.

In terms of extensions if I was to develop the game further, I would like to add cooperative multiplayer; specifically local multiplayer but perhaps also online multiplayer. Cooperative multiplayer would benefit from having the kind of player chasing behaviour noted in the deviations section as it would open the game up to strategy whereby two players could work together to lure enemies away from their partner and the barrels whilst the other attacked.

A more significant multiplayer extension could be competitive multiplayer where each player has a team: their own tank and a few AI enemies. There would either be one set of barrels and the players would compete to protect their AI team members while they try to steal the barrels, or a Capture the Flag-style game mode where each team has different coloured barrels and must protect their own whilst trying to capture the enemy’s.

As I implemented local leaderboards, I would also have liked to set up online leaderboards where your top local score would be synced with other players’ scores to create a global leaderboard.

**Video demonstration of the game**

<https://www.youtube.com/watch?v=ka_oTBEzZYE>