|  |  |
| --- | --- |
| **Visual Assets:** | All sprits are in */Assets/images* folder.  *Cursor* is the cursor in the game scene.  *Explosion* is the sprite occurring when two missiles colliding together.  *Ground* is the ground spite in the game scene.  *House* is the object that player needs to protect.  *missileLauncher* launches the missile. |
| Audio Assets: | CollisionBGM played when two missiles colliding together  GameBGM played when playing main game  GameOverBGM played when player is destroyed  MainMenuBGM played when showing the main menu  MissileLaunchGGM played when player missile is launched |
| Recreated Level Scene | A close up of a sign  Description automatically generated  Figure Main Game  Figure 1 is a screen short of the main game scene. At the top there are several enemy missiles with red particle effect. There are three houses locating on the ground and the middle one with a triangular roof is the missile launcher releasing the defending missile like the missile in the right middle of the scene. |
| Initial Movement Development | As we can see in Figure 1, enemy missiles are randomly generated from the top scene and targeted to a random house on the ground. When mouse curser click, player missile will be released from the missile launcher to the mouse position. |
| Project Organization |  |
| Git Repository | A screenshot of a cell phone  Description automatically generated  Figure Git branches with commits  A screenshot of a social media post  Description automatically generated  Figure git commits history  There are three branches totally in my project, which are master, development and feature. At the moment, all three branches are the same. |
| Plan for Coded System | A screenshot of a cell phone  Description automatically generated  Figure Code files  Figure 4 listed the scripts at the moment this project contains.  **Movement**: In *EnemyMissiles.cs* and *EnemyMissileSpawner.cs*, enemy missiles are generated and target to the housed on the groud.  **User-input**: In *CursorController.cs,* courser position is recorded when left click, then trigger the missile releasing function.  **Object behaviours**: Launching defending missiles are handled in the *MissileLauncher.cs* and *PlayerMissileController*.*cs*  **Game rules and resources**: Game rules are mainly built up in the *GameConroller*.*cs*.  **Menu screens**: *MainMenuManager*.*cs* is responsible for managing the main menu. |
| Design Proposal | A close up of a logo  Description automatically generated  Figure Missiles  **Innovation**  Missile Command is the game that I decided to recreate. First of all, I reviewed the classic game how it is played and some game rules. I find that the missiles in most of the games are just dots. So, I plan to replace the dot missile to the real looked missiles in Figure 5.  **Project Plan**  My project plan is to find the required assets such as visual assets and audio assets, then begin to code. There are a lot of functions and rules in the original game. It is impossible to finish all these during such short term. So, I decided to finish some very basic functions primarily. If time is permitted, I will add more advanced features and refine the visual assets.  **Challenges and self-learning**  The challenge is to rotate the missile arrows to target the objects. So, I need to learn the essential mathematics knowledge such as triangular formulas.  Unity Coroutine is the new knowledge that I have never encountered before. It is hard to understand. At first, I want to make a detour to use other functions to replace the coroutine. However, I find that this will make my code messier and more complex. So, I learned coroutine from the very beginning. |