# CS 512 Final Project

`Tower defense` is a subgenre of strategy game where the goal is to defend a player's territories or possessions by obstructing enemy attackers, usually by placing defensive structures along their path of attack. Tower defense plays on the human psychological need for security, ownership, and desire to protect the people closest to us.

## Group Members:

`Yi Ren` (002269013) `Wentao Lu` (002276355)

---

## Dependency

- Microsoft .NET Framework
- Unity3d
- Jetbrains Rider

---

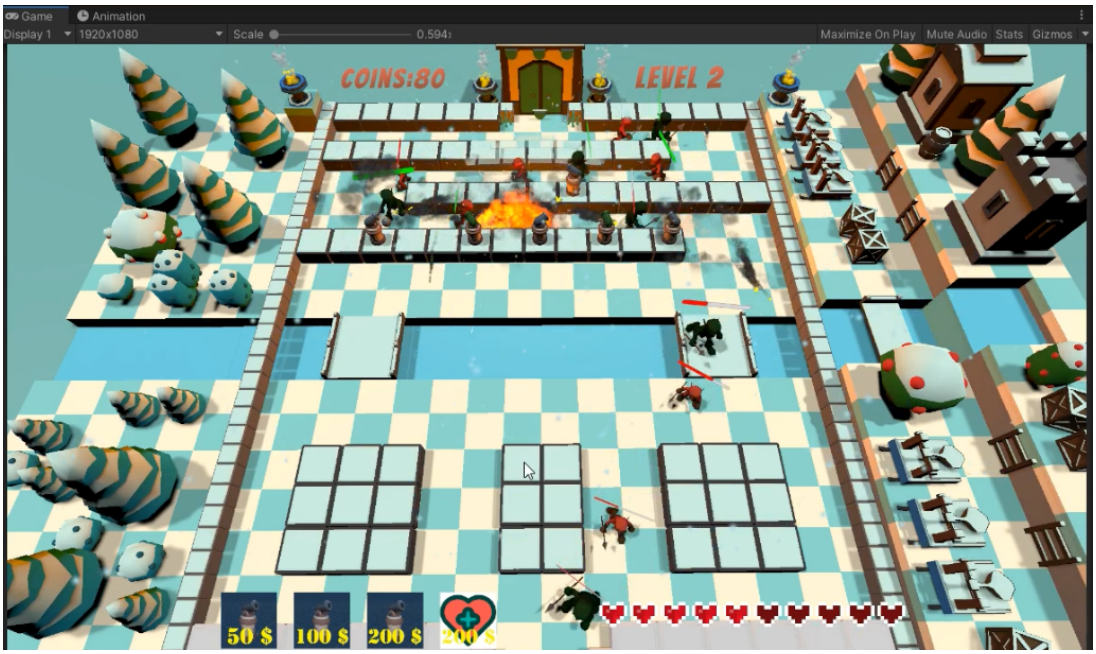
## Video demo

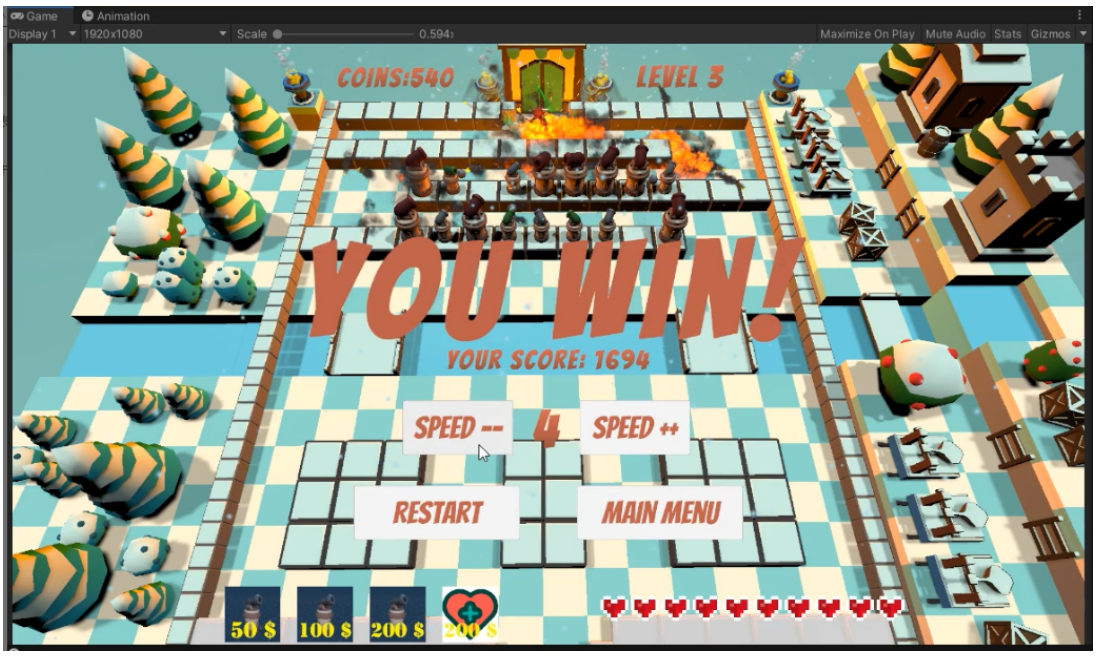
**Please check Viedo Demo - Final for more information.**

---

## Basic functions

### Part 1 - Gameplay




- **a. Must have at least 3 levels**

  For the game play, we designed 3 levels which are composed of different kind of enemies.

  - `Level` 1: Only normal goblins
  - `Level` 2: Normal goblins and elite goblins
  - `Level` 3: Normal goblins, elite goblins and Boss goblins


▶ Code

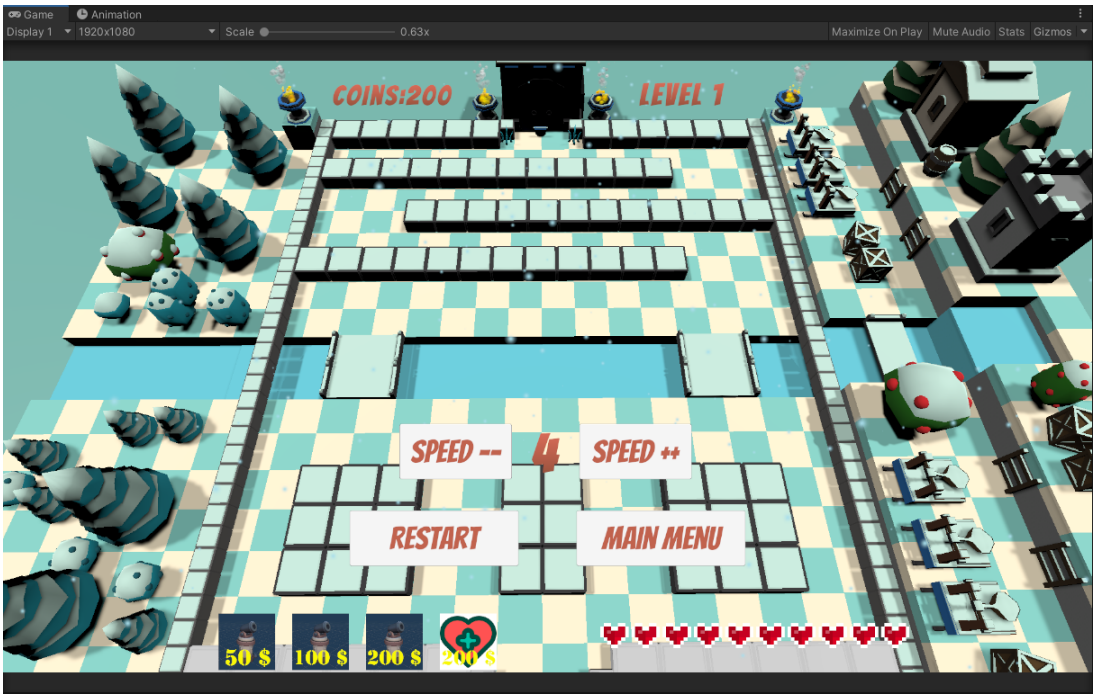
```
    public static InitConfig SharedInstance;
    // Define board size
    public const int BoardWidth = 16;
    public const int BoardHeight = 20;
    // Define cannon type
    public const int CannonType1 = 101;
    public const int CannonType2 = 102;
    public const int CannonType3 = 103;
    // Define enemy number for each level
    public Dictionary<string, int> enemyCount;
    // Define base cost to build cannon
    public const int BaseCost = 50;

    public const string TagObstacle = "Obstacle";
    // Define current game level
    public int gameLevel = 1;
    // Define current selected cannon type
    public int CANNON_TYPE = 0;
    public int DifficultLevel = 1;
    // Score, or money gained by kill enemy
    public int score;
    // Number of enemy passed
    public int passCount;

    private int Level1Enemy1, Level2Enemy1, Level3Enemy1, Level2Enemy2,
Level3Enemy2, Level3Enemy3;
```

- **b. can also add an option to 'speed up' the game**

we designed the UI for speed control and implemented the according function with `Time.timeScale`




▶ Code

**Handle piece click**

```
public void SpeedUp()
{
    speed++;
    timeScale = 1 + speed * 0.3f;
    MeshProSpeed.text = (speed + 4).ToString();
    Time.timeScale = timeScale;
}

public void SpeedDown()
{
    if(0.3 * (speed - 1) < -1)
        return;
    speed--;
    timeScale = 1 + speed * 0.3f;
    MeshProSpeed.text = (speed + 4).ToString();
    Time.timeScale = timeScale;
}
```

- **c. Learning curve – game should be easy to start and learn.**
  Quite easy for starters, since this game is designed for mobile phone and only have a few buttons.

Specifically, we only have 4 buttons in the main UI, and 4 buttons in the submenu when you press `Esc`. Also, we implemented some hint for new players, in order to provide a better gameplay.


**Main UI**

- `Button1` : Cannon type 1
- `Button2` : Cannon type 2
- `Button3` : Cannon type 3
- `Button4` : Heal (Add 1 heart in the player life counter)

**Submenu**

- `Button1` : Speed up
- `Button2` : Speed down
- `Button3` : Restart the game
- `Button4` : Go back to main scene

▶ Code

```
        BtnRestart.onClick.AddListener(RestartGame);
        BtnMain.onClick.AddListener(LoadMainMenu);
        BtnSpeedUp.onClick.AddListener(SpeedUp);
        BtnSpeedDown.onClick.AddListener(SpeedDown);
```

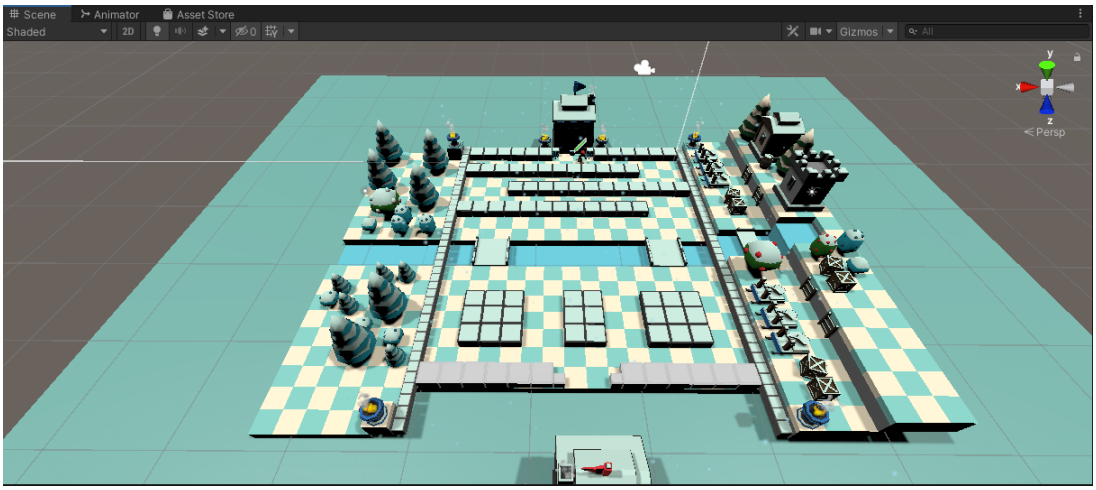
▶ Code

```
    public void Heal()
    {
        if(InitConfig.SharedInstance.score < 200)
            return;
        InitConfig.SharedInstance.score -= 200;
        ScreenTextManager.SharedInstance.MeshProScore.text = "COINS:" +
InitConfig.SharedInstance.score;
        InitConfig.SharedInstance.passCount -= 1;
        PlayerStats.Instance.Heal(1);
    }
    public void ShowAvailableSlice(int type)
    {
        Debug.Log("ShowAvailableSlice");
        InitConfig.SharedInstance.CANNON_TYPE = type;
        foreach (var item in obsList)
        {
            item.HighLight();
        }
    }
    public void RestartGame()
    {
        Resources.UnloadUnusedAssets();
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex,
LoadSceneMode.Single);
    }
    public void LoadMainMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }
```

---

## Part 2 - Look and feel

- **Graphics and animation – game must be 3D and animated**
  Please check our video demo or application for more details.



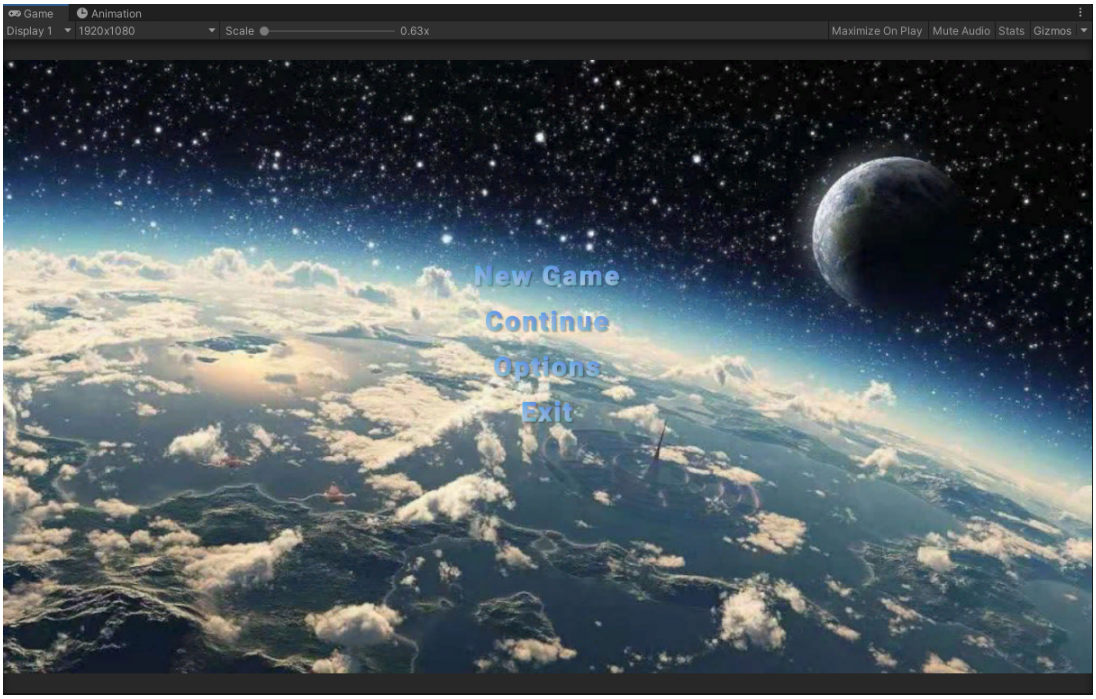

- **Sound – game must have sound, including some type of theme music**
  Please check our video demo or application for more details.
- **The graphics, animation, sound, etc., should all match the theme of your game.**
  Please check our video demo or application for more details.

---

## Part 3 - Controls and UI

- **Game must be designed with mobile gameplay in mind**
  To work properly on mobile phone, we designed our game with only 4 main buttons, which is convenient for moible users, since the screen size of mobile devices are limited(no more than 7 inches for most cases). Also, we implemented the scene based on landscape, which will make full use of mobile screen.

- **Main menu with at least the following 4 buttons:**



- **The in-game UI should be intuitive (player can figure out what a button does by the button's icon)**