**TASKS**

**\*\*Start level with Machine gun:\*\***

As idea is to reuse the code. Just figure out how a spawning and placement of a tower is done then copy some lines of code and place in a function in GameUI.cs.

//Main logic that build random tower

void BuildRandomTower()

{

state = State.Building;

IPlacementArea[] area = GameObject.FindObjectsOfType<TowerPlacementGrid>();

Tower towerToBuild = null;

//You can also directly assign the machine gun without using foreach

//You can also use towerToBuild = tower[0]; if you want

//making it with for loop is also better then foreach

foreach (Tower tower in LevelManager.instance.towerLibrary)

{

towerToBuild = tower;//Since first tower is maching gun

//There are different way you can assign any tower from Tower Library

break;

}

m\_CurrentTower = Instantiate(towerToBuild.towerGhostPrefab);

m\_CurrentTower.Initialize(towerToBuild);

m\_CurrentTower.Hide();

List<PlacementTile> possibleFitTiles = new List<PlacementTile>();

List<int> areaId = new List<int>();

List<IntVector2> gridAreaList = new List<IntVector2>();

TowerFitStatus fits = TowerFitStatus.Overlaps;

for (int j = 0; j < area.Length; j++)

{

this.m\_CurrentArea = area[j];

PlacementTile[] allTiles = m\_CurrentArea.transform.GetComponentsInChildren<PlacementTile>();

for (int i = 0; i < allTiles.Length; i++)

{

PlacementTile anyTile = allTiles[i];

Vector3 pointToFit = anyTile.transform.position;

m\_GridPosition = this.m\_CurrentArea.WorldToGrid(pointToFit, m\_CurrentTower.controller.dimensions);

fits = this.m\_CurrentArea.Fits(m\_GridPosition, m\_CurrentTower.controller.dimensions);

if (fits == TowerFitStatus.Fits)

{

possibleFitTiles.Add(anyTile);

areaId.Add(j);

gridAreaList.Add(m\_GridPosition);

}

}

}

if (possibleFitTiles.Count <= 0)

{

CancelGhostPlacement();

Debug.Log("No Possible To Fit Tiles");

return;

}

//Taking random

int randomFitTile = UnityEngine.Random.Range(0, possibleFitTiles.Count);

m\_GridPosition = gridAreaList[randomFitTile];

this.m\_CurrentArea = area[areaId[randomFitTile]];

m\_CurrentTower.Show();

m\_GhostPlacementPossible = fits == TowerFitStatus.Fits && IsValidPurchase();

m\_CurrentTower.Move(this.m\_CurrentArea.GridToWorld(m\_GridPosition, m\_CurrentTower.controller.dimensions),

this.m\_CurrentArea.transform.rotation,

m\_GhostPlacementPossible);

if (m\_CurrentArea != null)

{

TowerFitStatus newfits = m\_CurrentArea.Fits(m\_GridPosition, m\_CurrentTower.controller.dimensions);

if (newfits == TowerFitStatus.Fits)

{

// Place the ghost

Tower controller = m\_CurrentTower.controller;

Tower createdTower = Instantiate(controller);

createdTower.Initialize(m\_CurrentArea, m\_GridPosition);

}

CancelGhostPlacement();

}

state = State.Normal;

}

Now as we need to call it at start. I have created the Start function in GameUI.cs and place as follow

public void Start()

{

SetRandomTowerAtStart();

}

//Just taken another function to loop with n Number

//Uses Invoke function to give delay between spawn

[Range(0,5)]

public float nNumberOfTowerToBuild;

void SetRandomTowerAtStart()

{

for(int i=0;i<nNumberOfTowerToBuild;i++)

{

Invoke("BuildRandomTower",0.25f);

}

}

Here idea to use this nNumberOfTowerToBuild variable is to spawn multiple tower not just single.

Use Range attribute to show slider and camp the range from 0-5

**\*\*New defense tower:\*\***

Created a new tower and uses the few feature which is already exist.

Duplicated Rocket and rename to MRocket to use the same model.

Feature change as follow.

* Projectile change to Super Tower Projectile
* Tier 1 will only shoot the ground enemies
* Tier 2 and Tier 3 will shoot all
* Multiple Shoot at a time base on n Projectile points.

Changes icon to



Duplicated the Rocket Level Data and rename and made changes to description, cost and charge.

As I want this in every level just added to Tower library

In order to give variation while shooting added few lines of code in AttackAffector.cs

if (isMultiAtTime)

{

for (int i = 0; i < projectilePoints.Length; i++)

{

List<Targetable> enemies = towerTargetter.GetAllTargets();

m\_Launcher.Launch(enemies, damagerProjectile.gameObject, projectilePoints,i);

}

}

Created variable public bool isMultiAtTime; similart to public bool isMultiAttack; just to Launch projectile as n number of Porjectile Points.

**\*\*New enemy:\*\***

As I wanted to create ground base enemy uses HoverTank and made some design changes and name it as NewTank.

Added to the wave system (WaveManager to spawn in the wave sequence)

Changes healths from the inspector.

Now to as per the idea we need to create HoverTank when it destroyed.

Implemented as follow (Reuse the Spawn code from WaveManager and implemented in DamageableBehaviour.cs as we need to check health is less then or equal to zero)

Logic:

//Which one to spawn

public AgentConfiguration agentConfiguration;

void CreateSomeMoreL()

{

if (agentConfiguration==null || GetComponent<Agent>().isSpawnedOnDestroy)

return;

GetComponent<Agent>().isSpawnedOnDestroy = true;

var poolable = Poolable.TryGetPoolable<Poolable>(agentConfiguration.agentPrefab.gameObject);

if (poolable == null)

{

return;

}

var agentInstance = poolable.GetComponent<Agent>();

agentInstance.transform.position =transform.position ;

agentInstance.Initialize();

agentInstance.SetNode(GetComponent<Agent>().m\_CurrentNode);

agentInstance.MoveToNode();

agentInstance.transform.rotation = transform.rotation;

}

using variable public **AgentConfiguration** agentConfiguration; is to not only spawn HoverTank but also any Enemy.

Created another bool varaible **isSpawnedOnDestroy** in **Agent** just to control multiple spawn bug which will be reset on Initialize fucntion of **Agent**.

Now in the TakeDamage function added this line which will call when the current enemy dies

if (isDead)

{

CreateSomeMoreL();

}

**\*\*Panning:\*\***

Added two logic for panning

* using Mouse Scroll
* using Middle mouse button and drag over the screen

Added few lines for Mouse Scroll in KeyboardMouseInput.cs

currentScrollDelta = (int)UnityInput.mouseScrollDelta.y;

if (lastScrollDelta != currentScrollDelta && scrollDirection==0)

{

lastScrollDelta = currentScrollDelta lastStopScroll = Time.time;

scrollDirection = (int)UnityInput.mouseScrollDelta.y;

}

and Added one more condition in

// Down

if (UnityInput.GetKey(KeyCode.DownArrow) || UnityInput.GetKey(KeyCode.S) || scrollDirection<=-1)

{

cameraRig.PanCamera(Vector3.back \* Time.deltaTime \* mouseEdgePanSpeed \* zoomRatio);

cameraRig.StopTracking();

}

// Up

if (UnityInput.GetKey(KeyCode.UpArrow) || UnityInput.GetKey(KeyCode.W) || scrollDirection >= 1)

{

cameraRig.PanCamera(Vector3.forward \* Time.deltaTime \* mouseEdgePanSpeed \* zoomRatio);

cameraRig.StopTracking();

}

For Mouse Middle Button click added just one condition to the function as below

if (mouseInside && UnityInput.GetMouseButton(2))

{

PanWithScreenCoordinates(mousePos, screenPanThreshold,mouseEdgePanSpeed);

}

**\*\*Tooltip:\*\***

For tool tip added a seperated script and only using in level 1

Just copy some UI from Game Ui and then added the script to that copied (Duplicated UI)

Named it as TopTipScroll

Code is as follow

// Start is called before the first frame update

//Gameobject as panel for tip

public GameObject tipPanel;

void Start()

{

//We will show the tip after 1 sec so using Invoke function (we can also use IEnumerator)

Invoke("ShowTip", 1);

}

void ShowTip()

{

tipPanel.SetActive(true);

//After enabling tip panel we hide it after 5 Sec

Invoke("HideTip", 5);

}

void HideTip()

{

tipPanel.SetActive(false);

}

**\*\*Scoreboard:\*\***

In order to get the time taken to complete the wave added the logic as follow

In this case uses GameManager.cs and added one variable to store the start time information

//To Store start time of the wave

public float startTime;

//Whenever start wave button click we say new level started and calculate from this time

public void NewLevelStarted()

{

startTime = Time.time;

}

As the function to start wave is in LevelManager.cs I just call NewLevelStarted() function of GameManager.cs from it as follow.

public virtual void BuildingCompleted()

{

ChangeLevelState(LevelState.SpawningEnemies);

GameManager.instance.NewLevelStarted();//Added this

}

Showing and Store Time information.

In order to show the Time added one Text in Game Over Page below the Game Status and Added this line in ScorePane.cs as its used to show stars I have uses it to also show time and 5 Best times

//Minus Current time from Start time to get actual time

float timeTaken =Time.time-GameManager.instance.startTime;

int min = Mathf.FloorToInt(timeTaken / 60);

int sec = Mathf.FloorToInt(timeTaken % 60);

timeTakenText.text = "Time Taken: " + string.Format("{0:00}:{1:00}", min, sec);

//In ordet to create 5 Best Time list

Create one Panel in the Game Over

(Added Layouts and Text in order to adjust base on n number of text)

After setting the UI for 5 Best Time created script GameData.cs in order to add static function and use from this.

The GameData.cs will contain some PlayperPref logic as follow

//Passing id in order to get level best

// Start is called before the first frame update

//As we need to store information for each level we use one more parameter and use that in PlayerPref key

//This will Get Best Time

public static float GetBestTime(int id,string level)

{

return PlayerPrefs.GetFloat("BestTime" + id+"OfLevel"+ level);

}

//This will Set Best Time for the level

public static void SetBestTime(int id, string level, float val)

{

PlayerPrefs.SetFloat("BestTime" + id + "OfLevel" + level, val);

}

//Whenever level completed we increase the count and save that information to playerpref

//Get Level Completed Counts

public static int GetBestTimeMaxCount(string level)

{

return PlayerPrefs.GetInt("BesTimeMaxCountOf"+level);

}

//Set Level Completed Counts

public static void SetBestTimeMaxCountAdded(int val,string level)

{

PlayerPrefs.SetInt("BesTimeMaxCountOf"+level, val);

}

//Now the final logic of ScorePanel.cs will be as follow

public void SetStars(int score,string resultText)

{

//We will disable the 5 Best Time list and Time Taken if Level Failed

bestTimeListText[0].transform.parent.gameObject.SetActive(false);

if (resultText.Contains("FAILED"))

{

timeTakenText.gameObject.SetActive(false);

}

if (score <= 0)

{

return;

}

//Minus Current time from Start time to get actual time

float timeTaken =Time.time-GameManager.instance.startTime;

//Saving to PlayerPref in ordet to show in 5 Best Time

string currentLevel = Application.loadedLevelName;

GameData.SetBestTimeMaxCountAdded(GameData.GetBestTimeMaxCount(currentLevel) + 1, currentLevel);

GameData.SetBestTime(GameData.GetBestTimeMaxCount(currentLevel), currentLevel, timeTaken);

int min = Mathf.FloorToInt(timeTaken / 60);

int sec = Mathf.FloorToInt(timeTaken % 60);

timeTakenText.text = "Time Taken: " + string.Format("{0:00}:{1:00}", min, sec);

score = Mathf.Clamp(score, 0, starImages.Length);

for (int i = 0; i < score; i++)

{

starImages[i].sprite = achievedStarSprite;

}

bestTimeList.Clear();

//We will check through the PlayerPref using For loop with count and store in a list

for(int i=1;i<=GameData.GetBestTimeMaxCount(currentLevel);i++)

{

if(GameData.GetBestTime(i,currentLevel)>0)

{

bestTimeList.Add(GameData.GetBestTime(i,currentLevel));

}

}

bestTimeList.Sort();

//Displaying the Best Time using For loop with length bestTimeListText count

//We will only show when its greater then 1

if (bestTimeList.Count > 1)

{

bestTimeListText[0].transform.parent.gameObject.SetActive(true);

}

for (int i=0;i< bestTimeListText.Count;i++)

{

if (i < bestTimeList.Count)

{

min = Mathf.FloorToInt(bestTimeList[i] / 60);

sec = Mathf.FloorToInt(bestTimeList[i] % 60);

bestTimeListText[i].text = string.Format("{0:00}:{1:00}", min, sec);

bestTimeListText[i].gameObject.SetActive(true);

}

}

}

Time Taken to do this above is around 6 Hours + half Hour for documenting

\*\*\*\*\*\*\*\*Thank You \*\*\*\*\*\*\*