CW2 Programming Game:

Planned to use RPG Maker MZ to create our game, but due to assignment constraints, while the program is a viable, industry engine it wasn’t suitable to use in this scenario, so we switched to unity.

-due to Christmas holidays, the intended deadline for assessment gave us less than three weeks to work on the game proper, while managing the other assessments due in at same time.

-we had to plan a schedule to enable us to work together

-we struggled to achieve the specific collision detection but using an “if” statement we defined the collision object.

-ran into a syntax error while writing the “target Rigidbody” code – fixed by reviewing contents and adding a semicolon and removing an unnecessary bracket.

-while coding the health system, we missed 9ut on an “int” until we reviewed another piece of code and fixed the mistake.

Vector3 = UnityEngine.Vector3 = use to make operable code on unity engine

Code for game: Complete version~:

Keeping score:

Textmeshpro

Import tmp essential – import if needed.

-hierarchy, gui, tmp

Rename object “score text”

-change anchor point

Code/ Variable~:

-gameManager script

Using System.Collections; //imports libraries of methods and classes in unity TMP imp-orts another package

Using TMPro;

Publix TextMeshProUGUI scoreText;

Private int score;

-health system

-fail states

Game over code:

public TextMeshProUGUI gameOverText;

void Start() {

gameOverText.gameObject.SetActive(true); }

public void GameOver() {

gameOverText.gameObject.SetActive(true); }

private void OnTriggerEnter(Collider other) {

Destroy(gameObject);

If (!gameObject.CompareTag(“Bad”)) { gameManager.GameOver(); } }

STOP SPAWNING AND SCORE ON GAME OVER.

public bool isGameActive;

void Start() { … isGameActive = true; }

public void GameOver() { … isGameActive = false; }

IEnumerator SpawnTarget() { while (//True//isGameActive) { … }

🡨🡪

private void OnMouseDown() {

if (gameManager.isGameActive) { … [all function code moved inside] }}

RESTART BUTTON method:

using UnityEngine.SceneManagement;

public void RestartGame() {

SceneManager.LoadScene(SceneManager.GetActiveScene().name); }

-difficulty options:

DIFFICULTY SCRIPT:

using UnityEngine.UI;

private Button button;

void Start() {

button = GetComponent<Button>(); }

SET DIFFICULTY:

void Start() {

button = GetComponent<Button>();

button.onClick.AddListener(SetDifficulty); }

void SetDifficulty() {

Debug.Log(gameObject.name + “was clicked”); }

START IN GAME DIFFICULTY:

GameManager.cs

void Start() { … }

public void StartGame() {

isGameActive = true;

score = 0;

StartCoroutine(SpawnTarget());

UpdateScore(0);

}

DifficultyButton.cs

private GameManager gameManager;

void Start () {

…

gameManager = Game~Object.Find(“Game Manager”).GetComponent<GameManager>();

}

void SetDifficulty() {

…

gameManager.StartGame();

}

title screen:

public GameObject titleScreen;

StartGame() {

… titleScreen.gameObject.SetActive(false); }

PARAMETERS:

public int difficulty;

void SetDifficulty() {

… gameManager.startGame4(difficulty); }

🡨🡪

public void StartGame(int difficulty); {

spawnRate /= difficulty; }

New Concepts and Skills:

* AddListener()
* Passing parameters between scripts
* Divide/Assign (/=) operator
* Grouping child objects

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Target : MonoBehaviour

{

private Rigidbody targetRb;

// Start is called before the first frame update

void Start()

{

targetRb = GetComponent<Rigidbody>();

targetRb.AddForce(UnityEngine.Vector3.up \* Random.Range(12, 16), ForceMode.Impulse);

targetRb.AddTorque(Random.Range(-10, 10), Random.Range(-10, 10), (Random.Range(-10, 10) ForceMode.Impulse);

transform.position = new UnityEngine.Vector3(Random.Range(-4, 4), -6);

}

// Update is called once per frame

void Update()

{

}

}