

Project Documentation

COMP.6214



November 9, 2017

Corban Hirawani

2008003866

# Player Controls.

## Movement.

Player can move their plane with the WASD keys

void movementController()

// this method gets user input and translates it into movement velocity

{

// Input.GetAxis gets user input and converts it to a value from 0 to 1

float moveHorizontal = (float)Math.Round(Input.GetAxis("Horizontal"), 2); //rounding it to 2 decimal values

float moveVertical = (float)Math.Round(Input.GetAxis("Vertical"), 2);

//to make animation less smooth

Vector2 movement = new Vector2(moveHorizontal, moveVertical);

GetComponent<Rigidbody2D>().velocity = movement \* speed;

boundaryClamper();

}

## [[1]](#endnote-1)Collision.

If the player collides with an enemy or obstacles they will take damage, player has 3 lives, if lives gets to 0, the player dies and is de-activated until the game is restarted

void OnTriggerEnter2D(Collider2D collision)

{

//if player is hit by an enemy or obstacle and time since last death is greater than 1 second

if (Time.time > deathBuffer && (collision.tag == EnemyShot.tag || collision.tag == Enemy.tag))

{

PlayerTookDamage(collision);

Destroy(collision.gameObject);

}

if (Time.time > deathBuffer && !onUpperLayer && (collision.tag == Land1.tag || collision.tag == Land2.tag))

{

PlayerTookDamage(collision);

}

}

This even is triggered when the player collides with another game objects collider component.

The Playertookdamage method is then called if the player takes damage:

void PlayerTookDamage(Collider2D collision)

//method for when player takes damage from any source.

{

playerLives--; //decrement playerlives

StartCoroutine (FlashDamage(GetComponent<SpriteRenderer>(), 3)); //flashes sprite 3 times

if (PlayerScript.playerLives < 1) //if player lives < 1 the player has died.

PlayerDied();

transform.position = new Vector2(0, -5); //reset player position

deathBuffer = Time.time + 1f; //player cant die in quick succession, addds 1 second to death buffer

}

## Layer.

The player can only collide with obstacles if they are on the upper layer, they can only shoot enemy planes on the upper layer, and can only shoot turrets if on the lower layer.

void layerController()

{

if (Input.GetKeyDown(KeyCode.F) && onUpperLayer)

// if F is pressed then move to lower layer

{

newScale = minScale;

onUpperLayer = false;

// this boolean value tells us whether or not player is on a certain layer

}

else if (Input.GetKeyDown(KeyCode.R) && !onUpperLayer)

// if r is pressed then move to Upper layer

{

newScale = maxScale;

onUpperLayer = true;

}

// Lerp function to smooth the transition-resize animation

GetComponent<Rigidbody2D>().transform.localScale = Vector2.Lerp(transform.localScale, newScale, Time.deltaTime \* layerTransitionSpeed);

boundaryClamper(); //clamp player to boundary after resize

}

## Shoot

Play can shoot two bullets if the fire button is pressed and or held

void shootController()

{

// if time passed is greater than the nextFire wait buffer (0.15 seconds)

// and the fire button is pressed

if (Time.time > nextFire && Input.GetButton("Fire1"))

{

nextFire = Time.time + fireRate;

//shoot by instantiating two bullets

Instantiate(SingleShot, ShotSpawn.position - new Vector3(0.3f, 0, 0), ShotSpawn.rotation);

Instantiate(SingleShot, ShotSpawn.position - new Vector3(-0.3f, 0, 0), ShotSpawn.rotation);

}

}

# Spawn points

Enemies(Planes) and Obstacles(Islands and Battleships) both spawn from two spawn points located at the top of the screen. Turrets are fixed to the land objects in the prefab and are not spawned by themselves via scripts

## Enemy spawning

*Enemy planes are spawned with this Coroutine method:*

IEnumerator EnemySpawner()

{

yield return new WaitForSeconds(3); // 3 seconds breather before first spawn

int difficulty = 12; //adds to the spawnTimer variable via the random class

float spawnTimer = 0f; //speed of which enemies spawn in seconds

Vector3 spacer = new Vector3(0, 11, 0);

// this vector3 object "spacer" will be added to the spawn locations transform position

while (true) //infinite loop until game restart

{

//left side spawns first

spawnTimer = R.Next(difficulty, difficulty+1);

spacer.x = 0;

//reset x value in spacer object

spacer.x += R.Next(1, 4);

// adds to the x value to spawn in random location on left side

Instantiate(enemy, spacer, enemy.transform.rotation); //instantiate enemy

yield return new WaitForSeconds(spawnTimer);

//right side second

spawnTimer = R.Next(difficulty, difficulty+1);

spacer.x = 0;

//reset x value in spacer object

spacer.x -= R.Next(1, 4);

//minus from the x value to spawn in random location on right side

Instantiate(enemy, spacer, enemy.transform.rotation); //instantiate enemy

yield return new WaitForSeconds(spawnTimer);

//reset

if (difficulty > 3) // dont go lower/faster than 3 difficulty

{

difficulty--;

}

}

}

## Obstacle spawning

Land objects are spawned with these two methods:

IEnumerator ObstacleSpawner()

{

yield return new WaitForSeconds(3); // tiny breather before starting wave 1

while (true) //this repeats infinitely until player dies or game is restarted/exited

{

float spawnTimer = R.Next(10, 13);

//timer is set to a random value between 10 and 13

//spawns a random land object on the left side, random number 1 or 2

InstantiateRandomLandObject(island, battleship, Spawn1, Spawn1, R.Next(1, 3));

yield return new WaitForSeconds(spawnTimer);

// waits for the duration of spawnTimer,

thanks to the coroutine function waitforseconds.

//same as above by for the right side.

InstantiateRandomLandObject(island, battleship, Spawn2, Spawn2, R.Next(1, 3));

yield return new WaitForSeconds(spawnTimer);

}

}

void InstantiateRandomLandObject(GameObject gameObject1, GameObject gameObject2, Transform pos, Transform rot, int rNum)

{

// takes this parameter from the R.Next(1,3) function in the ObstacleSpawner Coroutine

switch (rNum)

{

case 1: // if the random number is 1, spawn island

Instantiate(gameObject1, pos.position, rot.rotation);

break;

case 2: // if the random number is 2, spawn battleship

Instantiate(gameObject2, pos.position, rot.rotation);

break;

}

}

# **Enemy logic.**

## Planes

*Planes shoot 3 bullets that spread, every X seconds*.

IEnumerator EnemyShoot() //instantiates 3 bullets with different rotations

{

yield return new WaitForSeconds(1); //1 second initial buffer

while(true) //infinite coroutine loop until game restart or enemy death.

{

Instantiate(EnemyShot, ShotSpawn.position - bulletOffset, bulletRotation1);

Instantiate(EnemyShot, ShotSpawn.position, ShotSpawn.rotation);

Instantiate(EnemyShot, ShotSpawn.position + bulletOffset, bulletRotation2);

yield return new WaitForSeconds(2.5f); //every 2.5 seconds

}

}

*Bullets move downwards with this piece of code attached to the bullet prefab:*

void Start()

{

//sets bullet into motion

GetComponent<Rigidbody2D>().velocity = GetComponent<Rigidbody2D>().transform.up \* bulletSpeed;

}

*Enemy moves downwards and left or right to cross middle depending on spawn location:*

IEnumerator EnemyMotion()

{

enemy.velocity = transform.up \* enemySpeed; //sets motion downwards

yield return new WaitForSeconds(1);

while (true)

{

enemy.velocity -= new Vector2(0, 0.1f) \* enemySpeed; //resets x velocity

enemy.velocity += new Vector2(-enemy.position.x \* 0.25f, 0) \* enemySpeed;

// adds x velocity towards the middle of screen

yield return new WaitForSeconds(1); //wait 1 second till next x motion change

}

}

## Turrets

Turrets face player and shoot 1 bullet every X seconds and follow their attached land object

IEnumerator TurretShoot()

{

yield return new WaitForSeconds(1);

while (true)

{

Instantiate(EnemyShot, transform.position, transform.rotation);

yield return new WaitForSeconds(2);

}

}

## Collision

The following method is used for both the Plane and Turret Enemies:

private void OnTriggerEnter2D(Collider2D collision)

{

// if collision is SingleShot, and player is on the upper layer

if (collision.name == "SingleShot(Clone)" && PlayerScript.onUpperLayer == false)

{

turretLives--; // takes damage

StartCoroutine (FlashDamage(GetComponent<SpriteRenderer>(), 1)); // flash

if (turretLives < 1)

// if lives less than 1, instantiate explosion and destroy gameobject

{

Instantiate(Explosion, transform.position, transform.rotation);

Destroy(gameObject);

PlayerScript.score += 100; //add points to score

}

Destroy(collision.gameObject); //destroy bullet

}

}

# Obstacles.

Land obstacles move straight downwards and carry turrets, This is the land mover script:

private float landSpeed;

void Awake()

{

landSpeed = -1.786f;

//controls how fast the object moves,

//set to negative value so that the transform.up method moves downwards instead of up

GetComponent<Rigidbody2D>().velocity = GetComponent<Rigidbody2D>().transform.up \* landSpeed;

//sets the objects velocity to transform.up \* landspeed

}

# Particle effects:

Flash Animation, called every time an agent takes damage:

IEnumerator FlashDamage(SpriteRenderer sr, int times)

{

for (int i = 0; i < times; i++) //flash N times

{

sr.color = new Color (1f, 1f, 1f, 0.3f); //turns opacity down

yield return new WaitForSeconds (.1f); // flash every 0.1 seconds

sr.color = Color.white; //resets colour value

yield return new WaitForSeconds (.1f);

}

}

Explosions are called in collision detection methods, when agents die.

# Menu System

Pause Menu methods:

public void ExitToMainMenu() //Exits to main menu

{

if (PlayerScript.playerLives > 0)

{

GetComponent<WriteNewScore>().GetNewScore();

GetComponent<WriteNewScore>().UpdateScore();

}

SceneManager.LoadScene("MainMenu", LoadSceneMode.Single);

Time.timeScale = 1;

}

public void Resume() //exits pause menu

{

canvas.gameObject.SetActive(false);

Time.timeScale = 1;

}

public void Restart() //restarts game

{

respawnCanvas.gameObject.SetActive(false);

Time.timeScale = 1;

SceneManager.LoadScene("SceneOne", LoadSceneMode.Single);

}

Main Menu methods:

public void PlayGame() //start new game

{

SceneManager.LoadScene("SceneOne", LoadSceneMode.Single);

}

public void HighScores() //show highscores

{

SceneManager.LoadScene("HighScores", LoadSceneMode.Single);

}

public void QuitGame() //exit to desktop

{

Application.Quit();

}

High Scores Menu and scoring logic:

//this method reads from the highscores.txt file and stores it into the topTenScores array

void ReadHighScoresFromFile()

{

try

{

StreamReader sr = new StreamReader("HighScores.txt");

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

{

String line = sr.ReadLine();

topTenScores[i] = string.IsNullOrEmpty(line) ? 0 : Convert.ToInt32(line);

}

sr.Close();

}

catch (Exception e)

{

Debug.Log("The file could not be read:");

Debug.Log(e.Message);

}

}

//displays highscores to text ui element

public void DisplayHighScores()

{

hsText.text = "";

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

{

hsText.text += topTenScores[i] + "\n";

}

}

public void Return()

{

SceneManager.LoadScene("MainMenu", LoadSceneMode.Single);

}

Death Screen and logic

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

// pauses game on escape key, resumes on second press

canvas.gameObject.SetActive(!canvas.gameObject.activeInHierarchy ? true : false);

Time.timeScale = !canvas.gameObject.activeInHierarchy ? 1 : 0;

}

if (PlayerScript.playerLives <= 0)

// if player lives less than 0, waitthengameover coroutine

{

StartCoroutine("WaitThenGameOver");

}

}

IEnumerator WaitThenGameOver()

// wait 2 seconds then bring up respawn screen canvas

{

yield return new WaitForSeconds(2);

respawnCanvas.gameObject.SetActive(true);

Time.timeScale = 0;

}

# GUI Score and Life display

This method displays live and score values to the screen, updates 5 times per second

IEnumerator UpdateText()

{

while (PlayerScript.playerLives > 0) // if player is alive, update text

{

scoreText.text = "Score: " + PlayerScript.score.ToString(); //static variables from Playerscript class.

livesText.text = "Lives: " + PlayerScript.playerLives.ToString();

yield return new WaitForSeconds(0.2f); // 5 times per second

}

}

1. [↑](#endnote-ref-1)