**CameraFollow.cs**

//Jean-Luc Hayes 11/25/15

//Used to make an object follow a target

using UnityEngine;

using System.Collections;

public class CameraFollow : MonoBehaviour

{

public Transform target;

public float smoothing = 5f;

Vector3 offset;

// Use this for initialization

void Start ()

{

offset = transform.position - target.position;

}

// Update is called once per frame

void FixedUpdate ()

{

Vector3 targetCamPos = target.position + offset;

transform.position = Vector3.Lerp(transform.position, targetCamPos, smoothing \* Time.deltaTime);

}

}

**GunPoint.cs**

//Jean-Luc Hayes 11/25/15

//used to make an object point at the mouse cursor's position

using UnityEngine;

using System.Collections;

public class GunPoint : MonoBehaviour

{

float zDistance = 10.0f;

// Use this for initialization

void Start ()

{

}

// Update is called once per frame

void Update ()

{

Vector3 mousePos = Input.mousePosition;

transform.LookAt(Camera.main.ScreenToWorldPoint(new Vector3(mousePos.x, mousePos.y, zDistance)));

}

}

**Minimap.cs**

//Jean-Luc Hayes 01/31/16

//used to make the Minimap follow the movement of the Player

using UnityEngine;

using System.Collections;

public class Minimap : MonoBehaviour

{

public GameObject player;

// Use this for initialization

void Start ()

{

}

// Update is called once per frame

void Update ()

{

transform.position = new Vector3(player.transform.position.x, transform.position.y, player.transform.position.z);

}

}

**MouseLook.cs**

//Jean-Luc Hayes and Online FPS tutorial script 11/26/15

//used to make the rotation of the camera move relative to the mouse position

using UnityEngine;

using System.Collections;

/// MouseLook rotates the transform based on the mouse delta.

/// Minimum and Maximum values can be used to constrain the possible rotation

/// To make an FPS style character:

/// - Create a capsule.

/// - Add the MouseLook script to the capsule.

/// -> Set the mouse look to use LookX. (You want to only turn character but not tilt it)

/// - Add FPSInputController script to the capsule

/// -> A CharacterMotor and a CharacterController component will be automatically added.

/// - Create a camera. Make the camera a child of the capsule. Reset it's transform.

/// - Add a MouseLook script to the camera.

/// -> Set the mouse look to use LookY. (You want the camera to tilt up and down like a head. The character already turns.)

[AddComponentMenu("Camera-Control/Mouse Look")]

public class MouseLook : MonoBehaviour {

public PlayerHealth playerHealth;

public enum RotationAxes { MouseXAndY = 0, MouseX = 1, MouseY = 2, MouseShoot = 3 }

public RotationAxes axes = RotationAxes.MouseXAndY;

public float sensitivityX = 15F;

public float sensitivityY = 15F;

public float minimumX = -360F;

public float maximumX = 360F;

public float minimumY = -60F;

public float maximumY = 60F;

float rotationY = 0F;

void Update ()

{

if (axes == RotationAxes.MouseXAndY)

{

float rotationX = transform.localEulerAngles.y + Input.GetAxis("Mouse X") \* sensitivityX;

rotationY += Input.GetAxis("Mouse Y") \* sensitivityY;

rotationY = Mathf.Clamp (rotationY, minimumY, maximumY);

transform.localEulerAngles = new Vector3(-rotationY, rotationX, 0);

}

if (axes == RotationAxes.MouseX)

{

transform.Rotate(0, Input.GetAxis("Mouse X") \* sensitivityX, 0);

}

if (axes == RotationAxes.MouseShoot)

{

transform.Rotate(Input.GetAxis("Mouse Y") \* -sensitivityY, 0, 0);

}

else

{

rotationY += Input.GetAxis("Mouse Y") \* sensitivityY;

rotationY = Mathf.Clamp(rotationY, minimumY, maximumY);

transform.localEulerAngles = new Vector3(-rotationY, transform.localEulerAngles.y, 0);

}

/\*if (playerHealth.isDead)

{

this.enabled = !playerHealth.isDead;

}\*/

}

void Start ()

{

// Make the rigid body not change rotation

if (GetComponent<Rigidbody>())

GetComponent<Rigidbody>().freezeRotation = true;

}

}

**WaypointLocate.cs**

//Jean-Luc Hayes 01/31/16

//Finds the angle between the arrow and the next waypoint

//if close enough to the waypoint, set nextwayPoint.

using UnityEngine;

using System.Collections;

public class WaypointLocate : MonoBehaviour

{

public GameObject target = null;

public static bool nextWayPoint = false;

private GameObject nextWayPoints = null;

// Use this for initialization

void Awake ()

{

}

// Update is called once per frame

void FixedUpdate ()

{

if (nextWayPoint == true)

{

float angle = signedAngle(nextWayPoints, Camera.main);

//interpret the angle change slowly.

transform.localRotation = Quaternion.Slerp(transform.localRotation, Quaternion.Euler(90, angle, 0), 0.1f);

}

if (target)

{

float angle = signedAngle(target, Camera.main);

transform.localRotation = Quaternion.Slerp(transform.localRotation, Quaternion.Euler(90, angle, 0), 0.1f);

}

}

public static float signedAngle(GameObject wayPoints, Camera camera)

{

Vector3 dist = calcDist(wayPoints, camera);

float angle = calcAngle(dist, camera);

Vector3 cross = Vector3.Cross(dist, camera.transform.forward);

if (cross.y < 0)

{

angle = -angle;

}

return angle;

}

public static float calcPoint(GameObject waypoint, Camera camera)

{

Vector3 dist = calcDist(waypoint, camera);

float angle = calcAngle(dist, camera.transform.forward);

return angle;

}

public static float calcAngle(Vector3 target, Vector3 orig)

{

return Vector3.Angle(target, orig);

}

public static float calcAngle(Vector3 target, Camera camera)

{

return Vector3.Angle(target, camera.transform.forward);

}

public static Vector3 calcDist(GameObject target, Camera orig)

{

return target.transform.position - orig.transform.position;

}

public void locate(GameObject wayPoint)

{

nextWayPoint = true;

nextWayPoints = wayPoint;

}

}

**EnemyAttack.cs**

//Jean-Luc Hayes 11/27/15

//Attack manager for an enemy

//Detects collisions and attacks if colider is a person

using UnityEngine;

using System.Collections;

public class EnemyAttack : MonoBehaviour

{

public float timeBetweenAttacks = 1.0f;

public int attackDamage = 10;

Animator anim;

GameObject player;

public bool playerInFront;

PlayerHealth playerHealth;

EnemyHealth enemyHealth;

float timer;

private bool isBoss = false;

public bool IsBoss

{

get { return isBoss; }

set{isBoss = value; }

}

public float TimeBetweenAttacks

{

get{ return timeBetweenAttacks; }

set{ timeBetweenAttacks = value; }

}

public int AttackDamage

{

get { return attackDamage; }

set { attackDamage = value; }

}

// Use this for initialization

void Awake ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerHealth>();

enemyHealth = GetComponent<EnemyHealth>();

anim = GetComponent<Animator>();

}

// Update is called once per frame

void FixedUpdate()

{

timer += Time.deltaTime;

if (timer > timeBetweenAttacks && playerInFront && enemyHealth.CurrentHealth > 0)

{

Attack();

timer = 0;

}

/\*if (LevelManager.bossBattleStarted == true)

{

timeBetweenAttacks = bossBattle.TimeBetweenAttacks;

attackDamage = bossBattle.AttackDamage;

}\*/

}

void OnTriggerEnter(Collider other)

{

if (other.gameObject == player)

{

playerInFront = true;

anim.SetBool("PlayerInFront", playerInFront);

}

}

void OnTriggerExit(Collider other)

{

if (other.gameObject == player)

{

playerInFront = false;

anim.SetBool("PlayerInFront", playerInFront);

}

}

void Attack()

{

if (playerHealth.CurrentHealth > 0)

{

playerHealth.takeDamage(attackDamage);

}

}

}

**EnemyHealth.cs**

//Jean-Luc Hayes 11/28/15

//Health for an enemy

//Controls death and damage given to the enemy

using UnityEngine;

using System.Collections;

public class EnemyHealth : MonoBehaviour

{

public int startingHealth = 25;

public int currentHealth;

public float sinkSpeed = 2.5f;

public float sinkOffset = 10f;

public int scoreValue = 10;

Animator anim;

private bool isDead;

bool isSinking;

float timer;

public bool isBoss = false;

public bool IsBoss

{

get { return isBoss; }

set { isBoss = value; }

}

public bool IsDead

{

get{return isDead;}

set{isDead = value;}

}

public int StartingHealth

{

get { return startingHealth; }

set { startingHealth = value; }

}

public int CurrentHealth

{

get { return currentHealth; }

set { currentHealth = value; }

}

public int ScoreValue

{

get { return scoreValue; }

set { scoreValue = value; }

}

public float SinkSpeed

{

get { return sinkSpeed; }

set { sinkSpeed = value; }

}

public float SinkOffset

{

get { return sinkOffset; }

set { sinkOffset = value; }

}

void Awake ()

{

anim = GetComponent<Animator>();

currentHealth = startingHealth;

}

// Update is called once per frame

void Update ()

{

timer += Time.deltaTime;

if (isSinking && timer >= sinkOffset)

{

Sink();

}

}

public void takeDamage(int amount, Vector3 hitPoint)

{

if (isDead)

return;

currentHealth -= amount;

if (currentHealth <= 0)

{

Death();

}

}

public void Death()

{

isDead = true;

anim.SetTrigger("Die");

StartSinking();

}

public void StartSinking()

{

GetComponent<NavMeshAgent>().enabled = false;

GetComponent<Rigidbody>().isKinematic = true;

isSinking = true;

ScoreManager.score += scoreValue;

}

public void Sink()

{

if(isBoss)

{

transform.Translate(-Vector3.up \* sinkSpeed \* Time.deltaTime);

}

else

{

transform.Translate(-Vector3.up \* sinkSpeed \* Time.deltaTime);

Destroy(gameObject, 2f);

}

}

}

**EnemyMovement.cs**

//Jean-Luc Hayes 11/28/15

//Controls the movement of an enemy

//Sets destination is the collision is a player

using UnityEngine;

using System.Collections;

public class EnemyMovement : MonoBehaviour

{

GameObject player;

PlayerHealth playerHealth;

EnemyHealth enemyHealth;

public NavMeshAgent nav;

Animator anim;

public bool isBoss = false;

public bool IsBoss

{

get { return isBoss; }

set { isBoss = value; }

}

// Use this for initialization

void Awake ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

enemyHealth = GetComponent<EnemyHealth>();

nav = GetComponent<NavMeshAgent>();

anim = GetComponent<Animator>();

}

// Update is called once per frame

void Update ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

enemyHealth = GetComponent<EnemyHealth>();

if (player != null)

{

if (enemyHealth.CurrentHealth > 0 && playerHealth.CurrentHealth > 0 && isBoss == false)

{

nav.SetDestination(player.transform.position);

anim.SetBool("PlayerNoticed", true);

}

else

{

anim.SetBool("PlayerNoticed", true);

}

}

}

public void setNavPoint(GameObject target)

{

nav.SetDestination(target.transform.position);

}

public void setNavPoint(Transform target)

{

nav.SetDestination(target.position);

}

}

**BossBattle.cs**

//Jean-Luc Hayes 02/02/16

//Manager of a Boss Battle

//uses previous classes that were put on an enemy

using UnityEngine;

using System.Collections;

public class BossBattle : MonoBehaviour

{

public GameObject boss;

private EnemyMovement enemyMovement;

private EnemyHealth enemyHealth;

private GameObject player;

private PlayerHealth playerHealth;

private LevelManager levelManager;

float timer;

public Transform bossPoint;

public Transform[] minions = new Transform[5];

public int attackDamage = 20;

public float timeBetweenAttacks = 10f;

public int startingHealth = 200;

public float sinkSpeed = 1.0f;

public float sinkOffset = 12f;

public int scoreValue = 150;

private bool isBoss = true;

private bool isDead = false;

public bool IsDead

{

get { return isDead; }

set { isDead = value; }

}

public float TimeBetweenAttacks

{

get { return timeBetweenAttacks; }

set { timeBetweenAttacks = value; }

}

public int AttackDamage

{

get { return attackDamage; }

set { attackDamage = value; }

}

public bool IsBoss

{

get { return isBoss; }

set { isBoss = value; }

}

public int StartingHealth

{

get { return startingHealth; }

set { startingHealth = value; }

}

public int ScoreValue

{

get { return scoreValue; }

set { scoreValue = value; }

}

public float SinkSpeed

{

get { return sinkSpeed; }

set { sinkSpeed = value; }

}

public float SinkOffset

{

get { return sinkOffset; }

set { sinkOffset = value; }

}

// Use this for initialization

void Awake()

{

GameObject levelman = GameObject.FindGameObjectWithTag("LevelManager");

levelManager = levelman.GetComponent<LevelManager>();

enemyMovement = GetComponent<EnemyMovement>();

enemyHealth = GetComponent<EnemyHealth>();

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

InvokeRepeating("startMovement", 0, timeBetweenAttacks \* 2);

//setToBoss();

}

// Update is called once per frame

void Update ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

timer += Time.deltaTime;

if(LevelManager.bossBattleStarted == true)

{

if (enemyHealth.IsDead)

{

LevelManager.bossBattleStarted = false;

Death();

}

}

}

void Death()

{

isDead = true;

levelManager.isDead();

}

public void bossBattleStart()

{

LevelManager.bossBattleStarted = true;

}

void startMovement()

{

if (player != null)

{

if (playerHealth.CurrentHealth <= 0f || LevelManager.bossBattleStarted == false)

{

return;

}

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

{

int k = Random.Range(0, minions.Length);

enemyMovement.nav.SetDestination(minions[k].position);

}

}

}

}

**GameManagerDeath.cs**

//Jean-Luc Hayes 02/01/16

//The Manager of the UI's for Death instantiation

// Managers entire scene's Death screens

using UnityEngine;

using System.Collections;

public class GameManagerDeath : MonoBehaviour

{

public GameObject player;

public PlayerHealth playerHealth;

public Canvas canvas;

public UnityEngine.UI.Text lives;

public UnityEngine.UI.Text deathTextText;

public UnityEngine.UI.Text gameOver;

public UnityEngine.UI.Button startGame;

public UnityEngine.UI.Button exitGame;

public UnityEngine.UI.Button reentryGame;

public UnityEngine.UI.Text winGame;

// Use this for initialization

void Awake()

{

}

// Update is called once per frame

void Update()

{

lives.text = "Lives: " + playerHealth.CurrentNumLives;

}

public void deathScreenIntial()

{

deathText();

UnityEngine.UI.Button reentryLevels = Instantiate(reentryGame, reentryGame.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Button;

reentryLevels.transform.SetParent(canvas.transform, false);

}

public void deathText()

{

UnityEngine.UI.Text deathTexts = Instantiate(deathTextText, deathTextText.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Text;

deathTexts.transform.SetParent(canvas.transform, false);

Invoke("reentryLevel", 5f);

}

public void wonText()

{

UnityEngine.UI.Text winGames = Instantiate(winGame, winGame.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Text;

winGames.transform.SetParent(canvas.transform, false);

Invoke("endGame", 5f);

}

public void gameOverText()

{

UnityEngine.UI.Text gameOvers = Instantiate(gameOver, gameOver.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Text;

gameOvers.transform.SetParent(canvas.transform, false);

Invoke("endGame", 5f);

}

public void deathScreenFinal()

{

deathText();

UnityEngine.UI.Button startGames = Instantiate(startGame, startGame.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Button;

startGames.transform.SetParent(GameObject.Find("Canvas").transform, false);

startGames.enabled = true;

UnityEngine.UI.Button exitGames = Instantiate(exitGame, exitGame.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Button;

exitGames.transform.SetParent(GameObject.Find("Canvas").transform, false);

exitGames.enabled = true;

}

public void reentryLevel()

{

LevelManager.reentryLevel = true;

}

public void endGame()

{

LevelManager.endGame = true;

}

}

**GameManagerPlay.cs**

//Jean-Luc Hayes 01/30/16

//Game Manager Play controls aspects of gameplay integral to gameplay

//Spawns minions and controls spawnTime

using UnityEngine;

using System.Collections;

public class GameManagerPlay : MonoBehaviour

{

public int waves = 1;

UnityEngine.UI.Text wave;

public static float timeLeft = 60;

public GameObject player;

public PlayerHealth playerHealth;

public GameObject enemy;

public float spawnTime = 10f;

public Transform[] spawnPoints;

private Transform[] spawnPointsClose;

// Use this for initialization

void Awake ()

{

player = GameObject.FindGameObjectWithTag("Player");

spawnPointsClose = new Transform[4];

//wave = GetComponent<UnityEngine.UI.Text>();

//waves = 1;

}

void FixedUpdate()

{

//wave.text = "Wave: " + waves;

/\*timeLeft -= Time.deltaTime;

WaveSelector();

if (timeLeft <= 0)

{

waves++;

timeLeft = 60;

}\*/

player = GameObject.FindGameObjectWithTag("Player");

if (Time.deltaTime > 60f || Time.deltaTime > 120f || Time.deltaTime > 240 || Time.deltaTime > 480)

{

spawnTime = spawnTime \* Mathf.Exp(-0.111f);

}

if (Input.GetKeyDown(KeyCode.Escape))

{

Application.Quit();

}

}

void Start ()

{

InvokeRepeating("Spawn", 0, spawnTime);

}

/\*void WaveSelector()

{

float[] spawnTimes = new float[100];

for (int i = 0; i < 100; i++)

{

float start = 5f;

spawnTimes[i] = (start / 2.0f);

if (waves == i)

{

spawnTime = spawnTimes[i];

}

}

}\*/

void Spawn()

{

if (playerHealth.CurrentHealth <= 0f || LevelManager.bossBattleStarted == true)

{

return;

}

int j = 0;

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

{

if (WayPoints.closeEnough(player, spawnPoints[i]) <= 125f && j != 4)

{

spawnPointsClose[j] = spawnPoints[i];

j++;

}

}

int k = Random.Range(0, spawnPointsClose.Length);

Instantiate(enemy, spawnPointsClose[k].position, spawnPointsClose[k].rotation);

}

}

**LevelManager.cs**

//Jean-Luc Hayes 02/01/16

//Manager for the whole Game - Managers the Loading of the Levels

//also resets Player's position if died

using UnityEngine;

using System.Collections;

public class LevelManager : MonoBehaviour

{

private GameObject player;

private PlayerHealth playerHealth;

private PlayerMovement playerMovement;

private GameObject deathManagerObject;

private GameManagerDeath deathManager;

private GameObject boss;

private BossBattle bossBattle;

private GameObject wayPointObject;

private WayPoints wayPoints;

public Canvas canvas;

public static bool reentryLevel = false;

public static bool endGame = false;

public static bool bossBattleStarted = false;

public static int wayPoint = 0;

public static Vector3 deathPoint = new Vector3(529, 0, 310);

public static int numLives = 3;

public static int score = 0;

// Use this for initialization

void Awake()

{

DontDestroyOnLoad(this);

if (GameObject.FindGameObjectsWithTag(gameObject.tag).Length > 1)

{

Destroy(gameObject);

}

setUpLinks();

}

void setUpLinks()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

playerMovement = player.GetComponent<PlayerMovement>();

boss = GameObject.FindGameObjectWithTag("Boss");

bossBattle = boss.GetComponent<BossBattle>();

deathManagerObject = GameObject.FindGameObjectWithTag("Respawn");

deathManager = deathManagerObject.GetComponent<GameManagerDeath>();

wayPointObject = GameObject.FindGameObjectWithTag("WayPoint");

wayPoints = wayPointObject.GetComponent<WayPoints>();

}

// Update is called once per frame

void Update()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

playerMovement = player.GetComponent<PlayerMovement>();

boss = GameObject.FindGameObjectWithTag("Boss");

bossBattle = boss.GetComponent<BossBattle>();

deathManagerObject = GameObject.FindGameObjectWithTag("Respawn");

deathManager = deathManagerObject.GetComponent<GameManagerDeath>();

wayPointObject = GameObject.FindGameObjectWithTag("WayPoint");

wayPoints = wayPointObject.GetComponent<WayPoints>();

if (wayPoints.getLastWayPoint() == true)

{

bossBattle.bossBattleStart();

wayPoints.setLastWayPoint(false);

}

if (reentryLevel == true)

{

Application.LoadLevel(Application.loadedLevel);

Invoke("SetPreviousLife", 0.5f);

reentryLevel = false;

}

if (endGame == true)

{

Application.LoadLevel("Level 0");

endGame = false;

Destroy(this);

}

}

public void SetPreviousLife()

{

playerHealth.setPreviousLife();

playerMovement.setPreviousLife();

wayPoints.setPreviousLife();

}

void checkPosition()

{

deathPoint = player.transform.position;

numLives = playerHealth.CurrentNumLives;

score = ScoreManager.score;

wayPoint = wayPoints.wayPoint;

}

public void isDead()

{

if (playerHealth.IsDead == true && playerHealth.CurrentNumLives != 0 && !(player.transform.position.y < -5))

{

checkPosition();

deathManager.deathText();

}

else if (playerHealth.IsDead == true && playerHealth.CurrentNumLives != 0 && (player.transform.position.y < -5))

{

deathManager.deathText();

}

else if (playerHealth.IsDead == true && playerHealth.CurrentNumLives == 0)

{

deathManager.gameOverText();

}

else if (playerHealth.IsDead == false && bossBattle.IsDead)

{

deathManager.wonText();

}

}

}

**MinionManager.cs**

//Jean-Luc Hayes 02/02/16

//Controls the Spawning of the Boss's Minions

//Done every 20 seconds, or the variable minionTime seconds

using UnityEngine;

using System.Collections;

public class MinionManager : MonoBehaviour

{

public float minionTime = 20f;

private GameObject player;

private PlayerHealth playerHealth;

public GameObject enemy;

private GameObject boss;

private Transform[] minions = new Transform[5];

// Use this for initialization

void Start ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

boss = GameObject.FindGameObjectWithTag("Boss");

minions = boss.GetComponent<BossBattle>().minions;

InvokeRepeating("spawnMinions", 0, minionTime);

}

// Update is called once per frame

void Update ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

}

public void spawnMinions()

{

if (player != null)

{

if (playerHealth.CurrentHealth <= 0f || LevelManager.bossBattleStarted == false)

{

return;

}

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

{

int k = Random.Range(0, minions.Length);

Instantiate(enemy, minions[k].transform.position, minions[k].transform.rotation);

}

}

}

}

**ScoreManager.cs**

//Jean-Luc Hayes 11/29/16

//Controls the Score of the game through a static variable

using UnityEngine;

using System.Collections;

public class ScoreManager : MonoBehaviour

{

public static int score;

UnityEngine.UI.Text text;

// Use this for initialization

void Awake ()

{

text = GetComponent<UnityEngine.UI.Text>();

score = 0;

}

// Update is called once per frame

void FixedUpdate ()

{

text.text = "Score: " + score;

}

}

**WayPoints.cs**

//Jean-Luc Hayes 02/02/16

//WayPoint Manager

//Makes sure that the Waypoints are the current one when a player dies

using UnityEngine;

using System.Collections;

public class WayPoints : MonoBehaviour

{

public GameObject[] wayPoints = new GameObject[14];

public WaypointLocate wayPointLocate;

private GameObject player;

private PlayerHealth playerHealth;

private bool lastWayPoint = false;

public Canvas canvas;

public UnityEngine.UI.Text hitWayPoint;

public int wayPoint = 0;

private float minDist;

public bool getLastWayPoint()

{

return lastWayPoint;

}

public void setLastWayPoint(bool way)

{

lastWayPoint = way;

}

// Use this for initialization

void Awake ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

}

// Update is called once per frame

void Update ()

{

player = GameObject.FindGameObjectWithTag("Player");

playerHealth = player.GetComponent<PlayerHealth>();

if (player != null)

{

if (playerHealth.CurrentHealth > 0 && wayPoint < wayPoints.Length)

{

wayPointLocate.locate(wayPoints[wayPoint]);

if (closeEnough(player, wayPoints[wayPoint]) <= 8f)

{

UnityEngine.UI.Text hitWayPoints = Instantiate(hitWayPoint, hitWayPoint.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Text;

hitWayPoints.transform.SetParent(canvas.transform, false);

Destroy(hitWayPoints, 1f);

wayPoint++;

}

}

if (wayPoint == wayPoints.Length)

{

lastWayPoint = true;

}

}

}

public void setPreviousLife()

{

wayPoint = LevelManager.wayPoint;

}

public static float closeEnough(GameObject player, GameObject wayPoints)

{

return Vector3.Distance(player.transform.position, wayPoints.transform.position);

}

public static float closeEnough(GameObject player, Transform wayPoints)

{

return Vector3.Distance(player.transform.position, wayPoints.transform.position);

}

}

**PlayerHealth.cs**

//Jean-Luc Hayes 11/26/15

//Managers Health of PlayerHealth

//Also will call neccessary UI

using UnityEngine;

using System.Collections;

public class PlayerHealth : MonoBehaviour

{

private LevelManager levelManager;

private int startingHealth = 100;

private int currentHealth;

public GameObject health;

Animation anim;

PlayerMovement playerMovement;

PlayerShooting playerShooting;

private bool isDead;

private int startingNumLives = 3;

private int currentNumLives;

public AudioClip[] sounds = new AudioClip[2];

public int StartingHealth

{

get { return startingHealth; }

set { startingHealth = value; }

}

public int CurrentHealth

{

get { return currentHealth; }

set { currentHealth = value; }

}

public bool IsDead

{

get { return isDead; }

set { isDead = value; }

}

public int CurrentNumLives

{

get { return currentNumLives; }

set { currentNumLives = value; }

}

// Use this for initialization

void Awake()

{

anim = GetComponent<Animation>();

playerMovement = GetComponent<PlayerMovement>();

playerShooting = GetComponentInChildren<PlayerShooting>();

currentHealth = startingHealth;

currentNumLives = startingNumLives;

levelManager = GameObject.FindGameObjectWithTag("LevelManager").GetComponent<LevelManager>();

}

void Start()

{

}

// Update is called once per frame

void FixedUpdate()

{

health.GetComponentInChildren<UnityEngine.UI.Slider>().value = currentHealth;

if (transform.position.y < -5)

{

deathBelow();

}

}

public void setPreviousLife()

{

currentNumLives = LevelManager.numLives;

}

public void takeDamage(int amount)

{

//damaged();

currentHealth -= amount;

if (currentHealth <= 0 && !isDead)

{

Death();

}

}

/\*void damaged()

{

AudioSource.PlayClipAtPoint(sounds[0], transform.position);

}\*/

void deathBelow()

{

isDead = true;

levelManager.isDead();

playerShooting.DisableEffects(!isDead);

anim.CrossFade("soldierDieBack");

playerMovement.enabled = !isDead;

playerShooting.enabled = !isDead;

}

void Death()

{

isDead = true;

currentNumLives -= 1;

levelManager.isDead();

//AudioSource.PlayClipAtPoint(sounds[1], transform.position);

playerShooting.DisableEffects(!isDead);

anim.CrossFade("soldierDieFront");

playerMovement.enabled = !isDead;

playerShooting.enabled = !isDead;

}

public void rebirth()

{

isDead = false;

//gameResetManager.isDead();

currentHealth = startingHealth;

playerShooting.DisableEffects(!isDead);

anim.CrossFade("soldierIdle");

playerMovement.enabled = !isDead;

playerShooting.enabled = !isDead;

}

}

**PlayerMovement.cs**

//Jean-Luc Hayes 11/28/15

//Controls the movement of Player

//many booleans to detect the state of the animation of the player

using UnityEngine;

using System.Collections;

public class PlayerMovement : MonoBehaviour

{

//public static bool inFront = false;

private float walkingSpeed = 8f;

private float runningSpeed = 12f;

private float rotateSpeed = 5;

public float fadeTime = 0.5f;

private float timer;

private bool walking;

private bool running;

private bool standing;

private bool strafeRight;

private bool strafeLeft;

private bool jumping;

public AudioClip[] sounds = new AudioClip[2];

Animation anim;

//Inizilation of Game

void Awake()

{

anim = GetComponent<Animation>();

//checkPosition();

}

// Update is not called once per frame, but evenly for all frame rates

void FixedUpdate()

{

//timer = 0;

timer += Time.deltaTime;

float h = Input.GetAxisRaw("Horizontal");

float v = Input.GetAxisRaw("Vertical");

walking = h != 0f || v != 0f;

standing = !walking;

strafeRight = h > 0f;

strafeLeft = h < 0f;

running = Input.GetKey(KeyCode.LeftShift);

standing = Input.GetMouseButton(0);

//Turn();

Move(h, v);

Animating(h, v);

}

public void setPreviousLife()

{

transform.position = LevelManager.deathPoint;

ScoreManager.score = LevelManager.score;

}

void Move(float h, float v)

{

if (walking)

{

moveFoward(walkingSpeed, h, v);

}

if (running)

{

moveFoward(runningSpeed, h, v);

}

}

void moveFoward(float speed, float h, float v)

{

//InvokeRepeating("footsteps", 0, 1f);

transform.position += transform.forward \* v \* speed \* Time.deltaTime;

transform.position += transform.right \* h \* speed \* Time.deltaTime;

}

/\*void gunshot()

{

AudioSource.PlayClipAtPoint(sounds[1], transform.position);

}\*/

/\*void footsteps()

{

AudioSource.PlayClipAtPoint(sounds[0], transform.position);

}\*/

void moveUpward(float height)

{

transform.position += transform.up \* height;

}

void Animating(float h, float v)

{

anim.CrossFade("soldierIdleRelaxed", fadeTime);

if (walking)

{

anim.CrossFade("soldierWalk", fadeTime);

}

if (standing)

{

//InvokeRepeating("gunshot", 0, 1f);

anim.CrossFade("soldierIdle", fadeTime);

}

if (running)

{

anim.CrossFade("soldierSprint", fadeTime);

}

if (strafeRight)

{

anim.CrossFade("soldierStrafeRight", fadeTime);

}

if (strafeLeft)

{

anim.CrossFade("soldierStrafeLeft", fadeTime);

}

}

void Turn()

{

var ray = Camera.main.ScreenPointToRay(Input.mousePosition);

float hitdist = 0.0f; ;

Plane playerPlane = new Plane(Vector3.up, transform.position);

if (playerPlane.Raycast(ray, out hitdist))

{

Vector3 targetPoint = ray.GetPoint(hitdist);

Quaternion targetRotation = Quaternion.LookRotation(targetPoint - transform.position);

transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, rotateSpeed \* Time.deltaTime);

}

}

}

**PlayerShooting.cs**

//Jean-Luc Hayes 11/29/15

//Managers the player's shooting abilities

//If dead - disables shooting effects

using UnityEngine;

using System.Collections;

public class PlayerShooting : MonoBehaviour

{

private int damagePerShot = 5;

private float timeBetweenBullets = 0.15f;

private float range = 100f;

private float effectsDisplayTime = 0.2f;

float timer;

Ray shootRay;

RaycastHit shootHit;

ParticleSystem gunParticles;

LineRenderer gunLine;

Light gunLight;

// Use this for initialization

void Awake ()

{

gunParticles = GetComponent<ParticleSystem>();

gunLine = GetComponentInChildren<LineRenderer>();

gunLight = GetComponentInChildren<Light>();

}

// Update is called once per frame

void Update ()

{

timer += Time.deltaTime;

if (Input.GetButton("Fire1") && timer >= timeBetweenBullets)

{

Shoot();

}

if (timer >= timeBetweenBullets \* effectsDisplayTime)

{

DisableEffects(false);

}

}

public void DisableEffects(bool effects)

{

gunLine.enabled = effects;

gunLight.enabled = effects;

}

void Shoot()

{

timer = 0f;

gunLight.enabled = true;

gunParticles.Stop();

gunParticles.Play();

gunLine.enabled = true;

gunLine.SetPosition(0, transform.position);

shootRay.origin = transform.position;

shootRay.direction = transform.forward;

if (Physics.Raycast(shootRay, out shootHit, range))

{

EnemyHealth enemyHealth = shootHit.collider.GetComponent<EnemyHealth>();

if (enemyHealth != null)

{

enemyHealth.takeDamage(damagePerShot, shootHit.point);

}

gunLine.SetPosition(1, shootHit.point);

}

else

{

gunLine.SetPosition(1, shootRay.origin + shootRay.direction \* range);

}

}

}

**DepthUI.cs**

//Jean-Luc Hayes 02/01/16

//Online tutorial - //blog.manapebbles.com/world-space-overlay-camera-in-unity

//used to define the length of an enemy from the screen

using UnityEngine;

using System.Collections;

public class DepthUI : MonoBehaviour

{

public float depth;

// Use this for initialization

void Awake ()

{

}

// Update is called once per frame

void Update ()

{

}

}

**EnemyUI.cs**

//01/31/16 Jean-Luc Hayes - Some Help from Online -

//blog.manapebbles.com/world-space-overlay-camera-in-unity

//Enemy UI compares the game state to the enemy state to

//draw the enemyHealthBar above the enemy

using UnityEngine;

using System.Collections;

public class EnemyUI : MonoBehaviour

{

private float maxDrawDistance = 100f;

private EnemyHealth enemyHealth;

private GameObject player;

private GameObject canvasObject;

public Canvas canvas;

public GameObject healthPrefab;

private Vector3 worldPos;

private GameObject wayPointObject;

private WayPoints wayPoints;

public AudioClip[] fight = new AudioClip[2];

public bool isBoss = false;

private DepthUI depthUI;

public float panelOffset = 5f;

private GameObject healthPanel;

private UnityEngine.UI.Slider healthSlider;

// Use this for initialization

void Awake()

{

player = GameObject.FindGameObjectWithTag("Player");

canvasObject = GameObject.FindGameObjectWithTag("Canvas");

canvas = canvasObject.GetComponent<Canvas>();

enemyHealth = GetComponent<EnemyHealth>();

healthPanel = Instantiate(healthPrefab) as GameObject;

healthPanel.transform.SetParent(canvas.transform, false);

healthSlider = healthPanel.GetComponentInChildren<UnityEngine.UI.Slider>();

depthUI = healthPanel.GetComponent<DepthUI>();

canvas.GetComponent<ScreenSpaceCanvas>().AddToCanvas(healthPanel);

wayPointObject = GameObject.FindGameObjectWithTag("WayPoint");

wayPoints = wayPointObject.GetComponent<WayPoints>();

}

// Update is called once per frame

void Update()

{

if (!enemyHealth.IsDead)

{

float distance = (worldPos - Camera.main.transform.position).magnitude;

depthUI.depth = -distance;

healthSlider.value = enemyHealth.CurrentHealth;

worldPos = new Vector3(transform.position.x, transform.position.y + panelOffset, transform.position.z);

Vector3 screenPos = Camera.main.WorldToScreenPoint(worldPos);

healthPanel.transform.position = new Vector3(screenPos.x, screenPos.y, screenPos.z);

Vector3 dir = Camera.main.WorldToViewportPoint(transform.position);

//bool onScreen = dir.z > 0 && dir.x > 0 && dir.x < 1 && dir.y > 0 && dir.y < 1;

healthPanel.SetActive(true);

if (!(dir.z > 0 && dir.x > 0 && dir.x < 1 && dir.y > 0 && dir.y < 1) || Vector3.Distance(transform.position, player.transform.position) > maxDrawDistance)

{

healthPanel.SetActive(false);

}

else if (isBoss && LevelManager.bossBattleStarted == false)

{

healthPanel.SetActive(false);

}

else if (isBoss && LevelManager.bossBattleStarted == true)

{

healthPanel.SetActive(true);

//AudioSource.PlayClipAtPoint(fight[1], transform.position);

//AudioSource.PlayClipAtPoint(fight[2], transform.position);

wayPoints.setLastWayPoint(false);

}

}

else if(enemyHealth.IsDead)

{

Destroy(healthPanel);

}

}

}

**ExitButton.cs**

//Jean-Luc Hayes 11/30/16

//Exit button for the main menu

using UnityEngine;

using System.Collections;

using System;

using UnityEngine.EventSystems;

public class ExitButton : MonoBehaviour

{

// Use this for initialization

void Start ()

{

}

// Update is called once per frame

void FixedUpdate ()

{

}

public void exitGame()

{

Application.Quit();

}

}

**ExitInstruct.cs**

//Jean-Luc Hayes 02/02/16

//Exit instructions main menu button

using UnityEngine;

using System.Collections;

public class ExitInstruct : MonoBehaviour

{

public UnityEngine.UI.Image instruct;

public UnityEngine.UI.Image instructB;

private GameObject[] pieces;

private GameObject[] piecesB;

private GameObject[] piecesBu;

// Use this for initialization

void Awake ()

{

}

// Update is called once per frame

void Update ()

{

pieces = GameObject.FindGameObjectsWithTag("instruct");

piecesB = GameObject.FindGameObjectsWithTag("instructB");

piecesBu = GameObject.FindGameObjectsWithTag("instructBu");

}

public void exitInstruct()

{

Destroy(this);

Destroy(pieces[1]);

Destroy(piecesB[1]);

}

}

**InstructButton.cs**

//Jean-Luc Hayes 02/02/16

//Main Menu Instructions button

//instantiates the Instructions UI

using UnityEngine;

using System.Collections;

public class InstructButton : MonoBehaviour

{

public Canvas canvas;

public UnityEngine.UI.Image instruct;

public UnityEngine.UI.Button instructExit;

public UnityEngine.UI.Button exit;

public UnityEngine.UI.Image background;

// Use this for initialization

void Awake ()

{

}

// Update is called once per frame

void Update ()

{

}

public void instructions()

{

UnityEngine.UI.Image backgrounds = Instantiate(background, background.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Image;

backgrounds.transform.SetParent(canvas.transform, false);

UnityEngine.UI.Image instructs = Instantiate(instruct, instruct.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Image;

instructs.transform.SetParent(canvas.transform, false);

UnityEngine.UI.Button instructExits = Instantiate(instructExit, instructExit.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Button;

instructExits.transform.SetParent(canvas.transform, false);

/\*UnityEngine.UI.Button exits = Instantiate(exit, exit.transform.position, Quaternion.Euler(Vector3.zero)) as UnityEngine.UI.Button;

exits.transform.SetParent(canvas.transform, false);\*/

}

}

**ReentryLevelButton.cs**

//Jean-Luc Hayes 01/31/15

//Button to reentry the level

using UnityEngine;

using System.Collections;

using System;

using UnityEngine.EventSystems;

public class ReentryLevelButton : MonoBehaviour

{

private UnityEngine.UI.Button myButton;

// Use this for initialization

void Start()

{

myButton = GetComponent<UnityEngine.UI.Button>();

myButton.onClick.AddListener(() => reentryLevel());

}

// Update is called once per frame

void FixedUpdate()

{

}

public void reentryLevel()

{

LevelManager.reentryLevel = true;

}

}

**ScreenSpaceCanvas.cs**

//Jean-Luc Hayes 02/01/16

////blog.manapebbles.com/world-space-overlay-camera-in-unity

//Sorts the depth of the enemies from the screen with compare function

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

public class ScreenSpaceCanvas : MonoBehaviour

{

List<DepthUI> panels = new List<DepthUI>();

// Use this for initialization

void Awake ()

{

panels.Clear();

}

// Update is called once per frame

void Update ()

{

Sort();

}

public void AddToCanvas(GameObject objects)

{

panels.Add(objects.GetComponent<DepthUI>());

}

void Sort()

{

panels.Sort((x, y) => x.depth.CompareTo(y.depth));

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

{

panels[i].transform.SetSiblingIndex(i);

}

}

}

**StartButton.cs**

//Jean-Luc Hayes 01/30/16

//Button to enter the level

using UnityEngine;

using System.Collections;

using System;

using UnityEngine.EventSystems;

public class StartButton : MonoBehaviour

{

// Use this for initialization

void Start ()

{

}

// Update is called once per frame

void FixedUpdate()

{

}

public void startGame()

{

Application.LoadLevel("Level 1");

}

}

**WaveSetup.cs**

//Jean-Luc Hayes 01/30/16

//Controls Wave Time text

//Depreciated as I removed the wave UI

using UnityEngine;

using System.Collections;

public class UIManager : MonoBehaviour

{

UnityEngine.UI.Text waveTime;

// Use this for initialization

void Awake ()

{

waveTime = GetComponent<UnityEngine.UI.Text>();

}

// Update is called once per frame

void FixedUpdate ()

{

waveTime.text = "Time Till Next Wave: " + GameManagerPlay.timeLeft;

}

}