**Introduction:**

**Simple Description of the game:**

The main idea of the game is to protect the territory of the player from enemies. The game contains three levels of different hardness. There are two kinds of soldiers in level one, three kinds of soldiers in level two and four kinds of soldiers in level three. The player can choose any kind of soldiers at a cost and place them in the scene to confront the enemies throwing different objects like stones. The player can put multiple soldiers until he/she costs all of the points. If the game scene time is over before any enemy entering the player border, the player wins. Else if any of the enemy enters the player territory, the player loses the game. There are multiple kinds of enemies too, with different capabilities. The fox can jump over the soldiers and move fast towards the player area to win it. Level one is the easiest level and level three is the hardest level of the game. The player can also change the game hardness or the game sound from the options.

**Tools:**

**About Unity:**

There are some sections in the unity opening window,

**The scene section**: the texts, pictures, different objects, etc. of the game are put and edited here,

**The game section**: this section shows the output for the inputs in the other four sections,

**The project section**: the fodder structure is shown here and it can be edited, any file can be added, deleted and some other file operations,

**The hierarchy section**: to show the scenes and edit any of them, like editing the music, texts, pictures, adding objects, attaching scripts to a scene, etc. are done here.

**The inspector section**: it shows all the properties of any component object, like transformations, camera views, audio-video settings, sound settings, event setting, I/O settings of a component, graphics, any a lot of other properties.

**Asset store:**

**Animator section:**

**Console section:**

**Note: And describe other properties that are frequently used to make a game within 2 pages. I wrote some description you write more from net.**

**Implementation:**

Opening the unity window with the Glitch Garden project folder the following screen show up. This is the working window where we built our game. We already described how it works above.

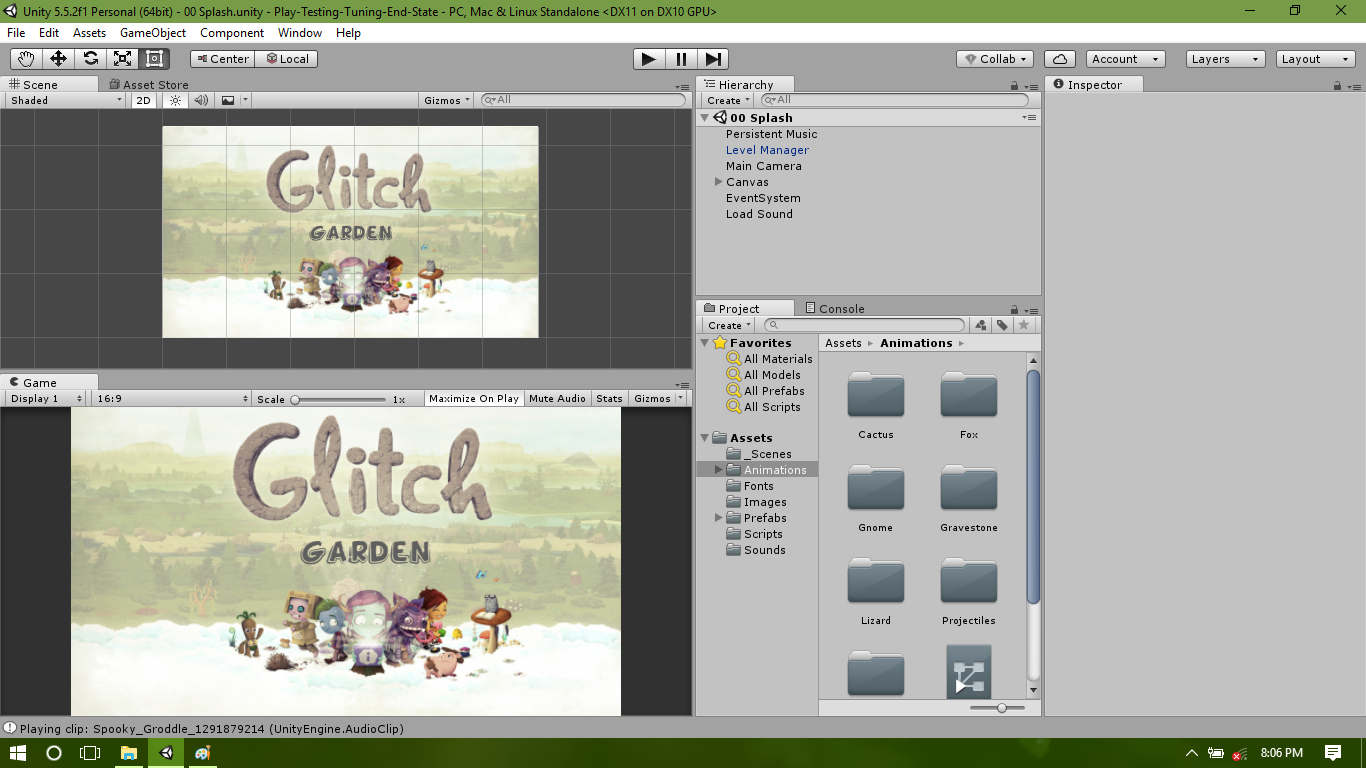


Figure 1: Unity project window

This unity project window becomes reddish as the project play button is pressed and the game window starts the game.



Figure 2: The game splash window

This the start window of the game. Here are 3 options to play game, options and quit. If the play button (orange circled) is pressed, the game starts at level one.

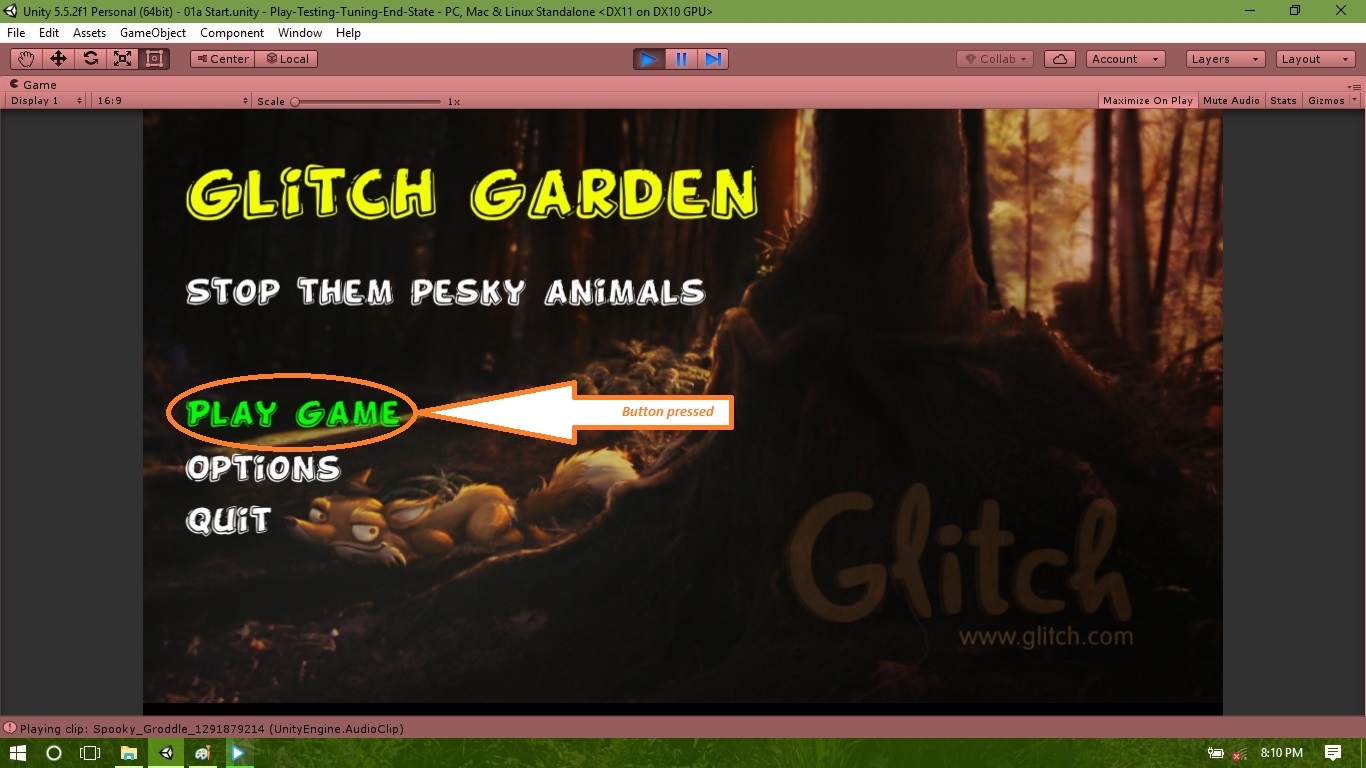


Figure 3: Game start screen

The following is the game window for level one. Two enemies enters in the game scene.



Figure 4: Level one game scene with enemies

The player puts his/her soldiers to confront the enemies in the battle field.



Figure 5: Level one game scene with soldiers fighting enemies

Soldiers fighting and throwing stones towards enemies to protect the player territory.



Figure 6: Level one game scene with soldiers throwing stone towards enemies

The player loses the game, since an enemy enters (orange marked) the player’s area in the game scene.



Figure 7: Player game losing scene

As the player loses the game he/she then taken to the play again screen. If the player presses the start again button, the game starts from the level one again.

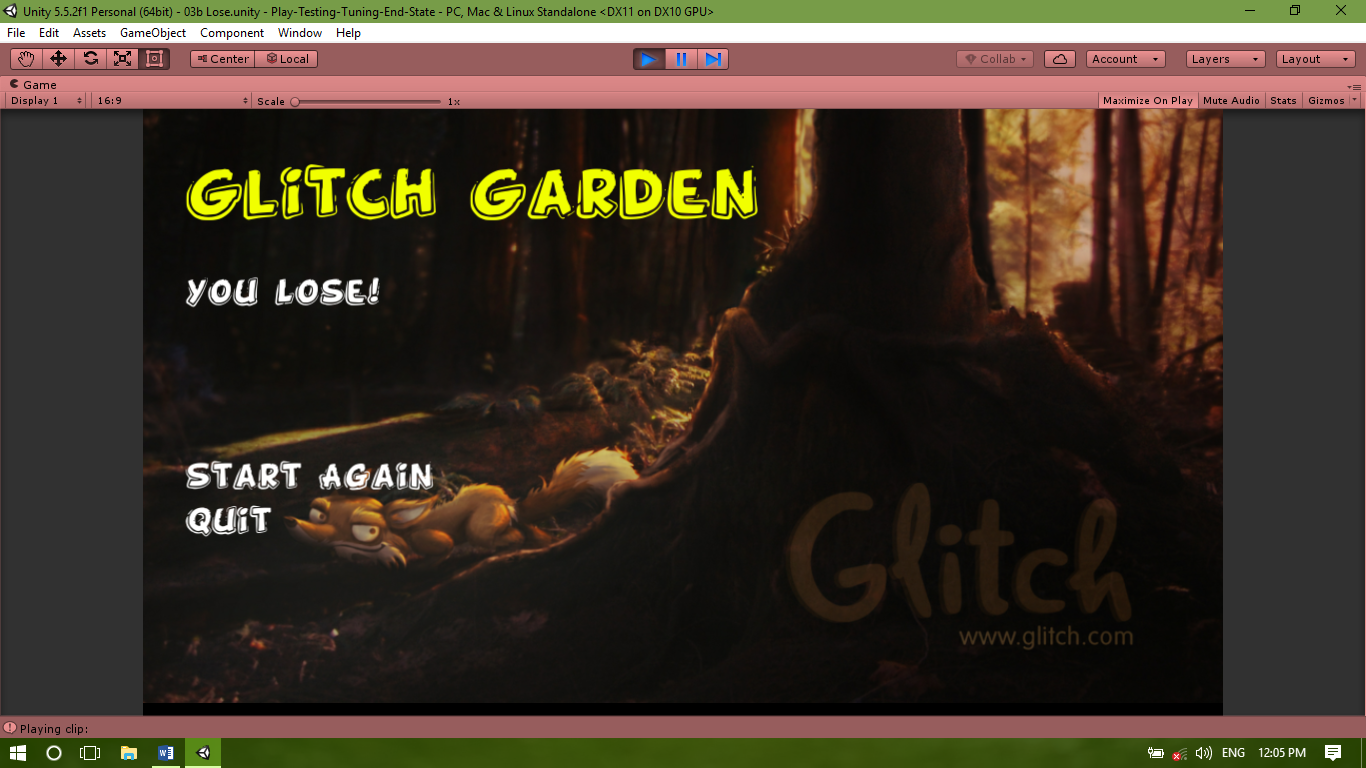


Figure 8: The game start again screen

As the player plays the game again and no enemy enters the player’s area within the game scene play time, the player wins this time.



Figure 9: The game winning scene

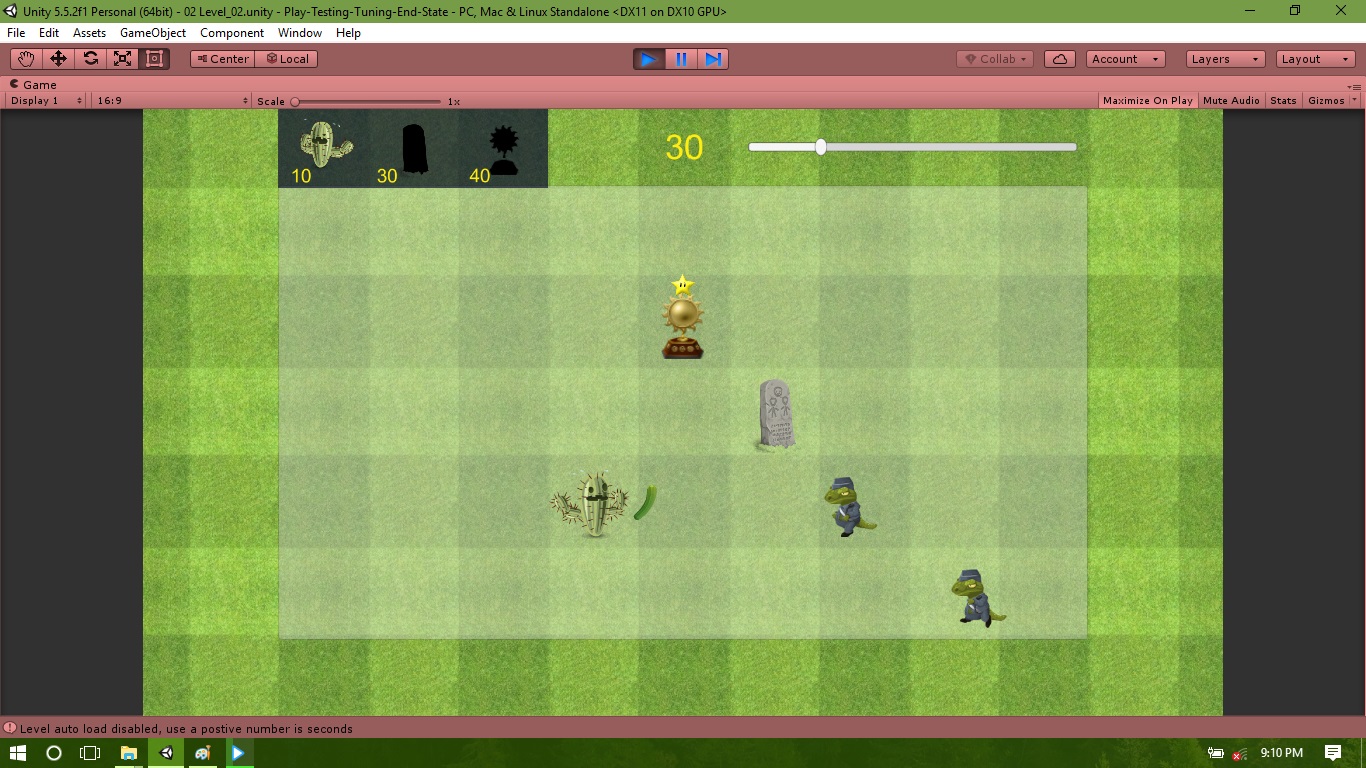


Figure 10: Level two game scene



Figure 11: Level two losing scene



Figure 12: Level two winning scene



Figure 13: Level three game start scene

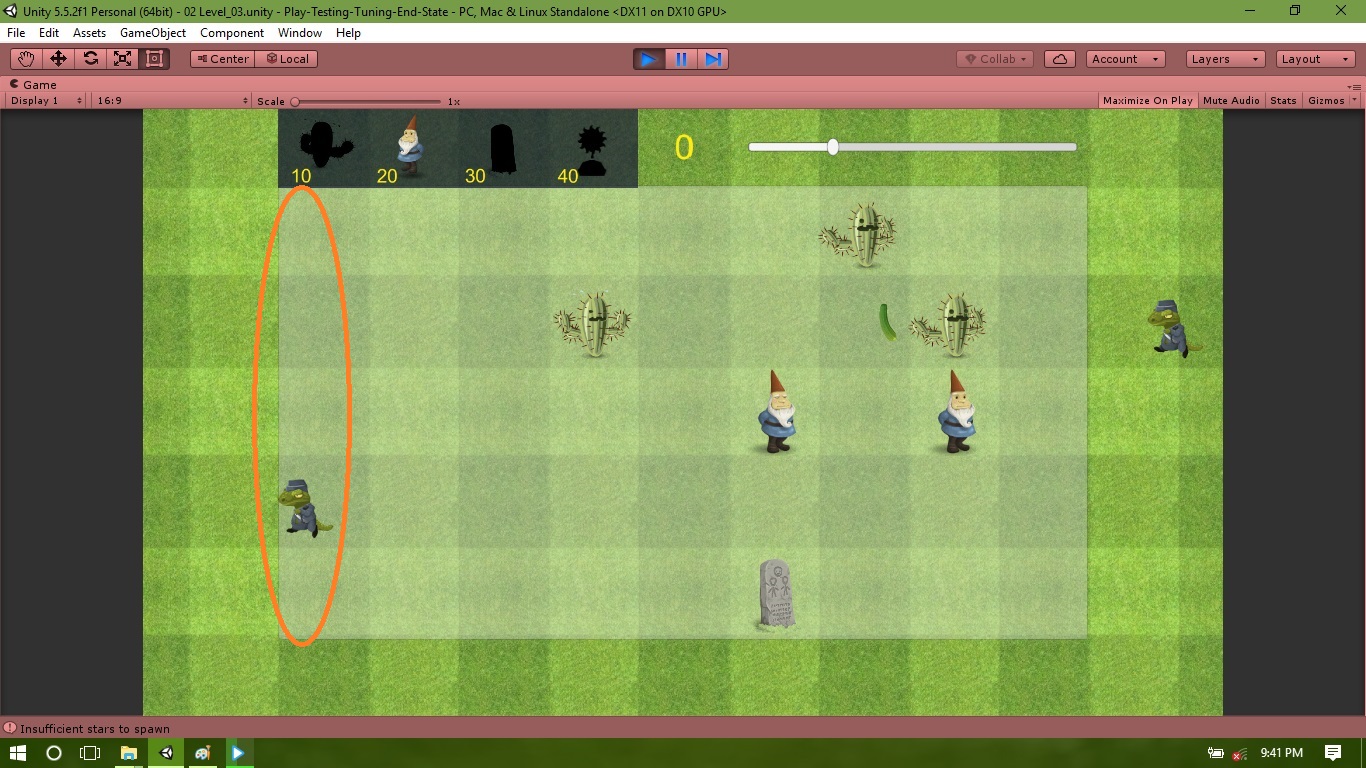


Figure 14: Level three game losing scene



Figure 15: Level three winning scene

Winning all levels, the player wins the game and the win screen show up.

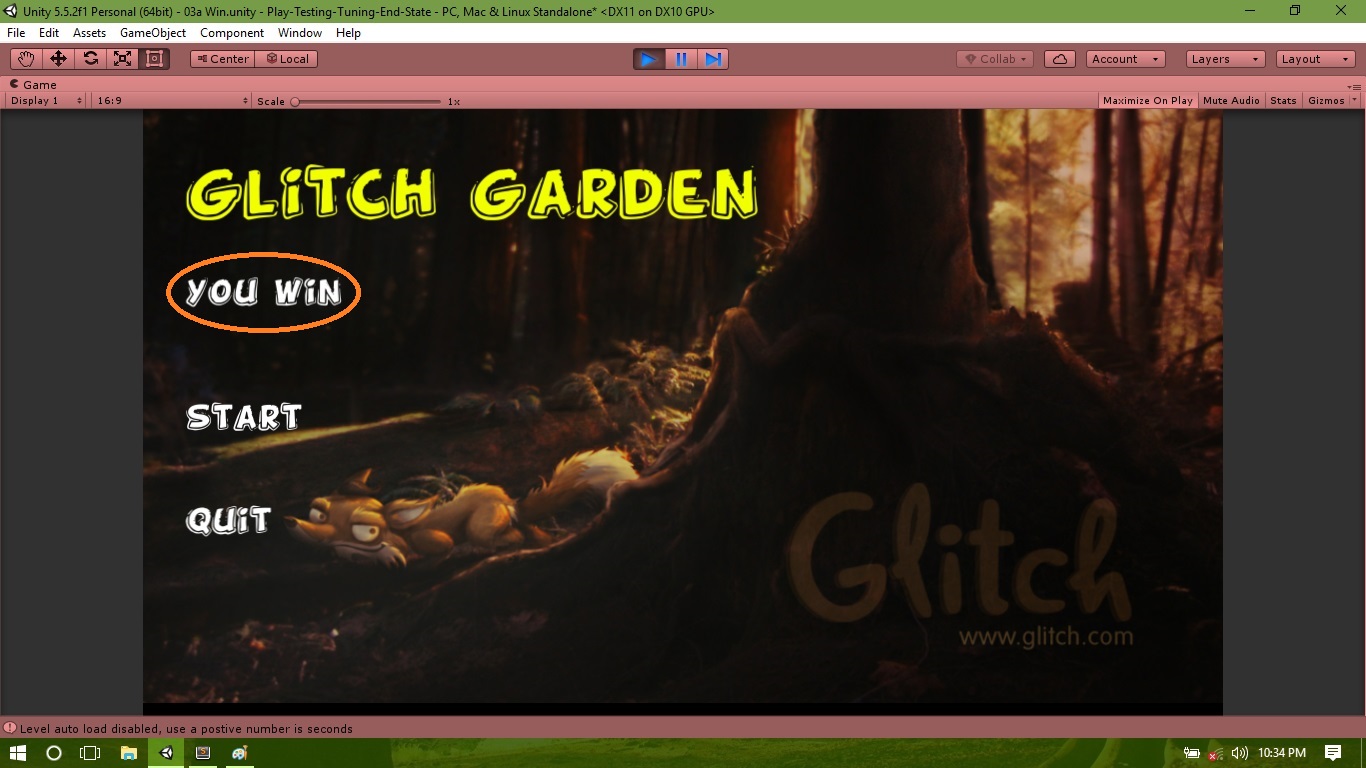


Figure 16: The game winning screen

Pressing the options button (orange marked), the player can go to options screen to change game settings.

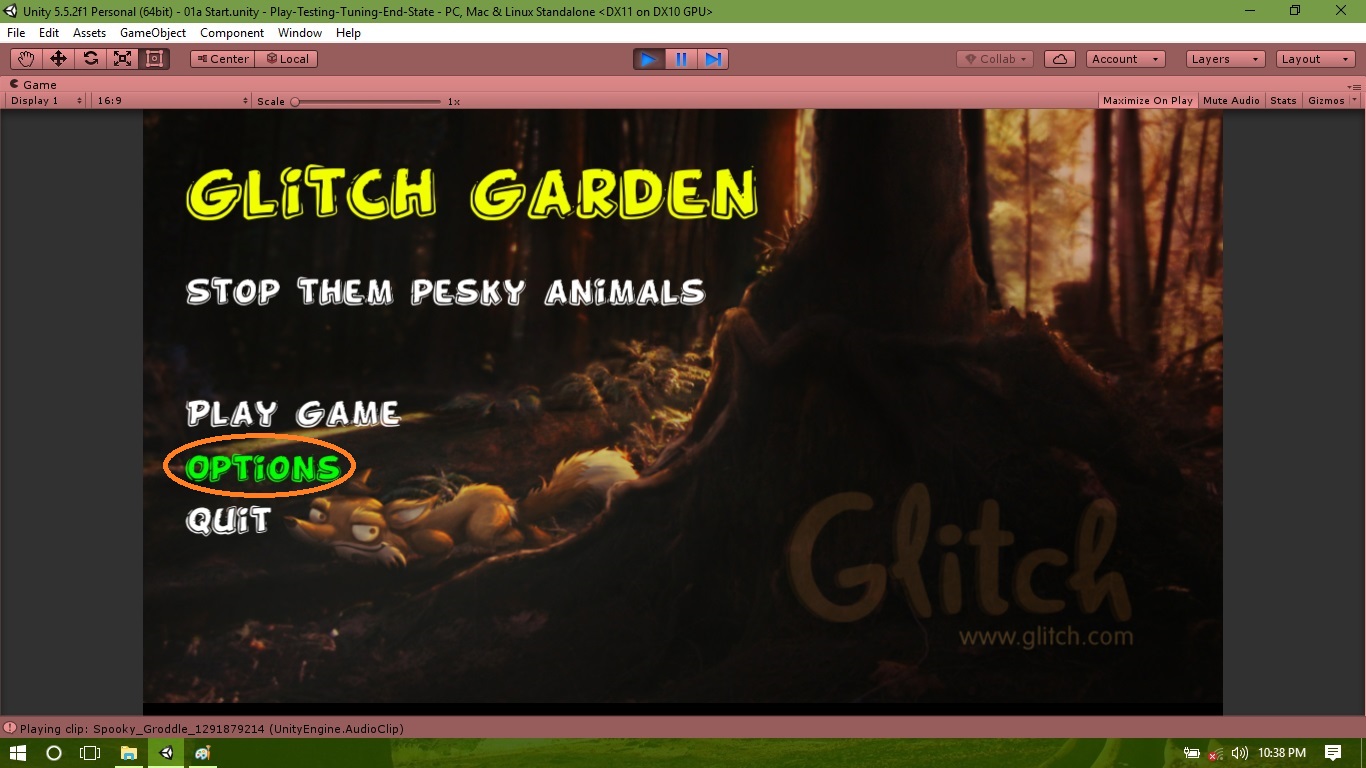


Figure 17: The game options screen

Option screen with player’s custom settings.

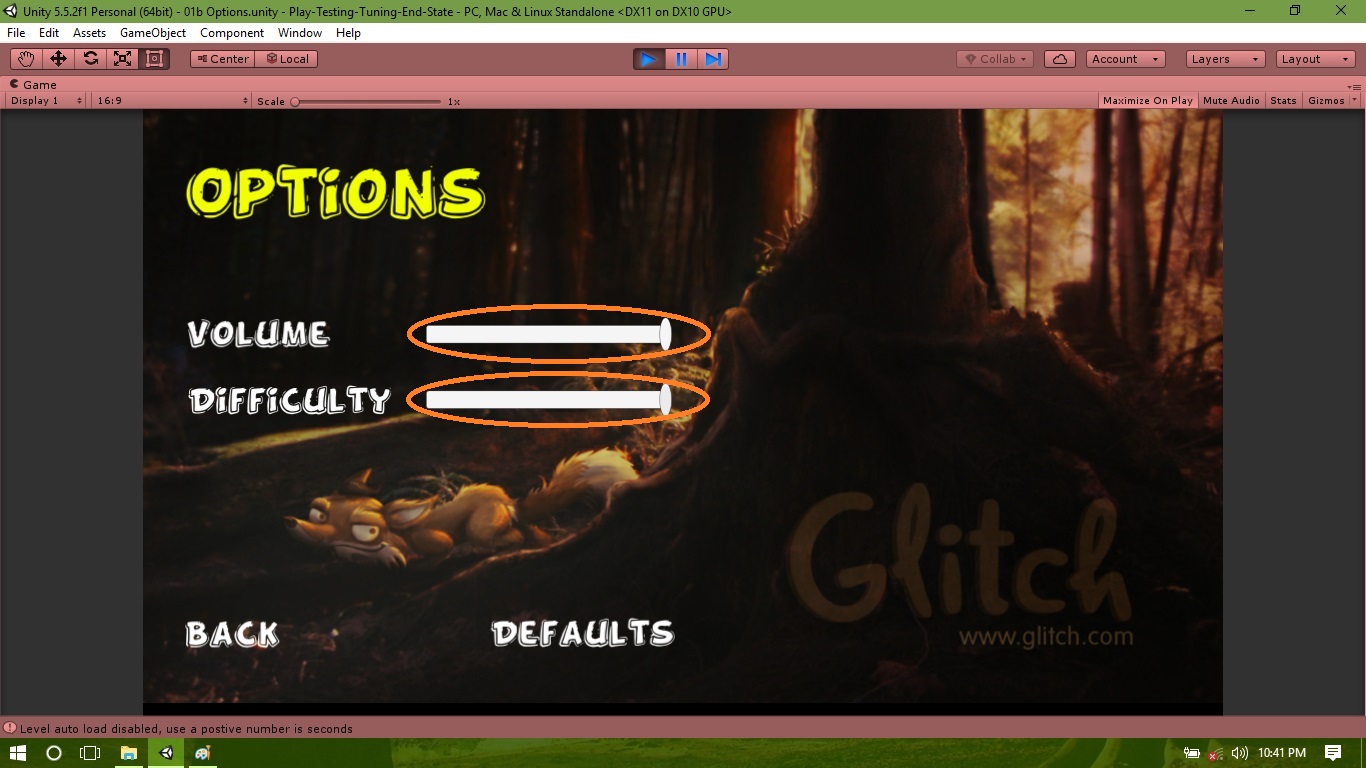


Figure 18: Game custom options

Pressing the default button (orange marked), the player can set the game settings to defaults. Pressing the back button the player can go back to the start screen.

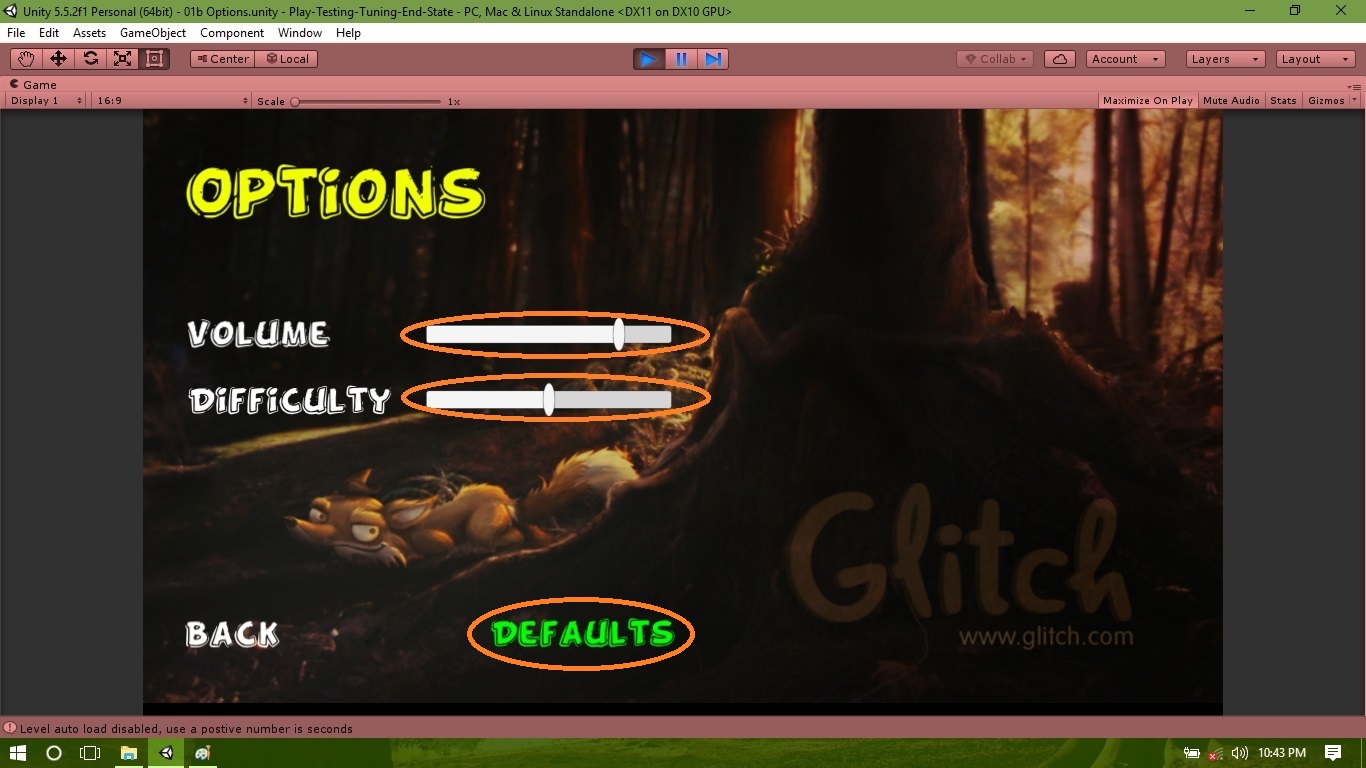


Figure 19: Game default options

Pressing the quit button (orange marked), the player can quit the game.

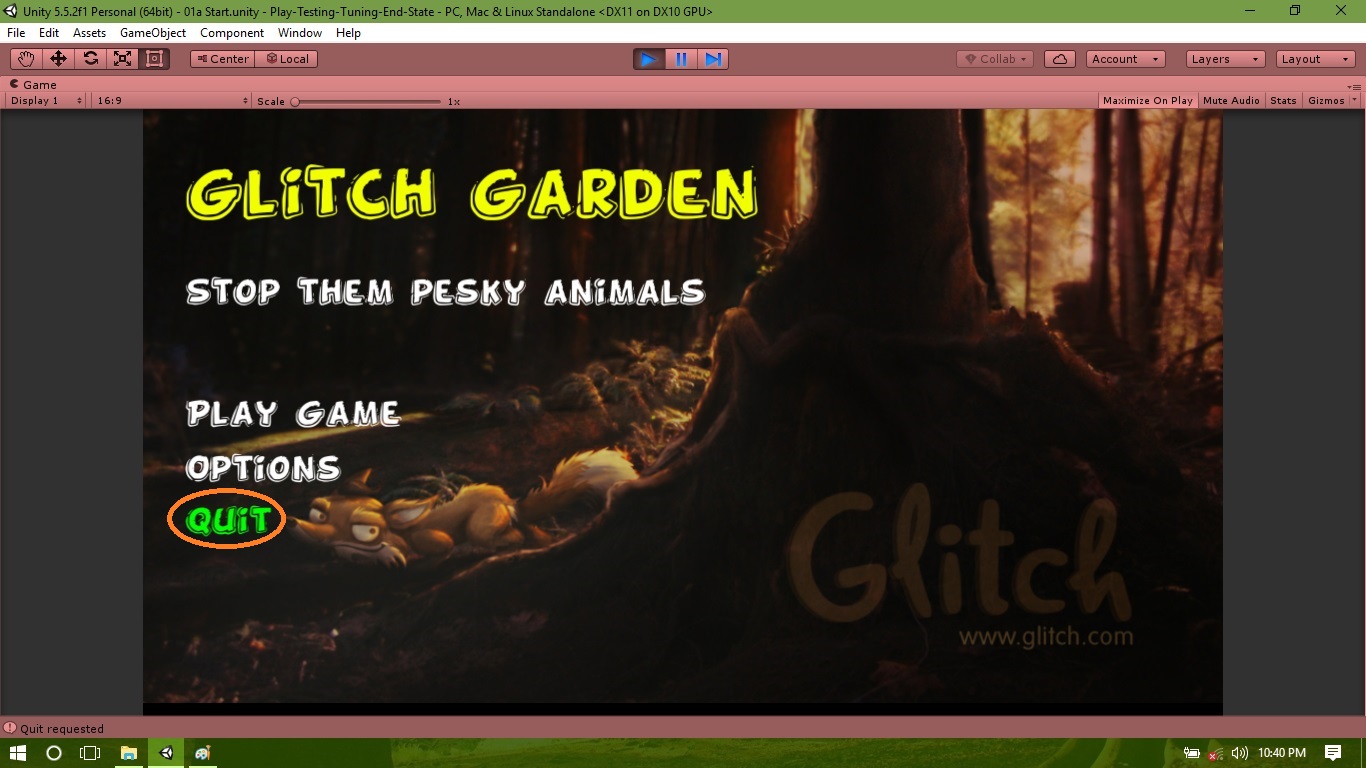


Figure 20: Game quit screen

**Conclusion:** at least half of a page or one page

**Appendix:**

**Code:**

1. ***Attacker***

using UnityEngine;

using System.Collections;

[RequireComponent (typeof (Rigidbody2D))]

public class Attacker : MonoBehaviour {

[Tooltip ("Average number of seconds between appearances")]

public float seenEverySeconds;

private float currentSpeed;

private GameObject currentTarget;

private Animator animator;

void Start () {

animator = GetComponent<Animator>();

}

// Update is called once per frame

void Update () {

transform.Translate (Vector3.left \* currentSpeed \* Time.deltaTime);

if (!currentTarget) {

animator.SetBool ("isAttacking", false);

}

}

void OnTriggerEnter2D () {

}

public void SetSpeed (float speed) {

currentSpeed = speed;

}

// Called from the animator at time of actual blow

public void StrikeCurrentTarget (float damage) {

if (currentTarget) {

Health health = currentTarget.GetComponent<Health>();

if (health) {

health.DealDamage (damage);

}

}

}

public void Attack (GameObject obj) {

currentTarget = obj;

}

}

1. ***Button***

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

public class Button : MonoBehaviour {

public GameObject defenderPrefab;

public static GameObject selectedDefender;

private Button[] buttonArray;

private Text costText;

// Use this for initialization

void Start () {

buttonArray = GameObject.FindObjectsOfType<Button>();

costText = GetComponentInChildren<Text>();

if (!costText) {Debug.LogWarning (name + " has no cost ");}

costText.text = defenderPrefab.GetComponent<Defender>().starCost.ToString();

}

// Update is called once per frame

void Update () {

}

void OnMouseDown () {

foreach (Button thisButton in buttonArray) {

thisButton.GetComponent<SpriteRenderer>().color = Color.black;

}

GetComponent<SpriteRenderer>().color = Color.white;

selectedDefender = defenderPrefab;

}

}

1. ***Defender***

using UnityEngine;

using System.Collections;

public class Defender : MonoBehaviour {

public int starCost = 100;

private StarDisplay starDisplay;

void Start () {

starDisplay = GameObject.FindObjectOfType<StarDisplay>();

}

// Only being used as a tag for now!

public void AddStars (int amount) {

starDisplay.AddStars (amount);

}

}

1. ***DefenderSpawner***

using UnityEngine;

using System.Collections;

public class DefenderSpawner : MonoBehaviour {

public Camera myCamera;

private GameObject parent;

private StarDisplay starDisplay;

void Start () {

parent = GameObject.Find ("Defenders");

starDisplay = GameObject.FindObjectOfType<StarDisplay>();

if (!parent) {

parent = new GameObject("Defenders");

}

}

void OnMouseDown () {

Vector2 rawPos = CalculateWorldPointOfMouseClick();

Vector2 roundedPos = SnapToGrid (rawPos);

GameObject defender = Button.selectedDefender;

int defenderCost = defender.GetComponent<Defender>().starCost;

if (starDisplay.UseStars(defenderCost) == StarDisplay.Status.SUCCESS) {

SpawnDefender (roundedPos, defender);

} else {

Debug.Log ("Insufficient stars to spawn");

}

}

void SpawnDefender (Vector2 roundedPos, GameObject defender)

{

Quaternion zeroRot = Quaternion.identity;

GameObject newDef = Instantiate (defender, roundedPos, zeroRot) as GameObject;

newDef.transform.parent = parent.transform;

}

Vector2 SnapToGrid (Vector2 rawWorldPos) {

float newX = Mathf.RoundToInt (rawWorldPos.x);

float newY = Mathf.RoundToInt (rawWorldPos.y);

return new Vector2 (newX, newY);

}

Vector2 CalculateWorldPointOfMouseClick () {

float mouseX = Input.mousePosition.x;

float mouseY = Input.mousePosition.y;

float distanceFromCamera = 10f;

Vector3 weirdTriplet = new Vector3 (mouseX, mouseY, distanceFromCamera);

Vector2 worldPos = myCamera.ScreenToWorldPoint (weirdTriplet);

return worldPos;

}

}

1. ***FadeIn***

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

public class FadeIn : MonoBehaviour {

public float fadeInTime;

private Image fadePanel;

private Color currentColor = Color.black;

// Use this for initialization

void Start () {

fadePanel = GetComponent<Image>();

}

// Update is called once per frame

void Update () {

if (Time.timeSinceLevelLoad < fadeInTime) {

// Fade in

float alphaChange = Time.deltaTime / fadeInTime;

currentColor.a -= alphaChange;

fadePanel.color = currentColor;

} else {

gameObject.SetActive (false);

}

}

}

1. ***Fox***

using UnityEngine;

using System.Collections;

[RequireComponent (typeof (Attacker))]

public class Fox : MonoBehaviour {

private Animator anim;

private Attacker attacker;

// Use this for initialization

void Start () {

anim = GetComponent<Animator>();

attacker = GetComponent<Attacker>();

}

// Update is called once per frame

void Update () {

}

void OnTriggerEnter2D (Collider2D collider) {

GameObject obj = collider.gameObject;

// Leave the method if not colliding with defender

if (!obj.GetComponent<Defender>()) {

return;

}

if (obj.GetComponent<Stone>()) {

anim.SetTrigger ("jump trigger");

} else {

anim.SetBool ("isAttacking", true);

attacker.Attack (obj);

}

}

}

1. ***GameTimer***

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

public class GameTimer : MonoBehaviour {

public float levelSeconds = 100;

private Slider slider;

private AudioSource audioSource;

private bool isEndOfLevel = false;

private LevelManager levelManager;

private GameObject winLabel;

// Use this for initialization

void Start () {

slider = GetComponent<Slider>();

audioSource = GetComponent<AudioSource>();

levelManager = GameObject.FindObjectOfType<LevelManager>();

FindYouWin ();

winLabel.SetActive(false);

}

void FindYouWin ()

{

winLabel = GameObject.Find ("You Win");

if (!winLabel) {

Debug.LogWarning ("Please create You Win object");

}

}

// Update is called once per frame

void Update () {

slider.value = Time.timeSinceLevelLoad / levelSeconds;

bool timeIsUp = (Time.timeSinceLevelLoad >= levelSeconds);

if (timeIsUp && !isEndOfLevel) {

audioSource.Play ();

winLabel.SetActive(true);

Invoke ("LoadNextLevel", audioSource.clip.length);

isEndOfLevel = true;

}

}

void LoadNextLevel () {

levelManager.LoadNextLevel();

}

}

1. ***Health***

using UnityEngine;

using System.Collections;

public class Health : MonoBehaviour {

public float health = 100f;

public void DealDamage (float damage) {

health -= damage;

if (health < 0) {

// Optionally trigger animation

DestoryObject ();

}

}

public void DestoryObject () {

Destroy (gameObject);

}

}

1. ***LevelManager***

using UnityEngine;

using System.Collections;

public class LevelManager : MonoBehaviour {

public float autoLoadNextLevelAfter;

void Start () {

if (autoLoadNextLevelAfter <= 0) {

Debug.Log ("Level auto load disabled, use a postive number is seconds");

} else {

Invoke ("LoadNextLevel", autoLoadNextLevelAfter);

}

}

public void LoadLevel(string name){

Debug.Log ("New Level load: " + name);

Application.LoadLevel (name);

}

public void QuitRequest(){

Debug.Log ("Quit requested");

Application.Quit ();

}

public void LoadNextLevel() {

Application.LoadLevel(Application.loadedLevel + 1);

}

}

1. ***Lizard***

using UnityEngine;

using System.Collections;

[RequireComponent (typeof (Attacker))]

public class Lizard : MonoBehaviour {

private Animator anim;

private Attacker attacker;

// Use this for initialization

void Start () {

anim = GetComponent<Animator>();

attacker = GetComponent<Attacker>();

}

// Update is called once per frame

void Update () {

}

void OnTriggerEnter2D (Collider2D collider) {

GameObject obj = collider.gameObject;

// Leave the method if not colliding with defender

if (!obj.GetComponent<Defender>()) {

return;

}

anim.SetBool ("isAttacking", true);

attacker.Attack (obj);

}

}

1. ***LoseCollider***

using UnityEngine;

using System.Collections;

public class LoseCollider : MonoBehaviour {

private LevelManager levelManager;

// Use this for initialization

void Start () {

levelManager = GameObject.FindObjectOfType<LevelManager>();

}

void OnTriggerEnter2D () {

levelManager.LoadLevel ("03b Lose");

}

}

1. ***MusicManager***

using UnityEngine;

using System.Collections;

public class MusicManager : MonoBehaviour {

public AudioClip[] levelMusicChangeArray;

private AudioSource audioSource;

void Awake() {

DontDestroyOnLoad (gameObject);

Debug.Log ("Don't destory on load: " + name);

}

void Start () {

audioSource = GetComponent<AudioSource>();

}

void OnLevelWasLoaded (int level) {

AudioClip thisLevelMusic = levelMusicChangeArray[level];

Debug.Log ("Playing clip: " + thisLevelMusic);

if (thisLevelMusic) { // If there's some music attached

audioSource.clip = thisLevelMusic;

audioSource.loop = true;

audioSource.Play ();

}

}

public void SetVolume (float volume) {

audioSource.volume = volume;

}

}

1. ***OptionsController***

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

public class OptionsController : MonoBehaviour {

public Slider volumeSlider, diffSlider;

public LevelManager levelManager;

private MusicManager musicManager;

// Use this for initialization

void Start () {

musicManager = GameObject.FindObjectOfType<MusicManager>();

volumeSlider.value = PlayerPrefsManager.GetMasterVolume ();

diffSlider.value = PlayerPrefsManager.GetDifficulty ();

}

// Update is called once per frame

void Update () {

musicManager.SetVolume (volumeSlider.value);

}

public void SaveAndExit () {

PlayerPrefsManager.SetMasterVolume (volumeSlider.value);

PlayerPrefsManager.SetDifficulty (diffSlider.value);

levelManager.LoadLevel ("01a Start");

}

public void SetDefaults () {

volumeSlider.value = 0.8f;

diffSlider.value = 2f;

}

}

1. ***PlayerPrefsManager***

using UnityEngine;

using System.Collections;

public class PlayerPrefsManager : MonoBehaviour {

const string MASTER\_VOLUME\_KEY = "master\_volume";

const string DIFFICULTY\_KEY = "difficulty";

const string LEVEL\_KEY = "level\_unlocked\_";

public static void SetMasterVolume (float volume) {

if (volume >= 0f && volume <= 1f) {

PlayerPrefs.SetFloat (MASTER\_VOLUME\_KEY, volume);

} else {

Debug.LogError ("Master volume out of range");

}

}

public static float GetMasterVolume () {

return PlayerPrefs.GetFloat (MASTER\_VOLUME\_KEY);

}

public static void UnlockLevel (int level) {

if (level <= Application.levelCount - 1) {

PlayerPrefs.SetInt (LEVEL\_KEY + level.ToString(), 1); // Use 1 for true

} else {

Debug.LogError ("Trying to unlock level not in build order");

}

}

public static bool IsLevelUnlocked (int level) {

int levelValue = PlayerPrefs.GetInt (LEVEL\_KEY + level.ToString());

bool isLevelUnlocked = (levelValue == 1);

if (level <= Application.levelCount - 1) {

return isLevelUnlocked;

} else {

Debug.LogError ("Trying to query level not in build order");

return false;

}

}

public static void SetDifficulty (float difficulty) {

if (difficulty >= 1f && difficulty <= 3f) {

PlayerPrefs.SetFloat (DIFFICULTY\_KEY, difficulty);

} else {

Debug.LogError ("Difficulty out of range");

}

}

public static float GetDifficulty () {

return PlayerPrefs.GetFloat (DIFFICULTY\_KEY);

}

}

1. ***Projectile***

using UnityEngine;

using System.Collections;

public class Projectile : MonoBehaviour {

public float speed, damage;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

transform.Translate (Vector3.right \* speed \* Time.deltaTime);

}

void OnTriggerEnter2D (Collider2D collider) {

Attacker attacker = collider.gameObject.GetComponent<Attacker>();

Health health = collider.gameObject.GetComponent<Health>();

if (attacker && health) {

health.DealDamage (damage);

Destroy (gameObject);

}

}

}

1. ***SetStartVolume***

using UnityEngine;

using System.Collections;

public class SetStartVolume : MonoBehaviour {

private MusicManager musicManager;

// Use this for initialization

void Start () {

musicManager = GameObject.FindObjectOfType<MusicManager>();

if (musicManager) {

float volume = PlayerPrefsManager.GetMasterVolume();

musicManager.SetVolume (volume);

} else {

Debug.LogWarning ("No music manger found in Start scene, can't set volume");

}

}

// Update is called once per frame

void Update () {

}

}

1. ***Shooter***

using UnityEngine;

using System.Collections;

public class Shooter : MonoBehaviour {

public GameObject projectile, gun;

private GameObject projectileParent;

private Animator animator;

private Spawner myLaneSpawner;

void Start () {

animator = GameObject.FindObjectOfType<Animator>();

// Creates a parent if necessary

projectileParent = GameObject.Find ("Projectiles");

if (!projectileParent) {

projectileParent = new GameObject("Projectiles");

}

SetMyLaneSpawner();

}

void Update () {

if (IsAttackerAheadInLane()) {

animator.SetBool ("isAttacking", true);

} else {

animator.SetBool ("isAttacking", false);

}

}

// Look through all spawners, and set myLaneSpanwer if found

void SetMyLaneSpawner () {

Spawner[] spawnerArray = GameObject.FindObjectsOfType<Spawner>();

foreach (Spawner spawner in spawnerArray) {

if (spawner.transform.position.y == transform.position.y) {

myLaneSpawner = spawner;

return;

}

}

Debug.LogError (name + " can't find spawner in lane");

}

bool IsAttackerAheadInLane() {

// Exit if no attackers in lane

if (myLaneSpawner.transform.childCount <= 0) {

return false;

}

// If there are attackers, are they ahead?

foreach (Transform attacker in myLaneSpawner.transform) {

if (attacker.transform.position.x > transform.position.x) {

return true;

}

}

// Attacker in lane, but behind us.

return false;

}

private void Fire () {

GameObject newProjectile = Instantiate (projectile) as GameObject;

newProjectile.transform.parent = projectileParent.transform;

newProjectile.transform.position = gun.transform.position;

}

}

1. ***Shreddder***

using UnityEngine;

using System.Collections;

public class Shreddder : MonoBehaviour {

void OnTriggerEnter2D (Collider2D collider) {

Destroy (collider.gameObject);

}

}

1. ***Spawner***

using UnityEngine;

using System.Collections;

public class Spawner : MonoBehaviour {

public GameObject[] attackerPrefabArray;

// Update is called once per frame

void Update () {

foreach (GameObject thisAttacker in attackerPrefabArray) {

if (isTimeToSpawn (thisAttacker)) {

Spawn (thisAttacker);

}

}

}

void Spawn (GameObject myGameObject) {

GameObject myAttacker = Instantiate (myGameObject) as GameObject;

myAttacker.transform.parent = transform;

myAttacker.transform.position = transform.position;

}

bool isTimeToSpawn (GameObject attackerGameObject) {

Attacker attacker = attackerGameObject.GetComponent<Attacker>();

float meanSpawnDelay = attacker.seenEverySeconds;

float spawnsPerSecond = 1 / meanSpawnDelay;

if (Time.deltaTime > meanSpawnDelay) {

Debug.LogWarning ("Spwan rate capped by frame rate");

}

float threshold = spawnsPerSecond \* Time.deltaTime / 5;

return (Random.value < threshold);

}

}

1. ***StarDisplay***

using UnityEngine;

using UnityEngine.UI;

using System.Collections;

[RequireComponent (typeof(Text))]

public class StarDisplay : MonoBehaviour {

private Text starText;

private int stars = 100;

public enum Status {SUCCESS, FAILURE};

// Use this for initialization

void Start () {

starText = GetComponent <Text>();

UpdateDisplay();

}

public void AddStars (int amount) {

stars += amount;

UpdateDisplay();

}

public Status UseStars (int amount) {

if (stars >= amount) {

stars -= amount;

UpdateDisplay();

return Status.SUCCESS;

}

return Status.FAILURE;

}

private void UpdateDisplay () {

starText.text =stars.ToString();

}

}

1. ***Stone***

using UnityEngine;

using System.Collections;

public class Stone : MonoBehaviour {

private Animator animator;

void Start () {

animator = GetComponent<Animator>();

}

void Update () {

}

void OnTriggerStay2D (Collider2D collider) {

Attacker attacker = collider.gameObject.GetComponent<Attacker>();

if (attacker) {

animator.SetTrigger ("underAttack trigger");

}

}

}