Portfolio Project

February 2020

By Charles Thomas Merrick-Bisset

# Contents

[Contents 1](#_Toc25930457)

[Introduction 3](#_Toc25930458)

[Team Roles 3](#_Toc25930459)

[Game Design 3](#_Toc25930460)

[Game Concept 4](#_Toc25930461)

[Backstory 5](#_Toc25930462)

[Feature Goals 5](#_Toc25930463)

[Top Priority 5](#_Toc25930464)

[Mid Priority 5](#_Toc25930465)

[Low Priority 6](#_Toc25930466)

[Controls 6](#_Toc25930467)

[Software Used 6](#_Toc25930468)

[Scrapped Ideas 6](#_Toc25930469)

[2D RPG 6](#_Toc25930470)

[Requirements Specification 7](#_Toc25930471)

[Functional Requirements 7](#_Toc25930472)

[Non-Functional Requirements 7](#_Toc25930473)

[Assets Required 7](#_Toc25930474)

[Player Interface 8](#_Toc25930475)

[Arena 8](#_Toc25930476)

[Player 8](#_Toc25930477)

[Enemies 8](#_Toc25930478)

[Undead - Melee 8](#_Toc25930479)

[Turrets - Shooter 9](#_Toc25930480)

[Boss - ??? 9](#_Toc25930481)

[Weapons 9](#_Toc25930482)

[Sword 9](#_Toc25930483)

[Bow 9](#_Toc25930484)

[Business Model/Games Development Life Cycle 10](#_Toc25930485)

[Usability testing 10](#_Toc25930486)

[Functionality testing 10](#_Toc25930487)

[Testing Methods 10](#_Toc25930488)

[Sound Design 11](#_Toc25930489)

[Inspiration 11](#_Toc25930490)

[Mechanics 11](#_Toc25930491)

[Story/Backstory 12](#_Toc25930492)

[Enemy Design 12](#_Toc25930493)

[Aesthetic 12](#_Toc25930494)

[Concept Art - Enemies 13](#_Toc25930495)

[ 13](#_Toc25930496)

[Concept Art - Player HUD 14](#_Toc25930497)

[Final Design 15](#_Toc25930498)

[Enemy 15](#_Toc25930499)

[Turret 15](#_Toc25930500)

[Player 16](#_Toc25930501)

[UML Diagrams 16](#_Toc25930502)

[Test Log 17](#_Toc25930503)

[Difficulties 18](#_Toc25930504)

[Sources/Credit 19](#_Toc25930505)

[Technical Design 19](#_Toc25930506)

[Mechanics and Gameplay 19](#_Toc25930507)

[Showing my work online 20](#_Toc25930508)

# Introduction

I’m Charles Thomas Merrick Bisset this is the documentation for my portfolio project the development team are as followed Charles Thomas Merrick Bisset. I’m the only team member I chose to work alone so I take responsibility for everything and every role. The Second half of this portfolio is for the 28th February 2020 and will be assessed by Gavin Baxter. I will be making this game in unity either from home or in the university labs. I know working yourself usually makes things more risky but I kind of wanted to work on a kind of dream project so I think its less risky because I have more passion for it.

## Team Roles

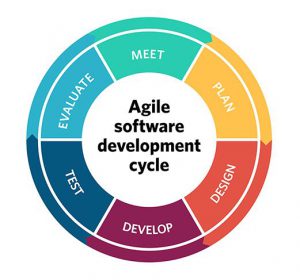
Charles Thomas Merrick Bisset Everything.

B00314125

# Business Model/Games Development Life Cycle

I settled on using the Agile life cycle as I had already applied some of its aspects by myself, such as our prioritisation of features, or mandatory vs optional requirements initial documentation.

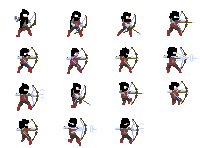
The agile methodology is similar to the standard waterfall methodology as it follows the same sequence of stages – Planning, Analysis, Design, Implementation, Testing and Maintenance. However, this method aims to catch problems early by releasing multiple iterations of the same the program rapidly with small incremental changes for testing, catching issues early in development before they become a bigger one further down the line. This allows me to make changes if we know something in the game won’t work before it’s too late.

  
  
  
I feel the Agile life cycle will help me greatly in working towards my goal as I am working myself in the labs I go down the priority list and move down each time I’m done.

## Assets and Source code Used

Player Assets

## 



## Enemy assets







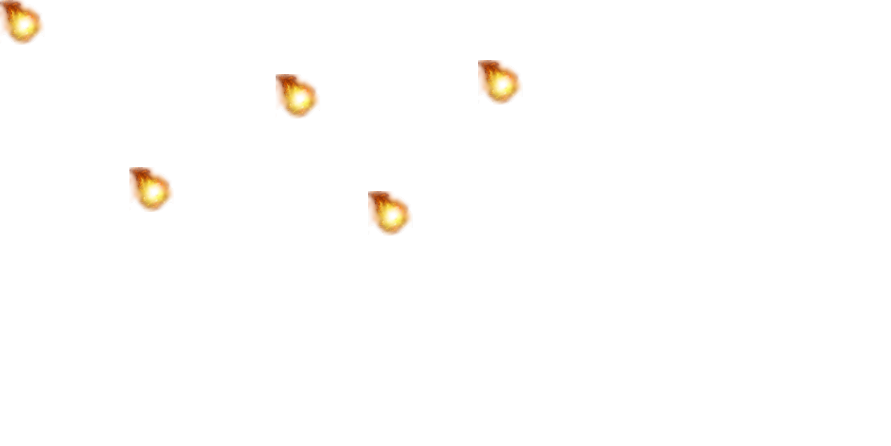






Background







## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 







**Player code**

**Player controller**

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

namespace Platformer

{

public class PlayerController : MonoBehaviour, IController

{

//Components

Stats stats;

AnimatorController animatorController;

PlayerCombat playerCombat;

public InputManager inputManager = new InputManager();

[Header("Variables")]

public float moveSpeed;

public float jumpForce;

public float rollForce;

Transform playerSprite;

[HideInInspector] public Rigidbody2D rigid2D { get; set; }

[SerializeField] bool isGrounded;

[HideInInspector] public bool isAttacking { get; set; }

[HideInInspector] public bool onLadder;

bool isClimping;

private void Start()

{

stats = GetComponent<Stats>();

rigid2D = GetComponent<Rigidbody2D>();

playerSprite = transform.GetChild(0);

animatorController = GetComponentInChildren<AnimatorController>();

inputManager.inputConfig.UpdateDictionary(); //

//add Death event

stats.OnDeath += Death;

}

public void FixedUpdate()

{

if (GameManager.Instance.isGame && !GameManager.Instance.isPause) //check Game status

{

Move();

LadderClimb();

GroundCheck();

}

}

private void Update()

{

if (GameManager.Instance.isGame && !GameManager.Instance.isPause)

{

Rotation();

Jump();

Roll();

Attack();

Animation();

}

}

//Move method

public void Move()

{

if (!isAttacking)

transform.Translate(new Vector2(inputManager.Horizontal \* moveSpeed \* Time.deltaTime, 0));

}

//Rotation method

public void Rotation()

{

if (inputManager.Horizontal != 0) //if player any horizontal side move

{

if (inputManager.Horizontal < 0)

playerSprite.localScale = new Vector3(-1, 1, 1);

else

playerSprite.localScale = new Vector3(1, 1, 1);

}

}

//Roll method

void Roll()

{

if (isGrounded && inputManager.Roll && !isAttacking && !isClimping) //Check for availability rollback

{

if (Mathf.Round(inputManager.Horizontal) != 0)

{

animatorController.SetTrigger("Roll");

rigid2D.velocity = Vector2.right \* inputManager.Horizontal \* rollForce;

}

}

}

//Jump method

void Jump()

{

if (inputManager.Jump && isGrounded && !isAttacking && !isClimping)

{

animatorController.SetTrigger("Jump");

rigid2D.velocity = Vector2.up \* jumpForce;

GetComponent<AudioSource>().Play();

}

}

//Attack method

public void Attack()

{

if (isGrounded && (inputManager.MeleeAttack || inputManager.RangeAttack) && !isAttacking && !isClimping)

{

isAttacking = true; //attack status

if (inputManager.MeleeAttack)

{

animatorController.SetBool("MeleeAttack", isAttacking); //Set animator bool

}

else if (inputManager.RangeAttack)

{

animatorController.SetBool("RangeAttack", isAttacking);

GetComponent<AudioSource>().Play();

}

}

}

//Animator method

public void Animation()

{

if (!isClimping) //Lader check

if (inputManager.Horizontal != 0)

animatorController.SetBool("Move", true);

else

animatorController.SetBool("Move", false);

else

{

animatorController.SetBool("Move", false);

}

animatorController.SetBool("isGrounded", isGrounded);

}

void LadderClimb()

{

if (onLadder) //check lader status

{

if (inputManager.Vertical != 0)

{

isClimping = true;

rigid2D.velocity = Vector2.up \* inputManager.Vertical \* moveSpeed; //move up or down

}

else

{

rigid2D.velocity = Vector2.zero;

}

}

else //if leave ladder

{

isClimping = false;

}

if (onLadder)

rigid2D.gravityScale = 0;

else

rigid2D.gravityScale = 1;

}

//Death event

public void Death()

{

animatorController.SetTrigger("Death");

GameManager.Instance.GameOver(); //set game state to game over

}

//Check ground

void GroundCheck()

{

RaycastHit2D raycastHit2D = Physics2D.Raycast(transform.position, Vector2.down, 0.15f);

if (raycastHit2D.collider != null)

{

if (Vector2.Distance(transform.position, raycastHit2D.point) <= raycastHit2D.distance) //if raycast

{

isGrounded = true; //is ground

Debug.DrawLine(transform.position, raycastHit2D.point); //draw line only in editor

}

}

else

{

isGrounded = false;

}

}

}

}

**Player combat**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

namespace Platformer

{

public class PlayerCombat : Combat, ICombat

{

//Components

Rigidbody2D rigid2D;

AnimatorController animatorController;

InputManager inputManager;

PlayerController playerController;

public AudioSource Crash1;

bool canCombo;

public override void Start()

{

base.Start();

playerController = GetComponentInParent<PlayerController>();

rigid2D = GetComponentInParent<Rigidbody2D>();

animatorController = GetComponent<AnimatorController>();

inputManager = playerController.inputManager;

//Add hit event

hitInfo += HitDetected;

}

//Hit method

public override void HitDetected()

{

if (ColliderDetected.gameObject.tag == "Enemy")//if is enemy

{

Stats enemyStats = ColliderDetected.GetComponent<Stats>(); //Get data component from object

//Make visual hit effect

HitEffect hitEffect = ColliderDetected.GetComponentInChildren<HitEffect>();

hitEffect.PlayEffect();

float damage = damageRange.Random(); //get 1 random value damage of 2 (min,max)

MeleeAttack(enemyStats, damage);

CameraManager.Instance.cameraShake.Shake(); //Damage

}

}

//Method for animator, when player melee attack begin

public void OnMeleeAttackBegin(float timeToCombo)

{

StartCoroutine(ICombo(timeToCombo)); //Start combo system

}

public void AttackForce(float forcePower)

{

rigid2D.AddForce(Vector2.right \* transform.localScale.x \* forcePower); //Makes jerk when attacking

}

//Method for animator, when player melee attack end

public void OnMeleeAttackEnd()

{

StopCoroutine(ICombo(0)); //Stop combo

//Animator update

animatorController.ResetTrigger("AttackCombo");

animatorController.SetBool("MeleeAttack", false);

canCombo = false; //block combo

playerController.isAttacking = false;

}

public void OnRangeAttackEnd()

{

animatorController.SetBool("RangeAttack", false);

playerController.isAttacking = false;

}

//Combo system

IEnumerator ICombo(float comboTimer)

{

canCombo = false;

yield return new WaitForSeconds(comboTimer);

canCombo = true;

while (canCombo)

{

if (inputManager.MeleeAttack)

{

canCombo = false;

animatorController.SetTrigger("AttackCombo");

StopCoroutine(ICombo(0));

}

yield return null;

}

}

}

}

**Enemy Code**

**Enemy eye trigger**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

namespace Platformer

{

public class EnemyEyesTrigger : MonoBehaviour

{

EnemyController enemyController; //Cached local Enemy Controller component

private void Start()

{

enemyController = GetComponentInParent<EnemyController>(); //Get component from object

}

private void OnTriggerEnter2D(Collider2D collision) //if some object entrer in trigger area

{

if (collision.gameObject.tag == "Player") //if that object tap is equal to Player

{

if (GameManager.Instance.isGame) //is game status is not equal to gameover

enemyController.Follow(collision.transform); //start follow object

}

}

}

}

**Enemy controller**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

namespace Platformer

{

public class EnemyController : MonoBehaviour, IController

{

//Components

Stats stats;

AnimatorController animatorController;

EnemyCombat enemyCombat;

CapsuleCollider2D capsuleCollider;

float colliderOffsetX; //Collider position

Transform enemySprite;

[HideInInspector] public Rigidbody2D rigid2D { get; set; }

[Header("Variables")]

public float moveSpeed; //move speed

bool isMove;

bool isGround;

[HideInInspector] public bool isAttacking { get; set; }

public float returnRadius = 2; //radius for start to return when player exit

bool isReturning;

Vector3 starterPos;

public bool isPatrolling;

public float patrollRadius;

//Patrol variables

float patrollSin;

float patrollTimer = 0.02f;

bool isFollow;

[HideInInspector] public Transform followTarget;

float returnTimer = 3;

public float attackRadius;

bool canMove;

private void Start()

{

canMove = true;

starterPos = transform.position;

stats = GetComponent<Stats>();

rigid2D = GetComponent<Rigidbody2D>();

enemySprite = transform.GetChild(0);

enemyCombat = GetComponentInChildren<EnemyCombat>();

animatorController = GetComponentInChildren<AnimatorController>();

capsuleCollider = GetComponent<CapsuleCollider2D>();

colliderOffsetX = capsuleCollider.offset.x;

//Add Death method to event when enemy die

stats.OnDeath += Death;

}

public void FixedUpdate()

{

if (GameManager.Instance.isGame && !GameManager.Instance.isPause) //Check pause and gameover

{

if (!canMove)

return; //Block move

Move();

}

}

private void Update()

{

if (GameManager.Instance.isGame && !GameManager.Instance.isPause)

{

if (!canMove)

return;

Rotation();

Animation();

}

}

public void Move()

{

CheckGround();

if (!enemyCombat.isAttacking)

isMove = true;

if (isPatrolling && !isFollow && !isReturning)

{

PatrollSin();

Patroll();

}

}

//The patrol is made using a sinusoid

void PatrollSin()

{

patrollSin += patrollTimer \* moveSpeed;

if (patrollSin >= 1)

{

patrollTimer = -0.02f;

}

else if (patrollSin <= -1)

{

patrollTimer = 0.02f;

}

}

//Follow method

public void Follow(Transform target)

{

followTarget = target; //set target to follow

if (!isFollow)

{

isFollow = true;

StartCoroutine(IFollow());

}

}

void Patroll()

{

float sin = patrollSin; //local sin varible

float x = patrollRadius \* sin + starterPos.x; //set x of enemy position

float y = transform.position.y;

float z = transform.position.z;

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

}

public void Rotation()

{

if (isPatrolling)

{

if (patrollTimer > 0) //Side of patroll

{

enemySprite.transform.localScale = new Vector3(1, 1, 1);

capsuleCollider.offset = new Vector2(colliderOffsetX, capsuleCollider.offset.y);

}

else //reverse

{

enemySprite.transform.localScale = new Vector3(-1, 1, 1);

capsuleCollider.offset = new Vector2(-colliderOffsetX, capsuleCollider.offset.y);

}

}

if (isFollow) //get target and check the difference between them

{

if (transform.position.x > followTarget.position.x)

{

enemySprite.transform.localScale = new Vector3(-1, 1, 1);

capsuleCollider.offset = new Vector2(-colliderOffsetX, capsuleCollider.offset.y);

}

else

{

enemySprite.transform.localScale = new Vector3(1, 1, 1);

capsuleCollider.offset = new Vector2(colliderOffsetX, capsuleCollider.offset.y);

}

}

if (isReturning) //side of stater position

{

if (transform.position.x > starterPos.x)

{

enemySprite.transform.localScale = new Vector3(-1, 1, 1);

capsuleCollider.offset = new Vector2(-colliderOffsetX, capsuleCollider.offset.y);

}

else

{

enemySprite.transform.localScale = new Vector3(1, 1, 1);

capsuleCollider.offset = new Vector2(colliderOffsetX, capsuleCollider.offset.y);

}

}

}

public void Attack()

{

enemyCombat.isAttacking = true;

}

public void Animation()

{

if (isMove)

animatorController.SetBool("Move", true);

else

animatorController.SetBool("Move", false);

}

//Death method

public void Death()

{

canMove = false; //block move

enemyCombat.enabled = false; //disable combat

rigid2D.simulated = false;//disable physics

animatorController.SetTrigger("Death");//animator death

}

public void CheckGround()

{

float rayStartXoffset = 0.14f; //Offset ray to check ground

float rayStartX;

if (patrollTimer > 0) //get rotate

{

rayStartX = transform.position.x + rayStartXoffset;

}

else

{

rayStartX = transform.position.x - rayStartXoffset;

}

Vector3 rayStartPos = new Vector3(rayStartX, transform.position.y, transform.position.z); //Raycast start position

RaycastHit2D raycastHit2D = Physics2D.Raycast(rayStartPos, Vector2.down, 0.18f); //Raycast

if (raycastHit2D.collider != null) //If ray hit object

{

if (Vector2.Distance(rayStartPos, raycastHit2D.point) <= raycastHit2D.distance) //if distans between object and enemy < local distance variable

{

Debug.DrawLine(rayStartPos, raycastHit2D.point);

isGround = true; //set is ground

}

if (raycastHit2D.collider.gameObject.tag != "Ground")

{

isGround = false;

patrollTimer = -patrollTimer;

}

}

else

{

isGround = false;

patrollTimer = -patrollTimer; //turn enemy in the opposite direction

}

}

IEnumerator IFollow()

{

float timer = returnTimer; //timer to return after palyer leave

while (isFollow)

{

if (Vector2.Distance(transform.position, followTarget.position) > returnRadius && timer <= 0) //if target leave

{

isFollow = false;

followTarget = null;

isReturning = true;

patrollSin = 0;

StartCoroutine(IReturnToStartPos()); //start returning

}

else if (!isGround) //if the target jumps over the gap

{

isFollow = false;

followTarget = null;

isReturning = true;

patrollSin = 0;

StartCoroutine(IReturnToStartPos());

}

else

{

if (Vector2.Distance(transform.position, followTarget.position) > attackRadius && !enemyCombat.isAttacking)

transform.position = Vector2.MoveTowards(transform.position, new Vector2(followTarget.position.x, transform.position.y), moveSpeed / 2 \* Time.deltaTime); //follow target

else //if target in attack radius

Attack(); //attack

timer -= Time.deltaTime;

}

yield return null;

}

yield return null;

}

IEnumerator IReturnToStartPos()

{

while (transform.position.x != starterPos.x)

{

transform.position = Vector2.MoveTowards(transform.position, new Vector2(starterPos.x, transform.position.y), moveSpeed / 2 \* Time.deltaTime); //move to start pos

if (stats.statsData.HP < enemyCombat.enemyConfig.HP)

{

stats.statsData.HP = enemyCombat.enemyConfig.HP; //reset enemy hp

}

yield return null;

}

isReturning = false;

yield return null;

}

//Just editor method for drawing variables

public void OnDrawGizmos()

{

Gizmos.DrawWireSphere(transform.position, attackRadius);

Gizmos.DrawWireSphere(transform.position, returnRadius);

}

}

}

**Enemy Combat**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

namespace Platformer

{

public class EnemyCombat : Combat, ICombat

{

[Header("Config")]

public EnemyConfig enemyConfig; //Config, drap and drop in inspector your config

//Components

Rigidbody2D rigid2D;

AnimatorController animatorController;

[Header("Variables")]

public float attackRateSpeed; //Rate of enemy attack

float attackRateTimer; //Rate local timer

[HideInInspector] public bool isAttacking; //check status of attack

public override void Start()

{

base.Start(); //parent start method

//Get components

rigid2D = GetComponentInParent<Rigidbody2D>();

animatorController = GetComponent<AnimatorController>();

//Set enemy data from config

damageRange = new DoubleFloat(enemyConfig.damageRange.min, enemyConfig.damageRange.max);

stats.statsData.HP = enemyConfig.HP;

//add method hitInfo to enent, need for setup logic when enemy get hit

hitInfo += HitDetected;

}

//Hit method

public override void HitDetected()

{

if (ColliderDetected.gameObject.tag == "Player") //if is player

{

Stats playerStats = ColliderDetected.GetComponent<Stats>(); //Get data component from object

//Make visual hit effect

HitEffect hitEffect = ColliderDetected.GetComponentInChildren<HitEffect>();

hitEffect.PlayEffect();

float damage = damageRange.Random(); //get 1 random value damage of 2 (min,max)

MeleeAttack(playerStats, damage); //Damage

CameraManager.Instance.cameraShake.Shake(); //Camera shake

}

}

//Method for animator, when enemy melee attack end

public void OnMeleeAttackEnd()

{

isAttacking = false; //Attack status

animatorController.SetBool("MeleeAttack", false); //Animator bool MeleeAttack = false

}

public void AttackForce(float forcePower)

{

// Here you can add any logic of the moment of impact, similar to that of a player.

}

public override void RangeAttack()

{

base.RangeAttack();

}

//Method for animator, when enemy range attack end

public void OnRangeAttackEnd()

{

animatorController.SetBool("RangeAttack", false);

}

void Update()

{

if (isAttacking)

{

if (attackRateTimer > 6) //attack rate timer

{

attackRateTimer -= Time.deltaTime; //attack rate timer - 1 every frame(you can change Update to FixedUpdate for everysecond)

}

else

{

attackRateTimer = attackRateSpeed; //set attack rate timer to attack rate time

animatorController.SetBool("MeleeAttack", true);

}

}

}

}

}

**Menus**

**Game manager**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

namespace Platformer

{

public class GameManager : MonoBehaviour

{

public static GameManager Instance;

[HideInInspector] public Stats playerStats;

[Header("Parameters")]

public bool isGame;

public bool isPause;

public AudioSource Crash1;

void SingletonInit()

{

if (Instance != null)

Destroy(gameObject);

else

Instance = this;

}

private void Awake()

{

SingletonInit();

playerStats = GameObject.FindGameObjectWithTag("Player").GetComponent<Stats>(); //find player data component

}

private void Start()

{

StartGame(); //when scene load game'll start

}

public void StartGame()

{

isGame = true;

}

//Lose method

public void GameOver()

{

isGame = false; //disable game

UIManager.Instance.ChangeScreen(UIManager.ScreenState.Lose); //change screen to lose

}

}

}

**Main menu manager**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

namespace Platformer

{

public class MainMenuManager : MonoBehaviour

{

//Simle main menu script

//Nothing unusual

public void NewGame()

{

SceneManager.LoadScene("GameScene");

}

public void Quit()

{

Application.Quit();

}

public void Story()

{

SceneManager.LoadScene("Story");

}

public void DarkStart()

{

SceneManager.LoadScene("Dark1");

}

public void Credits1()

{

SceneManager.LoadScene("Credits");

}

public void Credits2()

{

SceneManager.LoadScene("MainMenu");

}

}

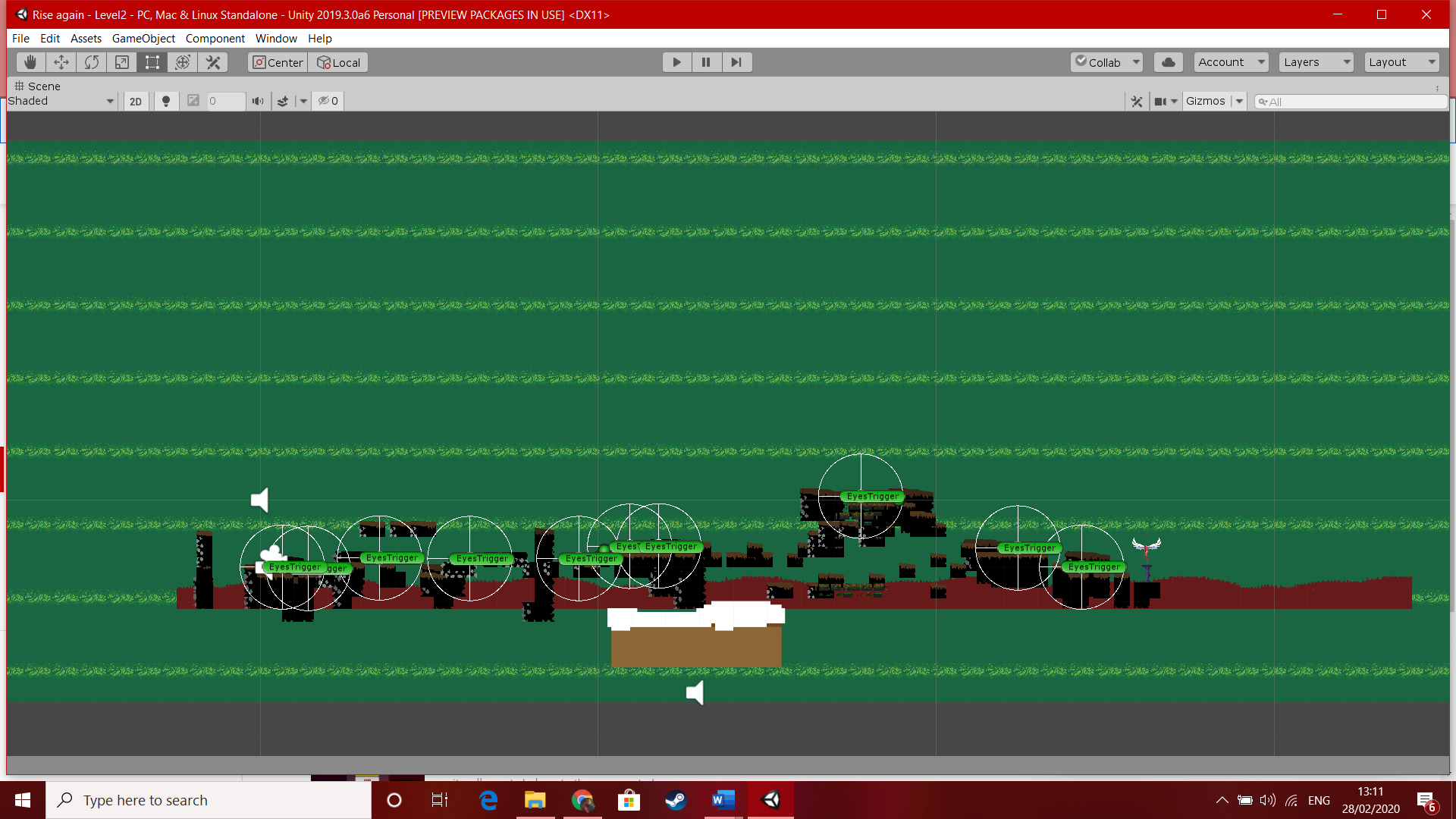
}

**Creation of developed levels and game maps**

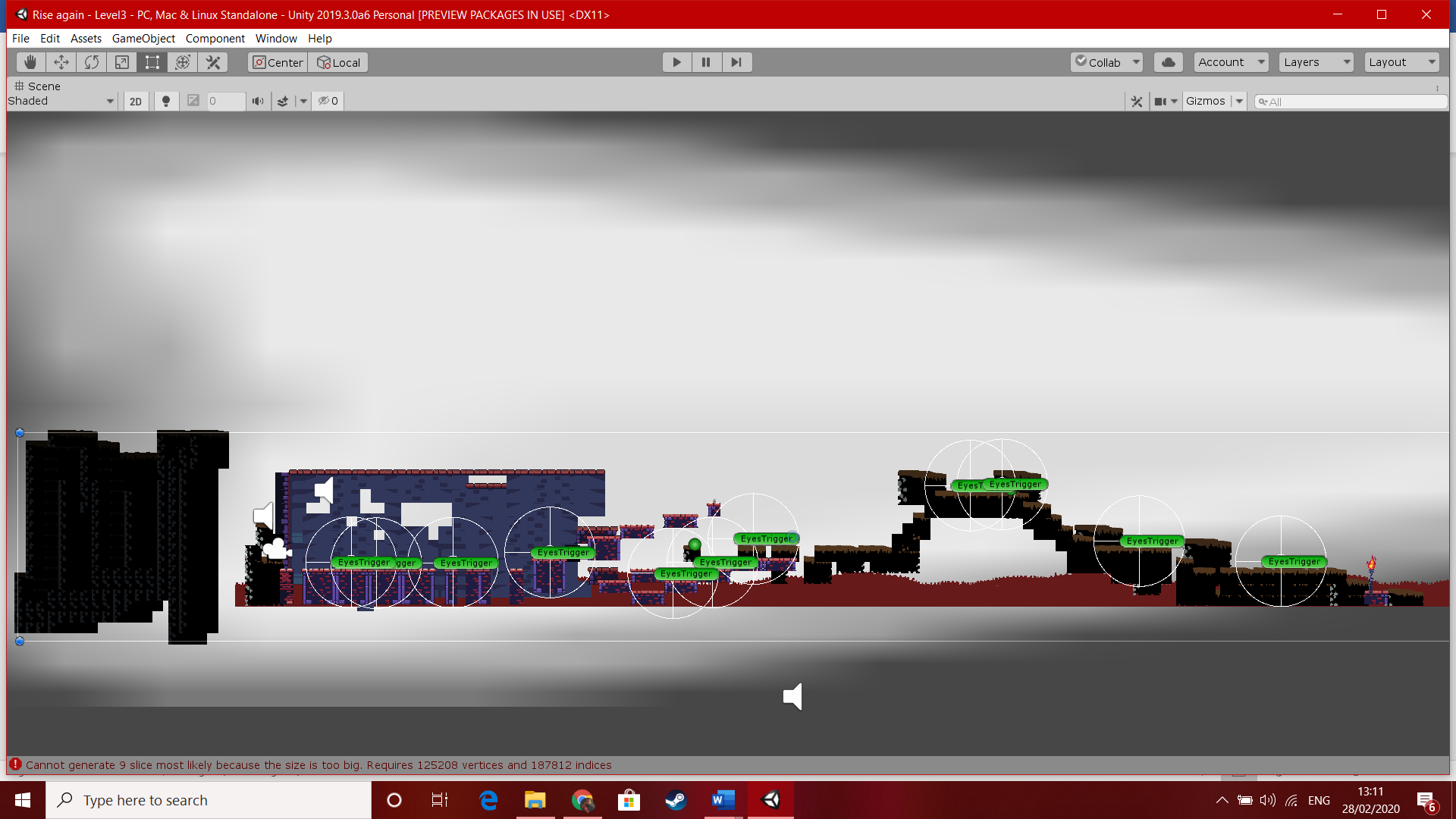
Each environment had to feel helpless and broken for the player to get a feeling of complete dread when playing, For example when the first light level begins there is still light throughout but the more you go into the game there is less natural sunlight covered by destruction and evil.

1st level 

2nd level



## 3rd level



## 4th



## The darker levels

These levels are mostly the same by design but have harder enemy’s and no restarting once you die that’s it.

1st level



## 2nd level



## 3rd level



4th level



**Audio used in game and menus**

Searching

We\_re the Resistors

Ascending

Come and Find Me - B mix

Arpanauts

Digital Native

Underclocked (underunderclocked mix)

Jump sound

Death sound

Arrow sound

**Other Code Used.**

To next scene

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class ToNextScene : MonoBehaviour

{

private int nextSceneToLoad;

// Start is called before the first frame update

private void Start()

{

nextSceneToLoad = SceneManager.GetActiveScene().buildIndex + 1;

}

// Update is called once per frame

private void OnTriggerEnter2D(Collider2D collision)

{

SceneManager.LoadScene(nextSceneToLoad);

}

}

This basically lets the player collide with the goal to then continue onto the next level.

**Bug fixes**

|  |  |
| --- | --- |
| Enemy unable to patrol or walks of ledge | This has been fixed by moving the environment which the enemy was clipping into and the patrol was changed so the enemy’s could only fall if chasing the player or being pushed by the player. |
| Enemys were unable to damage the player in levels beyond the first one | I fixed this by deleting the player then placing him back in the levels. |
| Music from other levels played when the player quit the game. | I re imported all the sounds and music and it seems to have fixed it. |
| Full screen not appearing when start up | This seemed to only be an issue on uni pcs it went full screen on other pcs. |

**User interface and hud**

The Hud only consist of the players health bar. Leaving the player to be able to see every platform and enemy on screen.

## 

## **Engagement of game**

Through out the whole game the enemies will come in more numbers and they increase in speed each level. It also doesn’t help the player that they attack on site and patrol the area while making it nearly impossible to preform large jumps while a pile of enemies are chasing you, and with turrets going for you during jumps making it more deadly the further you go.

The game it self does give the player a hard time making it nearly impossible at certain points but if you can time your jumps it makes the game easier throughout. This is the main advantage to the player if your good at jumping you really don’t need to kill a single enemy and this does get harder throughout as there health increases and their damage.

## Controls

WASD to move

Left click to use sword

Right click to use bow

Shift to role

E to interact

## Software Used

* Unity
* GitHub
* Google Drive
* Maya 2018
* Discord
* Audacity

# Requirements Specification

## Assets Required

|  |  |  |
| --- | --- | --- |
| **Models** | **Textures** | **Sounds** |
| * Player * Player Death * enemy * Enemy death * Sword * bow * arrow * turrets * bosses * flame * Level * Platforms * Potions | * Fire | * Player jump * Sword swing * Switching weapons * Enemy Idle * Enemy death * Player hurt * Player death * Turret sound * Level music * Menu music |

## Player Interface

* Health
* Corrupting Heart

## Arena

* Hexagon shaped
* Suspended in the air
* Falling off edge teleports player back to centre
* Tron aesthetic
* Neon lights

## Player

* Average HP
* Infinite ammo
* Average size
* Decent movement speed can also roll by pressing shift.

## Enemies

* Desire to kill player
* Becomes ash after death

### Undead - Melee

* Melee type enemy
* Fast movement speed
* Small size
* Aggressively follows the player
* Low health - 50 HP
* 25 points if killed
* Weapon - Claws
* Tries to swarm the player

### Turrets - Shooter

* Ranged, projectile type enemy
* Average movement speed
* Average size
* Tries to stay just within attacking range of the player - 8 meters
* Medium health - 100 HP
* 50 points if killed
* Weapon - Medium fire rate projectiles
* Will get within range of the player and circle around them, continuously shooting

### Boss - ???

* Ranged, projectile type enemy
* FAST movement speed
* NORMAL size
* Follows player
* High health - 300 HP
* Main Weapon - bow
* Extra Weapon - sword
* This is a shadow of the player if you can do it so can he.

## Weapons

### Sword

* This weapon only hits the ememy back you are an archer that is were the damage is
* Medium attack speed
* Medium damage

### Bow

* FAST fire rate
* High damage
* Needs to charge before firing - 1.5 seconds

### Usability testing

Usability testing will be covered by me and friends. The aim of this is to give me feedback on how the game runs and operates.

### Functionality testing

Functionality testing will be covered by me where I will look over each script, class and method for errors or mistakes. As part of functionality testing i will produce a test strategy and test log.

The methods of testing are discussed in greater detail below:-

## Testing Methods

Before our game is complete it is a good idea to lay out a test plan which will outline what approaches and methods we will use to test the functionality of our classes and the app’s overall usability. This will be done through the following methods:-

* **Unit testing** – the testing of individual components such as methods, functions and classes through ‘dummy’ code that won’t be used in the final prototype build of the game.
* **Integration testing** – the testing of how all of our classes work together without causing errors or crashes. Integration testing is usually done through ‘top-down’ and ‘bottom-up’ approaches. ‘Top down’ approach means to test elements at the top most level. Typically this means from the user’s/player’s point of view. ‘Bottom up’ approach means to test from the opposite end and start from the system’s functions upwards.
* **System testing** – System testing is done after most of the coding is done to see that your program matches the functional and non-functional requirements stated in the brief. This is as simple as making sure the program produces the correct results when run.

Additionally these methods can be categorized as either ‘White box’ or ‘Black box’ testing. ‘White box’ is a method of testing that tests the internal structure and workings of a program where as ‘Black box’ is a method which just covers the program’s functionality and output. For example, both my Unit testing and System testing will be considered as Black box testing and the Integration testing will be treated as white box testing.

# Sound Design

A lot of the sounds and music I got were from online and it allowed me to chose the right tone for each level and sound for the enemies and player. Were dark levels the music get more chaotic than the light levels were its more heroic sounding a slower based. The enemis I really wanted the player to get the effect that you struck down a person pretty much and the sense of dome and gloom from the player characters death.

### Mechanics

The game mechanics were mostly inspired by games I played in the past that being megaman x. This effects directly into my character being able to move easy and jump and role the same as well. In the final build I plan on implementing a wall jumping mechanic based on this. The bow and arrow mechanics are based on a more medieval version of the blaster as well ie the longer you hold the shoot button the stronger the attack I did the same with the bow and arrow. The sword is purely to move enemies out of your way. In the final build I may add an easy mode with instant sword kill but it kind of ruins the story. The rolling mechanic lets you quickly avoid and enemy or you can use it to cross a platform easier. The enemy’s only attack if they see you that includes the turrets so watch out. To chose between which level you want light or dark this was simply done by having separate portals for each in the first level then it will translon you to which ever choice you made.

### Story/Backstory

The story is very important to me and it certainly is a main focus of the game giving the player the choice and the reason to do what they wish. Although I am still debating if I want to add voices for these scenes or just have it be text the story can be found below.

You are a praised archer the best at what you do. You make your way to the castle to be knighted for your efforts but without any warning the sky becomes dark and grim with fire falling from the sky. You are hit knocked unconscious with the world around you going to hell

All you hear is a voice in your head saying only one thing “Follow the Flame”. You awaken with your face burned and demonic energy ingulfing you. You pick up a burned cloth and cover your scars then continue to pick up your bow and sword. The rest is up to you do you follow your heroic ideals which have brought you to where you are now, or do you follow the Flame.

In final version of the game you will be able to follow your own path save the town or follow the flame to choose a darker scenario.

### Enemy Design

The enemies a wanted them to look as undead as possible and you don’t get more than that with skeletons. I also wanted there death to look good as I want it to effect the player these were people once and you chose to do this as the enemy of the game has suffered the same fate as the player. The turrets I wanted a demonic looking crossbow and I think it could be better I might change it for the final version.

### Aesthetic

I like the medieval look and I like story’s about knights but I also like a good horror story and that’s how you get the world of zero. I wanted the towns to look like your basic rpg quest town but then when everything goes to hell so does the level. I want the player to look at the place afterwards and feel dread.

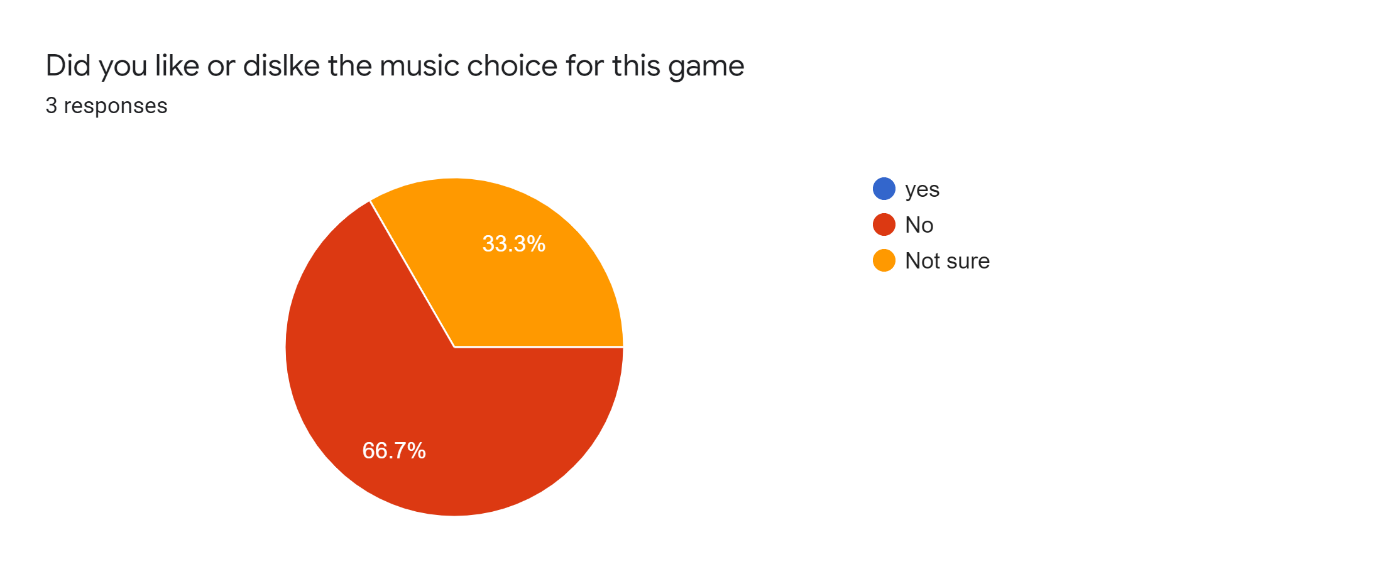
# ***Test Log***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Input** | **Expected Results** | **Actual Results** | **Comment** |
| Player movement | ‘W’ | Player moves forward | As expected |  |
|  | ‘S’ | Player moves backwards | As expected |  |
|  | ‘A’ | Player moves left | As expected |  |
|  | ‘D’ | Player moves right | As expected |  |
| Pause game | ‘Esc’ | Game freezes, Player HUD is disabled, Pause menu and controls panel enabled | As expected |  |
| Jump | ‘Space’ | Player jumps | As expected |  |
| Roll | Shift | Player roles as expected | As expected |  |
| Sword attack | Left click | Player slashes with their sword | As expected |  |
| Quick bow attack | Right click | Player quickly launches arrow | As expected |  |
| Long hold bow | Right click hold | Player holds the arrow for as long as they like | As expected |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Player hurt by enemies | Enemy attacks player | Player loses some health | As expected |  |
| Enemies hurt and killed by player | Attacking enemy | Enemy loses health until it reaches 0, death animation plays, enemy model disappears | As expected |  |
| Game Over | Player health reaches 0 | Game pauses, Game Over panel | As expected |  |

# C:\Users\Charles\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\3592D209.tmp

# C:\Users\Charles\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\6F99CFBB.tmp

# Difficulties



* Sound
* I cant seem to get the enemys to make there death sounds and the same for the player.
* Other sounds seem to work like the music so I don’t understand but I plan to fix it.
* Programming

Getting the enemy to stop falling of the side of the map I eventually fixed this by using colliders.

* Animation
* I don’t Think the main player looks bad but I feel some improvements are needed.

# Sources/Credit

* Brackeys YouTube channel
  + Video tutorials for basic game mechanics/concepts
* Freesound.org
  + Various sound effects
  + And music
  + Main menu theme
  + Gameplay theme
* Anonymous playtesters
* Unity asset store for providing some very useful assets.
  + Giving feedback on gameplay mechanics, difficulty scaling, overall look and feel of the game

### Technical Design

This game will be developed using the unity engine and will be programmed in c# with assets both developed and borrowed from the unity asset store page . The game will run on both windows pc and Linux system and should work on the most up to date versions.Most of the hard ware I use from home or bring in my laptop the current processor is and i7 and a nivida 1070 graphics card and 16gb of ram. Most files will be unitys meta files and sound will be wav files. Images will be both png and jpeg. I also use a tile map software which is realy easy to use with unity’s default software. Other software I will use are maya and github and software called piskel for models.

## Showing my work online

I mostly uploaded clips to my YouTube channel as its really easy to do

You can view that at this link

<https://www.youtube.com/channel/UC8TOz1Oj1LxmBeAT_OQeY-g?view_as=subscriber>

### 

# 