metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**ISTINYE UNIVERSITY**

**ENGINEERING AND NATURAL SCIENCES FACULTY**

**COMPUTER ENGINEERING DEPARTMENT**

**CAPSTONE PROJECT 1**

**PRELIMINARY REPORT**

16 December 2022

**PROJECT TITLE**

No Bugs Were Included

**PROJECT MEMBERS**

Ahmed Muaz Topalan - 190722089

Cankat Sezer - 190722064

Mehmet Naim Erdem - 190722065

**ADVISOR**

Dr. Öğretim Üyesi Buse Yılmaz

**ABSTRACT**

The project is a video game called “No Bugs Are Included”, written in “C#” [1], made with “Unity” [2]. It is a 2D platformer with rogue-like elements. It is based around insects. We have a class mechanic in the context of playable characters being insects such as ants, flies, bees etc.. We have a main villian - not going to spoil it - who is one step higher on the food chain. He/she is mind controlling other insects and evslaving for his own needs. Our characters try to basicly take him over and de-throne him.

**TABLE OF CONTENTS**

[**INTRODUCTION 1**](#_heading=h.gjdgxs)

[**LITERATURE REVIEW 2**](#_heading=h.30j0zll)

[**MATERIAL AND METHODS 3**](#_heading=h.1fob9te)

[**EXPECTED RESULTS UNTIL THE NEXT REPORT ON THE PROJECT 4**](#_heading=h.3znysh7)

[**GANTT CHART 5**](#_heading=h.tyjcwt)

[**REFERENCES 6**](#_heading=h.3dy6vkm)

[**LIST OF TABLES 7**](#_heading=h.1t3h5sf)

[**LIST OF FIGURES 8**](#_heading=h.4d34og8)

[**APPENDIX LIST 9**](#_heading=h.2s8eyo1)

# INTRODUCTION

The subject of our project is the fact that there is A lack of platformer – rogue-like types of games in the industry right now. Solution to the lack of games in the genre and industry is to create one.

Our motivation is to create a final product which the audience are going to find entertaining. We have a detailed roadmap for the game which we are going to follow accordingly.

Game offers unique gameplay mechanics, rich story telling, astonishing visuals and joyful soundtrack.

Design theme of the game is cartoon-themed, rogue-like platformer. Player is going to choose between multiple insects to play as, one of them, being an Ant.

Three of our playable characters are ;

* Ant’onio the Ant
* Beetlejuice the Beetle
* Buzz the Bee

After the player has chosen the character, game starts in the first level. Level design in the game is lineer, but they come with optional pathways at the end.

Game’s scenario is that, there is main villain who controls other insects and enslaves them into taking care of his/her needs. Other insects try to stops us while we are trying to make our way into the wast world of “No Bugs Were Included”.

Replay-ability is an important driving factor for our project, firstly because it is a rogue-like, and secondly for our audience to be able to play it on repeat and enjoy the game multiple times.

# LITERATURE REVIEW

There are good examples of these types of games in the industry, but they are scattered across a long time span, this leads to a scarce amount of games for the audience to play and experience.

To give an example, here are two games in the same category;

* Ori and The Blind Forest :Will of Wisps
* Dead Cells

**Dead Cells**

Gameplay in “Dead Cells” [3] involves navigating procedurally generated levels, battling various enemies and bosses, and collecting weapons and other equipment to improve the player's character. The game has a fast-paced, hack-and-slash combat system, and players can choose from a variety of weapons, including swords, bows, and shields, as well as unlock abilities and skills as they progress trough the game.

The game's environments are detailed and varied, ranging from dark and gloomy dungeons to bright and colorful gardens. The character designs are also highly detailed and expressive, with each enemy and boss having their own distinct appearance. In addition to the hand-drawn art style, “Dead Cells” also incorporates 3D elements, such as particle effects and lighting, which help to bring the world of the game to life and add depth to the visual experience.

“No Bugs Were Included” has a similar art style to “Dead Cells”, but the key difference between two games is the fact that, “Dead Cells” is a fast paced, hack-and-slash type of game ; “No Bugs Were Included” has a slow pace compared to it.

**Ori and The Blind Forest :Will of Wisps**

Gameplay in “Ori and the Blind Forest: Will of the Wisps” [4] combines platforming, exploration, and combat. Players must navigate through various environments, avoiding obstacles and enemies, while also collecting power-ups and abilities to improve Ori's abilities. The game features a range of abilities, including double jumping, wall-climbing, and dashing, which can, be used to solve challanges and overcome obstacles.

One of the key mechanics of “Ori and the Blind Forest: Will of the Wisps” is its focus on exploration and discovery. Players must explore the game's environments and uncover secrets and collectibles hidden throughout the world. The game also features a variety of enemies and bosses, which players must defeat using a combination of platforming skills and strategic combat.

“No Bugs Were Included” has a similar tempo and combat system to “Ori and the Blind Forest: Will of the Wisps”, but the key difference between two games is the fact that, “Ori and the Blind Forest: Will of the Wisps ” has much richer visuals and sound design.

# MATERIAL AND METHODS

* Programs we use for drawing characters and environment
* Programs we will use to make game soundtracks
* Real Time Lightning Components

“Aseprite” [5] will be used to create our sprites for the material we will be using. “Aseprite” is a proprietary, open-source image editor primarily used for pixel art drawing and animation.

 

Figure 2  
Assasin Bug

Figure 4  
Mantis

Figure 1



Figure 6  
Ant

Figure 5  
Ant

Figure 3  
Beetle

Figure 1  
Wasp

“FL Studio” [1] and other pixelated sound generators will be used for sound effects and music. It is a digital audio workstation.

The “Tilemap” [7] component is a system that stores and manages Tile Assets for the purpose of creating 2D levels. It transmits the necessary data from the Tiles positioned on it to other associated parts.

A “particle system” [8] simulates and produces many tiny pictures or Meshes, also known as particles, to create a visual effect. Each particle in a system represents a unique graphic component of the effect. The system simulates each particle collectively to provide a sense of the overall impact.

“Real-time Lights” [9] are Light Components which are rendered in Realtime. We use this feature for Lights that need to change their attributes or are generated by scripts while playing. In Unity, the lighting for each frame is calculated and updated in real time. They can change in response to the player's actions or events in the Scene. Real-time lights illuminate both static and dynamic game objects, casting realistic shadows. Realtime Global Illumination (Realtime GI) and real-time lights can also be used to provide indirect lighting for static and moving game objects.

# EXPECTED RESULTS UNTIL THE NEXT REPORT ON THE PROJECT

**What remains to be done until the final report ?**

* All levels are ready to be play tested in order to get feedback from players.
* Minimum of 3 playable characters.
* Weapons and a combat system.
* Health, checkpoint, death, in-game-economy systems.
* Bosses.
* Music and sound effects.
* Particle systems.
* In-game activities and interactable NPCs.

**What does it take to make these plans come true ?**

We have to constantly adapt our scripts because we anticipate the fact that there are going to be bugs. We will conduct numerous player tests and bug fixes as a result. We intend to use player feedback to fix bugs and enhance the game as a whole.

**What method is planned to be followed to solve the problems encountered ?**

Our method to solve the problems encountered involves a lot of tools. One of them is the “Git Desktop app” [10]. Thanks to the app, we will update the game in synchronicity. We will also have all the versions of the game in advance.

This step is important because if we ever make a mistake which can’t be reversed, we can go back to older versions to start where we left from. We will also use debug.log inside “Unity”. These logs will help us understand what and where the problem is. There are also debug tools for “Unity” and we will seek the aid of these debug tools too.

# TIME TABLE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project Step | Date started | Time it took | Date ended | E/R |
| Early phase of creating the idea | 8.11.2022 | 7 | 15.11.2022 | Expectation |
| 8.11.2022 | 5 | 13.11.2022 | Reality |
| Solidating the idea and creating some assets | 16.11.2022 | 14 | 30.11.2022 | Expectation |
| 14.11.2022 | 16 | 30.11.2022 | Reality |
| Creating basic scripts and in game design | 1.12.2022 | 7 | 8.12.2022 | Expectation |
| 1.12.2022 | 6 | 7.12.2022 | Reality |
| Writing Prelminiary Report | 9.12.2022 | 3 | 12.12.2022 | Expectation |
| 8.12.2022 | 4 | 12.12.2022 | Reality |

# REFERENCES

|  |  |
| --- | --- |
| [1] | «C#,» Microsoft, [Çevrimiçi]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/. [Erişildi: 16 12 2022]. |
| [2] | «Unity,» Unity Technologies, [Çevrimiçi]. Available: https://unity.com. [Erişildi: 16 12 2022]. |
| [3] | «Dead Cells,» Motion Twin, [Çevrimiçi]. Available: https://dead-cells.com. [Erişildi: 16 12 2022]. |
| [4] | «Ori-The-Will-Of-Wisp,» Moon Studios, [Çevrimiçi]. Available: https://orithegame.com. [Erişildi: 16 12 2022]. |
| [5] | «Aseprite,» Igara Studios, [Çevrimiçi]. Available: https://www.aseprite.org/. [Erişildi: 16 12 2022]. |
| [6] | «FL Studio,» [Çevrimiçi]. Available: https://www.image-line.com. [Erişildi: 16 12 2022]. |
| [7] | «Unity User Manual,» Unity Technologies, [Çevrimiçi]. Available: https://docs.unity3d.com/Manual/class-Tilemap.html. [Erişildi: 16 12 2022]. |
| [8] | «Unity User Manual,» Unity Technologies, [Çevrimiçi]. Available: https://docs.unity3d.com/Manual/ParticleSystems.html. [Erişildi: 16 12 2022]. |
| [9] | «Unity User Manual,» Unity Technologies, [Çevrimiçi]. Available: https://docs.unity3d.com/Manual/realtime-gi-using-enlighten.html. [Erişildi: 16 12 2022]. |
| [10] | «GithDesktop,» Microsoft, [Çevrimiçi]. Available: https://desktop.github.com/. [Erişildi: 16 12 2022]. |

# LIST OF TABLES

# 

# LIST OF FIGURES

Figure 1…………………………………………………………………….………………………Wasp Concept art

Figure 2……………………………………………………………….…….…………….Assasin Bug Concept art

Figure 3…………………………………………………………….……….……………………..Beetle Concept art

Figure 4………………………………………………………………….…….…………………Mantis Concept art

Figure 5………………………………………………………….……………….……………………Ant Concept art

Figure 6……….……………….……………….……………….……………….……………….……Ant Concept art

# APPENDIX LIST

##### We can’t include every script here because we'll use a variety of them. We will add all of our scripts into a git repository with the advisor's approval, and we'll provide the link to it here as well.

##### For a brief example our movement script that will be used to move the player character is shown below.

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEditor;

using UnityEngine;

public class MovementScript : MonoBehaviour

{

public float jumpForce = 10.0f;

public float speed = 1.0f;

private float moveDirection;

private bool jump;

private bool grounded = true;

private bool moving;

private Rigidbody2D \_rigidbody2dD;

private SpriteRenderer \_spriteRenderer;

void Start()

{

\_rigidbody2dD = GetComponent<Rigidbody2D>();

\_spriteRenderer = GetComponent<SpriteRenderer>();

}

private void FixedUpdate()

{

if (\_rigidbody2dD.velocity != Vector2.zero)

{

moving = true;

}

else

{

moving = false;

}

\_rigidbody2dD.velocity = new Vector2(speed \* moveDirection, \_rigidbody2dD.velocity.y);

if (jump == true)

{

\_rigidbody2dD.velocity = new Vector2(\_rigidbody2dD.velocity.x, jumpForce);

jump = false;

}

}

private void Update()

{

if (grounded == true && (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.D)))

{

if (Input.GetKey(KeyCode.A))

{

moveDirection = -1.0f;

\_spriteRenderer.flipX = true;

}

else if (Input.GetKey(KeyCode.D))

{

moveDirection = 1.0f;

\_spriteRenderer.flipX = false;

}

}

else if (grounded == true)

{

moveDirection = 0.0f;

}

if (grounded == true && Input.GetKey(KeyCode.W))

{

jump = true;

grounded = false;

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if (collision.gameObject.CompareTag("Ground"))

{

grounded = true;

}

}

}