Technical Design Document

Team Name: Rubbish Wakanda Ibis (RWI)

GAME: Mal’s Life

|  |  |
| --- | --- |
| Names | Roles |
| Daniel | Artist – Animations, Materials |
| Nathaniel | Artist – Rigging, Models |
| Ryan | Artist – Environment, Character Models |
| Mal | Designer – Level Design, Game Design |
| Blake | Designer - Level Design, Game Design |
| Sarthak | Programmer - Game Functionality |

Contents

[Development Environment 2](#_Toc22030489)

[Game Overview 2](#_Toc22030490)

[Genre, Perspective, Controls, and Platform 2](#_Toc22030491)

[Technical Goals and Features 2](#_Toc22030492)

[Technology Benchmark 2](#_Toc22030493)

[Game objects and logic 2](#_Toc22030494)

[Game Flow 2](#_Toc22030495)

[Custom Game Systems 2](#_Toc22030496)

[Mechanics 3](#_Toc22030497)

[Core Mechanics 3](#_Toc22030498)

[Graphics 3](#_Toc22030499)

[Artificial Intelligence <If applicable> 3](#_Toc22030500)

[Physics 3](#_Toc22030501)

[Game Flow 3](#_Toc22030502)

[Levels <If Applicable> 3](#_Toc22030503)

[Interface 3](#_Toc22030504)

[Asset List 3](#_Toc22030505)

[Technical Risk Management 3](#_Toc22030506)

# Development Environment

**Game engine –** Unity

**IDE –** Visual Studio 2017

**Source Control –** Multiple commits on the GitHub.

**Third Party Libraries –** Photoshop, MS Paint, Github (for source control)

# Game Overview

## Genre, Perspective, Controls, and Platform

**Genre –** The game will be a 2.5D platformer.

**Perspective –** 2.5D

**Controls -**

|  |  |
| --- | --- |
| Keys | Function |
| A | Moves player to the left of the screen |
| D | Moves player to the right of the screen |
| Space Bar | Moves player to the top |

**Platform –** The target platform is PC 32-bit and 64-bit.

## Technical Goals and Features

|  |  |
| --- | --- |
| Technical Goals | Features |
| Creating good transition between controls | Allows player to feel the character is more responsive |
| Good Jumping | Since the game is a platformer smooth jumping mechanics will allow the player to be more fun and intuitive. |
|  |  |

## Technology Benchmark

Minimum-

Maximum-

## Game objects and logic

|  |  |  |
| --- | --- | --- |
| Game Object | Behaviour | Purpose |
| Wall | Static | Does not allow the player to pass through and remove the ability to jump. |
| Platforms | Static | Does not allow the player to pass through and gives player ability to jump |
| Windows | Static | Turn gold if goal, then upon collision with player turn back to original colour. |

## Game Flow

(Start) -> Collect Input -> (perform player movement) -> <If Player on Window> -> (Reaches Goal) -> otherwise continue loop (<-).

## Custom Game Systems

Random Window System

(Start) -> <if Window not active> get Random Number -> (use found number to as index to find window in list) -> send window index data -> (turn window data to goal).

# Mechanics

## Core Mechanics

|  |  |
| --- | --- |
| Mechanics | Technical Implementation |
| Stress Meter (In Progress) | When completed will include a visual representation of the stress from GUI. The GUI will consist of the slider bar, and the increase in stress will be due to increase in time. The stress will decrease upon reaching goal and giving flyer. |

# Graphics

# Artificial Intelligence <If applicable>

Thoughts for adding in a game object of birds. The birds will have its own attach and flee behaviour.

# Physics

Player physics are depended on the original Unity physics system. The features include a rigid body on the player allowing an increase in speed when jumping and falling.

**Relation**

When player interact with wall, the wall will stay static not letting player collide through. The same relation is with platforms.

**Issues**

The players will get stuck on horizontal angle to platforms so that can be fixed by many tutorials on YouTube displaying better collision mechanics.

# Game Flow

For now, the game only consists of only one level.

**Objectives**

Player must make it to the golden window which for now is a representation of the goal.

**Evaluation of Player Progress**

The player progress will be described through the stress meter and timer which will contribute to the score. If the player reaches a certain score they can move to the next level. So far since it is one level once the player collides with the golden window (goal) the player will gain one score, and the timer will increase. The level is loaded in upon start, but the new goal is determined through runtime.

**Loading levels**

Already rendered in objects are loaded in when the game starts.

# Levels <If Applicable>

For now no level progression included.

# Interface

Currently there is no interface in the game.

**Camera Operation**

The camera has only one operating factor which is to follow the player.

(Start)->(follow Player) -> (Loop to Start)

**Player Controls**

|  |  |
| --- | --- |
| **Player Controls** | **Use explained** |
| **A** | Makes player move to the left of the screen. Adds force to the left on the horizontal axis. |
| **D** | Makes player moves toward right if the screen. Adds force to the right on the horizontal axis. |
| **Space Bar** | Makes player move up the screen at a parabolic curve. Adds force on a Vector3 up and decrease upon a range of velocity. |

# Asset List

**Folders in The Unity Project**

* Prefabs Folder – Will contain the prefabs of created items
* Scripts Folder – Will contain the scripts for functionality
* Animations – Contains all created animations
* Materials – Contains the materials of every object in the game

# Technical Risk Management

|  |  |
| --- | --- |
| **Potential Risk** | **Risk Avoidance Methods** |
| **Implementing other factors without diluting the stress factor.** | Will place the effect of stress on each new idea of a game object. This will make the stress factor become even more common, and not be forgotten. |