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 |

|  |  |
| --- | --- |
| Version | Added Features |
| **0.1** | Added Title Headings |
| **0.2** | Filled in all most all Title headers. |
| **0.3** | Changed Table Formatting, to be Blue instead of white |
| **0.4** | (In Game) Added Sample Movement and collision system |
| **0.5** | Fixed up TDD, where needed depending on review |

Contents

[Development Environment 2](#_Toc22129071)

[Game Overview 2](#_Toc22129072)

[Genre, Perspective, Controls, and Platform 2](#_Toc22129073)

[Technical Goals and Features 3](#_Toc22129074)

[Technology Benchmark 3](#_Toc22129075)

[Game objects and logic 3](#_Toc22129076)

[Game Flow 3](#_Toc22129077)

[Custom Game Systems 4](#_Toc22129078)

[Mechanics 5](#_Toc22129079)

[Core Mechanics 5](#_Toc22129080)

[Graphics 5](#_Toc22129081)

[Artificial Intelligence <If applicable> 5](#_Toc22129082)

[Physics 5](#_Toc22129083)

[Game Flow 6](#_Toc22129084)

[Levels <If Applicable> 6](#_Toc22129085)

[Interface 6](#_Toc22129086)

[Asset List 7](#_Toc22129087)

[Technical Risk Management 8](#_Toc22129088)

# Development Environment

**Game engine –** Unity

**IDE –** Visual Studio 2017

**Source Control –** Multiple commits on the GitHub on personal account. Multiple commits on GitKraken.

**Third Party Libraries –** Github (for personal source control), and GitKraken.

# 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-**

* **CPU:** Core i3 2.4GHz
* **RAM:** 4GB RAM
* **HDD:** 16GB Free Space
* **GPU:** Intel HD 4000 (integrated graphics)
* **OS:** Windows 7, 8, or Windows 10 64-bit
* **DirectX**: Version 11 or better

**Maximum-**

* **CPU:** Core i5 2.8GHz
* **RAM:** 8GB RAM
* **HDD:** 16GB Free Space
* **GPU:** Nvidia GeForce GTX 660 / AMD Radeon HD 7870 or equivalent video card with dedicated memory of 2GB or higher VRAM.
* **OS:** Windows 7, 8.1, or Windows 10 64-bit
* **DirectX**: Version 11 or better

## Game objects and logic

|  |  |  |
| --- | --- | --- |
| Game Object | Behaviour | Purpose |
| Wall | Static | Does not allow the player to pass through and removes 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 (<-).

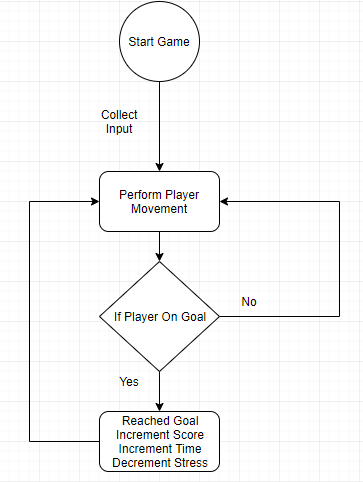


Figure 1. Displays Game Flow

## 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).

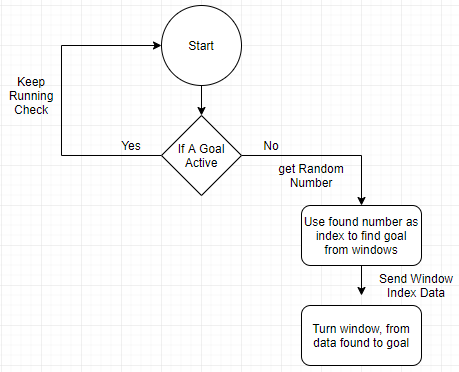


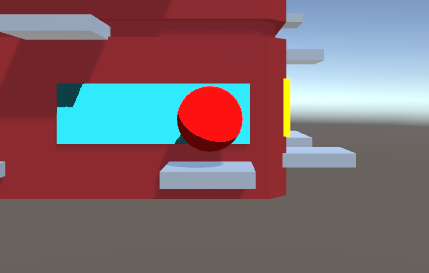
Figure 2. Sets Goal

# 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

**Perspective-** The 2.5D perspective is displayed in the referenced image below.   


**Art Style –** The team has decided on the art style to contain a toon shader, since the art style is cartoony and stylized.

**Graphic features –** The graphic features include stress meter and timer. The stress meter will be placed on the bottom left hand side of the screen, filling up red to indicate the stress, as being bad. The timer will be displayed at the top right of the screen, counting down.

**3D detail expected poly count –** The expected poly count for the model will range around 10-20k. So taking that as reference the entire project poly count should be near 30-40k.

# 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

**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. At the current moment there are ideas of multiple levels but in current builds we have only one 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, and the player will reach the win screen upon a certain score. 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 specific behaviour is intended.

# 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 forever)

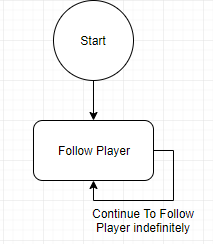


Figure 3. Camera Operation

**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

**Asset List**  
•    Art Assets  
o    Building  
o    Mal (player character)  
o    Cat  
o    Birds  
o    Pot plants  
o    Bin  
o    Building plants  
o    Vines  
o    Window sill/ledges  
o    Paper stack/Flyers  
o    Flag Pole  
o    Window Cleaning scaffold  
•    Scripts  
o    3rd person Player controller script  
o    Player engine  
o    Toon Shader  
o    Toon Mat  
o    Jump Physics  
o    Jump Script  
o    Attach Player  
o    Detach  
o    Check If Goal  
•    Text & Sounds  
o    Stress font text  
o    UI text  
o    Sound effects  
o    Background music

# 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. |
| **Team has no previous knowledge of push and pulling on Git. This may cause some data to be potentially deleted or corrupted, if not managed.** | Have them call out each time they are pulling or pushing something, so that no one merges to files and end up breaking them. As well as going through Git push, pull methods to each teammate. |