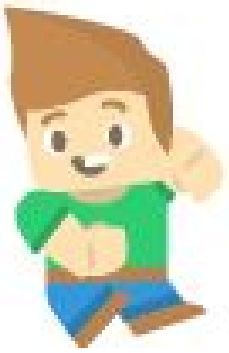


Design Document for:

Kenny The Hopper

The Educational Platforming Game

Raiders of the Lost Taco 

Game Available at: https://goo.gl/mMViGR or https://tinyurl.com/KennyTheHopper

Written by Daniel Magro, Nathan Galea, Luke Ellul, Stephanie Chetcuti and Kevin Tonna

Version # 1.00

19 December 2017

Table of Contents

Design History

Version 0.1

The basis of the game was created in this version.

- A basic level was designed with plain and static platforms was created
- A placeholder for the character was created and the basic game mechanics were implemented, i.e. the movement script (PlayerMovement.cs) as well as the colliders for the player and the platform.

Version 0.2

The decision platforms were added, introducing the educational aspect of this game.

- Correct Platforms which allow the player to progress through the level
- Incorrect Platforms which collapse when the player jumps on them

Version 0.3

An Enemy (Red Alien) was introduced into the game.

- A script (Damage.cs) was written for the enemy which follows the player throughout the level, attempting to throw them off platforms and deal damage. The script takes care of deducting the player's HP on collision, and indicating a 'Game Over' when the player's health reaches 0
- A script (Health.cs) was written which keeps track of the player's health and displays it on screen

Version 0.4

The playable level was changed from a static one to one which is dynamically generated based upon the inputted sequence of steps the user would like to learn, as well as making the level different for each playthrough, even given an identical sequence of steps (random platform placement and incorrect platform option)

- The workings of this mechanic are explained near the end of the document, in the 'Dynamic Level Generation' section.

Version 0.5

Graphics were added to the game.

- Sprite Sheets were downloaded for the player. The sprite sheets were sliced and then combined to create the animations. The animations were combined with the Moving.cs script so that the character displayed the relevant animation to the action it was performing.
- A sprite was also found for the Red and Green Alien, the enemies of this game.
- The platforms were also replaced with sprites.
- A background was created.

Version 0.6

A User Interface was introduced to the game.

- Main Menu and Level Selector were added which ties the two game modes into one game.
- Pause screen was also implemented
- Game Over screen was included which is displayed when the player's health reaches 0
- Level Complete screen was included for when the player successfully completes the level.

Version 0.7

The Green Aliens were introduced into the game

- Green Aliens were implemented such that one would spawn for every pair of decision platforms. Each Alien would slide back and forth between each platform, trying to throw the player off of a platform. The green aliens can also be used as intermediary platforms to jump onto higher platforms.

Version 1.0

General Polishing

- Background Music was added to each level
- Sound effects were added for several actions and events
- Particle Systems were added to the decision platform to reinforce correct choices
- Some game mechanics were revisited and polished
    - Jumping power and player speed were adjusted
    - Camera mechanics were changed to make it feel as if it were floating after the player rather than being fixed with the player in the centre
    - Difficulty was lowered slightly by reducing the distance between platforms

Game Overview

Philosophy

Aim

This aim of this game is to help students memorize sequences and/or lists. It's designed to serve as an educational game that can be adapted to any subject or topic.

.

Common Questions

What is the game?

The game is a simple 2D platformer that enables the user to jump onto platforms in order to climb higher until the final platform is reached. At times the user will have two platforms at the same y-coordinate to choose from, each containing one step of the sequence, one will contain the correct option, while the other will contain a different incorrect step. If the user jumps onto the incorrect platform, the platform will collapse along with the user. The number and content of the platforms depends on the sequence that is inputted by the teacher, which the student will be learning/memorising.

Why create this game?

As stated before, this game aims to be an educational game, to help students memorise sequences. It takes the form of a simple but fun and addicting game with the hope that the task of memorizing sequences by heart will seem less boring and daunting.

Where does the game take place?

As previously mentioned, the game takes place in a 2D platform world. The user has to simply move upwards until the last platform is reached. The last platform can be identified from the rest because it has a different colour and has a cactus on the edge of it. At times the player will have a choice between two platforms, if the player goes onto the right platform a green light will flash to indicate that it is the correct platform. On the other hand, if the player goes onto the wrong platform, a red light will flash to indicate that it is the wrong platform. The platform will also fall down and thus the player has to restart the sequence.

There exists two types of enemy in the world, green aliens and red aliens. The green aliens simply exist to obstruct you while the red alien will follow you around and decrease your health by 10 points if it hits you.

What do I control?

The player will be able to control a single character. In order to move to the right the player has to press the 'D' key and to move to the left the player has to press the 'A' key. While the space bar will enable the character to jump or fly using the jetpack. When the user presses the 'P' key the game will pause and can be unpaused by repressing the 'P' key.

The user will be able to navigate through the menu screen with the use of the mouse or touchpad.

How many characters do I control?

In this game, the user will only be able to control one character.

What is the main focus?

The main focus is to go through all the steps in a sequence in order. The game is pre-set to the System Development Life Cycle (SDLC) but this can be easily changed to any sequence that the teacher or student wishes.

What's different?

What sets this game apart from others is that it is not fixed, the order of steps can be changed to anything, making the game versatile.

Feature Set

General Features

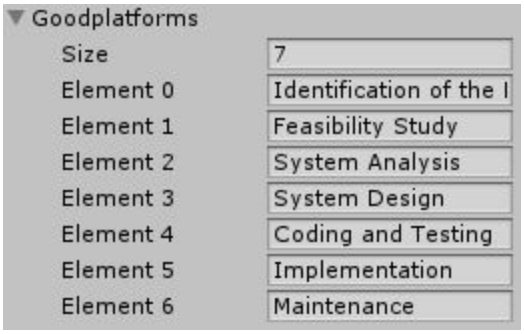
The game is an addictive and fast-paced platformer. The player mustn't take too long to move or else the red alien will deplete the character's health and the player will have to start over. Apart from that, when standing on a platform, the green alien can knock the player off if they take too long to jump to the next platform, or if the player doesn't successfully dodge the alien. Since to win the game, the player must jump onto the correct answer platforms, without even realising it, the player is developing a strong knowledge of the sequence of steps that is being studied. Moreover, correct decisions are reinforced and acknowledged by the game with a green flash, while stepping on an incorrect platform will flash a red light and the platform will collapse along with the player.
Another great feature of this game is that the sequence being studied can be inputted into the editor, and the level is generated based upon that sequence, i.e. the number of decision platforms, correct platforms names and incorrect platforms will be based upon the given input.
Furthermore, even with the same sequence input, each time a level is played, it is randomly generated, meaning that the distance between platforms, incorrect decision platform names etc will be different for each playthrough.

Multiplayer Features

While no multiplayer features were included in the game thus far, one very simple feature that can be added is a score counter, and consequently a High-Scores Table or Leaderboard, with say daily, monthly and all-time high scores for groups of friends or schools/courses.

Editor

Since this is an educational game, an interface is provided to input the sequence which is desired to be taught to players. For example, in the case of the Systems Development Life Cycle (SDLC) the input is as follows:



Gameplay

The gameplay is designed to give the player an objective while simultaneously learning a sequence of steps. The main fun features of the game are the jumping and flying from one platform to the other. We tested different dynamics on the player to provide the user not only with an enjoyable gameplay but also with an addictive experience. The game provides the user with different challenges and includes enemies in order to make the game more competitive and challenging.

The Game World

Overview

In this game there was no Overworld included, the only connection between Levels is the Level Selector in the Main Menu. The level selector was implemented this way because of the time constraints of the Game Jam. Given more time, the level selector could have been implemented such that each level would be a door on a platform, such that the player can jump onto any platform, and when they are on their desired level they can press a key to start that level.

World Editing

Another feature which would have been implemented if not for the time constraints is that the user would be allowed to save the levels he creates using the level editor (i.e. saving the sequence of steps, the level will be dynamically generated as explained in the ‘Dynamic Level Generation’ section) and the level will be added to the overworld.

Rendering System

Overview

Give an overview of how your game will be rendered and then go into detail in the following paragraphs.

2D/3D Rendering

For our project Unity will be used. Unity is capable of both 3D and 2D rendering and in our case will be mostly used for 2D rendering as it deals with images as a gameobject. We also made use of layers in our scene to lay out the different entities of the game

Camera

Overview

The background is attached to the camera, such that the background always appears to stay fixed wherever the player moves. The camera follows the player, however with a slight delay to make it feel as if it were floating.

Camera Mechanics

The camera works using the CameraScript.cs script file.

Game Engine

Overview

The Game Engine manages all the Physics of the game. This is what allows the Player to move in a realistic way, jumping and falling (gravity), getting pushed off of platforms by the aliens etc.

Movement

The movement of the character is made possible, and made to look and feel realistic by the Game Engine.
This specific aspect of the game is managed by the PlayerMovement.cs script, which controls the movement speed, jumping force, and stops the player from double jumping (in mid-air). The movement of the jet-pack is managed by the JetPack.cs script.

Collision Detection

Collision Detection is a very important design element of this game.

- The Player, Platforms and the Enemies are all on separate Layers
- The player can collide with both the platforms and the enemies. This enables:
    - The player to stand on the normal platforms
    - The player to stand on the correct decision platforms
    - The incorrect decision platforms to give way when the player jumps on them
- The Enemies can pass through the platforms but can collide with the player. This enables:
    - The Red Alien to go through the platforms while chasing the player, however still knocking the player off the platform

- The Green Aliens to slide through the decision platforms, enabling them to throw the player off the platform.

Lighting Models

Overview

The project being a 2D and not a 3D model we did not make use of any 3D lighting. The default lighting of Unity 2D was used since it was deemed well suited enough. The entire screen is illuminated equally.

Improvements

Another level was planned to be added, that of a night level. The idea was to have a night scene, with the character holding a lantern, which would emit light. This way the user would only see a limited area of the level. This was not implemented due to the limited time, however could be implemented in the future.

Game Characters

Overview

The main character, which is controlled by the player, is Kenny.

Enemies and Monsters

There are 2 types of enemies in this game:

The Red Alien:

For each level, one Red Alien follows the player around the map, in order to push him off platforms, as well as decrease the player's health. Each time the alien bumps into the player, 10 HP are deducted out of a 100.

The Green Alien:

There are several Green Aliens on each level. Each one flies back and forth between each pair of decision platforms. The green aliens do not decrease the player's HP, however they can knock the player off of platforms. The green aliens can also be used as platforms, however it is quite hard not to fall off them.

User Interface

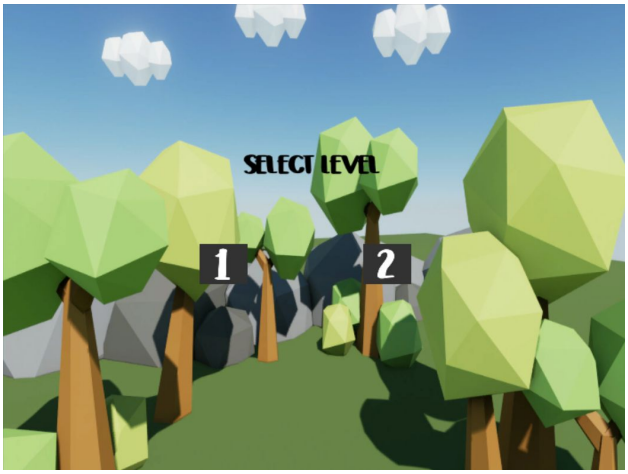
Overview

The user interface can be navigated through using the mouse or touchpad as well as some functionality from the keyboard during gameplay. These will be discussed in detail below.

Main Menu

When starting up the game the user will be met with the following screen, the main menu:

Using the mouse, the user simply clicks on the appropriate button; ‘play’ to start playing the game and ‘quit’ to exit the game entirely. Moving onto the next screen, if the user clicks ‘play’ a level select screen will appear where the user may choose between two options; level 1 and level 2.

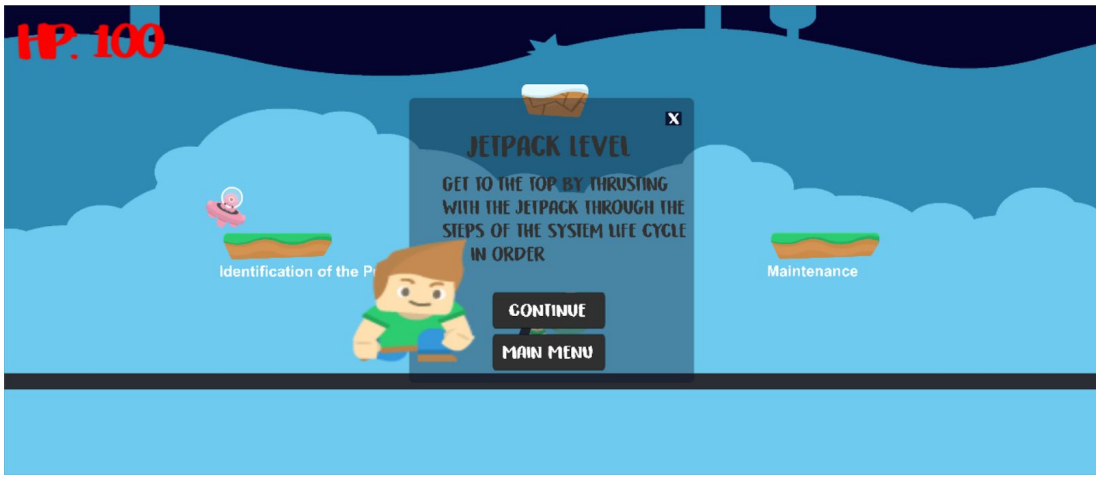


Level 1 is the level where the user navigates through the world using a jetpack while level 2 is the level where the user jumps onto the different platforms.

The background used in the above menus was obtained from the following website: https://nerd-time.com/learning-lowpoly-blender-ue4/

Pause Screen

Once selecting the level, the user will be met with the following menu before the gameplay actually starts:



This menu will display the level name and some information about the level. It also has options to either continue with the game or go back to the main menu. The small ‘x’ symbol at the top right corner works the same as the ‘continue’ button.

This menu will also serve as a pause menu. If the user wishes to pause the game at any time, they may press the ‘P’ key on their keyboard which will halt the gameplay and load up the menu shown above. Pressing the ‘continue’ button or repressing the key ‘P’ will continue the gameplay from where it stopped.

Game Over and End Game Screen

When the player runs out of health, a game over screen is loaded for 2 seconds then returns back to the main menu. This is done using the GameOver.cs script. Similarly, when the user completes a level, a screen is loaded to indicate that the level was completed. This is done using the EndGame.cs script.

Weapons

Overview

This game has no weapons as the user is only asked to evade incoming enemies.

Musical Scores and Sound Effects

Background Music

The game consists of two levels each containing a different background music song loop. The music tracks were obtained from https://opengameart.org/content/platformer-game-music-pack and http://ozzed.net/music/nackskott.shtml#tune06.

Sound Effects

The game is also consists of a number of sound effects to make it more immersive. Sound effects are located in a folder called *sfx* located in the *Assets* folder of the game. The most common used sound effect is probably the **jump** sound effect. This sound effect is played every time the player jumps (hits the space bar). Every time the player jumps on a bad platform (a platform which has the wrong answer) the sound effect *fall* is played. Whenever the player jumps on the right platform (a platform which has the correct answer) the sound effect *pickup* is played. When the player reaches the last platform hence and wins, the sound effect *win* is played.

In Level 1 the player flies on a jetpack. The sound effect *jetpack* is played while the player uses the jetpack.

The songs used for the background music are stored inside a directory called *Music* which is stored in the *sfx* directory.

Single-Player Game

Overview

This game is split up into two levels, each with a different game-mode. Given the limited time available to design and create this game, it was decided to create just two levels in order for them to be complete and well designed rather than having a large number of unpolished levels.

Single Player Game Detail #1

In the first level the player is given a jetpack. This makes the game accessible to a wider audience, i.e. it enables those who are not as good at video games to play and learn from the game. The jetpack makes jumping from one platform to another, and recovering from landing on an incorrect platform easier since the player can 'fly'

In the second level, the player can only jump, and thus incorrect choices are costlier as they usually mean the player has to start the level all over again. Furthermore the player now has to make use of the sliding platforms to visit the higher decision platform and thus the final platform.

Story

The story revolves around the player. Jumping or hovering over the platforms in the correct sequence of the 'System Development Life Cycle' with a few obstacles including the red alien and the green alien. When the player hits the green alien it will deal no damage however it will be quite annoying getting past them without getting stuck so you have to time each jump correctly. The other alien which is the red alien will chase you throughout the whole journey until you land on the final platform. Landing on incorrect platforms will result in the platform falling off.

First Level

In the first level, the player will be equipped with a jetpack that will allow the character to fly through the world and land on the correct platforms.

Second Level

The second level serves as a slightly harder level then the first. In this level, the player will only be able to jump between the consecutive platforms. On top of this, if the user lands on an incorrect platform, they will fall and will not be able to save themselves as easily.

Hours of Gameplay

The hours of gameplay depends on how good the player is and how long is the sequence the teacher inputs. With the implementation provided, i.e. the SDLC, an experienced user should be able to complete this in roughly 5 minutes. The game was designed to be a bit challenging to reach the end and at the same time addictive, as not to bore the user.

Victory Conditions

The game is won by reaching the final platform. To do this, the player must first land on all the correct platforms leading to the final platform.

Character Rendering

Overview

The main player character that the player uses to play is rendered using a spritesheet of the player's different positions. The sprite sheet is obtained from: http://imedia9.net/game-assets-2d-kenney-character-male.html.

The spritesheet is located in a directory *Sprite Sheets* located in the *Assets* directory. There is also another flipped sprite sheet of the main player's sprite sheet in order to handle the player's movement when the player moves in the opposite direction since this was not included in the original sprite sheet.

The player's animations were edited using Unity's built-in sprite sheet animator. Different sections of the sprite sheet were selected to handle animations for jumping, walking and falling. The animations are saved in a directory called *Animations* located in the *Sprite Sheets* directory.

The *Sprite Sheets* directory also includes a directory called *UFOs* which contains a sprite sheet of UFOs that are used to render the enemy UFOs which chase the player and prevent them from jumping to upper platforms. The UFOs sprite sheets are obtained from https://kenney.nl/assets/alien-ufo-pack.

We also used sprite sheets to render the platforms that the player jumps on. They were obtained from https://kenney.nl/assets/jumper-pack.

Dynamic Level Generation

At the start of every scene, the components in the scene are constructed dynamically using a formula defined in PlatformSpawner.cs. The script has an array of strings as one of its public variables. Each string is a step in the System Development Life Cycle process and are inputted manually through the Graphical User Interface of Unity. These strings/steps can be changed for any sequence the teacher/user would like the students/players to learn.

The script will then map these strings to the components in the scene. These components are defined in a public variable called *platform* of type GameObject. In our case, we defined the components as platforms which float in the air and are constantly moving on the x-axis. The player plays the game by jumping on these platforms.

After the strings in the array are mapped to their respective components, the script will randomly position the components within the boundaries of the scene. This way, whenever the player plays a new level they find a new challenge before them, thus decreasing monotony.

Besides the components which have mapped strings, the script also accepts another GameObject in one of its public variables which will be used as intermediary components that the player can jump or hop on in order to reach components which contain strings. These components are defined a public variable of type GameObject called *cupcake* (because in the game they look like cupcakes). Like platforms, the script positions cupcakes randomly within the boundaries of the scene and they also move on the x-axis. These add more variety to the game while making it more enjoyable.

To include more variety whilst making the game more challenging, the script also accepts a public variable of type GameObject called *alienPlatform*. This variable will act as an enemy alien which. The script also places the enemy aliens at different positions within the boundaries of the scene. The player can also jump on these aliens so they also serve as platforms, although it is difficult to proceed from there.

Every level scene contains a GameObject called *PlatformInstantiator* which has the script PlatformSpawner.cs as one of its components. In the *start()* method the script does all the above work.

Role of Each Member

Daniel Magro

- Character Animations
- Red Alien - Enemy script (Enemy.cs) + HP system (Health.cs)
- Dynamically Generated Levels (PlatformSpawner.cs)
- Green Aliens - moving alien platforms
- Particle Systems - when landing on decision platforms

Stephanie Chetcuti

- User interface
    - Main menu
    - Pause menu
    - Level selector
- Game over script
- End game and game over scripts

Nathan Galea

- Overall design of the game
- Assembling and giving components to game objects
- Background
- Enemy script
- Making new scenes and setting up the levels
- Jetpack script

Luke Ellul

- Dynamically generated platforms for each level (PlatformSpawner.cs)
- Mapping of text phrases to their respective platform
- Animations

Kevin Tonna

- Player Movement
- Making sure that the randomly generated levels are actually beatable