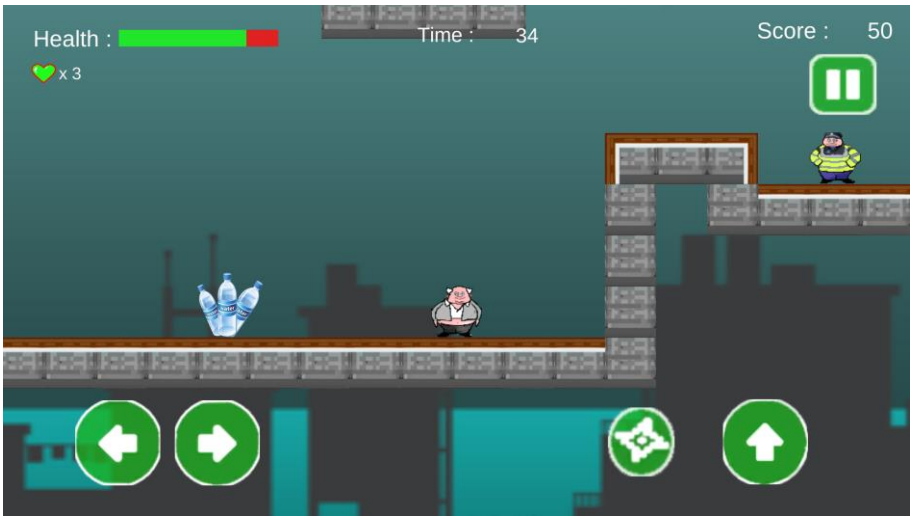


PINTMAN2D

Developer Diary

ABSTRACT

A diary tracking the development process of a comical 2D scrolling platformer developed using Unity.

KEVIN BARRY – G00339811

Mobile Application Development 3

PINTMAN2D

CONTENTS

Introduction 2
Build Options 2
The rate of increasing difficulty 2
Developer Log 3
Bug Log 6
Gameplay conditions 6
  Gameplay Systems 6
  Gain conditions: 7
  Lose conditions: 7
Controls 7
  Windows 7
  Touch/ Android 8
Storing Data 8
Screenshots of finished Android build 9
  Main Menu 9
  Level Select Menu 9
  Pause Menu 10
  In Game 10
The use of colours and Text elements on the user interface 10
Test Cases 11
  Movement Tests 11
  Menu & UI Tests 11
  Data Saving Tests 12
  Gameplay Tests 12
Known Issues 13
Conclusion 13

INTRODUCTION

Pintman2d is a comical 2D platformer where the user, Paddy 'Pintman' Losty, must navigate through the levels collecting Pints while avoiding bottles of water and enemies. This diary was developed to track the development process of this project. The Unity game engine was used to develop this project. Pintman2d is now available for Android and Windows.

Developer: Kevin Barry

Customer: Thomas Duffy

BUILD OPTIONS

The game has been developed to deploy on Android, PC, MAC LINUX and Standalone.

An apk for android and .exe is now available.

The controls for each differ and require a small change in configuration when changing from a touchscreen to a non-touchscreen device within unity. Each platform has its own advantages and disadvantages which I will discuss further on.

THE RATE OF INCREASING DIFFICULTY

The game has been designed to challenge player competence and confidence as they progress through the game. Each level is more difficult to complete than the previous level. In the 1st level it takes 2 shots to kill an enemy, 3 shots in the 2nd level and 5 shots in the 3rd level. As the user progresses each levels difficulty is also challenged by increasing the number of enemy's and enemy objects. Navigation of the levels becomes more difficult as the levels increase.

DEVELOPER LOG

Date	Task	Comment
04 /10/2018	Initial Set up of project	• Created blank project in Unity. • Set up basic folder structure.
05/10/18	Setting up player	• Created initial pixel art sprite sheet. • Added movement to player. • Added Jump to player.
13/10/18	Checkpoints and respawn	• Added CheckpointController.cs script that activates when a player passes checkpoint area. • Now respawns to the last activated checkpoint when player dies.
	Enemy Object	• Created enemy sprite and object as a prefab.
	Collectables	• Created a pint prefab as a collectable object.
22/10/18	Particle System	• Particle system now in place when player dies and respawns. • Particle object destroys after use.
23/10/18	Enemy behaviour	• Enemies behaviour implemented, and animation applied.
	Point System	• Basic point system put in place. • 50 points for killing an enemy. • 100 points for collecting collectable pint object.
	Camera	• Camera is now set to follow the player.
	Game boundaries	• A bounder collider set up under game map so when a player falls of the level a respawn is activated to latest checkpoint.
09/11/18	Firing weapon	• Added shooting action to player. • Control = V (will be changed). • Fires a star object. • +50 points kill enemy. • Particle effect added when contact is made with another object.

		• Star destroyed on contact to save memory space. • Star ignores checkpoints.
	Player Animation	• Added animations to play state. • Walking. • Idle. • Jumping • Flip (flips player horizontally based on movement direction).
	Level Management	• Set a door object to act as end of level. (Activates a menu) • Level menu implemented. • User control enters door (exact buttons yet to be confirmed). • Started design on level 2.
	Pints (Collectables)	• Rotation animation applied using pixel art sprite sheet.
	Menu	• Created Main menu as a level. • Has 3 buttons to select: 'New Game', 'Level Select', 'Quit Game'
10/11/18	Enemy	• Updated Sprite with new pixel art. • Added Animation to enemy state and script for enemy patrol.
	Firing weapon (Star)	• Refined star object. • Destroys star object on contact or after 3 seconds to increase performance.
12/11/18		
	Health System Enemy	• Initial health system put in place. • Takes 2 firing shots the kill enemy (will increase as game progresses)
	Health System Player	• Added Player health system. • Player loses health when collides with enemy.
	Player	• Player is knocked back on contact with enemy.
02/12/18		
	Life System	• Started player life system. • Carrying lives through level. • Reset lives when new game started
	Score	• Carry score through levels. • Reset score when new game started
	UI	• Set up Heads up Display UI to display player health, number of lives, score and countdown timer.
	Time system	• Set a countdown, counts down from one minute per level. • Timer is stopped when game paused. • When time < 0 a life is decremented.
	Sound System	• Sound system implemented.

		• Noise on Player contact with enemy. • When player shoots. • When shooting star hits any other object. • When pickup (pint or extra health) activated.
3/12/18	Health	• Health pickup (+3 health)
	Enemy	• Destroy object on impact. • Player bouncing off enemies corrected.
	Touch System	• Touch buttons UI created. • Mapped touch to the methods. • Scripted to perform different code if touch or pc input.
	Firing weapon	• Added rotation animation to shooting object. • Set delay time on shooting.
	Android	• Build tested on android device (need tunings).
	Level Select map	• Added level select sprites. • Added movement to player to navigate to selected level.
	Level Locking	• Levels are now locked until the previous level is completed.
6/12/18	Enemy objects	• Changed Water objects to polygon colliders.
	Menu	• Anchored menu to support different screen resolutions.
	Finishing	• Debug and ran test cases.
		• Finalised builds
		• Built APK
		• Built EXE
		• Met with client for final correspondence.
		• Project Finished
		• Finished Documentation

BUG LOG

Below is a brief log of some of the bugs encountered during the development process which I fixed later. As this was a new framework to me most bugs where fixed as soon as they occurred.

Date	Bug	Solution
22/10/18	Player Stick to walls	Increased gravity scale
23/10/18	Player falling of screen	Added a collider under game map, respawns player to last activated checkpoint.
26/11/18	Player scaling	Player size was scaling due to incorrectly configured sprites. Redesigned sprite sheet and learned better import methods.
3/12/18	Player firing multiple stars	Set counter to next shot (0.2 sec delay)
	Android controls - All Input Methods acting uncontrollable	Fixed moved kickback code.
4/12/18	UI sizes out of scale	Added scale to screen option on canvas and anchored elements.
	Can't exit door on touch	Added function to call level loader. Now uses jump touch button
	Level Selector Not saving unlocked levels	Used local storage to save which levels are unlocked.
	Level selector: On pause player not in right position	Use player preferences to save position.
6/12/18	Pause menu not display correct	Anchored canvas and set to scale

GAMEPLAY CONDITIONS

The Aim of the game is to progress through each level by avoiding and killing enemies whilst collect as much pints as possible.

GAMEPLAY SYSTEMS

Time Counter: (Initially one minute)

Player Health Counter: (UI scroller, represents from 1-5) Game starts with 5 health.

Score Counter: initially set to 0.

Lives Counter: initially set to 3.

Enemy Health: Health of enemies.

GAIN CONDITIONS:

Collect Pint object increases points by 100 

Killing an enemy increases point by 50.

Shooting an enemy or jumping on enemy head deducts 1 health from enemy.

Finishing a level unlocks the next level.

Collecting a heart increases life by 1.

Collecting a small paddy item increase health by 3 points.

LOSE CONDITIONS:

A collision with a water object decreases player health by 3/4/5 depending on level and decreases score by 50


.

The timer reaching 0 decreases one life from player.

When health is less than 0 player loses a life.

When player loses last life, the game is ended.

When player collides with enemy 1 health point is lost

CONTROLS

WINDOWS

All UI Menu and pause option buttons are controlled by mouse click.

Function	Keyboard	Xbox Game Pad
Move Left or right	Left / right arrow	Joystick axis X
Shoot	V	joystick button 2
Jump/ submit	Space bar	joystick button 0
Enter pause menu	Escape	start
Enter door (end of level)	Up arrow	Joystick axis Y
Enter Level (select menu)	Space bar	joystick button 2
Cancel	Escape	Joystick Button 1

TOUCH/ ANDROID

All UI Menu and pause option buttons are controlled by touch.

Button	Action
	Move Player Left
	Move Player Right
	Jump Enter end of level door Enter level select door
	Shoot
	Pause / Un pause game

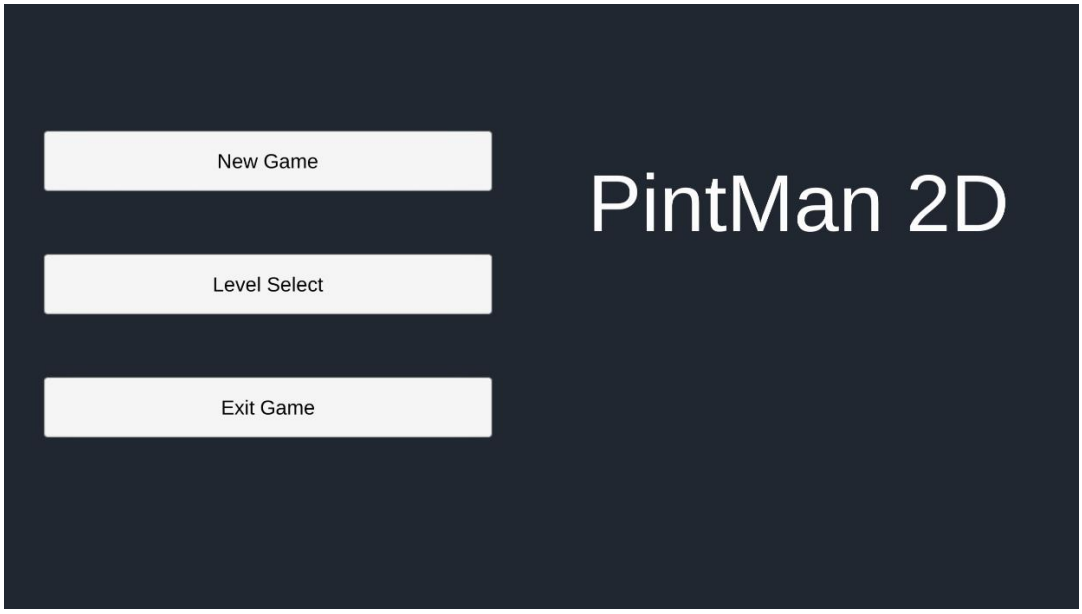
STORING DATA

In order to increase the game functionality some data is stored locally. This was achieved by using the class UnityEngine.PlayerPrefs. Using PlayerPrefs enabled data to be carried through levels while also allowing the game to be exited and re-opened in the same state.

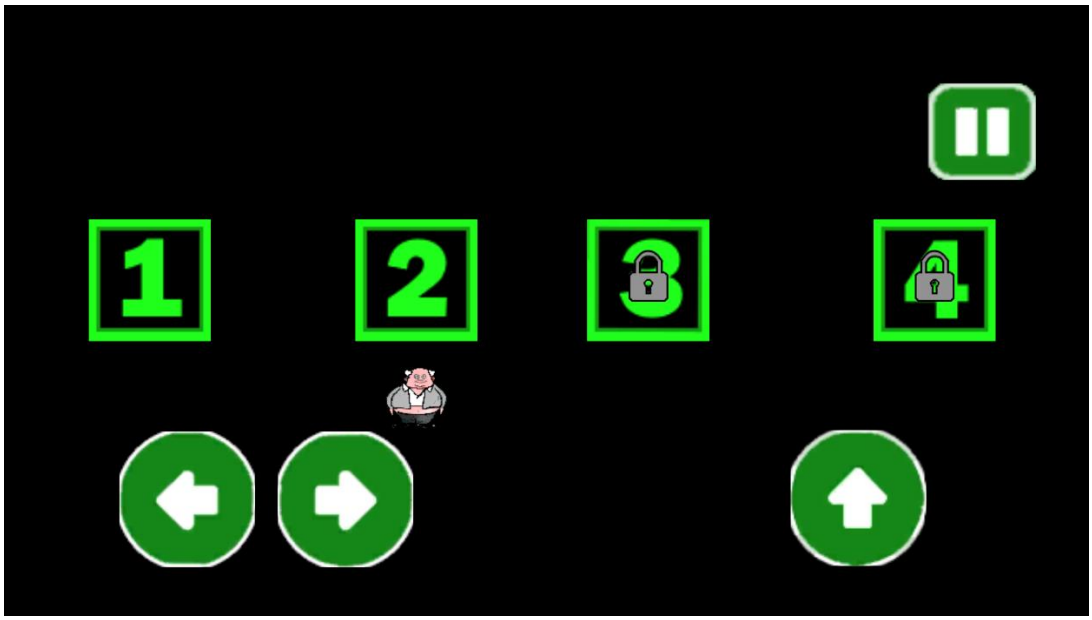
- **CurrentPlayerHealth:** Saves current play health so game can be reloaded with same health and health level can be carried through levels.
- **unlockedLevel**: This saves the level unlocked from finishing a level. unlockedLevel is the name of the lock which can be either level1locked, level2locked or level3locked and is set to a value of 1 if the level is unlocked.
- **PlayerLevelSelectPosition:** Is used to store the x value of the players position in the level select menu. This meaning when a player enters the level select menu the character will be located under the current level.
- **CurrentPlayerLives:** Stores the current value of player lives.
- **CurrentPlayerScore:** Stores the players current score and is reset to 0 on new game.

SCREENSHOTS OF FINISHED ANDROID BUILD

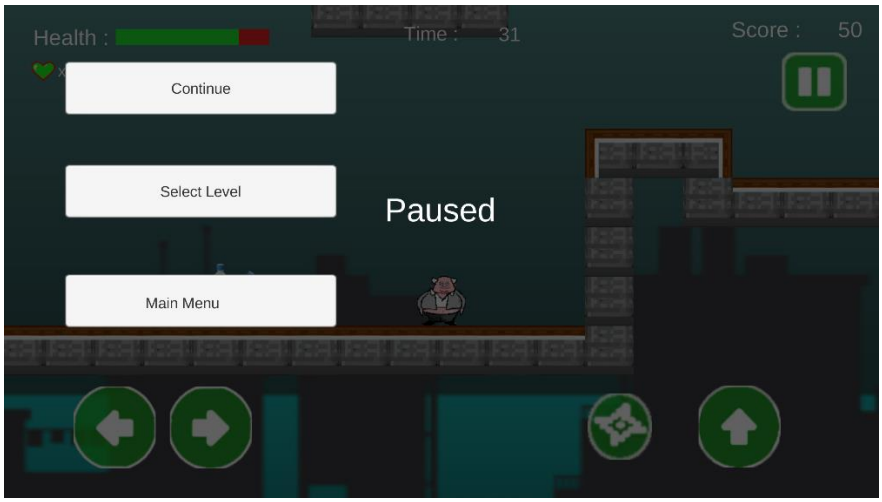
MAIN MENU



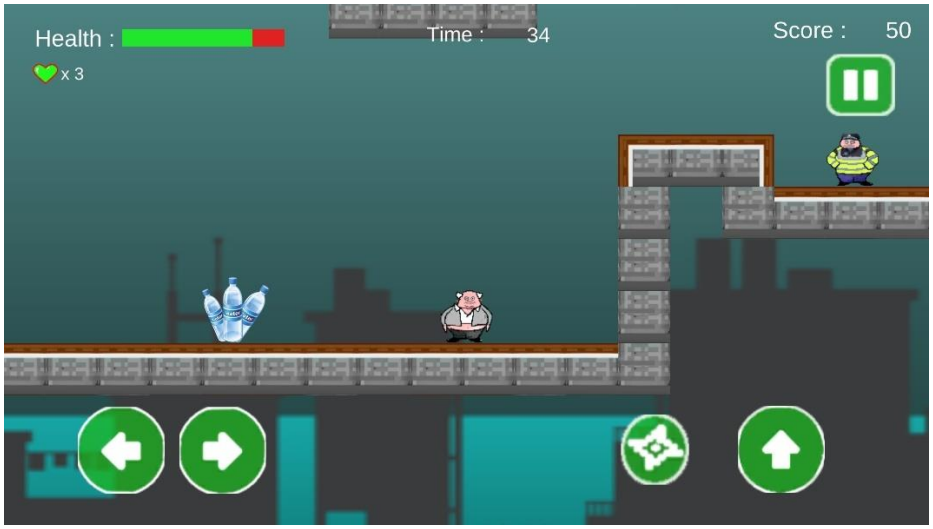
LEVEL SELECT MENU



PAUSE MENU



IN GAME



THE USE OF COLOURS AND TEXT ELEMENTS ON THE USER INTERFACE

As described in the design doc the aim was to create an easy to use, simple to understand user interface. I had discussed this with the client who instructed that the menus be left simple to allow for the focus be on the actual game play. The client also stated that “the user shouldn’t be spending any more time than is necessary in the game’s menu system”.

The game itself is based on the Irish character know as Paddy ‘Pintman’ Losty. To support this theme, we decided to use green as the base colour for UI buttons. All menu buttons are white and change to green when clicked.

TEST CASES

All tests were carried out on a Samsung Galaxy S6 with touch screen capabilities.

MOVEMENT TESTS

Test	Expected Result	Actual Result	Pass / Fail
Player can move left/right	Player moves left and right	Played moves left and right	Pass
Player can jump	Player jumps	Player jumped	Pass
Player can shoot	Player shoots star	Player shoots	Pass
Double Jump	Player can double jump by pressing jump twice	Player double jumped	Pass
Player restricted by walls	Player should not be able move through walls	Players movement was restricted when in contact with a wall	Pass

MENU & UI TESTS

Test	Expected Result	Actual Result	Pass / Fail
Selecting new game	Starts a new game	Launched a new game to level 1	Pass
Selecting Level	Brings User to select level menu	Game launched select level menu	Pass
Exit Game	Exits the game	Game shutdown	Pass
Screen scaling and positioning (Landscape)	Screen should render and display all elements clearly	Elements scaled and performed as expected	Pass
Screen scaling and positioning (Portrait)	Screen should render and display all elements clearly	Elements in select menu did not scale correctly	Fail
Pause Menu	Pause menu opens and pauses gameplay	Pause menu rendered and all counters paused	Pass
Select Levels locked until completed	Levels should show a lock symbol if user has not completed previous level	All levels after level 1 are locked	Pass

DATA SAVING TESTS

Test	Expected Result	Actual Result	Pass / Fail
Exiting and Resuming Game reloads current score	Score from previous game shown	Game loaded previous score	Pass
Level Locks saved when game shut down and reopened	Completing level 1 should unlock second level even after restarting game	Level 2 remained unlocked	Pass
Value of user lives should carry through levels	Same amount of lives finishing level 1 as starting level 2	Lives correctly saved	Pass
Value of user health should carry through levels	Same amount of health finishing level 1 as starting level 2	Health correctly saved	Pass

GAMEPLAY TESTS

Test	Expected Result	Actual Result	Pass / Fail
Colliding with water damages player	Players health reduced	Players health reduced	Pass
Colliding with enemy damages player	Players health reduced	Players health reduced	Pass
Shooting enemy (3times level one) destroys enemy	Enemy destroyed	The enemy was destroyed	Pass
Colliding with a heart object increases players life value by 1	Players life value increases by 1	Life value incremented as expected	Pass
Colliding with a mini Paddy object increases players health value by 3	Players health value increases by 3	Health value incremented as expected	Pass

KNOWN ISSUES

Developing the game to run on various platforms proved to be challenging. The game works on all platforms as expected. The Game passed most tests that it was put through. This being said a few minor issues where spotted when testing on different screen resolutions and platforms.

- **Select Level Menu**: Does not scale correctly at times as it was initially built as a level. This occurs when an Android device is rotated from the landscape to portrait view. To solve this a canvas would have to be created allowing elements to be anchored and scaled to size.

- **Main Menu Game Title**: As the client specified the layout should have menu options to the left with the game title to the right. Unfortunately, due to the limited size of an Android screen when the device is rotated to the portrait position the title over laps some of the buttons. I found it difficult to find a middle point where the text would be small enough not to overlap but big enough to be easily read by the user. To solve this, I would set the buttons to stack under the title when the device detects it has been changed to portrait view.

- **Configuring the project in Unity to run as a Windows application**. In order to configure the project to build and run as a Windows application a few minor adjustments must be made,
    - Go to **File > Build Settings > PC, MAC & LINUX standalone > switch platform**
    - Go to **Assets > _prefabs > HUD > Touch Controls** and toggle off the active select box. This disables the canvas that renders the touch screen buttons. To solve this issue, I would make a controller script that dynamically checks at runtime if the device is touch compatible or not. In the PlayerController.cs script I tried to address this by using "`if UNITY_STANDALONE || UNITY_WEBPLAYER`" which worked for specific devices but proved unsuccessful when running on multiple platforms.

CONCLUSION

After a final review with the client he is impressed with the finished product. I have met all the specified requirements and improved and exceeded them where necessary. I have explained the known issues of the game to the user and we feel that this is down to taking on too much of a task to have the game capable of running on multiple platforms.

The final product is now available as an APK for android or an .EXE for Windows

To conclude we are both happy with the overall outcome of the project.