2D Educational Platform Game Developed In Unity

Jonathan P. Ward

**Contents**

[Abstract 4](#_gjdgxs)

[Background Information and Useful Definitions 5](#_30j0zll)

[Background Information 5](#_1fob9te)

[Backstory 5](#_3znysh7)

[Game Mechanics and Rules 6](#_2et92p0)

[More Definitions 6](#_tyjcwt)

[Development Engine 8](#_3dy6vkm)

[Introduction and Project Overview 8](#_1t3h5sf)

[Problem 9](#_4d34og8)

[Objectives 9](#_2s8eyo1)

[Summary of Features 10](#_17dp8vu)

[Design, Development, and Testing 10](#_3rdcrjn)

[Design 11](#_26in1rg)

[Original Development Diagram 11](#_lnxbz9)

[Modules, Components, and Interfaces 12](#_35nkun2)

[Final Development Diagram 14](#_1ksv4uv)

[Development 14](#_44sinio)

[System Planning 15](#_2jxsxqh)

[System Analysis 15](#_z337ya)

[System Design 15](#_3j2qqm3)

[Implementation and Deployment 15](#_1y810tw)

[System Testing and Integration 15](#_4i7ojhp)

[System Maintenance 16](#_2xcytpi)

[Testing 16](#_2bn6wsx)

[Results 16](#_1ci93xb)

[References 17](#_3whwml4)

Abstract

This report provides insight and recommendations based on the development of a 2D Android educational-platform game titled Harambe Hustle. The report introduces the project, describes the design, development, testing, and results, as well as make some conclusions based on the entire experience. Instructors of grade school kids face the tough challenge of holding attention span. In society, today, kids are more connected to their phones than ever before. The development of this 2D Android platform game addresses that issue. An instructor can use this game to help get ideas or concepts across to their students in a more entertaining way. The main objectives are to have a working 2D platform game that runs on Android devices. Other, sub-objectives, are to have quiz-type questions implemented at the end of each level to educate the player as well as having a high score system to motivate them. In the design, development, and test phases of the project there are some issues that arise but they are quickly corrected. This project began strictly as a java based game which required implementation of a custom game engine. It then became apparent that this project is better suited for development in Unity. Finally, the results of the project show that Unity is a capable game engine which allowed for a multi-level educational game to be created in a relatively short period. Educational aspects of the game come in the form of true/false, multiple choice, and short answer questions. Coin collection and a score system help motivate the player to continue forward, even with the threat of quiz-type questions in between each level. The goal of developing a 2D Android platform game that can be used by educators was accomplished in a satisfactory fashion when considering the time constraints, team size, and knowledge of the developer.

2D Educational Platform Game Developed In Unity

## Background Information

Educational games are nothing new but there is always room for a fresh one. Examples of other educational game developed for Android devices are Toddler Tapping Zoo, Super WHY, and Alphabet Car. These are just a small portion of the thousands of other games available. Similar games on the Android market have features and strengths that Harambe Hustle cannot match because development constraints. The strength of Harambe Hustle comes from the platforming aspect (jumping from platform to platform). It is an action platformer at its core but has educational features sprinkled in. Most other games in the educational marketplace do not focus on gameplay, rather the educational features. Whether this is problematic is yet to be seen because this focus on gameplay may devalue the educational aspects. Weaknesses of the game stems from problems that occurred because of time constraints and lack of experience. The entire scope of this project is to demonstrate development of a 2D android educational-platform game and the problem is to develop a solution that can help instructors keep their teaching fresh.

## Backstory

The game follows the son of Harambe, Harambe Jr. To collect enough money for his father’s funeral he is collecting gold coins. He must avoid various enemies and other dangerous obstacles to be able to gathers all the funds.

## Game Mechanics and Rules

Mechanics (2) refers to the main actions that a player must make and become good at to be able to complete the game. Mastering the mobile controls is the first thing a user must deal with. The controls are touch-based buttons (left, right, and jump). The player must jump onto various platforms and avoid falling into holes or pits. Throughout each level are various amounts of coins that can be collected and added to the player’s score. After a certain threshold is met the scene changes to a quiz and asks questions based on the questions another user has entered. The questions can be any combination of true/false, short answer, and multiple choice. After the questions are completed the player continues through the game. The levels get progressively harder by adding differently sized platforms. If the player falls into a pit or off the designated map, then he/she will respawn at the start of the level. The term respawn (1) refers to the recreation of an entity after its death or destruction, perhaps after losing one of its lives. The game at this stage has three levels that must be completed to reach the end of the game.

## More Definitions

All definitions were obtained from www.wikipedia.com and cited in the reference section.

**Java** - Java is a programming language expressly designed for use in the [distributed](http://searchcio-midmarket.techtarget.com/definition/distributed) environment of the Internet. It was designed to have the "look and feel" of the [C++](http://searchsqlserver.techtarget.com/definition/C) language, but it is simpler to use than C++ and enforces an [object-oriented programming](http://searchsoa.techtarget.com/definition/object-oriented-programming) model.

**Android Studio** -  is the official integrated development environment(IDE) for Android platform development.

**Game(development) Engine** - A game engine is a software framework designed for the creation and development of video games. Developers use them to create games for consoles, mobile devices, and personal computers.

**Unity** - Unity is a cross-platform game engine developed by Unity Technologies and used to develop video games for PC, consoles, mobile devices, and websites.

**C#** - is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.

**Sprites** - In computer graphics, a sprite is a two-dimensional bitmap that is integrated into a larger scene.

**Platform Game** - Platform game (or platformer) is a video game which involves guiding an avatar to jump between suspended platforms and/or over obstacles to advance the game.

**GUI** - In computer science, a graphical user interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators.

**Script** - programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator

**Game Object** - are the fundamental objects in Unity that represent characters, props, and scenery.

**Game Scene** - scenes contain the objects of your game. They can be used to create a main menu, individual levels, and anything else. Think of each unique Scene file as a unique level.

**Animator Class** - Interface to control the Mecanim animation system in Unity.

**Collider** - A base class of all colliders. Collider components define the shape of an object for the purposes of physical collisions.

**BoxCollider** - A box-shaped primitive collider.

**Transform** - The Transform component determines the Position, Rotation, and Scale of each object in the scene. Every GameObject has a Transform.

**Anchor** - Rect Transforms include a layout concept called anchors. Anchors are shown as four small triangular handles in the Scene View and anchor information is also shown in the Inspector.

**Trigger** - are colliders that are capable of executing script when a collider touches the trigger.

## Development Engine

While the original project design called for using Java and Android Studio it was quickly realized that it would not be practical to create a custom game engine then have time to create a game in that engine. So, Unity 5 was used in the final development of Harambe Hustle. The features of Unity are more flexible and powerful than any custom engine that could have been developed. All development in Unity is done using a choice of C#, Boo, or JavaScript. Harambe Hustle was completely developed in C#.

# Introduction and Project Overview

This project was developed for the use of instructors, tutors, and parents. Harambe Hustle is a useful tool allowing for an alternative approach to learning. The main gameplay features mechanics are like other action platformer games. Coin collection is the primary focus of the gameplay along with traditional platformer elements. A unique educational twist is also featured which comes as a solution to the problem of being able to keep a young child’s attention. The following subsections will introduce the main problem in more detail as well as describe the original objectives of the entire project. Later, topics such as benefits of the application and problem scope are discussed. Lastly, this section will conclude with a summary of features and figures.

## Problem

Instructors, tutors, and parents face an uphill battle when it comes to entertaining, educating, and controlling young students. Whether the students are not paying attention or only interested in texting it can be disheartening for many. It is well documented that there are proven techniques to help teachers with keeping control of the classroom. Per Tristan de Frondeville, of Edutopia.com, it is important to keep kids vigilant by changing teaching styles (3). With mobile technology, already in a high percentage of kids’ hands, a solution that came as a mobile game seemed like a good idea. The solution had to be basic and simple enough to complete in a few months. After discussing ideas with the higher ups a decision was made to develop a platformer game with educational features.

## Objectives

Development of Harambe Hustle aims to help a teacher change their teaching style with minimal effort. Starting with no knowledge of Unity or the programming language C# the best solution to the problem was to be sure project aspirations were not set unreasonably high. Ideal solutions would require extended development time as well as a deeper understanding of Unity and C#. With the allotted time, developer knowledge base, and other circumstances a basic solution of a 2D Android based platform game with true/false, multiple choice, and short answer questions was developed. With the games strengths and weaknesses taken into consideration, the game does demonstrate a possible solution to the problem described earlier.

## Summary of Features

|  |  |  |
| --- | --- | --- |
| Multiple scenes with 2D Characters | Player movement | 2D Physics |
| Collision Detection | Moving Obstacles | Player Respawn Upon Death |
| Coin Collection | Score System | True/False Questions |
| Multiple Choice Questions | Short Answer Questions | Touch Based GUI Buttons |
| Sound Effects | Main Menu Scene | Sprites |
| Animations | Basic AI for Enemy | Buttons Trigger New Scenes |
| Coin Collection Triggers New Scenes | Runs on Android Devices |  |
|  |  |  |

# Design, Development, and Testing

The following section will cover topics that consider the design, development, and testing of Harambe Hustle. Beginning with the design, components like software, modules, and libraries are described. A small section below is also dedicated to the details of any interfaces and operations that were used in the development.

## Design

### Original Development Diagram



Figure 1. The diagram above shows the original development ideas for Harambe Hustle. Green squares are level scenes, red squares are quiz scenes, blue squares are user input scenes, yellow squares are options/information scenes. The original plans had to be modified slightly to meet deadlines (more on that later).

In figure 1, shown above, is the original development plan for Harambe Hustle. The exact implementation shown above was not achieved fully but about 85% of the desired features from this diagram are realized in the final implementation. Sections later in the report will discuss the features that were implemented in the final product. As mentioned in previous sections the entire game was developed in Unity 5. The scripts that are used in the by the game objects are all written in C#.

### Modules, Components, and Interfaces

In this section a description of the modules, components, and interfaces used in the design of Harambe Hustle are discussed. The game consists of 8 scenes each of which has many scripts, game objects, obstacles, and decorations connected to it. Over the course of development, 10 full scenes were created but because of various issues 8 made it into the final game. The scenes consist of a Main Menu, three levels scenes, Question/Answer Scenes (True/False, Short Answer, and Multiple Choice), and an End Demo Scene.

#### Main Menu Scene and Question/Answer Scenes

The main menu scene is the first scene a player will see when starting the game. The title “Harambe Hustle” is displayed at the top. There are buttons three buttons that will change the scene by the player’s touch. “Play” loads the first level, “Question” loads the user input question scene, and “Exit” will leave the game. It sounds very simple but in practice was much harder to program than originally predicted. The script attached to the Main Menu Scene (MenuManger) consists of 144 lines of code that provide the functionality to the three menu buttons as well as the “Back”, “MCQ”, “Short Question” and “True/False” buttons. The script also provides functionality for the reset buttons which will remove all saved questions from the specified question category. All the information from user input is stored by the PlayerPrefs class as well as text objects using the .GetComponent<Text> () method from the GameObect class. Another script that is used throughout the Menu and Question scenes is used to control the animator(AnimatorScript). This script is smaller than others but is used throughout the entire project to determine if a certain scene is open. The last important script that is used with these scenes is an anchor script (SetAnchorPresets). This script defines the default values of certain menu elements and buttons so that they are always anchored to a certain area of the screen.

#### Level Scenes

The level scenes are what players will see after hitting the “Play” button and after finishing a section of quiz questions. The scripts used in the levels are displayed in Figure 2 below.

**Table Showing Scripts Used in the Level Scenes**

|  |  |
| --- | --- |
| CameraFollow.cs | Sets camera attributes and sets camera to follow the player. |
| Enemy.cs | Class used to create enemy characters. |
| Player.cs | Class used to set statistics for the player. Settings for the touch buttons, jump force, and other aspects like that. |
| CoinCollision.cs | Class that onTrigger() event will destroy the coin object, play a sound for the coin being collected, and add points to the score by calling the ScoreManager.cs class. |
| EnemySight.cs | Class that simulates an enemy character’s sight. OnTriggerEnter2D and OnTriggerExit2D are used to determine if the enemy is targeting the player. |
| ScoreManager.cs | This class changes the display text of score every time a coin is collected and adds it to an accumulator. |
| IgnoreCollision.cs | Class that is called when the player needs to ignore a collision, such as a time when they are jumping up “through” a platform. |
| Character.cs | Abstract class that contains all the general functionality of the characters in the game. |
| CollisionTrigger.cs | This class handles the platform collision. |

Figure 2. Scripts that are used in level scenes of Harambe Hustle

### Final Development Diagram

Figure 3. Final development diagram. Most features were implemented as planned. Only two of the planned scenes are omitted.

As shown in Figure 3, most of the planned features were implemented. A couple of scenes and features were removed but the core is still intact.

## Development

This section of the document will describe the how the system was developed and talk about how risks were addressed early.

### System Planning

The planning stage is one of the most critical parts of any project. This project had a clear goal of being an educational Android game that would be beneficial in solving the problem of keeping a young child’s attention while simultaneously educating them. After meeting with the “higher up” this was determined to be an acceptable course of action for project development.

### System Analysis

End-user’s requirements were assumed. Communication was kept with the “higher up” to help determine if the project was staying on track and improving weekly.

### System Design

The design is mentioned in the previous section. In summary, a diagram was created to follow through to the end of development but a modified design had to be implemented for various reasons.

### Implementation and Deployment

The actual code was written during this stage.

### System Testing and Integration

The testing that occurred during the process of this project occurred during this phase. Most testing was done by directly testing features that were promised. Nearing the end of development, a volunteer was brought in to try the game out. The most beneficial testing was getting an outside party to test the game. After live user testing it was determined that the buttons did not always move smoothly so players would have to tap left or right just to move. A correction was made to allow for holding down the move buttons, rather than tap. Examples of the direct testing are in Figure 4 below:

**Table Showing Direct Testing by Observation**

|  |  |
| --- | --- |
| Have a fully fleshed out level with items | YouTube video shows “Full Level” with items (Coins). I still need to implement collection of the coins and add a bit more space to the level. |
| Platforms have been created | Platforms have been added. There is a box collider 2d on each platform that detects points on the Harambe Sprite to allow him to stand on them. |
| Character is able to fall off of the screen and respawn(Reset Position on screen) upon falling into water or off of level. | Youtube video demonstrates this. |
| Enemy Player detects when close to Harambe and will swing arms to attack. | Youtube video demonstrates this. |

Figure 4. Shows an example of the direct testing of features by showing they work.

Below, Figure 5 shows every major feature and how each of those features were tested to verify function.

**Table Showing All Features and How Each Feature Was Tested**

|  |  |  |
| --- | --- | --- |
| Multiple scenes with 2D Characters – This is an observable test. | Player movement – Tested by verifying touch controls and touchscreen button function. | 2D Physics – Tested by jumping on platforms and attempting to jump through platforms. Also, tested by jumping off the map and jumping. |
| Collision Detection – Tested by jumping on platforms. | Moving Obstacles – This is an observable test. Level 3 has a moving platform. | Player Respawn Upon Death- Tested by jumping into holes and off the map to determine if player respawns at start of level. |
| Coin Collection – Tested by moving player over coins and determining if score was incremented. | Score System - Tested by moving player over coins and determining if score was incremented. | True/False Questions – This is a observable test. True/False question scenes function. |
| Multiple Choice Questions – This is an observable test. MCQ function. | Short Answer Questions – This is an observable test. Short Answer Questions Function. | Touch Based GUI Buttons – Tested by pressing the buttons on an android device. |
| Sound Effects – this is an observable test. Coins do play sound effect when collected. | Main Menu Scene – This is an observable feature. Main Menu scene appears and functions. | Sprites – This is an observable feature. Sprite sheets were used and free elements obtained |
| Animations – Tested the animations by jumping, running, and landing. | Basic AI for Enemy – Tested enemy AI by approaching the enemy and checking the target of the enemy. | Buttons Trigger New Scenes – Tested feature by using touch based controls to change scenes. |
| Coin Collection Triggers New Scenes – Tested by verifying the threshold of coins required to be collected and if it changes the scene. | Runs on Android Devices – Tested by running on personal android device running Android 7 (Nougat). |  |
|  |  |  |

Figure 5. Shows the main features and how each feature was tested.

### 

### System Maintenance

There has not been an opportunity for maintenance because a complete version was just completed recently. In the future, there are some aspects that could be added and updated, those features will be mentioned later in the report.

# Results

This section will summarize the results of development and design of Harambe Hustle. Below, in Figure 6, is the Summary of Features section that shows the fully implemented features.

**Fully Implemented Features**

|  |  |  |
| --- | --- | --- |
| Multiple scenes with 2D Characters | Player movement | 2D Physics |
| Collision Detection | Moving Obstacles | Player Respawn Upon Death |
| Coin Collection | Score System | True/False Questions |
| Multiple Choice Questions | Short Answer Questions | Touch Based GUI Buttons |
| Sound Effects | Main Menu Scene | Sprites |
| Animations | Basic AI for Enemy | Buttons Trigger New Scenes |
| Coin Collection Triggers New Scenes | Runs on Android Devices |  |
|  |  |  |

Figure 6. Shows the fully implemented features in Harambe Hustle

The entire project did meet a lot of the expectations after the project was slightly modified. Some features that were planned and not implemented because the focus changed to an educational game are as follows:

* Weapon able to fire projectiles
* Character select screen
* Character Health Meter
* Music Implemented
* Death Animation for Main Characters

### Problems Encountered

Most problems that occurred during the development process of Harambe Hustle had do with project management. Simply figuring out the scope and genre of game was a difficult task. Pertaining to the scope it was very easy to be overly ambitious and unrealistic with the deadline that was given. For future projects, it would be wise to improve project tracking or use a custom workflow. Another problem was that some design decisions were made during the game production phase. It is better to do the preparation work and then begin working on the game production. Lastly, a problem that came up was exhaustion. It was easy to be excited at the start of the project but it was difficult to sustain this attitude for the whole duration of development.

## Conclusions and Future Work

This project set out to be a solution for teachers, tutors, and parents who want to engage their children in a different way. Mobile devices are everywhere and an educational game that is fun is beneficial in teaching young kids. The development of this project was approached in a “controlled storm” type of way. The environment the game was developed in simulated Agile methodologies but it was difficult to follow the exact process because of zero previous experience. The result is a game that can function as a solution to keeping a kid’s attention and teaching them basic concepts through various questioning styles.

### Areas of Further Study and/or Development

Ideally, if development continues this project, there are many areas of further study and development. Below in Figure 7 is a table of interesting features that could be added in the future:

**Possible Future Additions to Harambe Hustle**

|  |  |  |
| --- | --- | --- |
| Send Results to Instructor Option | Ability to add questions online and having users Login to access questions in the game. This would improve upon the current system of having to input questions on every device. | Online High Score Board |
| Full Soundtrack and full sound Effects | Larger Levels | Addition of Story Elements that expands on the backstory given in introduction. |
| Loading Scenes to help with pacing. It is odd for the scenes to change instantly after collecting the set amount of coins. | More enemies | Other gameplay mechanics such as puzzles. |

Figure 7. Table of possible additions to the game in the future.

References

1. Spawning (video gaming). (2016, November 1). In *Wikipedia, The Free Encyclopedia*. Retrieved 17:40, November 1, 2016, from <https://en.wikipedia.org/w/index.php?title=Spawning_(video_gaming)&oldid=747313119>
2. Game mechanics. (2016, September 27). In *Wikipedia, The Free Encyclopedia*. Retrieved 08:49, September 27, 2016, from <https://en.wikipedia.org/w/index.php?title=Game_mechanics&oldid=741404791>
3. Frondeville, T. D. (2009). How to Keep Kids Engaged in Class. Retrieved November 21, 2016, from https://www.edutopia.org/classroom-student-participation-tips
4. Definitions Section of Paper. (2016, September 27). In *Wikipedia, The Free Encyclopedia*. Retrieved 08:49, September 27, 2016, from *various*