University of Arkansas – CSCE Department
Capstone I – Preliminary Proposal – Spring 2019

Tilted

David Green, William Harris, Drew Rudasill, Connor Dean

Abstract

The objective of this project, is to give all of us more experience in programming. It will help expose us to more tools that are popular to use in game development today. Our approach is to use the Unity Engine in order to create a mobile game that is able to take advantage of a phone's gyroscopic capabilities.

1.0 Problem

Developing mobile games may not provide the world a solution to their dire problems, but what they can provide still serves to improve the quality of life for all around us. There can be many problems tied to the purpose for a mobile game; such as location, or accessibility. For instance, people that don't have access to their consoles or computers on campus. Or consider people that don't have any consoles at all. And, perhaps most common, something to pass the time. Mobile games provide something to people that is perhaps overlooked; entertainment. Many people use entertainment as a method of temporary freedom from their hardships, and it takes many forms. It could be from work, school, relationships, and many more things. The lack of entertainment would bring down the overall quality of life by a measurable amount, even if it is small.

2.0 Objective

The objective of the project is to implement a "marble platformer" in Android Studio. The player will control a marble and must navigate it through a course to reach the goal. There will also be a time limit they must overcome on top of other challenges. However, there are other sub-goals that we'd like to reach in doing this project. The first step is to decide how we wish to build our application and from there we can carry out our other objectives such as implementing a leaderboard system, creating a database schema for keeping track of users, and the game mechanics itself. Another primary objective of ours is to implement both gyro and touch controls.

3.0 Background

3.1 Key Concepts

Unity

- Unity is the most popular game engine there is for developers. It's not only a framework with which we run our code on, but it also acts as an edit tool. Unity utilizes Views that allows us to manipulate the objects as we are building. It also allows us to view our game in real time while making changes in the code. This being supported on Android is what makes the project feasible.

Database

- A database is how we will store information from our users such as their ID. We need this because there are certain features that we want to implement and require a table to pull information from. For instance, if we want to implement a high-score leaderboard for a certain level, we can pull from each user their best times of that level and display the top ten.

3.2 Related Work

Other mobile games, that involve a marble specifically, have a similar problem in that there is no actual control for the marble. In Marble Run, users are allowed to design their own levels where they watch marbles race on the track, but no interactive controls for the users, with a terrible user interface. Marble It Up is a great example of what we want to accomplish. It features platforming and free movement over the marble, with difficult levels. However, its only available on PC and thus, being mobile will provide a better gaming experience. Additionally, we will provide two different methods of control, gyro and touch.

4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

What we want by the end of this project is a competent, fully functional, basic android mobile game, which provides reasonable challenge, can be played offline, but implements some network features, and has at least two unique control schemes. Network features include things like high scores and other user information. The game's UI should be fairly basic, and made of only a few easy to navigate menus, which can be interacted with by touch.

The gameplay will be simple. You must navigate a ball to an end goal, using either tilt controls, or touch controls. Levels should be designed in such a way that they are challenging, but not overly punishing. Scores and other user information should be saved offline locally upon completion of a stage If the user is online, those scores will be submitted to a database, which the game will query to fill some sort of table for high scores, or best times. Control schemes *must include* both a tilt control option, and also some sort of touch based control. Mobile games can be played in transit, while on a bus, in an uber, etc. Gyroscopic functionality may be impaired while in transit.

4.2 High Level Architecture

Unity natively uses an Entity-Component architecture, but can support a sort of Model View Controller sub-architecture

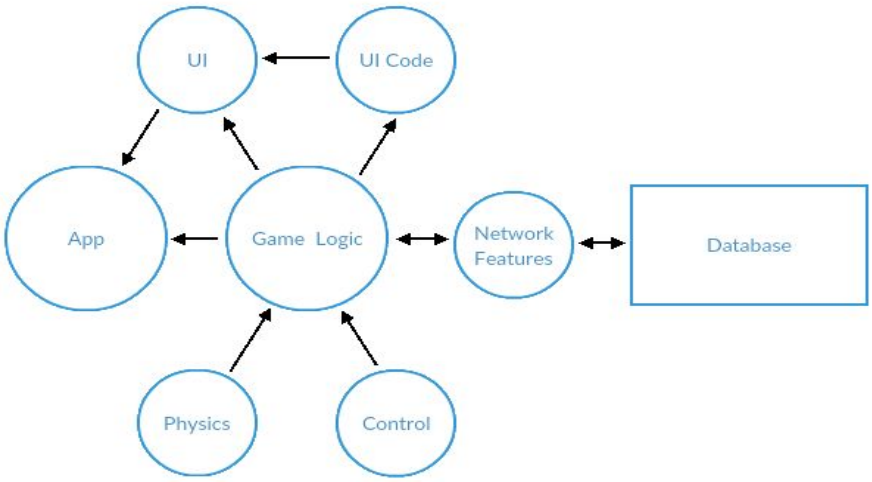
In concurrence with the Entity-Component architecture, we will be using a Model View Controller (MVC) architecture. Our Model will contain all the data of the game, such as, player location, goal location, pick-up locations. Anything that can be found in the game world will be put in here.

Our View will show the game to the player. This will load all the needed files for the player, based on what they can currently “see”, things behind them or out of view will either not be loaded or lose detail as to make the game more optimized.

Our Controller will be how the player navigates the game world and the menus. This will need to contain two different control schemes. One for tilt controls in order to navigate the world, and the other will be touch controls to navigate the world. This is to ensure that we avoid the problem of unwanted external input, such as being a passenger in a vehicle and the vehicle turns, which would also cause the tilt controls to shift.

There will also be The game engine Unity will be used to help us develop the game world.

For security, RSA will most likely be used in order to encrypt messages that need to remain secure. If we wanted an easier solution, MD5 is acceptable as well, although slightly less stable. In order to save scores/times of the player, a Database will be used. 000webhost is a free web hosting service which includes database management tools, and can be accessed and manipulated entirely from within a browser, so there’s no need to download any external software. After the completion of every level, the most recent score set, will be compared to a players all time best score for that specific level. If they beat it, a message will be sent to the Database holding the leaderboards telling it to update them by replacing the players old time with the new time. And placing in it in the leaderboard appropriately. On the front-end, a scrollable leaderboard will appear for the player to browse, so long as the game is connected to the internet.



4.3 Risks

Risk	Risk Reduction
Designing games is a potentially challenging task. It all depends what's in it. For instance, building your own game engine could be its own capstone project.	Instead of designing our own, we're using Unity; a well known, and *super* well documented engine.
Connection to the internet can cause all kinds of security issues. People finding information they aren't supposed to find and using it with malicious intent.	Secure messages through encryption and decryption when internet is required for a task.

4.4 Tasks

These tasks are more or less in the order they need to be completed, but there may be some performed out of order as needed.

1. First of all: Gain a better understanding of how Unity works. Run through some tutorials.
2. Design in detail game flow, as in, how menus are connected to levels, how they function, and in what order, etc.
3. Program basic game physics
4. Program basic UI interaction (basic menus, control options)
5. Initial device testing
6. Level design
7. Level implementation
8. Decide on and design network features
9. Implement network features
10. Secondary device testing
11. Bug fixing
12. Prepare app store version, and associated materials for storefront
13. Deploy to app store
14. Document

4.5 Schedule

The needed tasks can more or less be broken down into weekly goals, where some tasks may be given more time, and others less. This timeline fits almost perfectly into one semester.

Tasks	Dates
1. Learn Unity's ins and outs.	8/19-8/25
2. Design "Game flow"	8/26-9/1
3. Program game physics	9/2-9/8
4. Program UI interface	9/9-9/15
5. Test on actual android devices	9/16-9/22
6. Design actual levels	9/23-9/29
7. Build actual levels	9/30-10/6
8. Design network features	10/7-10/13
9. Implement network features	10/14-10/20
10. Secondary device testing	10/21-10/27
11. Fix anything that's still broken	10/28-11/03
12. Prep app & storefront materials	11/04-11/10
13. Deploy app to store	11/11-11/17
14. Document our process	11/18-11/24

4.6 Deliverables

The final product should have these associated deliverable items:

- A final, full app to be visible on the app store.
- If necessary, a design document visualizing work done in Unity.
- Full game code in C#.
- Database scheme, and associated code.
- Final Report

5.0 Key Personnel

David Green- Green is a Senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas.
He has completed: Software Engineering, Operating Systems, Database Management, Programming Paradigms.
Has experience with: Machine Learning, Android Studio, C++, Java, SQL, Hadoop, Spark
Is responsible for: Most likely responsible for the Android code.

William Harris- Harris is a Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas.
He has completed: Database management, computer graphics, programming paradigms.
Has experience with: C++, Java, SQL, Managing databases, game design, UI design.
Is responsible for: Very likely the database side of things. Maybe level design?

Drew Rudasill- Rudasill is a Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas.
He has completed: Software Engineering, Database Management Systems, Operating Systems, Programming Paradigms
Has experience with: C++, Java, SQL, Python
Is responsible for: Most likely “Game flow” and game physics

Connor Dean- Dean is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas.
He has completed: Programming Foundations I & II, Computer Paradigms
Has experience with: C++, Python, Java, JavaScript, C, artificial intelligence, game logic, and game design.
Is responsible for: Helping to make the game work.

6.0 Facilities and Equipment

- A computer with Android Studio for emulation, if needed
- Unity software
- Emulated/Physical Android phone
- Database host