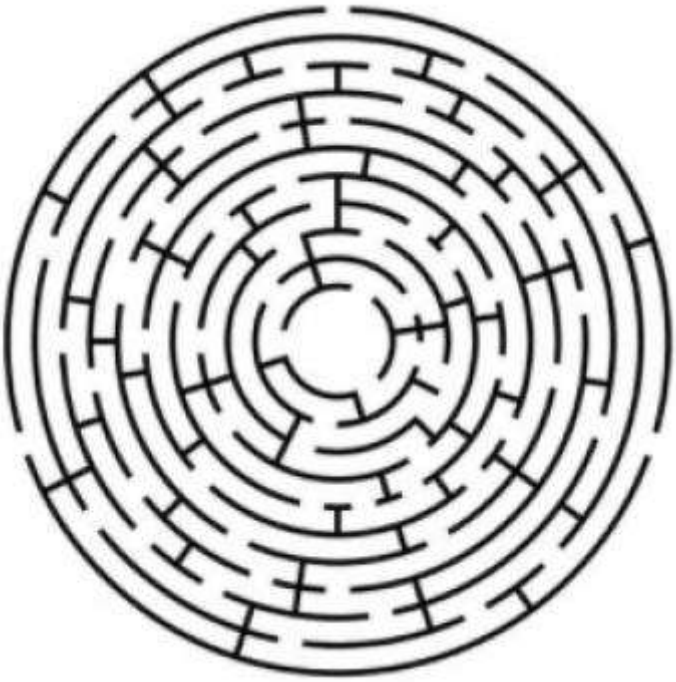
Doodle Amazing

By Ryan Jones and Anthony Klassen
CS 467 Spring 2021



I. Introduction

Our project was to create a phone game, which is called Doodle Amazing. As the name hints at, the app allows the user to take a photo of a “doodle” (hand drawn maze) that the user draws/creates and then the app generates a playable maze. We were both excited to work on this project as we wanted experience working with Unity and mobile development along with expanding our knowledge in the C# programming language. The added intrigue of computer vision, augmented reality and backend databases gave our group a lot to be curious about. Our work this quarter with Unity came full of surprises: some goals that we thought would take a long time were quickly solved with Unity's powerful game engine tools, such as creating a default maze with physics. Other items took more time than expected as Unity, while being a powerful development tool, also has its limits, such as generating maze objects based on an image or simple tasks such as downscaling the resolution of a PNG picture during runtime. Even with this learning curve, we believe we have made a game that is well rounded in terms of having an engaging UI, visuals, gameplay and an added interest of users being able to see their hand drawn mazes come to life on the screen.

As stated above, our game app allows the user to create whatever doodle maze they want, allowing the user’s creativity to show and bring their creation to the virtual world. We are excited for users to experience our quarter long project.

II. User’s Perspective, Setup and Usage

Technology needed:

1. Windows or Mac
    a. Mac is **highly** preferred
2. iOS device
    a. App: Unity Remote 5

Starting the Game:

1. With Unity installed, you can play through Unity by selecting our Doodle Maze Project folder. When Starting Unity, navigate and select the provided folder “DoodleMazeProject” and open this project with Unity.
2. In the Assets folder and then sub folder Scenes, if the Main Menu scene has not been selected, double click Main Menu.
3. Connect your iOS device to your computer and have Unity Remote 5 open.
4. To begin playing the game you may then select the play button at the top mid section of the Unity screen. The main menu should pop up on your phone and you will be able to select through the options.
5. If the game does not display on your phone, we recommend stopping the game in Unity, disconnect the phone from the computer, refresh Unity Remote 5, connect phone again, and press play game in Unity

- **Notes about using Windows:**

- You will be not be able to save a maze when running the iOS build as Firebase for iOS isn't supported on Windows
- You will still be able to take a picture and play the maze
- You will find this error in the console:
    - 

    - You still be able to play and use the game besides what has been mentioned above

Player Controls:

Doodle Amazing requires two main types of input:

Menu: Player navigates menu options using touch controls; simply press the option on the screen to use.

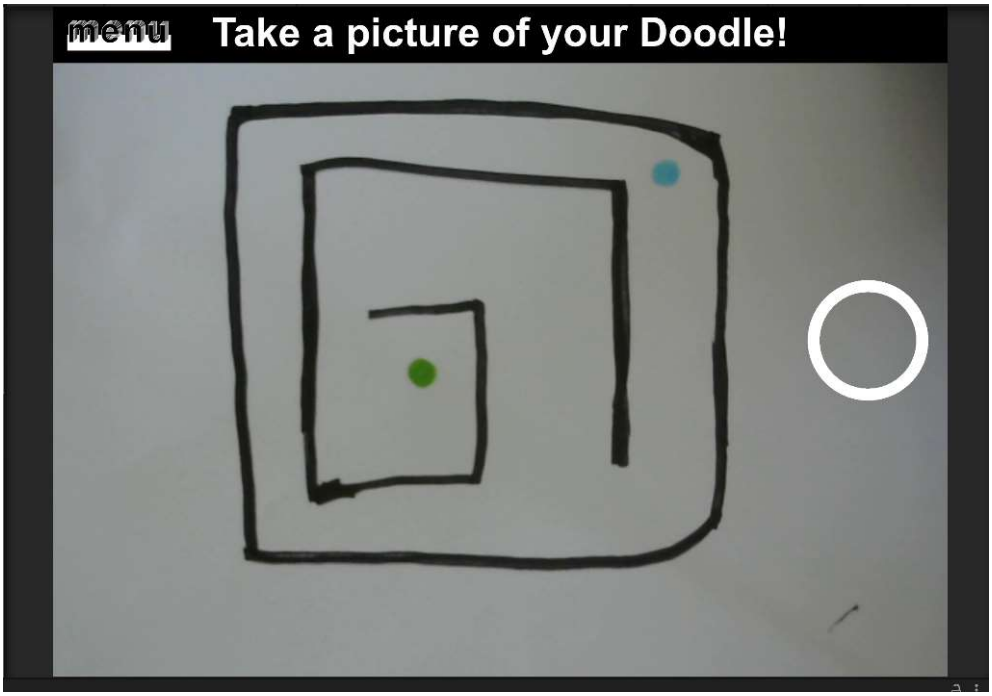
Maze game: Player controls a ball and moves the ball by tilting their phone's screen, therefore updating the accelerometer's input data.

Navigating the Game:

Start Menu:

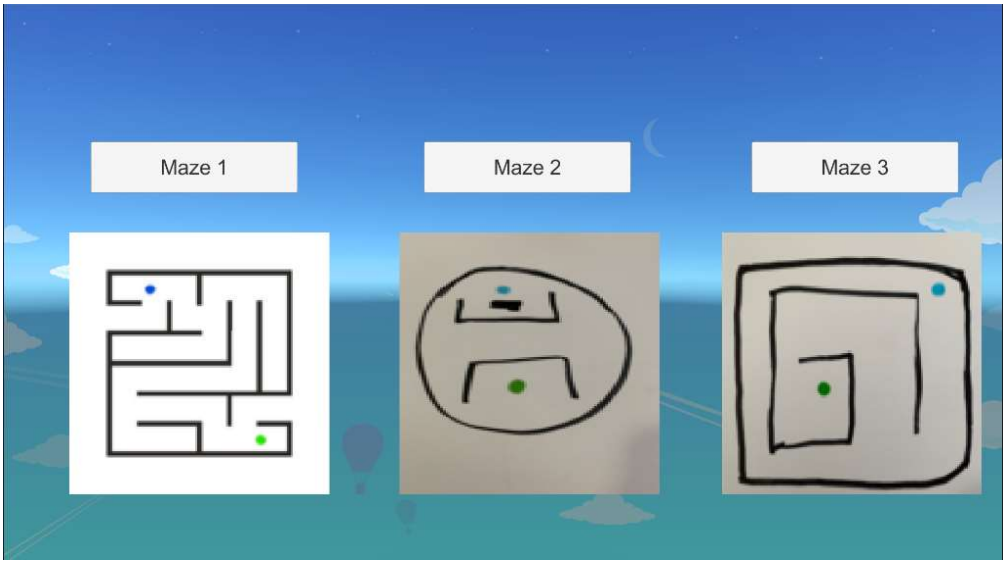
In the start menu you have the options to play our default maze and view how to play the game. Other options that are included are Make New Doodle, Play Existing Maze and How to Play.

Make New Doodle Maze:
Selecting this option will bring up the device camera and allow the user to take a picture of the doodle. Once selected, the user will be able to confirm that the correct picture has been taken, the maze will then be generated and also stored in “Play Existing Maze”.

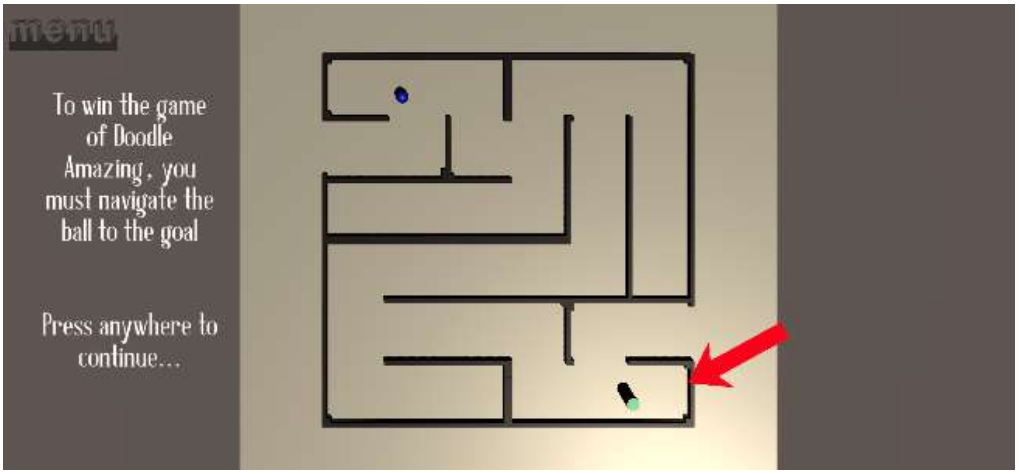


Play Existing Maze:
Users will be able to select from mazes that were taken from the “Make New Doodle Maze” scene.

Play Default Maze:
This option will bring up 3 default mazes to choose from. The original image doodles are displayed and when selected, the maze will be generated from those images.



How to Play:
Users will be shown a tutorial on how to control and complete a default maze.



III. Development Tools

Unity and Unity Collab:
Our main development tool was Unity and its built-in version control system called Collab. Unity is a widely popular game engine that can be used to create video games and applications for iOS, Android, MacOs and Windows. Its strongest features are its easy to navigate 3D graphical interactive toolset and its ability to simulate the game at any point in the development process. Unity Collab is an amazing and yet frustrating tool

to use for version control. It abstracts away normal software control functionality like branches, merging, merge conflicts and slow, repetitive code reviews in exchange for less control and sometimes unpredictable merging behavior.

Google Firebase Storage and Cloud Firestore
Firebase is a platform of various services that make storing data, managing data and analyzing your mobile and web applications easier than ever. In our case we used it to set up a backend database for storing all of the image data and pertinent associated metadata from the maze generation of Doodle Amazing. It provides an easy to use API for working with Unity projects that turns a few simple method calls into functioning database manipulations.

Visual Studio Code and C#
A powerful, yet lightweight and cross platform code editor, Visual Studio Code was our most commonly used tool after Unity itself. Any C# script editing was done within VS Code and the features of C# gave us a lot of flexibility of being to natively call methods for Unity based API's as well as the Google Firebase API. This combined with our previous experience in C like programming languages made the learning curve that much shorter.

IV. Team Member Responsibilities and Weekly Task Breakdown

Ryan Jones:

Tasks Completed	**Time (hrs)**
Week 3 - Research Unity - Watch Unity tutorials - Start User Interface - Created Main Menu	12
Week 4 - Continue User Interface dev - Implement "How to Play" scene - Implement timer for playing a maze	12
Week 5 - Continue User Interface dev - Implement placeholder scene for playing existing mazes - Add menu items for navigating scenes	14

Week 6 - Implement live image display - Implement image capture scene - Worked on MidPoint report	15
Week 7 - Create Firebase Cloud into Unity project - Implement image upload to Firebase Cloud Storage	12
Week 8 - Implement image download from Firebase Cloud storage - Optimize image resolution and maze generation	12
Week 9 - Contribute to Poster - Finish existing mazes scene including adding Firebase Firestore	10
Week 10 - Contribute to Final Report - Demonstrate Project - Clean up generation of mazes by modifying generation algorithm	10

Anthony Klassen:

Tasks Completed	**Time (hrs)**
Week 3 - Research Unity and Create Unity Project	10
Week 4 - Created Default Maze for testing - Ground objects, wall objects, ball object, goal objects	10

- Created Script for ball control	
Week 5 - Created script for goal (win condition) - Create UI for victory screen	15
Week 6 - Research Computer Algorithm or other maze generation possibilities - Worked on MidPoint Project Report	15
Week 7 - Created initial algorithm for Maze Generation with C# script and Unity - Added the ability for ground objects and walls to be generated in script	20
Week 8 - Added the ability for ball and goal to be generated in script - Optimized maze generation script for detecting RGB colors	10
Week 9 - Contribute to Poster - Continue optimizing maze generation - Complete default maze script and scenes - Added scenes to show 3 maze options	10
Week 10 - Contribute to Final Report - Demonstrate Project	10

V. Deviations from Original Plan

A major area where we deviated was our use of a Computer Vision library such as OpenCV. Originally we planned to use a library such as OpenCV but as we researched ways to generate a maze from an image, we decided to make our own C# script and use Unity's ability to detect and isolate pixel RGB values to then generate game objects. This allowed for a less complex codebase but resulted in a tradeoff of robustness of the maze generation. Given more time, we could have implemented object detection of a maze during live image capture (using a platform/tool like TensorFlow) and then only used the image data from that section of the full image to generate the game maze.

Another area where we deviated from our original plan was in what types of systems would be able to use our game. As we worked on our project, we started developing specifically for iOS because both of us had iPhones and did not own any Android devices. Not owning Android devices was the biggest factor in us focusing our efforts on developing and testing exclusively for iOS. The great part about a Unity project is that it is natively cross platform between iOS and Android, so it wouldn't be much extra effort to have a fully functioning Android application as well (we briefly tested the Android built version with an emulator to confirm it works).

VI. Conclusion

There is still a considerable amount that could be improved on this project, but we are proud of what we have been able to accomplish with the allotted time. We both chose this project to get experience working in Unity and we can confidently say that we made the most of our project and were able to dive deep into the Unity environment. Now that we have a taste of a Computer Vision application we can now more confidently assess when certain technologies would be useful. Doodle Amazing was a great project to work on as we were able to write C# scripts that interacted with the system devices such as the camera and manipulated PNG data to create 3D objects.