CS 467
Summer 2020

Doom: Project Plan

"Doom" Team Members

Edmund Dea
Martin Edmunds
Lee Rice

Introduction

The Project Doom Team will be building an Android app, named "Doodle Maze", that allows users to turn a picture of a drawing and play with that maze on their mobile device. This project will expose our team to a wide variety of technologies and frameworks, including Computer Vision, Machine Learning, mobile application development, Unity, and game development. As a team, our experience in these fields is initially limited but this project will allow us to develop highly marketable skills.

User's Perspective

The user starts by making a drawing they wish to see turned into a maze on their mobile device. The drawing must include at least 1 wall, an 'O' representing the starting position of a ball that will be guided through the maze, and an 'X' that indicates the end of the maze.

When the user starts Doodle Maze, they will be presented a Main screen with 4 options - (1) create a new maze, (2) load a saved maze, (3) credits, and (4) exit. If the user clicks (1), then they will be routed to a new screen with the options: (1a) Take a new picture or (2a) Use an existing picture. If the user chooses to take a picture of their drawing, then the camera app opens, identifies the 4 corners of their doodle, and takes a snapshot for the user. Then, the app will crop the new image, validate the image, generate a new maze, and present the user with a 2D height map representation of their drawing as a maze. If the app was unable to determine that the doodle was a valid maze, the app will report an error message and prompt the user with menu options (1) and (2). If the user clicks (3), then the app displays a list of credits and acknowledgements. When the user selects to load a saved maze, the user will be presented with a grid of mazes each with a selectable drop down box consisting of (2a) 'Play a Maze' and (2b) 'Score'. The Score button will show the user the top 10 scores for the selected maze.

Once the maze has been rendered, the game starts. A timer starts counting down and the user utilizes the tilt controls of their phone, controlled via the accelerometer sensor, to guide a ball through the maze to the goal location they designated as 'X' in the drawing. Along the way, the user must avoid randomly located traps, such as slow traps, lava traps, or teleport traps. Power-ups will also be randomly located throughout the maze that affect the ball in different ways, such as a speed boost item.

When the goal is reached, the user is given a final score that is calculated based on the time they took to solve the maze and the number of items they collected. This score is stored on a backend server and is associated with the user's maze. This allows users to compete for high scores.

Client

The instructor, William Pfeil, will be considered the client. The client's requirements will be the default requirements outlined in the project proposal provided by the instructor (Doodle aMazing Project Requirements).

Client Requirements

- This project is an app that converts the image of a doodle into a simple maze on a mobile device.
- Besides walls, the maze contains a 'o' which represents a ball and an 'x' which represents a hole.
- The user takes a picture of the doodle and the maze is converted to the same maze represented digitally.
- The ball is then steered through the maze towards the hole by manipulating the phone (accelerometer).
- Or maybe the ball is steered through the maze avoiding holes to the goal.
- The accelerometers in the phone report the orientation of the phone in space and the ball responds to the tilt of the phone.
- Possibility of using javascript HTML 5 game engine
- Possibility of using Unity or other platform

Sources

- Fair use: This software is for educational purposes only and makes use of non-commercial images under the "fair use" provisions of the copyright act. All sources will be credited.

Program Structure

The mobile device's camera stores the image. The image is then filtered by multiple digital imaging transformations in order to supply a reliable and robust picture that can be utilized by the machine learning module. Our chosen machine learning framework will then classify whether the image is a valid maze. If the learning model identifies that the image as a maze, additional image transformations and filters will be applied to build a maze that is rendered in the Unity game engine. As a fail-safe, if the machine learning framework fails, the user is given the opportunity either to approve the image for conversion or take another picture. Once the image is confirmed the image, along with additional object data, will be passed into the Unity module for maze rendering. The user is presented with a 2.5D perspective of the maze. The player controls a ball that starts at a point designated in the image with a circle and uses their mobile device's tilt sensor to guide the ball. The user attempts to move the ball into a hole that they had previously drawn as an X in their doodle. Once the player reaches the goal, the app stores their final score in an internal database. A scoreboard keeps track of the top 10 scores for each locally stored maze.

Technologies Used and Purpose

- Kotlin
    - Our team had some experience with Java, but decided to use Kotlin to develop our Android app. We believe implementing the app with Kotlin will be a good learning opportunity.
- OpenCV
    - We plan on using OpenCV to acquire an image that will be processed by the machine learning algorithm.
- Google Machine Learning Kit/Tensorflow Lite

        - The Project Doom team will be using Google's Machine Learning Kit and Tensorflow Lite for Maze detection/validation.
            - We will produce a dataset of drawn mazes that will be used to train our machine learning model
- Sobel edge detection algorithm
    - This algorithm will be used to detect the lines drawn by the user that will be turned into walls.
- Tensorflow Lite Support Library
    - This library makes interacting with Tensorflow Lite ByteBuffer tensors easier to work with.
- Java
    - We will be using the Java programming language to interact with the TensorFlow Lite Android Support Library.
- Python
    - Python will be used to write the maze generation program. Mazes created by this program will be used to train the machine learning model.
- Unity
    - Unity will be the game engine that renders the maze and handles game physics.
- GNU Image Manipulation Program
    - This program will be used to create an icon for our team's app.
- Android Nougat 8.1
    - We will be creating the android app using Android version 8.1 that was released on August 21, 2017.

Initial Plans

Android Dev

- Create interface
    - Main screen (Lee)
        - Add menu background soundtrack
        - Add button hover and button click sound effects
        - Options
            - Create Maze
            - Load Maze
            - Scores
            - Credits
            - Exit
    - Credits/Attributions screen (Edmund)
        - Displays a list of the creators of the app along with attribution to artists whose free assets we've used in the game
    - Create Maze screen (Edmund)
        - Capture new maze button
            - Open camera app, capture an image, initiate generating a maze, and trigger Unity app to load
        - Load new maze from photo gallery
            - Open user's photo gallery, select an image, initiate generating a maze, and Trigger Unity app to load
    - Load Maze screen (Lee)
        - Displays list of saved mazes
            - After clicking on a maze, user can select these options:
                - Play

            - Trigger Unity app to load
        - Scores
        - Share
        - Delete map
    - Scores screen (Martin)
        - Displays the top 10 scores for that maze
    - Scan doodle screen (Edmund)
        - ML algorithm automatically detects the 4 corners of the doodle paper, captures the image, crops the image, and initiates converting the image to a game object.
    - Post Game Menu Game Over/Replay/Main Menu screen (Martin)
        - Displays user score along with top 10 scores for that maze
        - Replay button
            - Takes user back to game start
        - Main Menu
            - Returns user to the welcome screen
- Style Android interface (Lee)
- Program reading/writing to local storage for map and score data (Martin)
- Create icon for the app product (Lee)
- Program ability for user to choose which power-ups and traps will be on the maze they uploaded (stretch requirement)
- Share Screen (stretch requirement)
    - Displays a screen that allows the user to enter an email for another user. If the user is new, then an email invitation is sent to the user to register. If the user is already stored in the backend database, then add the maze to the user's profile and send an email alerting them that a friend shared a maze with them.
    - Alternative implementation ideas: Asks the user to select a maze and attaches the maze to an email for sending.
        - Prevents need for backend database, which would require remote hosting, and user authentication.

<u>Computer Vision</u>

- Write a program that will randomly create mazes that can be used to train the machine learning model (Martin)
- Digital Image Processing
    - Write code to for edge detection (Martin)
    - Write code to pass processed image to the machine learning algorithm for validation as a maze (Martin)
    - Write code to filter noise and clamp image (Martin)
- Validate whether a new image is a valid maze (Lee/Ed)
    - Create training set for validating the convolutional neural network (CNN) model
    - Create a dataset for testing the convolutional neural network (CNN) model
    - Use dataset to train model
    - Develop CNN model
- Identify the four corners of the maze to identify edges used for cropping the image and perform cropping (Lee/Ed)
    - Create training set for validating the convolutional neural network (CNN) model
    - Create a dataset for testing the convolutional neural network (CNN) model.
    - Use dataset to train model
    - Develop CNN model
- Convert new images from JPEG to RGB888 for image preprocessing (Lee)

- Identify the starting position of the ball (O) and the end of the maze (X). Then, crop the X and O from the maze. (Lee)
    - Create training set for validating the convolutional neural network (CNN) model
    - Create a dataset for testing the convolutional neural network (CNN) model.
    - Use dataset to train model
    - Develop CNN model

Unity Dev

- Write code to build walls from detected lines (Lee/Martin)
    - Apply sound effects to game colliders
- Program accelerometer sensors to move ball (Martin)
- Code game start sequence (Edmund)
    - Place ball at start location
- Code end sequence (Lee)
    - End the game when the ball reaches the goal
- Sound Effects
    - Add sound effects for all game objects, including the ball and possibly power-ups and traps (Martin)
    - Add soundtrack for the maze (Lee)
- Program score calculations based on time to complete, powerups collected, and attempts
    - How to calculate score from time? (Martin)
- Code power-ups (stretch requirement)
    - Power-ups are placed in random locations in the maze
    - Speedball
        - Increases movement of the ball
- Code traps (stretch requirement)
    - Traps are placed in random locations in the maze
    - Ice traps
        - Slows the balls movement speed for a specified duration of time
    - Hole traps
        - Transports the ball back to the starting point
        - Code the ability for the ball to jump over holes by pressing anywhere on the screen

Final Testing

- Test on various emulated Android devices (All team members)
    - Android Tablet
    - Android Phone

Mid-point Project Check

- Write submission contents section of report (Edmund)
- Write project status portion of the report (Edmund)
- Write instructions section of report (Lee)
- Package instructions along with code and submit (Martin)

Final Report Assignment

- Write introduction (Lee)
- Write description of program from user perspective (Lee)
- Write usage instructions (Edmund)
    - Include two graphical examples (Martin)
- Write description of how software and systems function together (Martin)

- Write a listing of technologies used and descriptions (Lee)
- Write a description of each team member's accomplishments (All team members)
- Write conclusion (Edmund)

Team Member Task List

Edmund Dea

Week	**Tasks**	**Time Estimate (hrs)**
Week 2	Create Maze screen Scan doodle screen	20 hrs
Week 3	Crop image manually Initiate edge detection algorithm	20 hrs
Week 4	Code game start sequence	20 hrs
Week 5	Validate whether a new image is a valid maze	20 hrs
Week 6	Identify the four corners of the maze to identify edges used for cropping the image. Then, automatically crop the image.	20 hrs
Week 7	Fix showstopper bugs	10 hrs
Week 8	Work on final report / Final Testing	10 hrs

Martin Edmunds

Week	**Tasks**	**Time Estimate (hrs)**
Week 2	Finish MazeGen Program Unity Heightmap Generation	20 hrs
Week 3	Program controller inputs for game / Scoring Image noise, clamping and Edge Detection code for Computer Vision Passing CV data to ML module	20 hrs
Week 4	Research Save File Format Program reading / writing to local storage for map Scoring Data	20 hrs
Week 5	Post Game Menu Game Over Replay Main Menu screen	10 hrs

Week 6	Identify the user marked locations (X's O's) and cropping them out of the image	20 hrs
Week 7	Unity Sound Programming Stretch Requirements	20 hrs
Week 8	Work on final report Final Testing	20 hrs

Lee Rice

Week	**Tasks**	**Time Estimate (hrs)**
Week 2	Unity HeightMap Generation	20 hrs
Week 3	Load Maze Screen Allow for user to manually crop image	20 hrs
Week 4	Convert new images to JPEG from RGB88 Code end sequence Identify start and end of maze	30 hrs
Week 5	Validate if a new image is a maze	20 hrs
Week 6	Identify the four corners of the maze to identify edges used for cropping the image. Then, automatically crop the image.	20 hrs
Week 7	Maze soundtrack Bug hunt/Fix Create icon for app Stretch Requirements	20 hrs
Week 8	Style Android Interface Work on final report Prep for final presentation Create report	20 hrs

Graphical Examples

Figure 1.1: Game Architecture

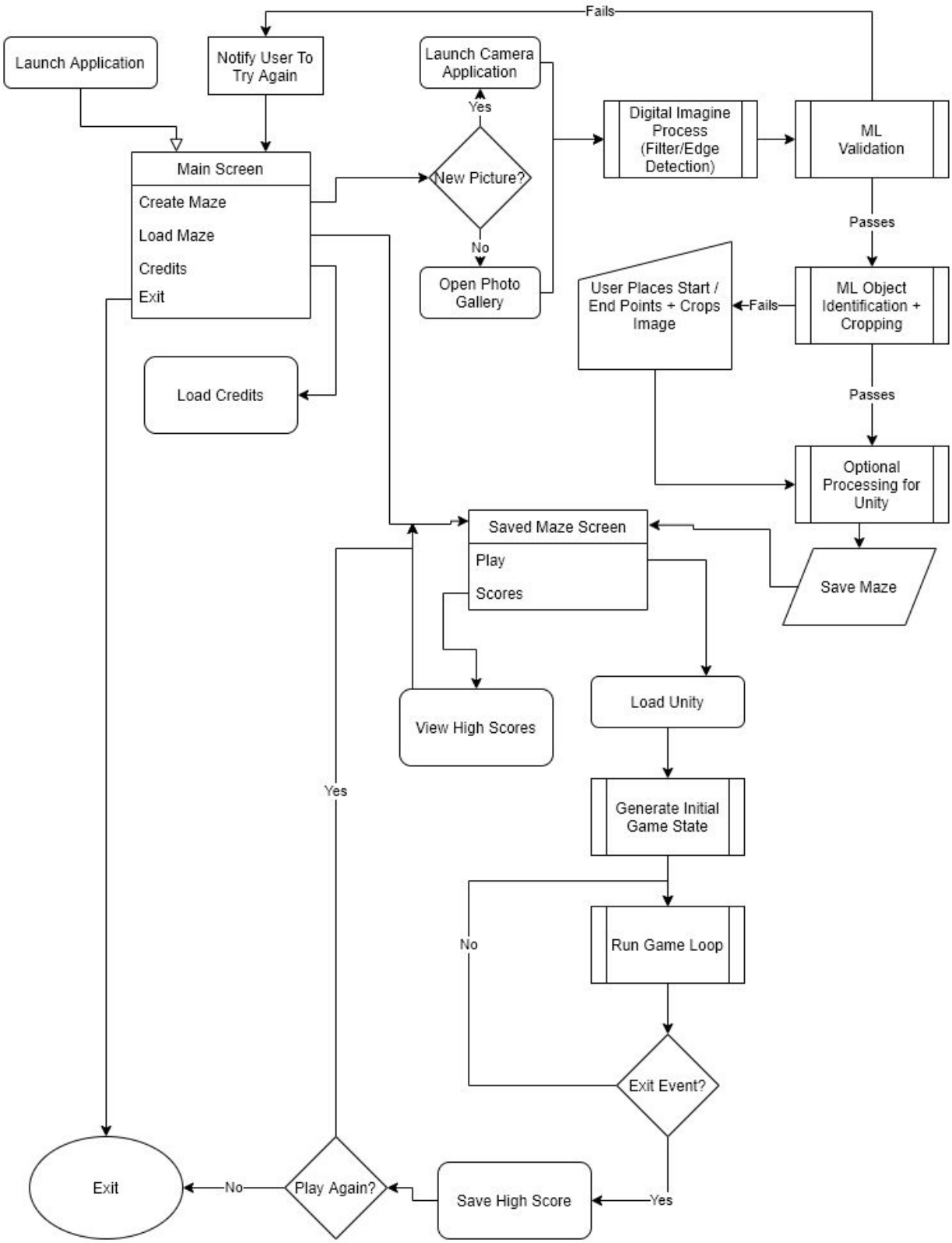

Figure 1.2: Interface Screen Prototype

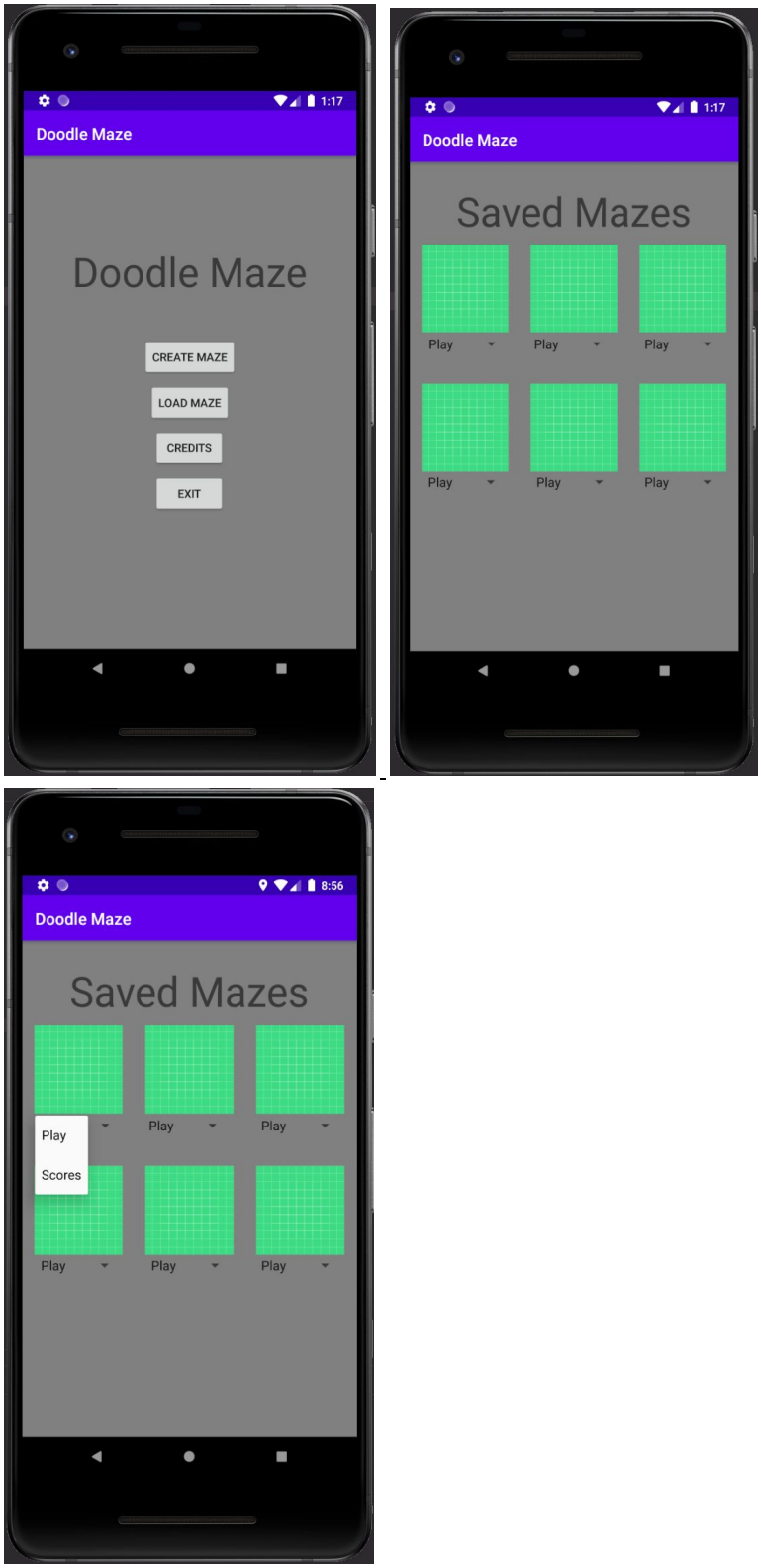

Figure 1.3: Game Interface Prototype

Conclusion

The Doom team plans to create an interactive mobile application that showcases multiple technologies to provide an intuitive and fun experience for the end user. By utilizing frameworks and applications such as Tensorflow Lite, Android Studio and Unity, the team expects development to take a minimum of 300 hours to complete.

References

1. https://www.tensorflow.org/lite/guide/lite_support
2. https://www.123rf.com/photo_63986374_stock-illustration-square-maze-top-view-3d-illustration-.html