4SPACE APPLICATION - FINAL REPORT

Kathleen Nguyendon
Shannon Hargrave
Khanh Dang
Myles Alcala

Sponsored by: Dr. Amy Hoover

IT/CS 491 CAPSTONE
Dr. Osama Eljabiri

Table of Contents

I. **INTRODUCTION** by Myles Alcala

4SPACE is a mobile music-concept teaching game our capstone team designed over the semester. Overseen by Dr. Amy Hoover an A.I. university specialist with experience in game design, the application focuses on teaching and music by utilizing *melodic mimicry*. This type of *interval training* is designed to help familiarize players with melodies, notes and scale functionality.

App Runthrough, Basic Gameplay

The app was designed for ease of play with intuitive (and original) UI design, such as simple menus and comforting color scheme. A player runs the app, taps the first page, choose a difficulty and are sent gameplay. The game is setup as steps and stars in a flat spacey landscape. Tapping on the stars and steps plays a specific note, and sliding the character to them submits that note as your 'answer' for what the melody segments are.

Problem Definition

We decided to use our semester's resources on this project because of its promise to attempt to aid people. During design, we thought it best the game not emphasize any particular instrument but rather a broader range of musical familiarity.

The problems this application seeks to address centers around music learning in an engaging way. These include: musicians of all caliber trying to improve their pitch or melody; people interested in music learning; and lastly gamers who simply enjoy playing games.

The game tends to naturally call for the player's attention during melodic segments and this is due to the game's design. It makes it slightly more 'focused' for the player as they need to truly pay attention to get perfect marks on the segments - at the same time, it discourages 'lazy' gaming where game vigilance isn't recognized or encouraged.

Iterative Design

Part of the challenge of crafting this app was the balancing of teaching elements but keeping it engaging. We thought many alternative apps fall too much on one side of the spectrum, and for good reason. It is difficult to find a harmonizing balance between the two and it took our team longer than expected to flesh out a design that was both fun and playable but unique and not dull. We constantly came back to the drawing board about educational and game design decisions and have explored 'musicified' designs that were similar *Temple Run*, 2D platformers, and a variety of theme and mascot concepts. Our design solidified with our current product: a simple melody-based simon-says style of game and a fun world to play in.

Glossary of Terms Used

1. **UI -** user interface; in games usually the focus of this interaction between users and the software is keeping things simple, understandable and engaging.
2. **pitch -** (1) : the property of a sound and especially a musical tone that is determined by the frequency of the waves producing it : highness or lowness of sound (2) : a standard frequency for tuning instruments
3. **melody -** A melody, also tune, voice, or line, is a linear succession of musical tones that the listener perceives as a single entity.
4. **melodic mimicry -** dealing with repeating melodies; in an educational aspect, helps reinforce melodic familiarity.
5. **interval training -**

II. **PROJECT MANAGEMENT** by Myles Alcala

Project management of this project over the course of the school semester has been long-winded but beneficial process for me personally. I was excited to get to work with all my team members shortly after our first meeting in the school atrium. We were all interested in games and design, and the opportunity to build an educational product from the ground up.

Task Analysis & Roles

During our forming processes in the atrium, when the capstone teams all first assembled, it was initially Khanh and I with a loose concept of a music teaching (or at least music-involved) game. My focus was in user experience, musical asset design, and gameplay while Khanh leaned heavier towards the programming functionality of video games. Shortly after, Shannon approached who was easily one of the best game programmers I've seen over the semesters and Kathleen joined later with an entire focus in UX/UI and design. Our roles and task analysis afterwards have been geared towards our strengths to push the best product we could.

We broke down tasks using based on our skill sets and familiarity with applications. Music asset and sound design went my way, on top of managerial tasks and facilitating communication between team members. Shannon and Khanh broke up the major tasks of Unity programming, mobile input, menu systems, scoring systems, player movement, step/star functionality. Kathleen worked the bulk of the animation style, fonts, menu designs and the educational component of the overall design.

Management

Management of the project was at times a monolithic task with keeping up with dates, meetings, and occasional missing members but it was overall a fulfilling experience at the end of the project. We used SCRUM style implementation within Trello board, an online board tracker where users can make lists, delegate tasks, set due-dates, and a lot of wonderful team-based stuff. We divided the project up into

four sprints, 2 - four-week sprints, and 2 - two-week sprints with the majority of our development being in the first half, and the bulk of our integrations being handled in the second. We proceeded with sprint summaries after each sprint, to gauge our progress and see how we could improve - or solve obstacles that came our way.

III. **DEFINE** by Khanh Dang

StakeHolders :

Our Team : (Developers, Artists, Sound Designers, Managers, etc.)
Our Sponsor : Professor Hoover
Players (Ages 10 and above)
Mobile Users(Android users)

While we were developing 4SPACE, these stakeholders were the main thoughts and focus for this capstone project and for future references as well. As we kept these stakeholders in our mind, 4SPACE developers can create a fun learning gaming experience. Our task was to develop the game and our sponsors provided us with important feedbacks and told us how to improve our project moving forward. During the final capstone showcase we were receiving a lot of good feedbacks on saying how good our game look and how fun it was to learn music through gamification. For the future, we potentially would love to publish this game to the Google Store and Apple Store for both iOS and Android mobile users to enjoy.

Requirement gathering techniques:

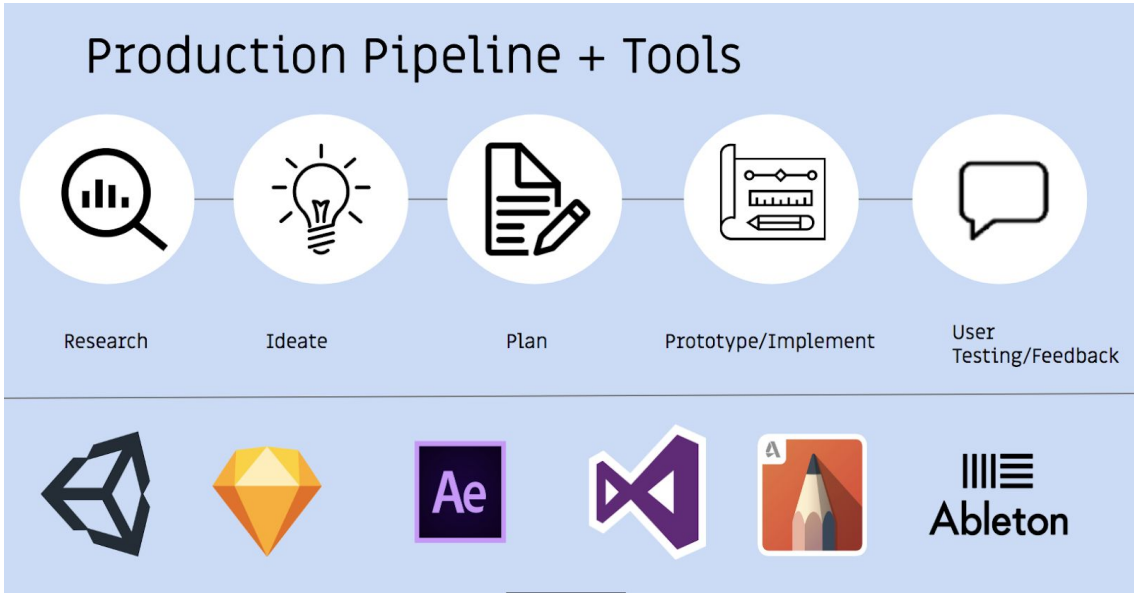
Our sponsor was working with a few doctors in the Netherlands wanting to conduct on research about how music games could potentially help people improve their musicality. This was our main goal was to come up with a game that could be both useful and fun in the process of improving one's musicality. Knowing this beforehand allow us to come up with a quick decision to make the game mobile and on Android since iOS development is quite time consuming. Since our team is gamifying the concept of music learning we always hit the roadblock of making sure there is an equilibrium between fun and learning.

Scope and Requirements:

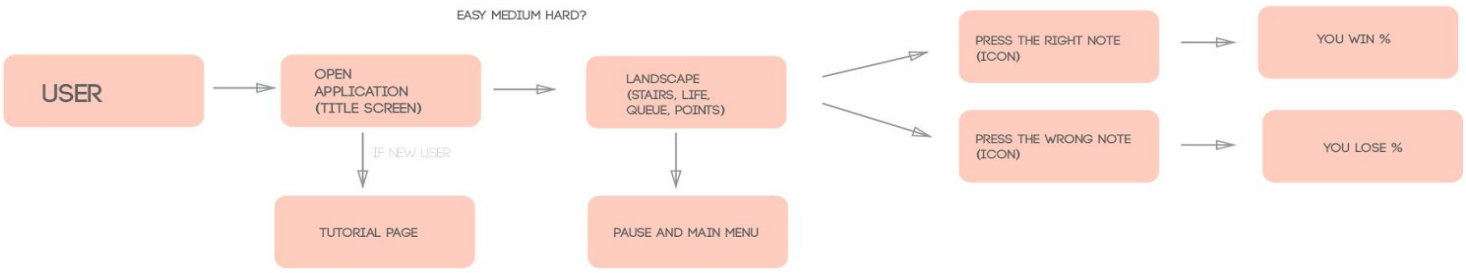
Our scope and goal is a single player experience where the player is immersed in the space theme musical world of 4SPACE. In 4SPACE the player get to experience the spacey melody and learn how to playback those melodies on top of their head. There are currently three songs and each song has their own difficulty.

The technologies that we use to create 4SPACE are:

HardWare: Laptops, Computers.
Software: Unity, Ableton Live, Sketch, Sketchbook Pro, After Effects



Use Case/Application Architecture

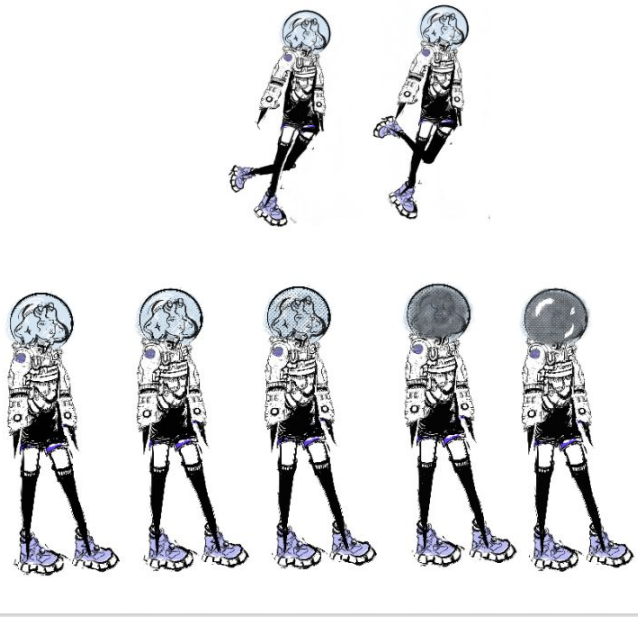


Our Gameplay breakdowns are as follow. The player follows listening and playback indicator this will let the player know when they are able to play and queue up their input. The listening part is there the player must pay attention to the melody and simply try to match along the melody by click on the stairs to hear if the notes are matching. During the Playback/Repeat Notes part this is where the player will try to copy the melody by sliding the player creating a jumping effect on the predicted appropriate note steps/stars. As mentioned before, players can

Tap steps/stars to hear that track's note, Slide to them to input what you think the melody's note was. The player gets a rating after each melody segment and at the end of the track will display the player's score result to show how well they did in the song through a percentage

IV. **DESIGN** by Kathleen Nguyendon

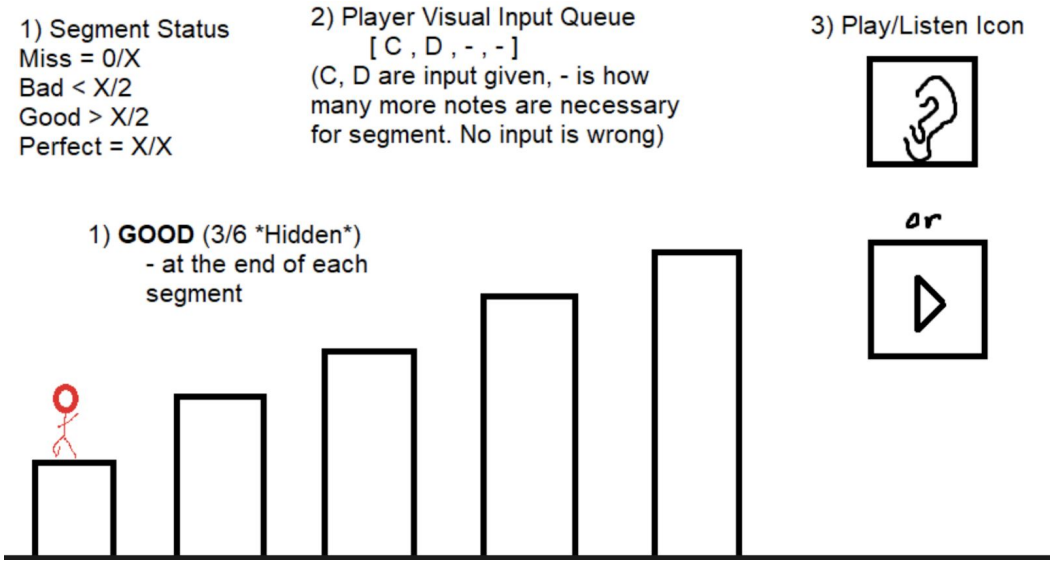
4SPACE is a educational music game that teaches musicality - meaning that it intends to teach music sensitivity and talent. The game was inherently created for mobile platforms, such as android in order to be more accessible, portable, and inclusive. 4SPACE uses interval training - training in which a musician alternates between different notes, typically requiring different rates of speed, degrees of effort, etc. Training players ages 10+ to recognize the gaps and pitches that our game plays for them. The application is similar to the classic game, Simon Says, in the sense that a melody is played to you and you have to repeat the short melody in order to move on in the game. 4SPACE'S Design tackles the issue of educational games being forced and not fun -- Our game is not only educational and deeply rooted in research, but also fun and engaging. The saturated art style and narrative/themes about space provides a less mundane environment to learn and improve the musicians/person who is interested in music's pitch sensitivity. You play as a fully animated space girl jumping from note to note in a space suit.



Game Design

4SPACE is a platformer where the note or notes are played first and the person has to repeat it. The player repeats the note by jumping on blocks that represent one of the 8 notes on the keyboard from middle c to high c. In early game designs we had visual cues to help the player, but we realized that using strictly audio cues aids the players more in being better at music in general. 4SPACE only uses audio cues the

player can repeat in order to fully train the ear of the player. There is Input queue that scores the player on how well they did at the end of each segment/song. The logic behind the queue and scoring is illustrated in the diagram below.



Levels of Difficulty

4SPACE was designed to have 3 levels of difficulty with 3 different tracks to correlate to each level. There is a hand composed track for each level - created and compiled in Ableton. Our music composer created the “easy” track to be lighthearted and simple to follow consisting of only 3 notes to repeat. “Medium” includes around 5 notes to repeat and is more complex in composition. “Hard” utilizes all of the notes and is the most difficult track and level in the game.

Mechanics

Movement:

The player moves by swiping in order to get from block to block. And occasionally star (representing the sharps and flats of notes) to star. There are colliders on each box to ensure that when the player swipes from one area to the other they land on the proper box which indicates a note.

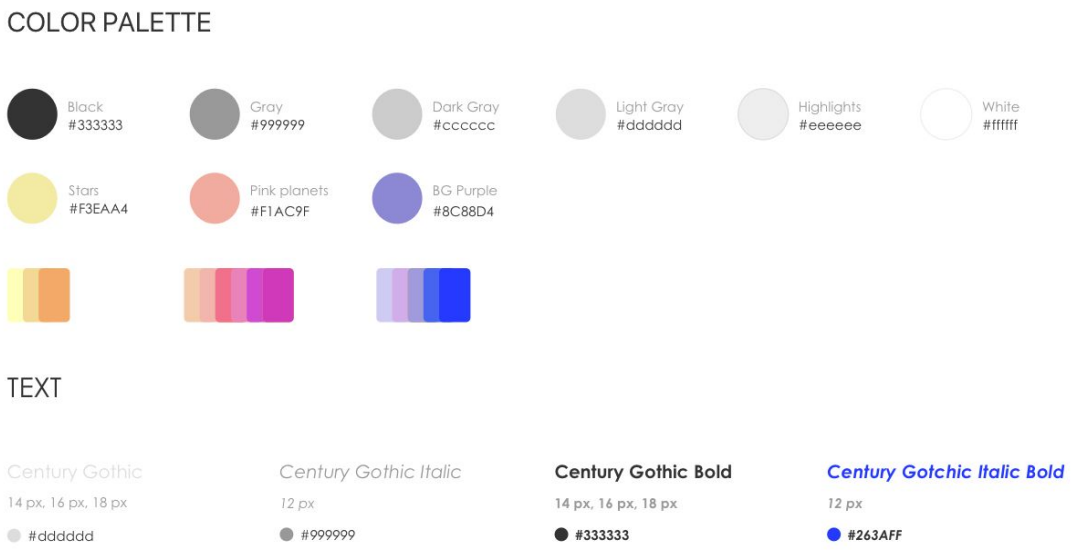
Input:

The player swipes in order to move from block to block or star to star when repeating notes. This adds their note to the queue and checks whether the answer was correct or not. The player also has the option to tap on blocks to hear the note before swiping and inputting their final answer.

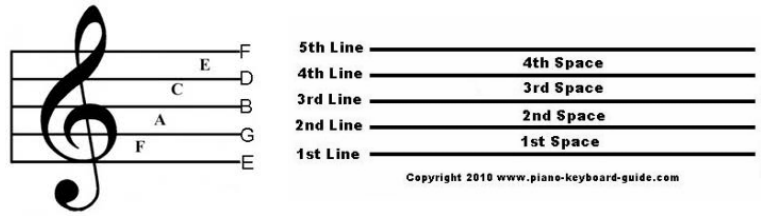
User Interface:

The user interface and experience is simple and intuitive. The UX/UI was first sketched/wireframed with a pen and paper/on sketchbook pro. It was later developed into a low fidelity hud in sketch and then

further refined and prototyped in sketch to create high fidelity final assets that were later integrated in unity. A style guide was created in order to dictate a clean, consistent, and precise interface.

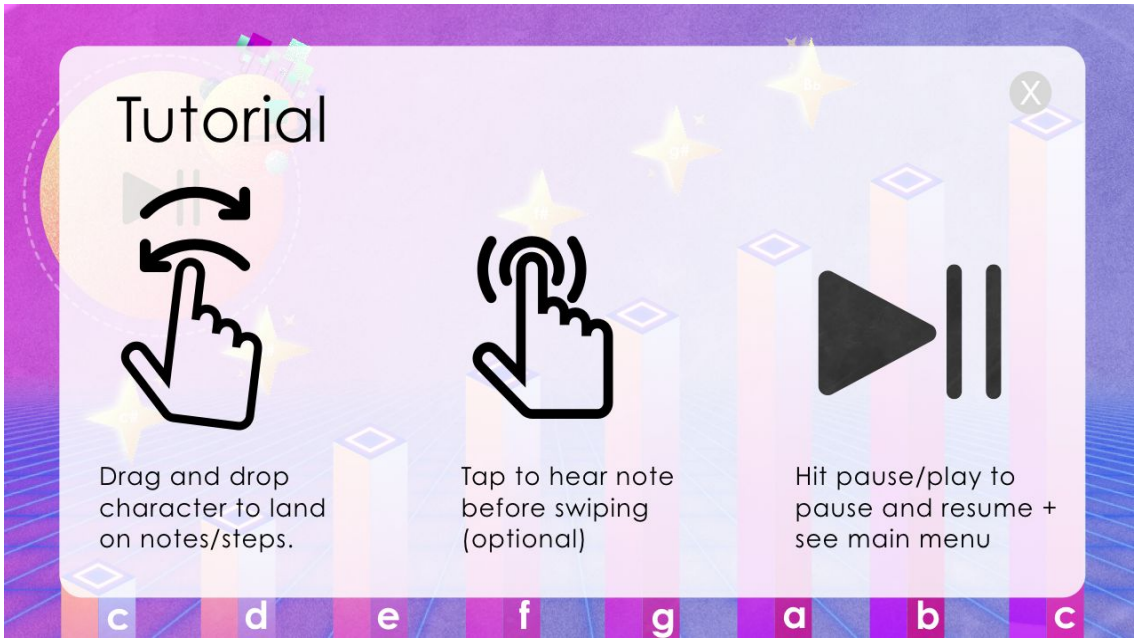


When you first open the application on your device, the logo loading screen and logo will show. The logo was created after various iterations on paper and digital mediums. The logo has 4 spaces in the E to represent the staff sheet music which has 4 spaces, and 5 lines.

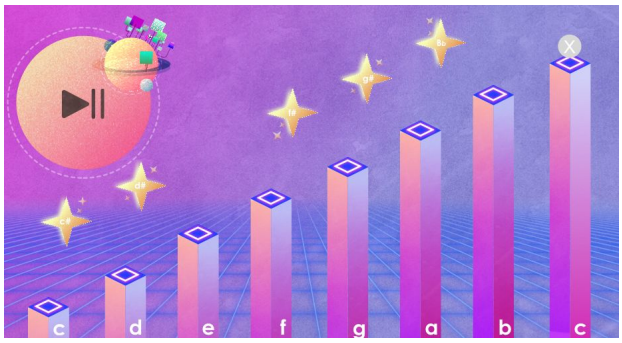


The option to select the level and difficulty of the song you will be tested has an animation that fades in like so.

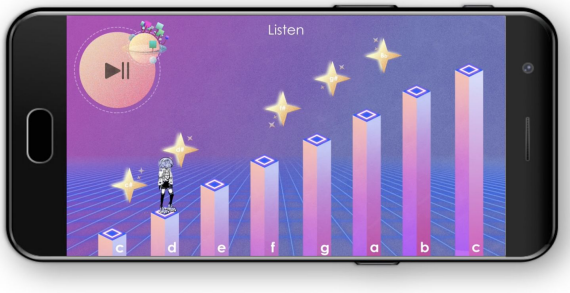
Then a tutorial page will show that can be exited out of at any time. This explains the game to the user and what they have to do in order to continue further into the game.



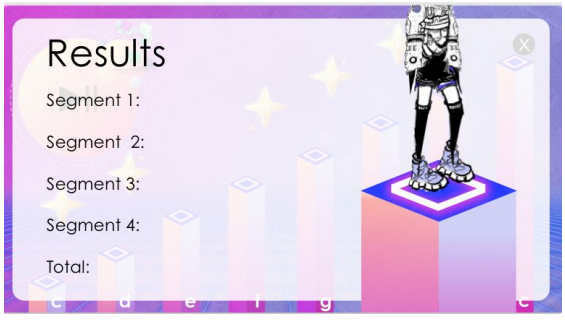
The final art/landscape has a planet on the left hand side that is equipped with a pause/play icon. If you press the pause/play icon you can access the main menu that allows the user to start over, or change difficulty.

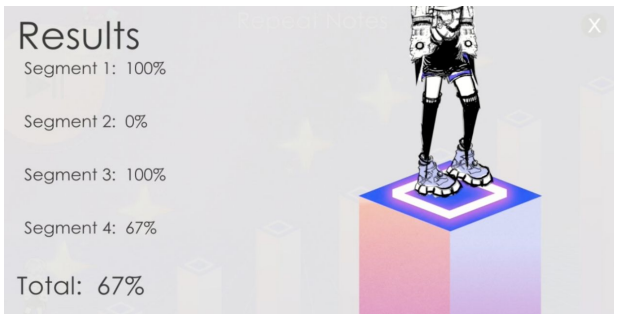


When you are supposed to “listen” to the song the text “listen” bobs/animates in. While you are supposed to repeat/implement, the text “Repeat Notes” bobs in.



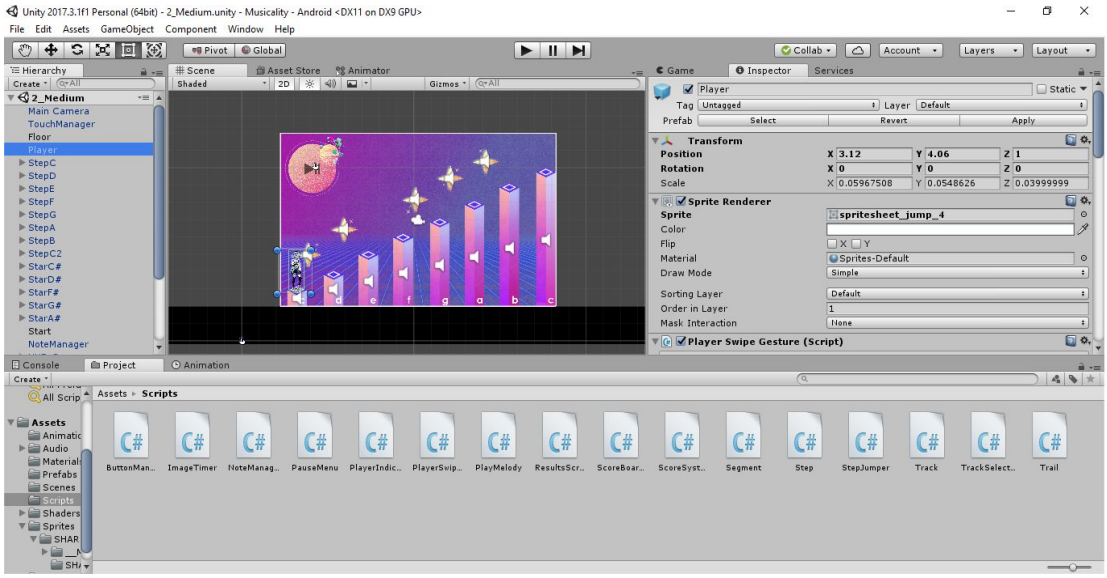
Finally, your final results are displayed based on your performance and accuracy.





V. **Development** by Shannon Hargrave

Our project was developed using Unity, which is a game engine used to develop both 2D and 3D games. In addition to Unity being a great tool for game development, it also makes version control very simple. Version control throughout this project was accomplished using Unity collaborate. Unity Collaborate is a feature that makes syncing up project changes very intuitive. It allows all of your changes to be uploaded to the cloud with the push of a single button. These changes can then be downloaded by all of the other members of the team. However, the service would occasionally not work as intended. Sometimes changes that were uploaded would not be correct when another team member downloaded them. To solve this, we would have to save a local copy of the thing we changed, return to an earlier state of the project, and then reimplement the feature. In addition to this, there were a few other major development hurdles.



Screenshot of the game in the Unity Editor

Detecting mobile gestures and touch input was another major hurdle. While Unity does have built in methods of detecting touch input, they are not very efficient. Due to this, a third party library had to be integrated into the project. The library we used was called TouchScript. This library made it significantly simpler to detect when a touch is a part of a gesture. In addition, the library made implementing callbacks for gestures extremely intuitive.

Character movement was a another major obstacle in completing this game. The steps and stars in each level contain an invisible point above them. When the player swipes from the character to one of the steps/stars, the character would then jump towards that invisible point right above it. The difficulty of getting this movement to work and look fluent was greatly

underestimated. The initial solution that was implemented was using Unity's built in spherically interpolation.

Spherical interpolation in Unity requires several steps. The first is obtaining the position halfway between the character and the target position. Next, the following two vectors must be obtained: one pointing from slightly below this halfway point to the target position (will be referred to as the target vector) and one pointing from slightly below this halfway point to the character position (will be referred to as the start vector). In addition, variables for the start time, and max time of the movement were maintained. By subtracting the start time from the current time, and then dividing by the max time, we obtained a variable representing whether or not the jump was finished (will be referred to as the completion time). After acquiring all of the aforementioned variables, Unity's Vector3.slerp function was used. This function spherically interpolates between two vectors by a certain amount of time. The two vectors used for this were the target vector and start vector variables. The completion time variable was used as the time variable. The character's position is then set to the vector that is returned from the spherical interpolation function.

The problem with this method is that the character would jump in an unnatural circular motion. This motion was less noticeable when jumping to a relatively close location. However, when jumping to a distant location, the player's character would move in a large circular motion that sometimes resulted in them going offscreen. This lead to a new solution having to be found, which ended up being cleaner and simpler.

The second method implemented to solve character movement was using the projectile motion equation for obtaining the max height.

$$h = v_0 t_h \sin(\theta) - \frac{1}{2} g t_h^2$$

$$h = \frac{v_0^2 \sin^2(\theta)}{2g} .$$

When the player is about to perform a jump, the distance and max height can be determined without the use of this equation. After those two variables are found, the first thing that is solved for is vertical velocity. This velocity is then used to solve for time, which is in turn used to retrieve horizontal velocity. These values are applied while the player's character is about to jump, but still making contact with the ground. Afterwards, these values are recalculated when the character has reached her max height. This time the vertical velocity will not be calculated, since it will be handled entirely by gravity.

`

REFERENCES

Based on research done by Dr. Hoover's colleagues in the Netherlands lead by Dr. Rafael Bidero.

1. **http://goldenhearing-app.herokuapp.com/#/music-test/**
2. **https://www.meludia.com/**