Asthma App

Manar Ajhar
manarajhar@hotmail.com

Dr. Suha Al-Naimi
sal-naimi@sharjah.ac.ae

October 10, 2024

Abstract

[Text Here]

1 Introduction

[Text Here]

2 Methodology and Procedures

[Text Here]

2.1 Programming and Mathematical Terms

There are several components and aspects in the fields of programming and mathematics that we need to discuss before heading to the methodology:

- Class: An asset that contains special properties and represents a specific object in a system or program.
- Game Engine: A computer program that is used to develop video games.
- Inheritance: The process of a class to derive the properties from another class to have all its properties and other more. For example, if "Class B" inherits from "Class A", then "Class B" is a sub-class of "Class A", and "Class B" gets access to all (or most) of the properties of "Class A", while also has its own. Inheritance represents an "is a" relationship.
- Singleton: A class that can only have one instance of it.
- Library: A collection of prewritten code that serves a unique purpose to help programmers enhance their work tasks and projects [1].
- Script: A computer file that contains code written in a specific programming language. Some scripts contain only one class, while others can have as many classes as possible.
- Encryption: The process of converting a file in a readable format to an unreadable format to prevent users from reading and tampering with the information [2].
- Decryption: The process of converting a file in an unreadable format back to its original (readable) format [2].
- Axis: A long, straight, and infinite line that represents a specific direction. In 3D graphics, there are three axes (each in their negative and positive directions, respectively):
    1. X-Axis: Left and Right
    2. Y-Axis: Down and Up
    3. Z-Axis: Backwards and Forward
- Transform: The geometrical measurement of an object in a 3D environment. There are three components of a transform:
    - Position
    - Rotation

  - Size

- Renderer: The component of making an object visible under certain properties or rules.
- Physicsbody: The component to make an object have gravity and other physics-based behaviours.

2.2 Unity

The program that the team agreed to build this endeavor with is Unity [3]. It is the main program used to develop the application. It is a game engine that has a friendly user interface and allows developers to flexibly program their games with as many utilizations as possible. The version that was used for this project was "2021.3.5f1". In Unity, the programming language that was used was C#. C# is an object-oriented programming language, which means we can create classes and assign them to certain objects.

The target platforms that were desired for this game are smartphones and tablets. They allow Augmented Reality (AR) technologies, having virtual scenes and objects "blend in" with the physical world. AR is an interesting technology that is being more accessible to consumers. Smartphones also have touch as their input method, which can simulate everyday actions, such as picking-up objects, pressing buttons, or shaking objects.

However, other tools were used to complete the application. For example, Photoshop was used in this project to design badges, buttons, icons, and other UI elements. It also helped change the color or hue of UI elements. The version of Photoshop that was used is "20.0.4". Another software that was used is Blender (Ver. 4.1). It was used to make simple modifications on 3D models and assets and make some 3D animations. A third tool is GitHub (Ver. 3.4.1), which was used to sync all my works and updates to all the other clients involved in the project, as well as keep track of the project's history and updates. More tools will be discussed throughout this article.

2.3 AR Package

The AR package that is implemented for the project is Google's ARCore package. The reason that this package is chosen for this project include:

1. It is compatible with iPhones and Android devices, which are the most dominant devices in their market[1]
2. Unity can let the user download the package natively from it
3. ARCore has the ability to let detect physical surfaces, such as floors, walls, ceilings, and tables. Therefore, no specific pattern is required, which are similar

[1] As of this document, according to Google's website, in terms of iPhones, ARCore supports up to iPhone 13 [4].

to traditional AR methods

Because we are using an AR environment, there needs to be special components to run the application in AR, such as AR cameras and sessions. These are assets that are within Unity. In the phone, when the camera detects a physical surface, an on-screen button will appear to tell the player to start the game. When the button is pressed the virtual scene will be placed on the very spot that the camera picked.

2.4 Loading and Saving

Loading and saving are very important when opening an application to avoid losing progress from any session. It is done using a ". json" file. It stores information in text-form. There is a C# script that read from the file and writes to it for loading and saving, respectively. It also encrypts the file to avoid users from easily tempering with the data. Each savable data has at least two variables: 1. Only one key variable and 2. at least one value variable. The key is basically an ID, which is a unique string that contains 32 characters. It can be generated via C#. The value can be of any type. Therefore, we needed to program what value do we want to save from what asset, and the asset needs a key as a reference to know where the value should be stored. The process of loading is:

1. Decrypt the file
2. Read the keys and values from it
3. Convert the values from text-form to the correct type
4. Find the variable that has its ID match the current one used in the search and assign its value to the variable
5. Keep doing this for all the data
6. Encrypt the file again

The process of saving is:

1. Decrypt the file
2. Read the keys and values from the classes
3. Convert the values from their current form to text
4. Write the information into the ".json" file
5. Keep doing this for all the data
6. Encrypt the ".json" file again

The information that we are saving are:

- The badges earned
- The answers on the action plan

- The settings
- The type of user

2.5 Programming and Designing the Action Plan

The Action Plan is a survey that prompts the user to ask questions about their status with asthma. It also has some questions about their personal information such as birthday, name, and gender. The overall number of questions are twenty. However, each question has a different type of answers:

1. Integer Answers
2. Float Answers
3. String Answers
4. Enum Answers
5. Answer with Two Strings
6. Date (DD/MM/YYYY Format, all integers)

Eventually, only if all questions are answered, the user will be able to make a PDF form of the plan (The PDF is explained in section "**The Action Plan's PDF Form**"). Also, they are saved, and if the user accesses the action plan again, the input fields will be filled with the previous answers.

Enums

There were two types enum questions: 1. The color of the inhaler; and 2. The user's gender. These are to ensure that the user do not provide various responses to questions that only need a specific answer. Visually, the answer of enum questions is presented in a dropdown menu. When the user taps on the menu, a dropdown box will appear below it, listing the possible choices based on the enum. The user can tap on a specific answer or accept the already selected answer.

Two Strings

There was only one question that needed two strings, which is the doctor's contact methods: phone number and e-mail. The phone number must be a string because if it is an integer (or any numerical variable type), the leading zeroes in the sequence will be discarded since they would be treated as numbers. Besides, phone numbers do not provide any mathematical purpose. Visually, there are two input field, each prompting for a contact method. For the phone number, only numerical values can be added. So, in a smartphone, when the input field for the phone number is selected, the on-screen number-pad will appear, rather than the whole on-screen keyboard.

Date

There are three questions that ask for the date: 1. The birthdate of the user; 2. The expiry date of the medication; and 3. The next appointment with the user's doctor. There are three input fields: day, month, and year, and each one is an integer. The former two only accepts two digits, while the latter accepts four.

The Action Plan's PDF Form

All the answers of the Action Plan can be presented in a PDF. This was done using the iTextSharp library [5], which is compatible with C# and Unity. There is a teal button called "Print PDF" in the action plan's UI, which is responsible for creating the PDF file. When the file is created, the very first attributes that are printed are 1. the date and time of printing; and 2. the time zone of the device's current location. Below them, a table is printed showing the personal information of the user, which are 1. Name, 2. Date of birth, 3. Age, and 4. Gender. The age is automatically calculated by the difference between the date of printing and the date of birth. The others are taken from the user's input from the app. After this, a list of all the questions and their answers (Other than the personal information) are printed across the document. As for the date and time formats:

- All the dates in the PDF are in the following format: "MMMM dd, yyyy (dddd)", for example "September 02, 2024 (Monday)"
- The time format used is in 12-hour format, for example "7:05 pm". Also, it measures the time zone by using "UTC" time zone as a base. For example, "UTC+4:00" in the time zone of the United Arab Emirates, or "UTC+1:59" in Croatia.[2]

2.6 The Exhibition

The exhibition is a unique experience that users can try. It is similar to seeing exhibits in a museum, and if there is a specific object or display that they want to know more about, they press the corresponding number that is displayed next to the object on their small digital companion. The exhibits that we are showing in the exhibition are the types and components of inhalers, as well as a whole assembled inhaler.

Preparing the Exhibition

In order to make the exhibition visually compelling in a 3D space (Even in AR), some math skills are required. The exhibits are displayed by being spread across in a perfect circle. To make the circle, we need to get the direction of each exhibit and choose a magnitude.

[2]Manar has tested printing the action plan in Croatia in late July of 2024. The time zone was accurate.

Programmatically, the whole process is done in a "`for`" loop. The loop has an integer variable, called "`_i`", which is the index of each iteration in the loop. Another integer that is involved in the loop is called "`_n`", which is the total number of exhibits. Initially, "`_i`" is equal to zero, and with each next iteration, it increases by one until it equals to "`_n`", exclusively. The following two subsections will describe how the exhibition is created, specifically what is happening in each iteration.

Step 1: Getting the Direction The first thing that we need to do is to instantiate (create a copy and put it in the scene) the current object. The rest of the work is done on that instantiated object, which will be the exhibit. Secondly, we need to get the angle for the current exhibit, which is done using the following equation: "$\theta_i = \left(\frac{i * 360}{n}\right)$".[3] In the equation, "$\theta_i$" is the angle of the current index (The angle is in degrees). "i" and "n" correspond to their respective values that were described in the heading section. Once we get the angle, we need to calculate its sine and cosine values to establish the direction in the z-axis and the x-axis, respectively, and store the values in a `Vector3` variable:

- $dir_{i,x} = cos\left(\theta_i * \frac{\pi}{180}\right)$
- $dir_{i,z} = sin\left(\theta_i * \frac{\pi}{180}\right)$

In Unity, the standard unit for angles is radians. Thus, we need to convert the angle from degrees to radians before getting the sine and cosine values, by multiplying it with the fraction $\left(\frac{\pi}{180}\right)$.[4] "$dir_{i,y}$" is equal to zero, since we do not want to spread the exhibits across the y-axis, regardless of the value of "i".

Step 2: Assigning the Position and Adding the Script After we get the direction, we multiply the values of "dir_i" with the radius to determine the position of the current exhibit (in meters, which is Unity's default unit for length or distance measurement). Therefore, the final function is: "$pos_i = dir_i * r$", where "r" is a positive float, and it represents the magnitude (or the radius of the circle). It will tell how far the exhibits should be from the center. We set the radius to be 500. While exhibits' positions are different from each other, their distances from the center are the same.

Lastly, before the iteration is complete, we add a C# component to the exhibit, called "`ExhibitionObjectScript`". This script will be further described in the section "**Showcasing the Exhibition**".

Showcasing the Exhibition

In Unity, there is a technique called "Raycasting", that is used to make the exhibition work. A raycast is a straight, invisible line that starts from one point and continues

[3] "θ" is a Greek symbol, pronounced "Theta". It is a variable used in geometry to represent angles.
[4] "π" is a Greek symbol, pronounced "Pi". It is a constant value, which equals to 3.141592

infinitely (by default) until it detects an object in the way.[5] It is, conceptually, similar to a laser beam. At the exhibition, the raycasting line/beam starts from the center of the user's camera. There is a special UI for the exhibition, which has the following components:

- A red circle, which is in the center of the screen, which is the same position as the starting point of the raycasting line
- A back button
- A microphone button which appears only when an exhibit is hit by the line and disappears otherwise
- A text field to display the name of an exhibit

All exhibits have one script in common, called "`ExhibitionObjectScript`". When the user places the center of his/her camera at an object, the raycasting line will detect if it contains the mentioned script. If so, the object will be highlighted in a specific color, the red circle will turn green, the name of the object will be displayed in the text field, and the microphone button will appear below. When the microphone button is pressed, Dr. Salem will discuss the highlighted object.

The Draggable Class

There are objects that need to be dragged by the user. To make this work, two classes were created: "`DraggableClass`" and "`DraggableManagerClass`". The former is a class that can be a component for any object in a scene. Basically, when the user taps on an object that has this script, the object will follow the finger's position and movements. When the user lets go of the screen, the object will stop following. The class has four possible states:

1. "Canceled" (The object is not being dragged)
2. "Starting" (The object has started being dragged)
3. "Following" (The object is being dragged)
4. "Ending" (The object has stopped being dragged)

The class contains a Boolean variable, called "`_draggableOn`", to turn on or off the ability to have the object be draggable, and its default value is '`true`'. "`DraggableManagerClass`" is a singleton class used to keep track of all the objects that contain "`DraggableClass`". It also allows other scripts to know which object is being currently dragged by the user.

The Shakable Inhaler

There is a simple feature to allow children to shake a virtual inhaler in the application. They can shake the object by flicking their finger up and down to simulate shaking the

[5] In Unity, you can set a limit for the raycast detection by a number, making it finite.

inhaler in real time. The inhaler object contains the "`DraggableClass`" component. We can also measure how many shakes were done and how fast each shake was.

2.7 Programming the Games

There are five mini-games established for the application:

1. Matching the letters
2. Card matching
3. Multiple Choice Questions
4. Matching the inhalers
5. Assembling the inhaler components

From a programming's perspective, there is a base-class for all the games that has variables which are applicable to all or most of the games, such as the progress percentage, the assigned badge for the game, or the spawning location of the game's components. However, each game also has its own sub-class, which derives from the base-class mentioned. These sub-classes have variables that are only needed for its specific game. Also in the games, we always provide positive feedback since it is an educational application that is targeted to an audience between the ages of 6 and 13 years, even if they did something incorrectly. Positive feedback gives these demographics encouragement to keep trying to get the answers correctly and learn more about a specific subject.

One special property that all the games have is a meter. It tells how the user progressed in the game so far. The meter has two factors: a counter and a percentage. The counter tells how many items are completed correctly out of the total of items. In the UI, the counter is displayed as "Completed / Total" fraction. The percentage is the mentioned fraction's ratio multiplied by 100. When something is done correctly in the game, the counter increments by one, and the percentage is updated with it automatically.

Block Matching Games

The matching games are games where the user is given a certain number of blocks and holes, and the user needs to drag the blocks into their correct holes. There are three of them: One for the letters, the other for the inhaler types and components, and the third is for assembling the inhaler. All the blocks have physicsbodies and the "DraggableScript" components. For the letters, there are six blocks and holes, and each block and hole are in a specific letter. The letters are: 'A', 'S', 'T', 'H', 'M', and 'A', which spells "ASTHMA". When a letter block is placed in the correct spot, the following will happen:

1. the hole's renderer will be invisible
2. the game's counter increments by one

3. Dr. Salem will talk about a word that starts with that letter
4. a particle effect of stars will appear once
5. the blocks will no longer have a physicsbody
6. the "`_draggableOn`" variable in "`DraggableClass`" will be '`false`'.

Also, the letter block will rotate indefinitely at 100 degrees per second in the y-axis. The user can drag any letter in no specific order. When the game is complete, Dr. Salem will say what asthma is, then reward the user with a badge. The badge is saved is saved in the ".json" file.

The other matching game, which involves the inhalers and their components instead, works the same way. However, there was a visual problem: There are three types of inhalers, which are the reliever, the preventer, and the hybrid. While they are in different color, they have the same physical shape and size. Meaning, while the blocks are different, the holes are identical in color and opacity. So, to solve this matter, there is a canvas above each hole, which displays the name of its corresponding item (whether it is a type of inhaler, a component of an inhaler, or the whole assembled object).

The assembly game is about putting the components of the inhaler together to make the whole object. Programmatically, it works that same way as the previous two game, where each block should be dragged to the correct hole. However, the difference is that this is a procedural game, meaning that the blocks must be dragged to their holes in a specific order. Each step has a textual description of what the user should do. With each step done correctly, the meter's counter increases by one, and the next step will be described. The steps of the procedure are:

1. Place the neck block into its hole
2. Place the medicine block into its hole
3. Remove the cap from the neck
4. Place the spacer block into its hole

Once the game is done, the player will be rewarded with a badge.

Card Matching Game

The card matching game is showing two cards and see if they are identical. The game itself is very well known. In this game's case, though, it is specifically for the triggers of asthma, such as pollens, cigarettes, and pollution. In each session of the game there are 12 cards, aligned in a 4×3 structure. However, there is something additional in this game: Two of the cards are information cards, which only provide random information about asthma. Their front side is a picture of a question mark. Thus, we have ten trigger cards (each two are identical), and two information cards. If the player reveals an information card, the game's meter increments by one, and

Dr. Salem will talk about an information about asthma. Otherwise, if the player reveals two identical cards about a specific trigger, the meter increments by two, and Dr. Salem will define the trigger. When the player completes the game, he/she is rewarded with a badge.

Multiple Choice Question Game

There is also a fourth game, which is multiple choice questions. We have made up to 24 questions, and each question has four choices, where one of them is correct. In a game session, only five of them are randomly selected. When a question is selected, their choices are randomly shuffled before they are displayed. Unlike the other game where they take place in the AR world, this is the only game that happens on-screen. When the player answers incorrectly, Dr. Salem provides positive feedback to encourage users to continue. When correctly:

- Dr. Salem praises the player and explains the answer
- The answer buttons are disabled for interaction
- A green "next" will appear below

[More Text Here]

The UI Canvas of the MCQ game The game has its own UI canvas, since it needs to be on-screen. It has:

- A slider to show progression
- A text to display the current question
- Four buttons, where each button shows one choice
- A text below the buttons for responses

Designing the canvas was challenging for a device as small as a smartphone and in portrait mode, specifically for the buttons. Aligning the buttons on top of each other could prevent space for other contents, such as the questions' text bar, the slider, the response text bar, and Dr. Salem's image. The most ideal structure of the group of buttons in a small display was to have two buttons above and two below to accommodate enough text for the choices, as well as the other UI elements.

3 Evaluation

[Text Here]

4 Conclusion and Future Work

[Text Here]

5 Definitions

- Cosine: The ratio of the adjacent side of a corner in a right triangle to the hypotenuse of the triangle: $cos(\theta) = \frac{adjacent}{hypotenuse}$.
- Sine: The ratio of the opposite side of a corner in a right triangle to the hypotenuse of the triangle: $sin(\theta) = \frac{opposite}{hypotenuse}$.
- Raycasting: using a straight line that starts from one point and continues infinitely (by default) until it detects an object in the way.
- Canvas: A UI element that contains other elements, including text, images, buttons, or other canvases.
  - On-Screen Canvas: A canvas that is always fixed in the display's screen.
  - World Space Canvas: A canvas that exist in a world with other objects.

5.1 Types of Programming Variables

- Enumerator: A variable type in programming that has a certain number of values and can only be assigned to one value.
- Integer: A whole number, such as: 1, 3, -8, 100, 2006.
- Float: A decimal number, such as: 1.0, 5.2, -10.33, 7604.099.
- String: A text or sequence of characters, such as: "Hello World, 12381 42!".
- `Vector2`: A collection/array of two float variables. The first one is named 'x', and the second one is named 'y'.
- `Vector3`: A collection/array of three float variables. In addition to the two variables in `Vector2`, the third one is named 'z'.
- Boolean: A variable type that only has two possible values: '`true`' or '`false`'.

6 Acronyms

- Enum: Enumerator

- AR: Augmented Reality
- UI: User Interface

References

[1] *What is a Programming Library? A Beginner's Guide*. Accessed on: October 4, 2024. Jan. 2023. URL: https://careerfoundry.com/en/blog/web-development/programming-library-guide/%5C#what-is-a-programming-library.

[2] *Encryption and Decryption*. Accessed on: October 4, 2024. 2010. URL: https://docs.oracle.com/cd/E19047-01/sunscreen151/806-5397/i996724/index.html%5C#:~:text=Encryption%5C%20is%5C%20the%5C%20process%5C%20by,its%5C%20original%5C%20(readable)%5C%20format..

[3] *Unity*. URL: https://unity.com/.

[4] *ARCore supported devices*. Accessed on: October 6, 2024. Jan. 2023. URL: https://developers.google.com/ar/devices.

[5] *iTextSharp*. URL: https://itextpdf.com/products/itextsharp.