Project3 Brief & Pseudo-code

Paul Plew

March 12, 2021

0.1 Brief

0.1.1 Introduction

For this project we are required to make a kind of Trivia game that uses inheretance, loading external files, and visual animations. I will code the trivia game to have question interface that will be used to represent a question. From this object there will be two subclasses: the first will be called Question and the second Answer. These classes will then be used to represent the questions and answers for the trivia game. In addition to the information such as the question String these will also hold a difficulty value that will be used to calculate the score of the player. I am going to create the score so that for each new question a timer will start counting down and the longer the user takes to answer the lower their score for that question will be. At the end of the game the program will save the user's score to another JSON file that can then be accessed in a high-scores menu. Throughout the game there will be a button to pause the game and in the pause menu the user can restart or resume the game (but beware the score will continue to count down!).

0.1.2 Risk Assessment

I have never used JSON or external file loading like this before so I am guessing it may be difficult to implement correctly. As a result of the questions being loaded externally I will have to be careful on how I display the text so it fits no matter the length of the question. Another difficult aspect of this assignment will accompany the dynamic elements of this game. In my last project I used some transformation to create animations, but this project will have many more animations and I want them to be as polished as possible.

0.2 Pseudocode

0.2.1 Global Variables

the `questions` array will be used to store objects that represent the questions in the trivia game. The `answers` array will be used similarly to the questions array except it will hold the answers to each question. The `currentQuestion` will be updated to reflect the current question that the user is on. `isAnswered` will be used to determine if the current fame should display a question or an answer.

```
let questions = [];
let answers = [];
let currentQuestion = 0;
let startState = true;
let playState = false;
let pauseState = false;
let scoreState = false
let endState = false;
```

0.2.2 preload()

this section will be used to instantiate the JSON file that holds the questions for the game.

```
LoadJSON("some file.json");
```

0.2.3 setup()

Setup will have a canvas of size: `CreateCanvas(windowWidth, windowHeight);`. It will make `noFill()` for the question boxes, as well as setting up the two arrays of questions and answers.

```
CreateCanvas(windowWidth, windowHeight);
noFill();
```

```
FOR
    intiialize the array of questions for the items in the json file
    initialize the array of answers for the intems in the json file
```

0.2.4 draw()

draw will determine the current state of the game and based on that will draw a different screen.

```
IF pause
    pauseScreen();
ELIF start
    drawStart();
ELIF play
    displayQuestions();
ELIF score
    displayScores();
ELIF end
    displayEnd();
```

0.2.5 drawStart()

this function will show a few rectangles that can be used to start the game or view the previous high scores.

```
rect(START)
rect(SCORES)
```

0.2.6 displayQuestion()

this will display a question box and the text inside it. Around the box will be some sort of animating line or a dynamic shape of some sort. When the mouse moves over the box I would like it to increase in size slightly.

```
Current QUESTION
rect();
if (MOUSE is inside Rect)
    increase the size
some animating element
```

0.2.7 displayScores()

this section will use the externally loaded JSON file and will parse previously recorded high scores from it. The top 10 scores will be displayed inside a leader board.

0.2.8 reset()

the function `reset()` will stop the game and go back to the start screen

```
currentQuestion = 1;
save the score to the JSON
startState = true;
playState = false;
scoreState = false;
endState = false;
```

0.2.9 mouseClicked()

determine the current state and call the appropriate mouseClicked function.

```
IF pause
    mouseClickedPause();
ELIF start
    mouseClickedStart();
ELIF play
    mouseClickedPlay();
```

```
ELIF scores
    mouseClickedScores();
```

0.2.10 mouseClickedPause()

If the mouse is clicked in the play box or reset box the game will reset or play depending on which box is clicked.

0.2.11 mouseClickedStart()

If the mouse is clicked in the start box or the high scores box the appropriate state will be activated.

0.2.12 mouseClickedPlay()

This one will be a little bit more involved because it will have to determine which answer was clicked in the multiple choice answer. The answer that is clicked will be the one that is compared with the correct answer. If the correct answer is clicked then the score will increase.

0.2.13 mouseCLickedScores()

If the mouse is clicked in the back button the screen will return to the start screen. There might be other functionality such as the ability to view breakdowns for the high scores but that is not a priority at the moment.

0.3 Classes

0.3.1 Answer

this class will house an answer as a String the answer will have string that is the text and a Boolean that determines if this is a correct answer. I am going to attempt to use cyclical data so that a Question object points to an answer and an answer object points to a Question.

0.3.2 Question

this class will represent a question in the game. It will have a list of answer objects inside it as well as a string that is the question itself.

0.3.3 Inheritance

because both the Question and Answer classes share some data they will both inherit a parent class that will have a method for comparison and a method for drawing the text and it's appropriate bounding box. The parent class will hold a string of text.