Report on the design of our “Guess the Answer” game

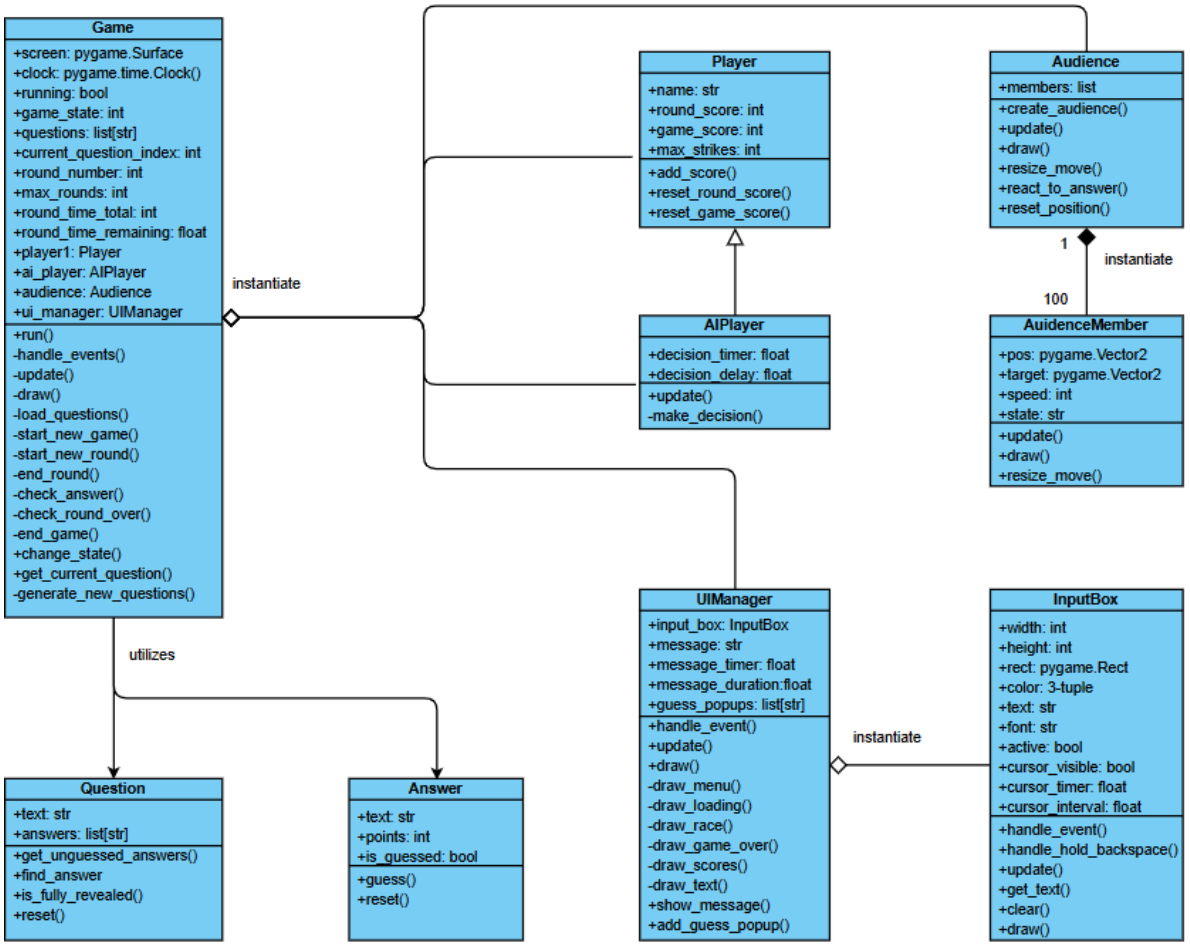
1. Introduction

Our design follows closely the baseline version of the game “guess their answer’ mentioned here:

- The program begins from main.py, where we first chat with ChatGPT-4o to obtain the 3 sets of questions, answers, and their respective points via an API call to openAI.
- After that, 3 rounds of games follow, with each of them showing a question and allowing the player to guess the 6 most popular answers by typing them, while an AI player also tries to guess an answer (P(correct) = 0.4) at random time intervals t (t = rand(3, 7) seconds). Both players get points by how popular the answer they guessed is.
- The round ends when one minute passes or all 6 answers are guessed by either player.
- The game ends when 3 rounds are passed, and scores of the player and AI player are kept tracked throughout the game and shown when the game ends.
- A new game can then be played following the game played and this process continues until the player quits.

2. OOP design of our program

The OOP design of our program can be described by the following UML diagram:



2.1 *Game* Class

At the beginning of the game, a *Game* object is instantiated from the *Game* class, which is responsible for creating the main game cycle, where the cycle checks the player's behaviour (including guessing answers by the player) continuously and updates the game status and screen correspondingly. In instantiation of this object, it utilizes other developed classes by instantiating these classes, namely, the *Player* (2.2), the *AIPlayer* (2.3), the *UIManager* (2.4), and the *Audience* (2.6) classes, which will be described below.

2.2 *Player* Class

This class is instantiated by the *Game* Class, and it mainly stores the player's scores throughout the 3 rounds of games to determine the winner at the end of the game against the AI player. Some setters are implemented in this class to update the player's scores.

2.3 *AIPlayer* Class

This class is instantiated by the *Game* Class and is inherited from the *Player* Class, which in addition to storing the AI player's scores and allowing the *Game* object to modify scores, it also stores a timer to guess an answer when the timer reaches the delay set. The delay is set to a random time between 3 to 7 seconds after each guess. Duck Typing is used here, since in handle_events() of the Game object, it adds the score of whoever player guesses a correct answer, regardless of the class (Player or AIPlayer) involved.

2.4 *UIManager* Class

This class is instantiated by the *Game* Class, and it implements various methods useful for creating the user interface of the game, mainly, drawing the game interface at different stages of the game. This class instantiates the *InputBox* (2.5) class (instead of creating the input box inside this class) as the input box of the game requires more handling than other drawing elements of the game.

2.5 *InputBox* Class

This class is instantiated by the *UIManager* Class, and it organizes all the attributes needed to create the input box of the game, as well as providing various methods for the input box that mainly handles the player's input and updates the visuals for the input box.

2.6 *Audience* Class

This class is instantiated by the *Game* Class, and it is used to create the visuals of the audience which moves to sides of the stage when either player guesses a correct answer. This class does not actually draw any audience, but instantiates 100 (sum of points for each question) *AudienceMember* (2.7) objects where each of them represent one drawn audience on screen. Methods provided by this class are used to centralize the movements of the 100 *AudienceMember* objects.

2.7 *AudienceMember* Class

This class is instantiated by the *Audience* Class, and it is used to draw 1 audience member on screen, as well as moving them to their desired places when a player guesses a correct answer, the screen is resized, or the game ends.

2.8 *Question* Class

This class stores the questions of the game in a centralized manner, and provides several methods (including getters) for the *Game* object to retrieve information about the questions.

2.9 *Answer* Class

This class stores the answers of the game, their points, and whether the answers are guessed in a centralized manner, and provides setters for the *Game* object to access and change whether the answers are guessed.

There are other OOP designs in our program that are not mentioned above:

2.10 Encapsulation

All the methods marked by '-' in front of the method names in the UML diagram are private, where although not enforced by Python, we avoid calling such methods outside of the class where the method is defined.

2.11 Using *Constant* and *GameState* Class

These two classes store some commonly used constants (including color's RGB tuples, screen height, width and frame per second (FPS)) and integers representing game state (such as at the menu, loading, in the game, the game ends) respectively. We rename such constants and integers to English words, which should increase the code's readability, and these constants and integers can be changed easily if we want. These two classes are used by many of the abovementioned classes.

3. Extra notes on this project

1. This project uses an external library "gtts" (Google text-to-speech), which is not covered in lectures (but covered in tutorial).
2. When guessing the answer, the player does not need to type the exact wording of the answer to be considered correct. As long as the guess is somewhat similar to an answer, it is considered a correct guess.

4. Resources used to create this project

1. openAI → chatGPT:
    - Used to generate questions for each game
2. Q1.mp3, Q2.mp3. Q3.mp3:
    - Played when a new round of game begins. Generated from Google Text-to-speech (gtts) in every run
3. background.mp3:
    - Used as background music for each round of the game
    - Sound source: https://www.youtube.com/watch?v=_6WlzEZ95BU
    - License: https://creativecommons.org/licenses/by/3.0/legalcode
4. cymbal.mp3:
    - Used as music when a round of game ends
    - Sound source: https://www.youtube.com/watch?v=RQmaTF161f8
    - License: https://creativecommons.org/licenses/by/3.0/legalcode
5. correct.mp3:
    - Played when any player guesses a correct popular answer

    - Sound source: https://www.youtube.com/watch?v=ymtpK5Eg8pQ
    - License: https://creativecommons.org/licenses/by/3.0/legalcode
6. incorrect.mp3:
    - Played when the player guesses an incorrect answer
    - Sound source: https://www.youtube.com/watch?v=RPidJ39lcLE
    - License: https://creativecommons.org/licenses/by/3.0/legalcode
7. background.png:
    - Background image of the game
    - Source: https://pixabay.com/illustrations/texture-background-graphic-arts-2072344/
    - License: https://pixabay.com/service/license-summary/
    - (the image was tuned darker)
8. audience1.png, audience2.png:
    - Image of the audience members
    - Image source 1: https://openmoji.org/library/emoji-1F9D1-200D-1F9B2/
    - Image source 2: https://openmoji.org/library/emoji-1F471-200D-2640-FE0F/
    - License: https://creativecommons.org/licenses/by-sa/4.0/
    - (as for audience1_green.png, audience1_red.png, audience2_green.png, and audience2_red.png, they are edited from the above 2 images by ourselves)