Assessment 02 - A Game Implemented with tKinter

Introduction

The aim of this coursework is to assess you on the material delivered so far. You will create a game using the `tkinter` Graphical User Interface library. The game should start off simple (but not too simple that the player gets bored) and increase in complexity as the player progresses.

In this assessment we are looking for your creativity as well as problem solving and coding skills. You can choose any game you wish to create, but your game should meet the criteria outlined below. Some students will opt to create a classic retro game such as space invaders. Other students may take a similar approach and create a classic retro game with a twist (added functionality or additional features for example). The more experienced students may decide to create an original game never seen before!

Important: The game must run using Python 3.8, so if you create it on your personal computer you must ensure it will work when being marked. The markers of the assessment will not install non-standard libraries or attempt to fix any issues with running the game. If the TA can't run your code successfully first time, you will be awarded a mark of 10, for attempt at creating a game.

Overview

This assessment has been designed to be ambiguous to allow for creativity. We are not just adhering to a tick box exercise like before. The marking team will play your game and will award you marks for their personal enjoyment of playing the game (they will rank your game from 0-3). This ranking is subjective for the markers and won't be challenged.

In addition to this the marking team also have a strict mandate to follow, i.e. they will be looking for the use of graphics, shapes, animations, complexity between stages or levels, how scoring is achieved, how the game provides incentives to the player to keep them engaged, how polished your graphics and/or shapes are, how you incorporated timing, etc.

You are also to use your given GitLab for all version control and must apply appropriate levels of commits to the project. The final commit you make for this assessment must be tagged as "Coursework_02" as we will search for the tag in your repositories and assess you work there. You should use the repository that has been created for you (i.e. *programming-practicals_[username]*, within this you should then create a new directory called *coursework_02*. This is where we will check your git history for the coursework when marking.

Objectives

The game should attempt to include the following features:

1. The use of images.
2. The use of shapes.
3. The use of text.
4. A scoring mechanism.
5. A leader board which is presented at appropriate places in the game (i.e. from a menu before the game begins and/or at the end of the game)
   a. The leader board must retain player names (or initials if you adopt a retro style), with their score and their position in the leader board.
   b. When the game is quit and reloaded, the leader board will reflect the history of the leaders of the game.
6. To avoid having to scale images for different screen resolutions you can use any of the following resolutions. 1920x1080, 1366x768, 1536x864, 1440x900, 1280x720 or 1600x900. Indicate the screen resolution at the top of the python source code as a comment just in case the marker needs to alter their screen resolution to best view your game.
7. The movement of objects.
8. The ability for the user to move an object.
9. Some form of collision detection.
10. The ability for the player to pause and unpause the game (so that they can make a cup of tea).
11. The ability to customise the experience for the user, for example, if the game has an object that can be moved the player should be able to define the keys that control that object.
12. Special cheat codes (i.e. if your game was a snake game, you might incorporate 'a shrink cheat' that decreases the length of the snake's body each time the code was entered).
    https://www.digitaltrends.com/gaming/famous-cheat-codes-in-video-games/
13. Save/Load game feature so that the player can resume playing where they left off tomorrow.
14. A 'boss key', we shouldn't play games at work. Some games introduced a boss key which allowed the game to flip to an image that gave the impression that the player was doing something work related quickly.
    https://en.wikipedia.org/wiki/Boss_key

Students with Little Programming Experience

We are very conscious that some students are new to programming and might find these instructions or this task very daunting. Therefore, we also accept games that you may have created in your laboratory exercises or introduced in the lecture, for example the snake or pong game. However, you must provide some customisation and enhancement. Submitting a solution to a step-by-step lab exercise does not demonstrate your creativity, problem solving or programming skills.

Can I use. . . ?

You can use any resources at your disposal, however, the game must be your game and not a game that is authored by somebody else. The code must be yours. Here some do's and don'ts.

Don't

- Use nonstandard libraries such as Pygame that requires a pip install.
- Take online code and just tweak it (we will spot this, we know what is out there).
- Use images that are copyrighted, or indeed anything that is copyrighted such as sprite sheets, audio files, graphics etc.

Do

- Look at what is already out there for ideas and inspiration.
- Make use of royalty free images, sprite sheets, audio files, graphics etc (please reference the original source in your code).
- Make use of more advanced concepts not covered (i.e. Object Oriented Programming)
- PLEASE. . . make sure your game works on Python 3.8.

Preparation and Submission

A single .zip file must be submitted to Blackboard in the appropriate place,
16321-Cwk2-S-Advanced Python

The marker will run your program though the terminal and test its functionality. Your code will also be checked for quality and appropriate use of programming principles (e.g. comments, white space, etc.).

Deadline:
18:00 on Friday the 4th December

Assessment Type:
This activity is subject to summative assessments, therefore your submission will be marked and you will receive the associated feedback. The marks you obtain (see below) count for up to 20% of your overall mark for this course unit.

Marks:
Marks are awarded on three main categories:

1. The running & testing of the game
2. Code quality
3. Git usage

If the marker finds that the file submitted is unreadable and/or does not compile then you will receive marks for the code quality & git sections only. Please be aware that we will not install external packages to make your code compile. We should be able to run your code with a basic python setup.

Please note: Your work will not be re-marked because you disagree with the mark you were awarded. The markers all follow the same rubric which has been designed to be as fair and comprehensive as possible. Any queries with your marking can be raised with either Gareth or Stewart within a week of the marks being released, any queries made after this time will not be considered.

End of Assessment

Marking Guide

Below is an extended version of how the marks will be awarded.

The running and enjoyable nature of the game	5%
The use of images/shapes/text within the game	9%
Scoring and leader boards	13%
Movement and collision detection	24%
Pausing, cheat codes and boss keys	18%
Saving and Loading game play	8%
Imagination used and the game meets professional standards	13%
Git usage	10%