Lab #10 Grading Rubric and Instructions

This is the tenth lab for the 'Introduction to Computer Programming' course.

Please see the syllabus for information about when the work in this lab is due.

Lab Objective

The purpose of this lab is to give you the opportunity to further explore the concepts of object oriented programming with classes, inheritance, and UML diagrams. It will also give you the opportunity to work with a programming project that has a more visualized output - as opposed to just working with output in the terminal.

Instructions/ Deliverables

NOTE: These tasks can be completed in any order you like. See the **Grading Items** section below for the point distribution.

CITATION: Some of the exercises found here are completely original to the instructor.

- The abbreviation 'MH' will be used to indicate these exercises. This citation will be placed next to the exercise title.
  - ex: [MH]

Lecture Notes/ Supplemental Readings:

- 'Check off' your notes in your Engineering Notebook for the following material with the TA/ Instructor. These notes should already be done before the start of the lab period.
  - Runestone chapters 17, 18, and 19
    - **NOTE**: You do not need to complete any of the exercises at the end of the chapter. However, it would be helpful to you in the long term if you were to do so.
    - **NOTE**: You only need to take notes on the Runestone readings if you did *not* take lecture notes in class. The contents of these chapters are what was covered in the lectures.
  - `UML_Diagrams.pptx`
    - **Available**: `'Files' -> 'Supplimental_Slides_and_Files'` folder on Canvas

NOTE: For this lab, you will need to install the `pygame-ce` framework if you have not done so already:

- (PC) `pip install pygame-ce`
- (mac) `pip3 install pygame-ce`

Game UML Diagram: [MH]

- Look at the included `inheritanceDemo.py` script to get an idea of how a UML diagram works in relation to working code.
- Either in a group or on your own, do the 'Group Exercise' at the end of the `UML_Diagrams.pptx` slides in your Engineering Notebook.

- **NOTE**: [thinkcspy] depicts 'composition' with a white diamond. As per the articles cited/ listed in the `UML_Diagrams.pptx` slides, I believe 'composition' should use a black diamond. [MH]
- **NOTE**: This is just an exercise - you do *not* have to program an actual game like this, nor do you need to encompass every single aspect of the game you choose.
- Once you have completed your UML diagram for your game, be sure to show it to the TA/ Instructor for credit.

`chimpRefactor.py`: [MH]

- Pygame is one of the main libraries used for making 2d video games in Python.
- Start by exploring how to initialize Pygame by typing in the two code examples in the "Quick start" section on the "Pygame-ce Front Page": https://pyga.me/docs/
    - These should help you understand how the basic setup of a Pygame project works.
- After you are comfortable setting up a Pygame project, explore some of the examples: https://pyga.me/docs/ref/examples.html
    - **NOTE**: These examples will be installed to your computer when you install `pygame-ce`.
    - As per the documentation, you can run a specific example with the following commands in the terminal:

```
python -m pygame.examples.<example name> <example arguments>
```

For example:

```
python -m pygame.examples.aliens
```

- You can find the location of these examples (and the `data` folder that contains their image/ sound files) on your computer by typing the following script into VS Code, and running it from the terminal:

```
import pygame.examples.chimp
print(pygame.examples.chimp.__file__)
```

- Here, the `chimp` example is just that - an example. You can use any of the other examples (aliens, etc.) and the above script should work just as well.
- After playing with a few of the examples and examining some of their source code, take a look at a few of the tutorials: https://pyga.me/docs/index.html#tutorials
    - Specifically, be sure to read and type in the code for the "Introduction to Pygame" tutorial: https://pyga.me/docs/tutorials/en/intro-to-pygame.html
        - Be sure to pay special attention to the concepts of a `Surface` and a `Rect`. A single Surface/ Rect pair can be used to hold your image data and the location to render that image to the screen.
    - After doing the "Introduction to Pygame" tutorial, go through the "Chimp Tutorial, Line by Line" tutorial: https://pyga.me/docs/tutorials/en/chimp-explanation.html
        - The full source code is available here: https://pyga.me/docs/tutorials/chimp.py.html
- Your task will be to refactor the code for the `chimp.py` game in several ways:
    - First, make a copy of `chimp.py` and rename it to be `chimpRefactor.py`.
    - Before you begin changing `chimpRefactor.py`, make sure that the `chimp.png`, `fist.png`, `punch.wav`, and `whiff.wav` files are in a folder called `data`. You can get this folder by downloading and unzipping the included `data.zip` folder.
        - This `data` folder should be in the same directory as your `chimpRefactor.py` script.
    - In `chimpRefactor.py`, create a parent class called `Entity`, which will descend from the

pygame.sprite.Sprite class.

        - Integrate the code found in the original load_image() function into the constructor for your new Entity class. Meaning - you will no longer have a load_image() function.
        - This Entity class should be able to load an image of *any* name, colorkey, and scale.
    - The original Fist and Chimp classes will now descend from your Entity class, instead of the pygame.sprite.Sprite class.
        - You will need to update Fist and Chimp so that they take the appropriate values in their constructors. Meaning, the values currently in Fist and Chimp should no longer be 'hard-coded' as they are originally.
        - **HINT**: Be sure to call the super().__init__() method inside your Fist and Chimp constructors, and be sure to include the appropriate parameters for the Entity superclass.
        - **HINT**: Be sure to pass in the appropriate arguments when you create the Fist and Chimp used in the game.
    - The load_sound() function should be modified to use a try/ except structure to accommodate for missing sound files.
        - For example, if punch.wav is missing, the program should not play any sound instead of crashing.
        - You will need to figure out exactly which exception is triggered when one of the sound files is missing.
            - **HINT**: You can do this by temporarily renaming one of the files.
        - For the except clause of the try/ except statement, you can just print out an error message, and then assign the sound variable to be a NoneSound object which you will return in lieu of the usual pygame.mixer.Sound object.
    - Move the following code from the top of the chimp.py file in to the main() function. Place this code below the # Initialize Everything code and above the # Create The Background code:

```
if not pygame.font:
        print("Warning, fonts disabled")
if not pygame.mixer:
        print("Warning, sound disabled")

main_dir = os.path.split(os.path.abspath(__file__))[0]
data_dir = os.path.join(main_dir, "data")
```

- Basically, all of the code for the entire game, except for the if __name__ == "__main__": line at the bottom, should be encapsulated inside functions or Class methods.
    - You will have to update your Entity, Fist, and Chimp classes to take in the data_dir variable as an argument when creating these objects, as data_dir will no longer be available at the global scope.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called chimpRefactor.py.

Attendance:

- If you have completed all of your tasks for the lab, you may work on any of the 'Additional Resources for Study' found in the Canvas announcement of the same name.
    - **NOTE**: If you leave early, you will not receive the 'attendance points' for the lab.

Optional Readings

NOTE: These readings are not required. However, they may provide a bit of interest/ insight into the broader world of Computer Science. Please complete the rest of your lab tasks before doing these readings. You do not need to take notes on these in your Engineering Notebook.

18 Ways to Secure Your Devices From Hackers - by: Max Freedman

- Available: https://www.businessnewsdaily.com/11213-secure-computer-from-hackers.html

What Is a Work Portfolio? (Plus How To Build One) - by: Indeed Editorial Team

- Available: https://www.indeed.com/career-advice/resumes-cover-letters/build-your-work-portfolio

When to Apply for Summer Internships: A Timeline - by: Zoe Kaplan - edited by: Emily Courtney

- Available: https://www.theforage.com/blog/basics/when-to-apply-for-summer-internships

The Means of Gaming Production - by: Masha Borak

- Available: https://slate.com/technology/2023/07/video-game-industry-labor-cooperatives.html

Files Provided

`inheritanceDemo.py`

`chimp.py`

`data.zip`, which contains:

- `chimp.png`
- `fist.png`
- `punch.wav`
- `whiff.wav`

Example Script

None

Example Output

None

Grading Items

- (**Lecture Notes/ Supplemental Readings**) Has the student taken notes on the listed material, and shown their notes in their Engineering Notebook to the TA/ Instructor?: ______ / 20

- (**Game UML Diagram**) Has the student completed the task above, and shown their drawing in their Engineering Notebook to the TA/ Instructor?: ______ / 30
- (**chimpRefactor.py**) Has the student completed the task above, and saved their work to a file called `chimpRefactor.py`?: ______ / 40
- (**Attendance**) Did the student attend the full lab meeting in person?: ______ / 10

TOTAL ______ / 100