CPSC 4160/6160 Fall 2022

PROJECT 1

Due: In-class demonstration on September 27, 2022; code delivered to handin by end of day, September 27, 2022

DESCRIPTION:

This project is to build a very basic and limited game engine, and implement a simple game called “Hangman”. The purpose of the game is to verify and demonstrate the performance and execution of the game engine. This assignment is worth 15 points, divided up broadly into three categories:

1. Does it work and are the specific requirements of the assignment implemented? (5 points)
2. Was the demonstration informative about the strengths and weaknesses of your project? (5 points)
3. Is the code clean, organized, and demonstrative of pride of craftsmanship, and easy to execute? (5 points)

On September 27, you will submit this project for grading in two ways:

1. In-class demonstration: You will have 1-2 minutes with either the instructor or TA to demonstrate, live, your game playing. You should show all of the features of gameplay asked for in the description below. This is your chance to present your project in the best light possible.
2. By the end of the day, September 27, you must submit the code for your project to the SoC handin system. The specifics for doing this are posted on the website for the class. Pay careful attention to this task: many students who were very competent have messed up this task and suffered point loss. We will expect to be able to run the game on our own machines, as well as inspect the code.

GAME ENGINE

As we have discussed in class, our python-based game engine must have the following organization of directories (`gamey` is the name of my game engine code - feel free to give yours any name that you like, while retaining this directory organization inside):

```
gamey
├── assets
│   ├── sounds
│   └── textures
├── engine
│   ├── actor
│   │   ├── action
│   │   └── entity
│   ├── asset
│   │   ├── action
│   │   └── entity
│   ├── crowd
│   │   ├── action
│   │   └── entity
│   ├── enviro
│   │   ├── action
│   │   └── entity
│   ├── fx
│   │   ├── action
│   │   └── entity
│   ├── physics
│   │   ├── action
│   │   └── entity
│   ├── play
│   │   ├── action
│   │   └── entity
│   ├── render
│   │   ├── action
│   │   └── entity
│   ├── sound
│   │   ├── action
│   │   └── entity
│   ├── story
│   │   ├── action
│   │   └── entity
│   ├── ui
│   │   ├── action
│   │   └── entity
│   └── utility
│       ├── action
│       └── entity
├── python
└── templates
```

Do not forget that inside each directory, there needs to be a valid python file called "__init__.py". An empty file is valid, but you can also use it to create utility or factor functions, for example.

For this assignment, we are only interested in two engines: the play and actor engines. The other engines shown (utility, ui, story, sound, render, physics, fx, enviro, crowd, and asset) may be in future projects. As a result, you may implement entities and actions for this assignment that would be implemented differently in the future when the game engine is more complete.

The individual engines have been divided into two kinds of things: entities, that describe game content, and actions, that execute the evolution of game content. This project focuses on just two engines: play and actor.

The Play Engine

The play engine should have two entities: A frame viewer and a game loop. We have discussed both of these extensively in class. You have freedom to name these entities as you wish, and to name and implement their data and methods.

The play engine should have one or more action(s) with type "event" that terminates play, quits pygame, and quits python, in response to two possible events: (1) the quit button on the window frame is pushed, and (2) the escape key is pressed.

There must also be an action with type "display" for updating the display, by clearing the screen's frame buffer to black, running all of the actions that are of type "display", and then flipping the frame buffer.

No other actions for the play engine are required for this project. In future projects, additional actions may be needed.

The Actor Engine

To support the hangman game, the actor engine much be more substantial that the play engine.

Entities:
1. A rectangular entity with dimensions, location, and color that are input to the entity.
2. A circle entity with radius, location, and color that are input to the entity.
3. A letter entity, with font size, location, color, and the specific letter, that are input to the entity.

Actions:
1. An action with type "display" that draws a rectangle entity using the dimensions, location, and color provided by the entity.
2. An action with type "display" that draws a circular entity using the radius, location, and color provided by the entity.
3. An action with type "display" that displays the letter entity using the font, size, and color provided by the entity.
4. An action with type "event" that detects a letter of the alphabet that has been selected on the keyboard, decides whether the letter is a correct guess or not, lets other action(s) decide which parts of the hanged man to display, and other action(s) decide which letters of the word to display and which boxes to display.

If you find that you want to implement more actions and/or entities, feel free to do so. However, you should be careful that more are actually needed for the game that must be implemented. More can certainly be useful in the long run for the game engine, but pace yourself and make sure you get the assigned project done.

Final word of advice, paraphrasing Gandalf: "Keep it simple, keep it safe".

HANGMAN GAME

On startup of the game, the window should be sized 1280 x 720, and remain that size throughout the game.

At any time, pressing the escape key or the quit button on the window should end the game and python should terminate.

The hangman game picks an English word at random and the player tries to guess the letters of the word. Each incorrect guess of a letter advances the drawing of a hanged man. If the drawing of the hanged man finishes before the letters of the word have been correctly guessed, the player loses. If the player guesses all of the letters of the word before the hanged man is completely drawn, the player wins.

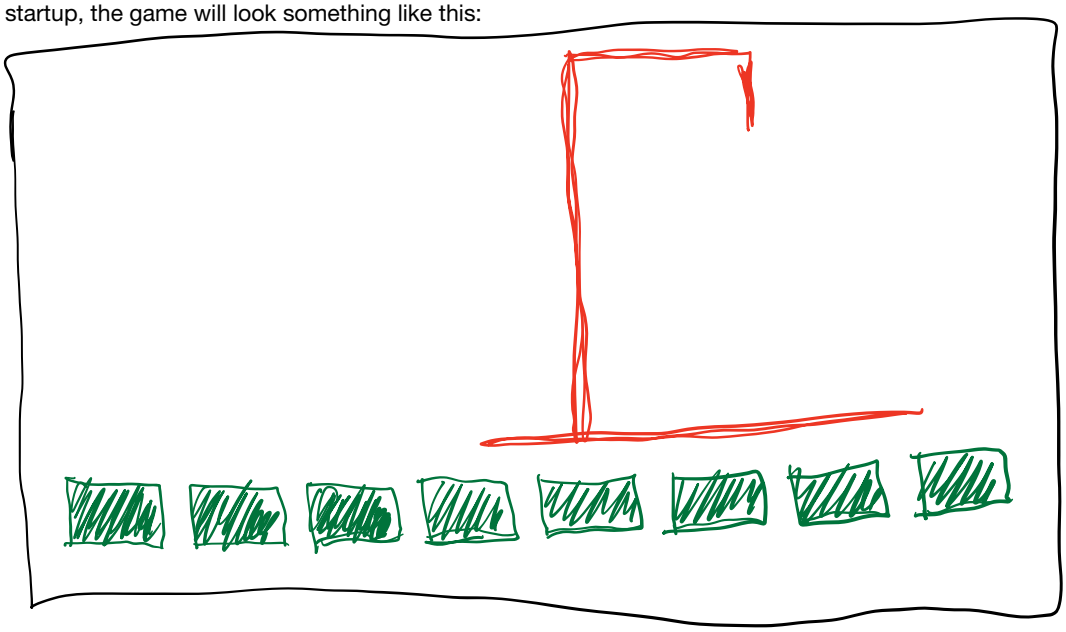
You must select words at random from a large collection of words. Specifically, the python package english-words provides a list of about 25,000 English words. You will need to install this python package. You can go to

https://pypi.org/project/english-words

for more information about the package and how to install it. You will use the set of words that the package designates `english_words_lower_alpha_set`. Your game will select a word at random, but only words that have at least 5 letters

and no more than 8 letters.
Note that the installation instructions use a command “pip” to do the installation. Depending on how you set up your python3 installation, you may need to use the command “pip3”.

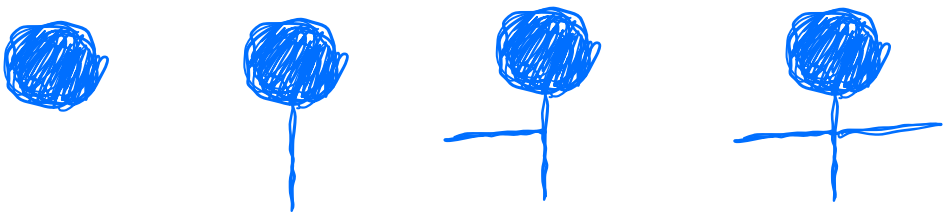
On startup, the game will look something like this:



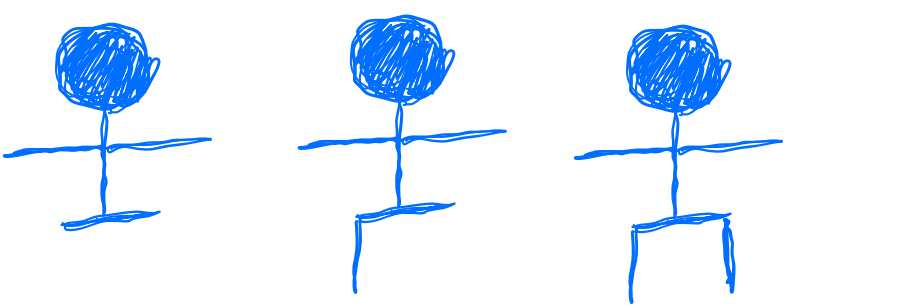
The red graphic is a collection of 4 rectangles representing the hanging stand. The green boxes are a collection of rectangles, one for each letter of the word that you randomly selected. Here it depicts 8 boxes for a word with 8 letters. If the randomly selected word has fewer letters (but no less than 5), a correspondingly fewer number of green boxes would be displayed.

Play begins with the user picking a key on the keyboard. If that letter occurs in the word, the green box(es) at the location of that letter disappear and the letter appears in its place. If that letter DOES NOT occur in the word, display of the hanged man is advanced to the next level, and the letter is added to a list of wrongly-guessed words in the upper left of the window.

Here is a progression of the updated drawing of the hanged man:

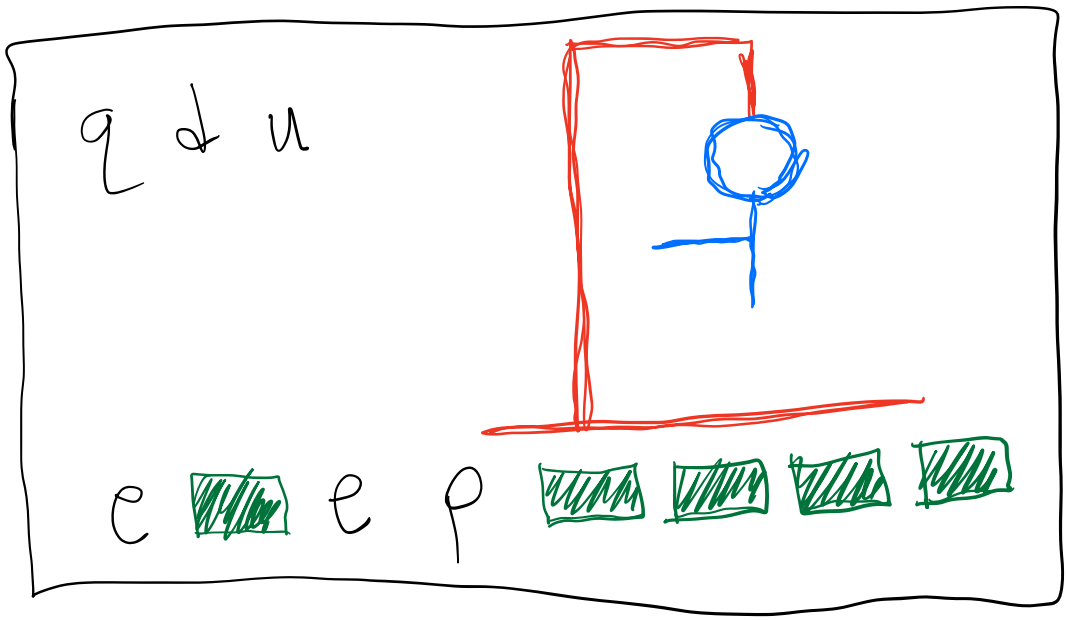


Start of game. First incorrect guess. Second incorrect guess. Third incorrect guess. Fourth incorrect guess



Fifth incorrect guess. Sixth incorrect guess. Seventh incorrect guess and end of game

The hanged man is a combination of a circle for the head and six rectangles for the torso, left and right arms, hips, and left and right legs. To give an idea of what the game should look like mid-play, here is a sample:



The word is “elephant”. The player has had three incorrect guesses “q”, “d”, “u”, and two correct guesses, “e” and “p”. The letter “e” occurs twice in the word, so one guess of the letter “e” should fill both spots. The incorrect guesses are displayed in the upper right area of the window.

The game code itself should be contained in the “python” directory, called “hangman.py”, as a python script that imports and uses the various elements of the game engine to initialized the game, entities, actions, and then runs the game loop.

STUDENTS IN 6160:

If you are in 6160, there are additional implementation elements. If you are in 4160, you may implement these also because you may find them useful. But 4160 grades will not be impacted by this additional detail.

There are many graphical elements in the display of the game. Three specific ones are: the hanging stand (red), the hanged man (blue), and the boxes for the word (green). In the project up to this point, these are assembled from many rectangular and circular actors. You must add an entity to group actor entities together. The hanging stand must be a group entity that contains the rectangle entities that make up the hanging stand. The hanged man must be a group entity that contains the rectangle and circle entities that make up the hanged man. The letter boxes must be a group entity that contains the individual rectangle entities. One or more actor actions will also need to be added that cascades actions down to the actions of the sub-entities. This means the individual sub-entities no longer respond to the keyboard event individually. Instead, the group entity action has type “event”, responds to the keyboard by making decisions of which sub-entities are to be drawn or not. You will find that there are many ways to do this, and many ways that look like they work, but actually dont work or dont work well. This part of the project is intended to give you experience with succeeding by failing first.