CS11 MP2: Mad Libs Programmer's Guide

PREPARED FOR

Sir Edgar Felizmenio

CS11 Professor at the University of the Philippines Diliman

PREPARED BY

Abdao, Regina Ciara

Angeles, Mikaela

Dela Cruz, Rome John

CS11 EF-M2-FRU AY 2018-2019

EXECUTIVE SUMMARY

The project deals with the development of a Python-based game with a Pyglet Graphical User Interface (GUI). The group has chosen to recreate a Mad Libs program, wherein a random story is given to the user, and he/she needs to fill out multiple entries to complete the story.

The program was divided into two modules, which includes the Main module and the GUI module. The Main module primarily deals with the initialization of the game itself, and contains functions which handles the base display. The GUI module contains both the functions which display the user interface and the functions which comprise the game engine itself.

For the Mad Libs program to function, a text file which functions as an external dictionary referred to as Templates is required for the program to work. This contains the list of all the stories which can be accessed randomly by the user when the game is played. In conjunction with this, another text file called Fill is a delimited version of the story, which is used to generated the final story.

Since the project also deals with the integration of the Pyglet GUI, included with the program are assets which include images that serve as the general menu display and background images for the stories.

1. Project Overview

aims, scope and intended operation]

The aim of the project is to develop a functional Mad Libs program which users outside the development team can enjoy and use with ease. When the game is played, the program first asks users for multiple inputs which is then stored into an array. Once all inputs are finished, the program outputs it into a pre-set template which will then be presented as a full story. Since the game is mostly for demonstration purposes, only a limited number of story templates was provided in the dictionary text. Also, the game is only limited to a single player functionality like similar Mad Libs programs, and does not have multiplayer support.

2. Obstacles

Majority of the obstacles that were encountered deal with the unfamiliarity with the use of the Pyglet GUI and problems that arose from object-oriented programming (OOP). Since the developers had some background in Python prior to the project, certain parts of the project were handled with ease. However, this was different during the development of the GUI because it needed to be learned first before it could be applied to the program. The Pyglet documentation was very helpful in this respect because it was very structured and was very easy to use. The problems related to OOP were similar in nature, and needed much debugging to solve in order for the final program to work properly.

3. Main Module

The Main module needs Pyglet and the GUI module as imports in order to work. Before any functions, the module initializes multiple variables which includes the starting window with dimensions 1280x720, the current screen, an empty holder variable for text, and an empty array for storage of inputs. A position variable is also initialized, which is needed to get the index of the random story and fill that will obtained from the two text dictionaries.

Four variables for the different window types were also initialized, which includes one for the Title, Input, Story, and End Windows. A random template is also acquired from the dictionary, of which variables are also initialized. This is where the position variable is used an index. The story is then delimited into a list and is also initialized into another variable template. Also includes main() to boot up the whole module.

Seven functions are included in the module, which includes the following:

- def change_screen(screen):

    - This updates the screen that is being displayed. A variable current_screen is assigned as a global variable. The input screen is then assigned to that variable.
- def makestory(template,inputs):
    - Each of the elements in the array inputs is inserted into the spaces within array template. This outputs into the initially empty variable story, and loops until the maximum length of the array template is reached. The story is then returned by the function.
- def update(dt):
    - Used to refresh the frames of the window.
- def main():
    - Uses function change_screen(screen) and initializes it as the Title Screen. Also sets the frequency at which the game refreshes, which clocks at 60 frames per second, and adds Pyglet GUI functionality.
- @window.event def on_key_press(symbol, modifiers):
    - The variable holder initialized as global to enable an empty space to type inputs. Contained in this are key commands one may use within the program.
        - pyglet.window.key.UP:
            - If within Title Screen, changes screen into Input Screen. If within End Screen, changes screen into Title Screen.
        - pyglet.window.key.BACKSPACE:
            - Replicates backspace functionality from text editors. Simply omits the last character input by reducing the holder variable width.
        - pyglet.window.key.ENTER:
            - If within Input Screen, resets holder variable and increments counter input_screen.counter by 1. Once input_screen.counter reaches the equivalent of maximum inputs, changes into Story Screen on click. If within Story Screen, uses makestory(template,inputs) function to return variable story. Also creates a story file into a text file, and changes into End Screen
- @window.event def on_text(text):
    - The variable holder initialized as global to enable an empty space to type inputs. Contained in this is the function which enables the user to type into variable similar to a text editor by refreshing the variable multiple times.
- @window.event def on_draw():
    - The variable current_screen declared global to be read by all functions within the program. Uses methods window.clear() to clear the window and current_screen.draw() to initialize the screen.

4. GUI Module

The GUI needs Pyglet and random as imports in order to work. For the purposes of this documentation, cosmetic choices such as font styles will not be discussed in detail. This module includes five classes, one for each window which includes:

- class BaseWindow:
    - Initializes the base window. Prerequisite for other windows to be displayed. Uses draw(self) function and uses a NotImplementedError if an instance arises that the initialization encounters an error identical to a RuntimeError.
- class TitleWindow(BaseWindow):
    - Within def __init__(self), sets variables self.label into “MAD LIBS!” and self.sublabel into “Press F to proceed.” by using pyglet.text.Label(). Also includes font style, font size, coordinates, and anchoring. Draws these into the window using draw(self) function.
- class InputWindow(BaseWindow):
    - Within def __init__(self), opens Fills Dictionary and assigns it to a variable self.fills method open('fill.txt', 'r').readlines(). Outputs string string of one fill using random.choice(self.fills) and assigns it to variable self.randomfill. Gets index of fill to match index of template using self.fills.index(self.randomfill) and assigns it to variable self.position. The list of word types (e.g. noun,adjective, etc.) for every empty space is then assigned to variable self.rfill using self.randomfill.split(','). The maximum number of fills is acquired through len(self.rfill) and is assigned to variable self.maximum. Variable self.counter is assigned to 0. Finally, sets variables self.label into ('Input {0}'.format(self.rfill[self.counter]) which prints the input counter and self.sublabel into an empty string by using pyglet.text.Label(). Also includes font style, font size, coordinates, and anchoring. Draws these into the window using draw(self) function.
- class StoryWindow(BaseWindow):
    - Within def __init__(self), sets variables self.label into “Press enter to finish your story!” by using pyglet.text.Label(). Also includes font style, font size, coordinates, and anchoring. Draws this into the window using draw(self) function.
- class EndWindow(BaseWindow):
    - Within def __init__(self), sets variables self.label into “YOUR STORY IS READY!” and self.sublabel into “You can see it in the folder. Do you wanna play again? Press up!” by using pyglet.text.Label(). Also includes font style, font size, coordinates, and anchoring. Draws these into the window using draw(self) function.

5. Text Files and Other Assets

The Templates and the Fills Text files contains data sufficient for 10 different stories. For the game to access them easily and to not mismatch them, the data for each Template text directly corresponds to the Fills text. Images that were used in the project were downloaded from the internet. Fonts on the other hand, were either built-in or downloaded.

References

- The official Python 3 documentation:
    - https://docs.python.org/3/
- The official Pyglet GUI documentation:
    - https://pyglet.readthedocs.io
- Lecture slides and sample code by Sir Edgar Felizmenio, CS11 Professor at the University of the Philippines Diliman
- Images were all found online, and can be downloaded from these links:
    - https://www.deviantart.com/singa2002/art/Aesthetic-clouds-682573151
    - https://www.flaticon.com/free-icon/down-arrow_626013#term=arrow&page=1&position=15

All files for this project can be accessed here:

https://drive.google.com/open?id=1Tqh0vsoW30IxcNyA1OFyk-I-3bYWe9Lh