PYTHON PROGRAMMING: MOOD AESTHETICS

Final Report

Submitted to

The Faculty of Operation Catapult 94

Rose-Hulman Institute of Technology

Terre Haute, Indiana

by

Group 44

Joanna Simwinga	Independence Senior High Independence, Kansas
Noah Verzani	The Loomis Chaffee School Maplewood , New Jersey
Alyssa Caganda	The British School Manila Manila, Philippines

July 24, 2013

Introduction to Python

Python is a programming language developed in the late 1980s by computer programmer, Guido van Rossum. Its simple syntax and emphasis on readable code led to its growth as one of the most popular languages used for both commercial and personal purposes. It is commonly used as a scripting language by large technological firms such as Google and Yahoo!, two of America's leading multinational corporations in Internet related products and services. These companies continue to use Python as an instrument to develop their web services and engines. However, Python can also function as a tool for game development, electronic design automation and formation of financial services.

Python possesses numerous features that make it easy to comprehend. It has a vast standard library, a stored collection of standard programs and routines that make it easier for programmers to execute functions; the user can access various functions without needing to download more modules. Python is considered a high level programming language. It often emphasizes usability over optimal program efficiency making it an ideal introductory language to the world of computer programming.

Background

Prior to Operation Catapult, all members of the project group had little to no background in computer programming. The equal footing and similar experience facilitated excellent teamwork among the members of the group. Further promoting harmony among the group members, a vision was shared of developing a program that focused more on showcasing the visual and graphic aspects of Python instead of creating a game as many of the other Python groups chose to do.

After initial conceptualization, the group opted to create a simple interactive audio visualizer. Despite choosing not to make a game, the group still utilized the Python module Pygame in making the program. Pygame is a set of Python libraries originally designed for writing videogames that contains numerous functions allowing for the development of animation, graphics and sounds.

Project Objective and Features

Mood Aesthetics, as the program is entitled, is a simple interactive audio visualizer centered around four human emotional states: anger, sadness, happiness and tranquility. It aims to enhance the user's current mood by pairing graphic animations and sonic melodies to the mood chosen by the user. The music and graphic animations were carefully chosen and designed to elicit a particular reaction from the user. The Angry function, for example, augments to the user's rage by making use of loud music, bright red colors, and edgy shapes flashing across the screen. This function also features an interactive component wherein the user clicks the mouse rapidly and the color of the screen consequently turns into a brighter shade of red.

Aside from the graphics and sounds, Mood Aesthetics was programmed to have a full *Menu* Screen, an *About* Page which displays a short description of the program, *Volume* Settings which can be changed by the User, and a *Pause* Function which allows the user to pause and resume the animations and sounds at any given time.

Design Process

To complete this program, the team followed a 3-step design process involving the making of Story Boards, User Stories, and CRC cards.

Story Boards

The first step involved initial conceptualization wherein the team formulated different ideas that would constitute the basic structure of the project. This process focused on the graphics and layout of the program instead of the actual code. As a result, the collaborators came up with a design for the menu screen and a draft of the basic features of the program. Finishing this process made it easier to visualize the product of the program and create a frame with which the team could work on.

User Story

The User Story is a written outline detailing the course of the program as viewed by the user. Making a user story aided the team in determining the different functions of the program with respect to the user, consequently providing organization and flow to the project. This process also permitted the team to analyze other needs of the user that may not have been addressed in the current plan, allowing more features to be added such as volume settings and an about page. By making a user story, the members of the team were able to create a more coherent flow between the different moods and the graphics that belong to each one.

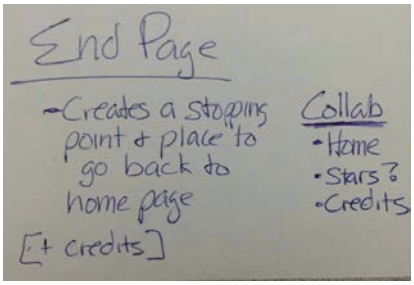
USER STORY

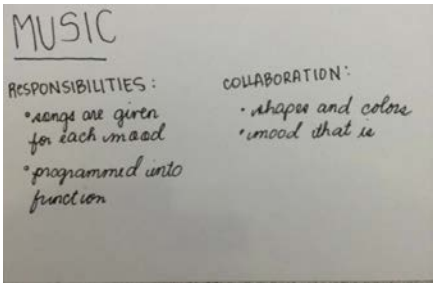
- *Screen Opens*
- *Scattered Star Background*
- *Menu Appears:*
    - *Stars move in a random pattern*
    - *Options list: volume*
    - *Moods: (Clickable)*
        - *Angry*
        - *Sad*
        - *Happy*
        - *Chill*
    - *Information/About (Clickable)*
- *User will choose mood (click)*
- *Screen will change:*
    - *Music related to mood chosen will play*
    - *Color scheme based on mood chosen*
    - *Shapes will move according to music*
        - *Shapes move with the mouse (cause ripple)*
- *Music will play on a loop/Shapes will change on a loop until time is finished*
- *After song end: Menu page appears: restart*

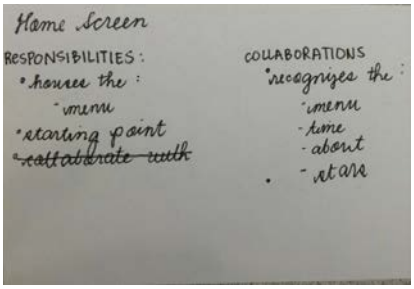
CRC cards

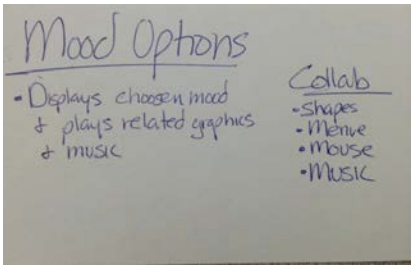
CRC cards or "Class, Responsibility and Collaboration" cards are brainstorming tools used to organize the different functions of the program. In each index card, the team wrote the name of the class, the responsibility of the class and the relation of this class with the other classes. This project had nine classes: The Home Screen, The Menu, The Stars (Background), Mood Options (Angry, Sad, Happy, and Chill), Music, Shape Patterns, Volume Settings, The About Page, and the End Page.

Examples of CRC Cards:









(Top Left-Clockwise: End Page CRC, Music CRC, Home Screen CRC, Mood Options CRC) The top left contains the name of the class. The left area defines its responsibilities and functions and the area on the right describes its relation with the other classes.

Difficulties Encountered

Having little background in programming was one of the biggest hurdles that the group had to overcome. Although there was an introductory course to Python programming in the first week of Catapult, learning to use the different functions and statements resulted in a lot of trial and error especially when it came to creating graphics. Counsel from Dr. Boutell, and his two programming assistants Katie and Brian, also helped in making the process of writing the code easier.

Another difficulty that was encountered was converting the program from Zellegraphics to Pygame. Zellegraphics is a separate graphic module that was taught at the beginning of the course. Because some animations were made using Zellegraphics, they had to be converted to Pygame in order to be executed correctly. Examples of the code necessary to convert text from Zellegraphics to Pygame can be found further in the report.

Another challenge was finding a way to make the animation move in sync with the song. Initially, the team wanted to incorporate MIDI files that the program could read and analyze. In the end however, the tempo of the song which is measured in beats per minute was converted into seconds and utilized as a delay by the clock module of Pygame, allowing the shapes to move and grow in sync with the music.

Discussion of Code

As discussed earlier, one problem tackled during the coding of Mood Aesthetics was converting certain graphical elements that used the Zellegraphics module into something capable of being run within a Pygame window. Pygame is packed with many more features than Zellegraphics, and its ability to play music and receive keyboard input was crucial to Mood Aesthetics. This made Pygame the clear choice. However, converting code from Zellegraphics to Pygame is not as simple as changing the syntax of the code; the two modules function in very different ways. Zellegraphics simply lets one create a geometric shape by first creating it and passing it variables, and then drawing it on to the window. For instance, creating a text box in Zellegraphics is as simple as

```
# Create a new text box at the point 600, 100 that says Click the Screen
instructions = zellegraphics.Text(zg.Point(600, 100), 'Click the screen')

# Draw the text onto the screen
instructions.draw(win)
```

One can then change certain variables about the text after it is drawn, without having to redraw it. For instance, to change the color and size of the text, one simply has to write

```
instructions.setSize(36)
instructions.setTextColor('Purple')
```

However, writing text is much more complicated in Pygame. One cannot simply draw text on to the screen; one must first create a font object out of the typeface and height.

```
# Create a font object in Helvetica 200 pixels tall
height = 200
font = pygame.font.Font('Helvetica.ttf', height)
```

One must then render the text in this font. Pygame cannot draw text onto the screen, it must create an image of the text, draw that onto a rectangle, and then “blit” that onto the graphics window.

```
# Create an image of the text ‘Hello World’, in our font, with antialiasing (the 1), in
# the rgb color (255, 255, 255) which is white.
text = font.render(‘Hello World’, 1, (255, 255, 255))
```

Next, one must create a rectangle out of this rendered text, and then set the center of that rectangle to a given point

```
textRect = text.get_rect()
        textRect.center = (200, 200)
```

Finally, one can “blit” the text onto the rectangle, showing it on the screen

```
screen.blit(text, textRect)
```

However, once the text is placed, it cannot simply be left on the screen. The text needs to be drawn continually by placing it inside a while loop

```
# For the duration of the background music, run the game loop
while pg.mixer.music.get_busy():

        # Graphics and Text code go here

        #Refresh the screen, at 30 frames per second
        pg.display.flip()
                pg.time.Clock().tick(30)
```

Not until after this arduous process can text be drawn on the screen. Similar difficulties were encountered with other shapes and figures when converting code from Zellegraphics.

Future Improvements

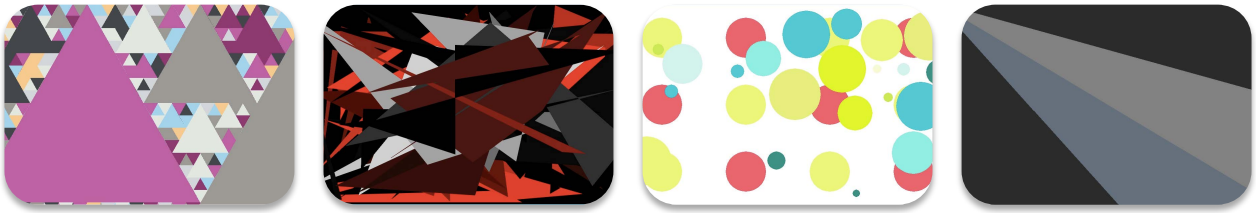
Future improvements, however, can still be made to enhance the efficacy of the project. One upgrade would be to increase the number of interactive elements. Currently, only the Angry Option and Happy Option have animations that include some sort of interaction or input from the user. Time constraint prevented the team from including more interactive options.

Another area to develop would be syncing the graphics with the music. Mood Aesthetics only plays one given song per mood option. This project can be enhanced to make the animations move in time with any given song by analyzing any musical file given.

Conclusion

Although the team had very little experience with programming prior to the making of this project, the result was a rousing success. Mood Aesthetics was completed with full graphics, settings and a menu page. More importantly, the team was able to learn the basics of programming and apply these lessons to the creation of Mood Aesthetics.

Figure 1.1: Animations of the Different Mood Visualizers



Chill Option | Angry Option | Happy Option | Sad Option