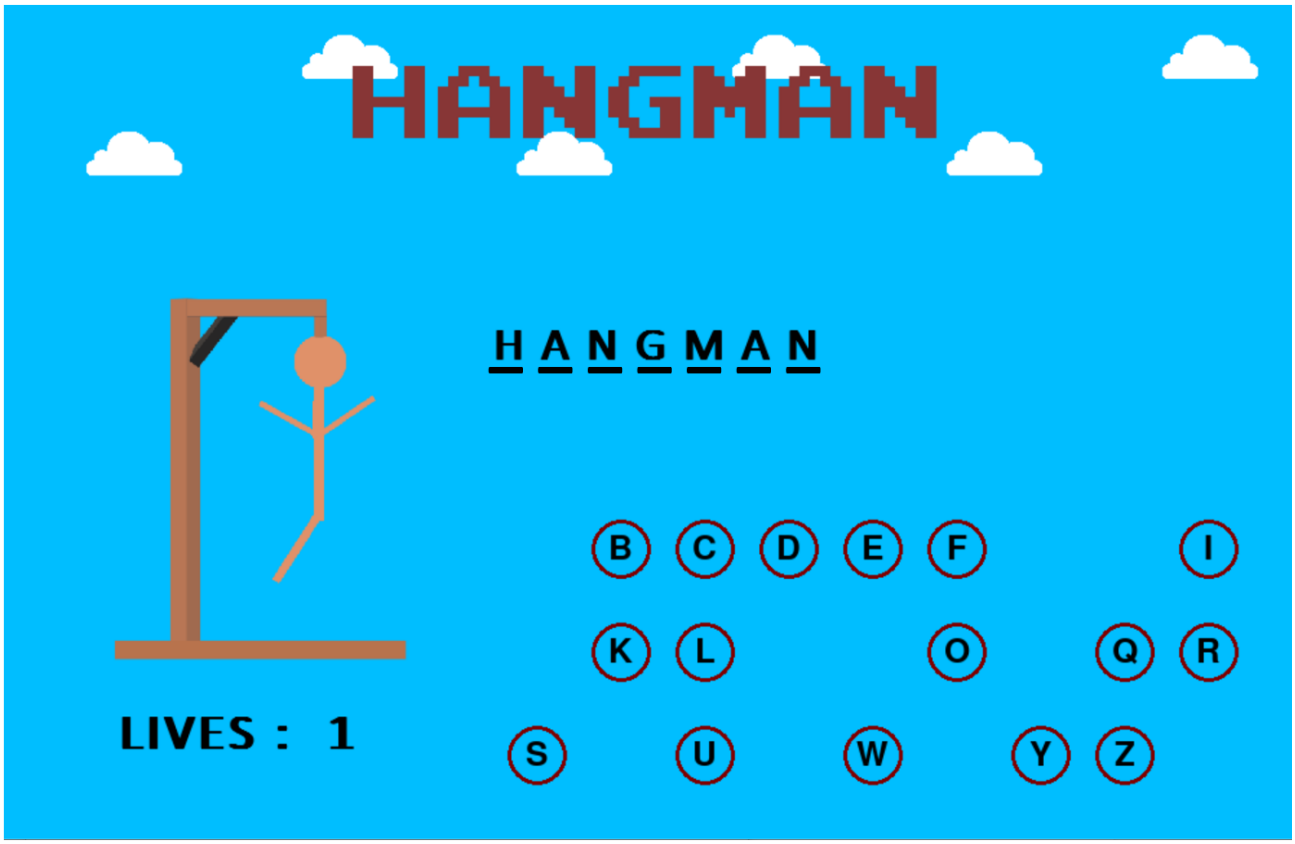
HANGMAN - PYGAME

SUMMER CAMP IT INTERNSHIP - Curtin University, Dubai



Faaizah Rajeev

29.07.2021

ACKNOWLEDGEMENTS

I express my gratitude to Ms Shahra Jafar, for her valuable advice and constant encouragement towards the fulfilment of my project. I thank her for providing access to a variety of resources that has ensured this project could come together.

I would like to extend my utmost gratitude to my family, especially my parents. Their unwavering belief in my abilities has ensured I aspire to do my very best throughout this endeavour.

INDEX

S.NO	TOPICS	PAGE
1	Abstract	3
2	Background	3
3	Methodology	4
4	Results	10
5	Conclusions	15
6	Future Work	15

ABSTRACT

The purpose of this project is to create a virtual game of Hangman. This classic word game relies on the random selection of words from a predefined list and guesswork by the player.

This game was made on Python with the help of the Pygame module to display real-time graphics and the Random module that chooses a random word from a list. I used concepts such as user-defined functions, while loops, for loops, if and else conditions, and methods from a few modules to complete this project.

BACKGROUND

Hangman, a simple spelling game, is a relic of the 19th century. The precise origin of the game remains unknown. However, it is speculated that the game emerged from a Victorian society where criminals were executed for committing severe crimes. Nowadays, this game is played by many for entertainment or to get a better grip of the English vocabulary.

The game aims to guess a random word by filling in letters into a series of blanks. With each incorrect letter guess, a new piece of the hangman figure (head, torso and each of the arms and legs) is drawn. The game ends when the player guesses the word or when he/she runs out of lives and the hangman stick figure is complete.

METHODOLOGY

In a classic game of Hangman, the word to guess can be any common noun or popular proper nouns such as movie, song, celebrity names. The words used in my program have been classified into four distinct categories: Movies, Sports, Singers, and Animals.

BASIC GAME FOUNDATION

Once the player chooses which category he/she wishes to play, a method from the 'Random' module: random.randint(start, end) ---> (returns a random integer value between the limits, both start and end included), is used. This helps to refer to a random word from the predefined lists through indexing.

The player gets a total of six lives. With each incorrect letter guess, the variable value (‘position’ - variable used in my program) is incremented and an updated image of the hangman is displayed on the screen. The letter inputted by the player, regardless of whether it is correct or not, is appended to a list called ‘guessed’ which is constantly compared to the individual characters of the correct word string-‘word’ by using a for loop.

Once all the characters of ‘word’ are part of ‘guessed’, the player wins. However, if the variable, ‘position’, is equal to six, the player has no lives left and loses the game.

REAL TIME GRAPHICS

The program consists of two main functions: main() and draw(). The main function consists of the main game loop and the basic game foundation, which has been described above. The draw function deals with displaying real time graphics, images, and text on the screen.

Before defining the functions, several images, fonts and music are to be loaded.

1) Display Screen: The display screen is to be initialised first.
   a=pygame.display.set_mode((width,height)) # used to initialize the pygame window or screen for display and set its size
   pygame.display.set_caption(display_screen_name) # names the display window pop up

2) Music: To play music, I used the pygame.mixer module.
   from pygame import mixer
   mixer.init() # initializes the mixer module
   mixer.music.load(file name) # loads the music file to be played
   mixer.music.play() # plays music in the pygame program

3) Images: To display images on the screen, we have to load them first.
   import pygame
   y=pygame.image.load(image_file_name.png) # loads the image required
   z=pygame.transform.scale(y, (width_of_the_image, height_of_the_image)) # resizes the image

4) Fonts: Several fonts were used in the program. They can be initialised through: Font1=pygame.font.SysFont(font_name ,size)

draw() Function

To display the images, texts, background colors, and shapes, the following methods were used:

1) Background colour: In Pygame, colors are expressed according to how much red, green, and blue they have in them. Three values are passed in a tuple (Red,Green,Blue) where each value can range from 0 to 255.
   Eg) (0,0,0) is black (absence of colour) and (255,255,255) is white.
   For my program, I used light blue shade with colour code (0,190,255). The background was filled by using:
   a.fill((0,190,255)) , where a is the object referring to the display screen.

2) Images: Stationary images like Hangman images, the Hangman title, etc. are displayed by using:

a.blit(z,(x_coordinate,y_coordinate)) # where z is the loaded image
Eg) a.blit(cloud, (0,0)), where cloud is an object containing an image of a cloud, will display the image at the upper left hand corner of the display window.

For moving images (mobile clouds in my program), the x and y coordinates need to be changed every time the screen is refreshed. In the program, the number of frames per second was set to 60. Therefore, the coordinates of the moving image will change 60 times in a second.

Under the main() function, rectangle objects that contain the clouds were created. Pygame uses Rect objects to store and manipulate rectangular areas.
Eg) cl1=pygame.Rect(0,25,75,35) # where (0,25) is the starting x and y coordinate and (75,35) is the width and height of the rectangle respectively.

As the clouds move from left to right, the y coordinate does not change. Everytime the screen is refreshed, the x coordinate is incremented by 1 unit through: cl1.x+=1 and the image is then blit in the draw() function:
a.blit(cloudsmall, (cl1.x, cl1.y)) # where cloudsmall is an object containing the loaded cloud image.

However, the movement of the cloud will terminate when it reaches the end of the screen. Here, an if condition can be used to change the position of the cloud to the beginning of the screen i.e.,
```
if cl1.x==925: # if the start of the cloud is approaching the end of the screen
        cl1.x=(-75) # changes the coordinate so that the start of the cloud will
```
have an x coordinate = 0 as the width = 75
This helps create the image of infinite floating clouds.

3) Shapes: Several shapes can be drawn in pygame. I used two shapes in the program: Circle (to display the alphabet buttons) and Rectangle (to display the try again button.
To draw a circle: pygame.draw.circle(a, (R,G,B), (x,y), radius, thickness),

where a is the display screen, (R,G,B) is the colour code, and (x,y) is the x and y coordinate.
To draw a rectangle: pygame.draw.rect(a,(R,G,B),[x,y,l,b],thickness), where a is the display screen, (R,G,B) is the colour code, x and y is the starting x and y coordinate, l is the length of the rectangle and b is the breadth of the rectangle.

4) Text: Before blitting text on the screen, it needs to be rendered first.
Eg) write=Font1.render('hello',1,(R,G,B)) # (Font1) was referenced above. 1 is the antialias and (R,G,B) is the colour code
a.blit(write,(x,y))

After blitting the images, graphics etc. , the screen has to be updated for the graphics to be visible. This is done by passing pygame.display.update()

BUTTON COLLISION

The button selected by the player has to be determined for the following functions:- Choosing a category, Choosing a letter, Choosing to play again.
To do this, the following commands are passed inside the main while loop:

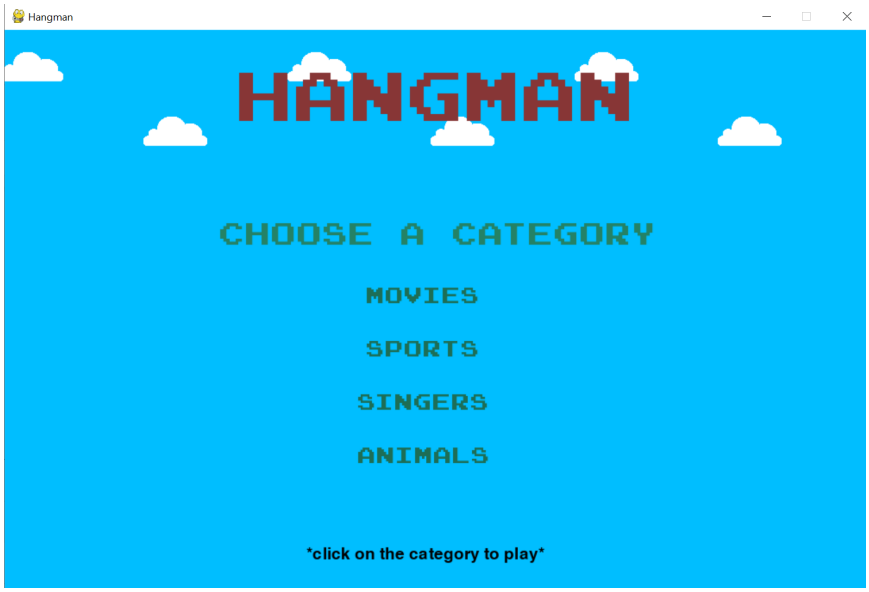
```
for event in pygame.event.get():
    if event.type==pygame.MOUSEBUTTONDOWN:
        px,py=pygame.mouse.get_pos()
```

px and py will return the x and y coordinate of a position that the player clicked. pygame.event.get gets events such as closing the pygame window, clicking a certain position on the screen, etc. from the queue.
if event.type==pygame.MOUSEBUTTONDOWN: checks if the user presses the mouse button. If the condition is satisfied, the coordinates of the position are retrieved.

SCREENS

To determine which screen is to be displayed, I used a variable 'f'.

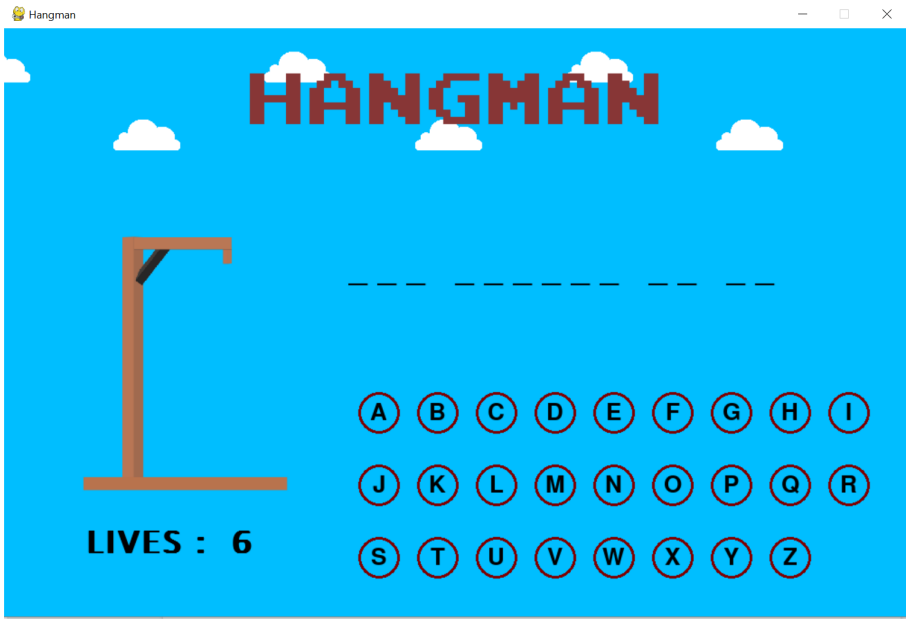
★ If f=0, the beginning screen, in which the player chooses the category, is displayed.



Screen 1 → f=0

Once the player chooses the category to play, ‘f’ is assigned to 1.

★ If f=1, the main game screen is displayed. The alphabets, number of lives, blanks and the hangman stand are displayed.



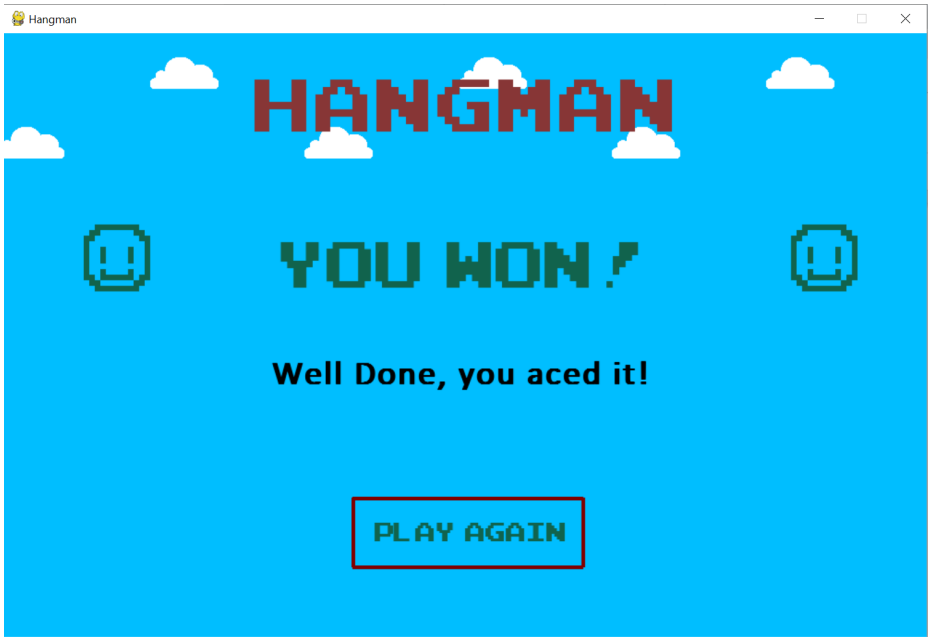
Screen 2 → f=1

According to the characters players guess, the hangman figure, number of lives, blank spaces, visible letters are updated.

If the player wins the game, ‘f’ is assigned to 2. If the player loses the game, ‘f’ is

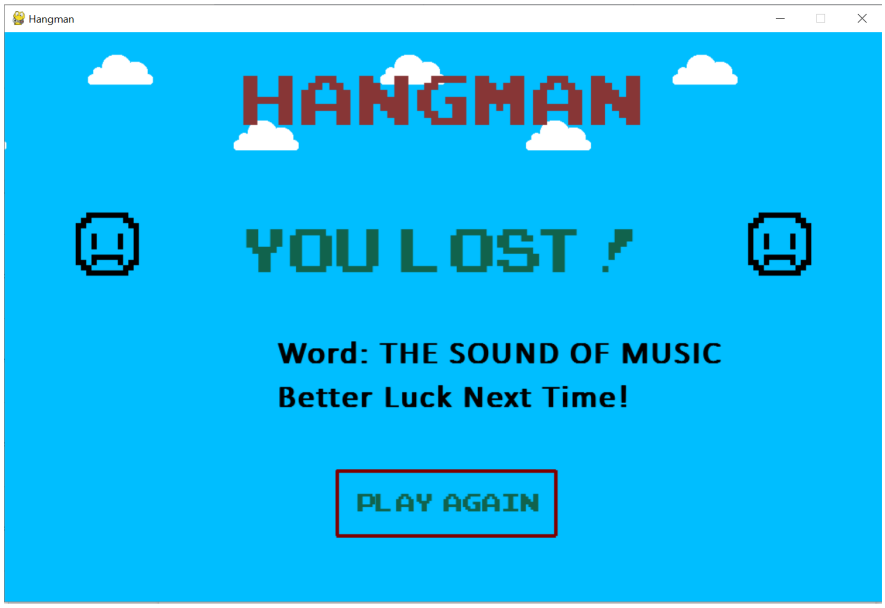
assigned to 3.

★ If f=2, a screen with ‘You won!’ and a play again option is displayed.



Screen 3 → f=2

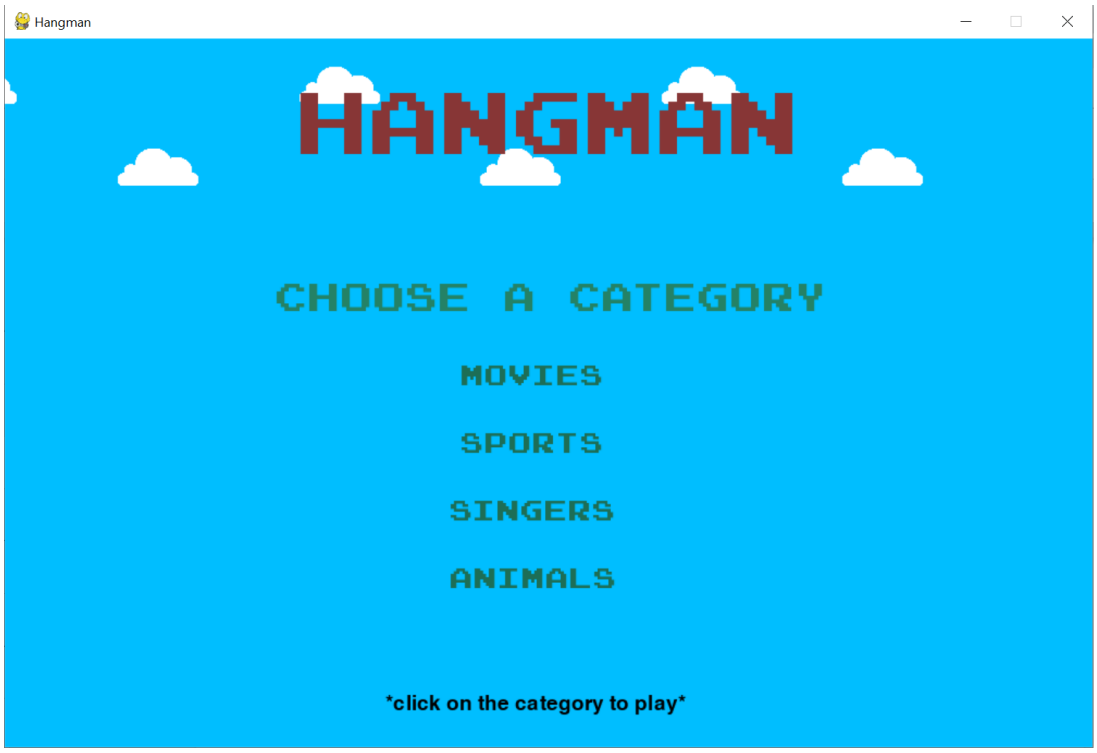
★ If f=3, a screen with ‘You lose’, the correct option and a play again option is displayed.



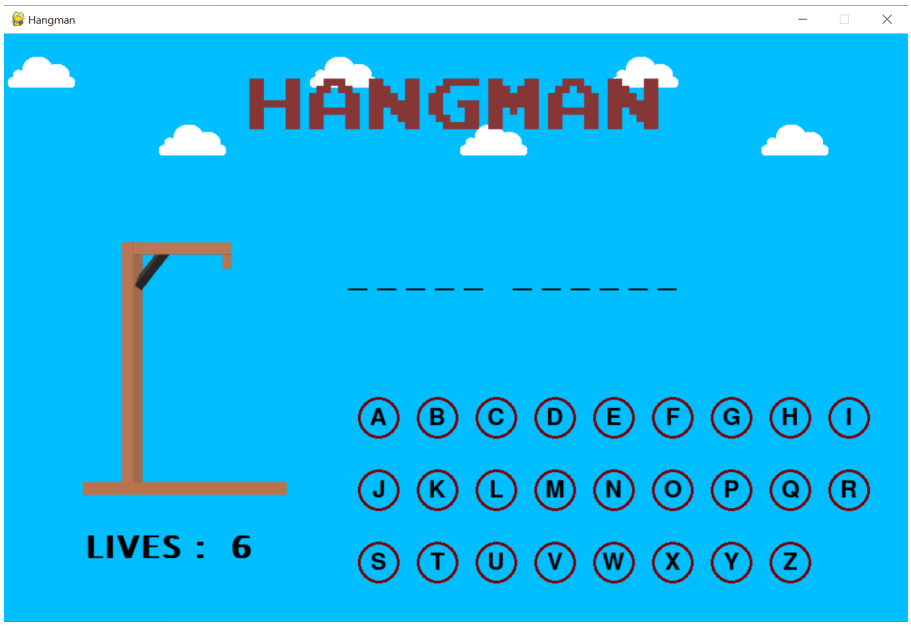
Screen 4 → f=3

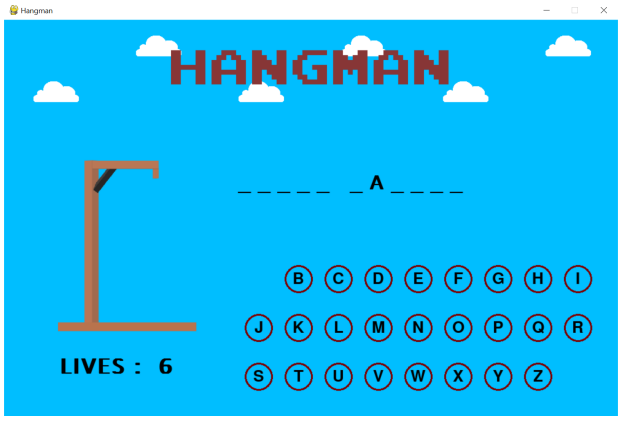
For f=2 and f=3, if the player chooses to play again, ‘f’ is assigned to 0 and Screen 1 is displayed.

RESULTS

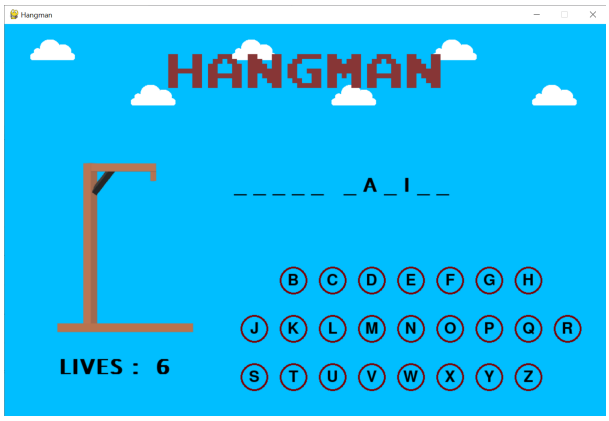


Player chooses Sports.

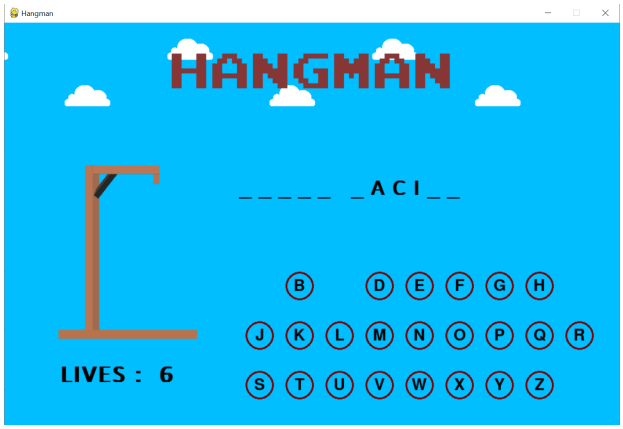




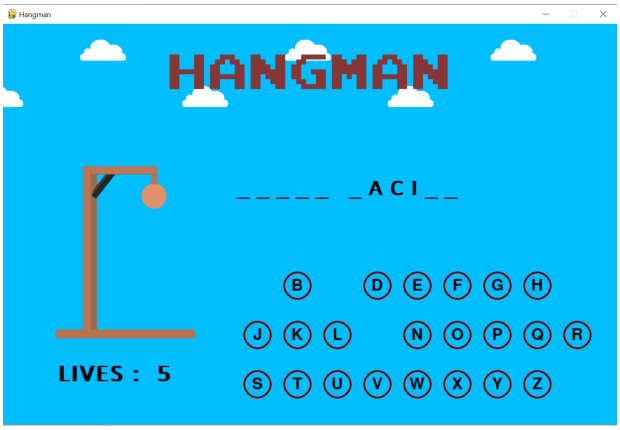
Player chose A



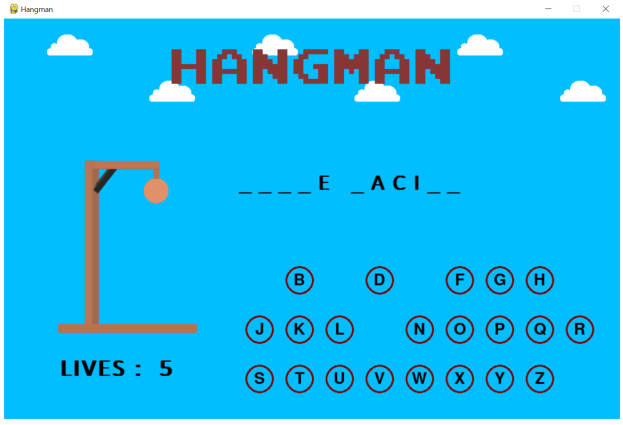
Player chose I



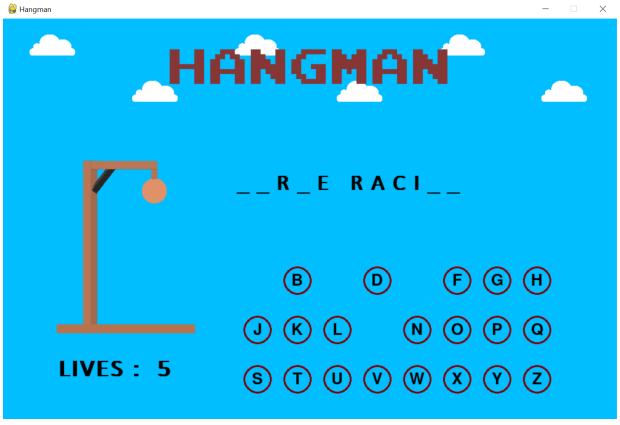
Player chose C



Player chose M



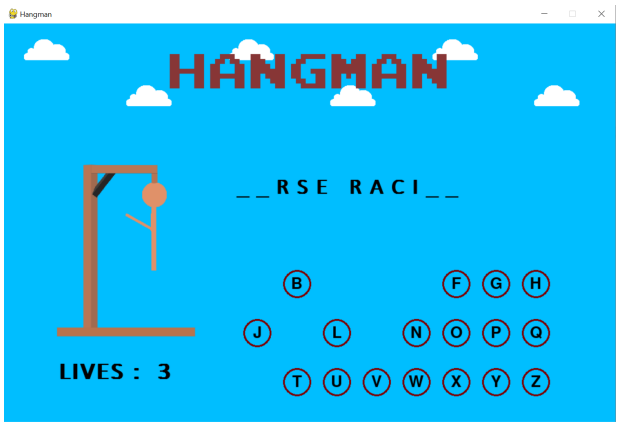
Player chose E



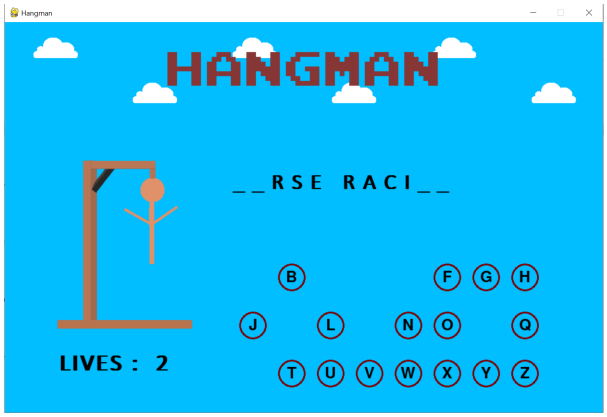
Player chose R

Player chose K

Player chose D

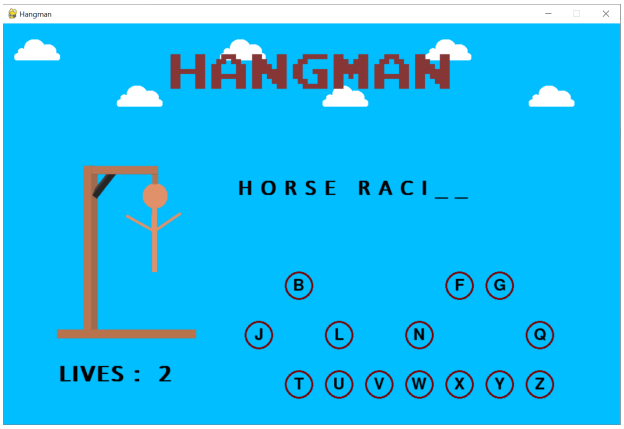


Player chose S



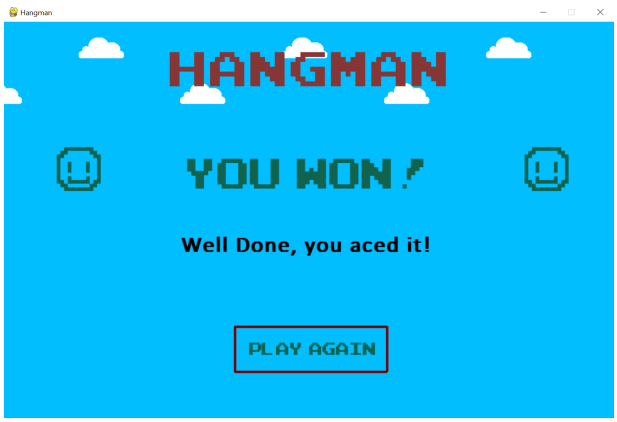
Player chose P

Player chose H

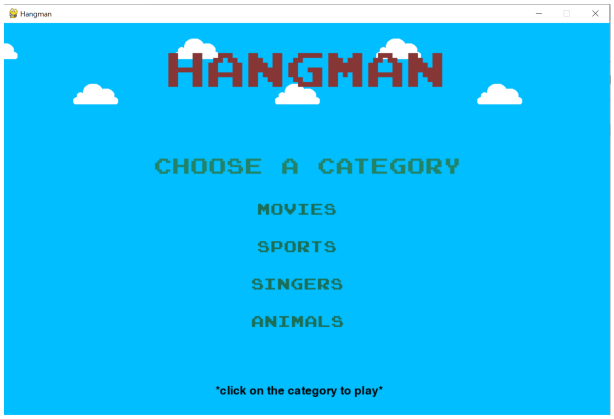


Player chose O

Player chose N

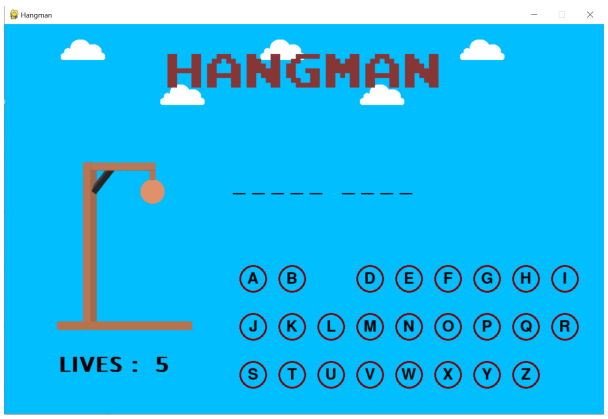


As soon as the player chose G.



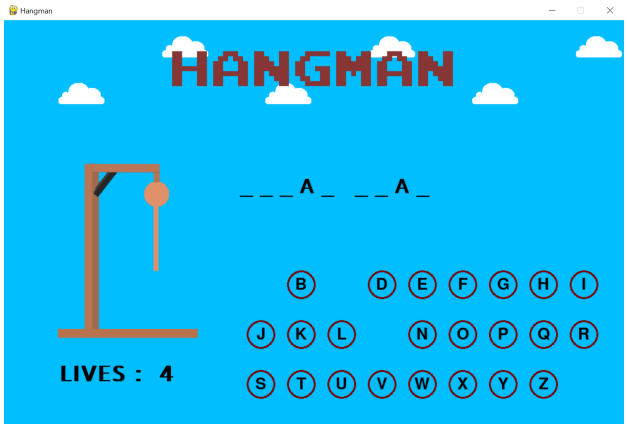
The player chose ‘Play Again’.

Player chose Animals

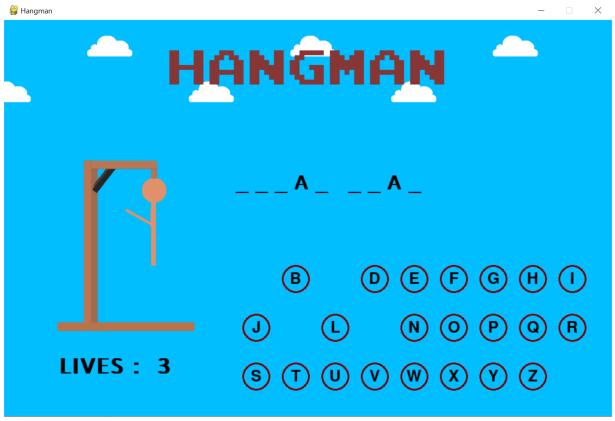


Player chose C

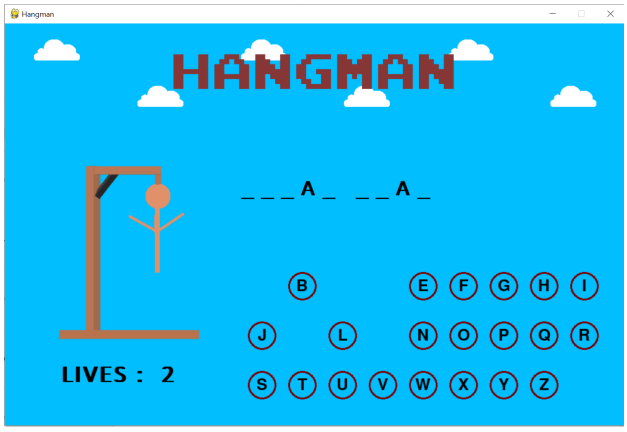
Player chose A



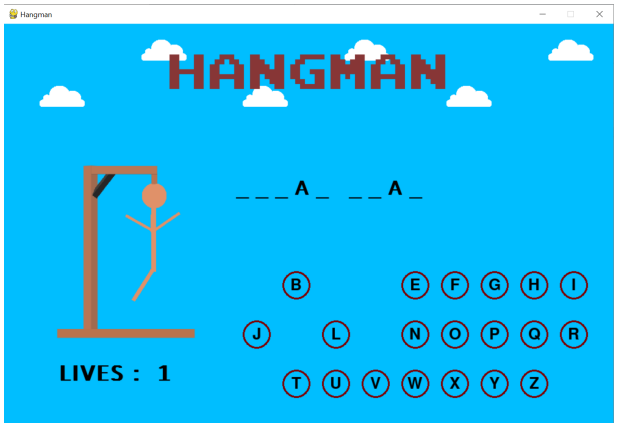
Player chose M



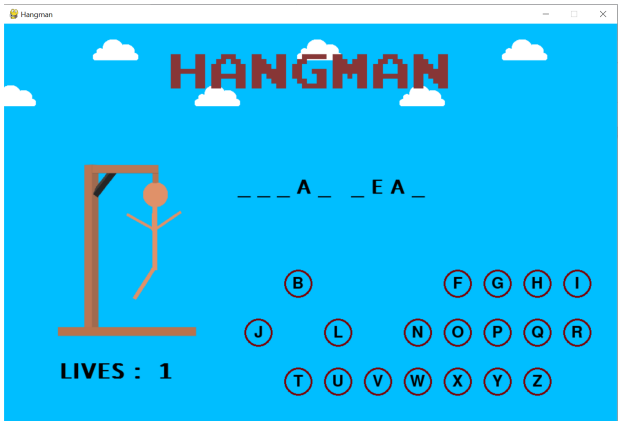
Player chose K



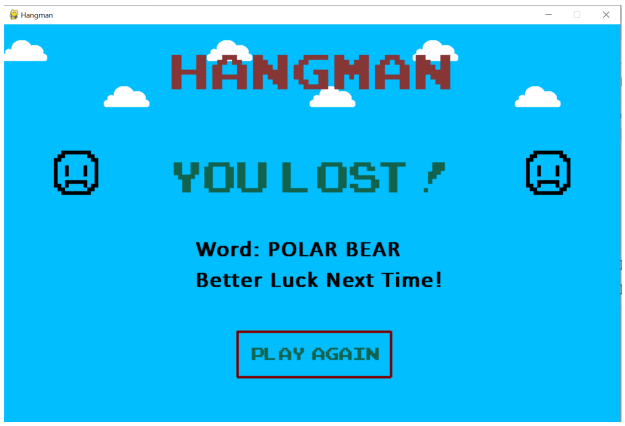
Player chose D



Player chose S



Player chose E



Player chose G and the game ended

CONCLUSION

The program is executed successfully with no errors. It displays the expected outputs in a minimum time span. The use of graphics helps make the program user friendly and fun to play.

This project taught me about game design, enhanced my programming skills and built-up confidence on what I can do with Python. I learnt the elements of making a fun game. I truly enjoyed the entire coding process.

FUTURE WORK

I believe the program can be made more interesting with a larger word library and a limited time for the player to guess the random word.

I would like to program Hangman in other programming languages such as Java or C++ or HTML. This would broaden my programming knowledge.

Furthermore, I wish to program other games such as the Snake game, Angry Birds, Flappy Birds etc.