**FINAL PROJECT**

ALGORITHM AND PROGRAMMING

**COMP 6047001**

PROJECT NAME **: UFO WORLD DOMINATION**

Name **: William Jonathan Mulyadi**

Student ID **: 2502045683**

Class **: L1CC**

**Project Specification**

UFO World Domination is a simple game which doesn’t require high logic and high skills; therefore, everyone can play it. The main aim of this project is to provide entertainment for everyone who play it. It is basically a shooting game, where you played as the UFO to attack the enemy by using the star bullet provided. The enemy will move downward each time it hits the edge of the screen and by the time it reaches ¾ of the screen size, the UFO will lose or in other words, it is game over. On the top left of the screen, score will be displayed. Each time you hit the skulls by using the star bullet, the score will increase by 1. However, the enemy will be respawned at a random position in range 1/3 of the top screen. ­­In the main page, there will be 3 menus which are skull, tako, and monster. Basically, it tells you the enemy you will be facing and how to choose it.

**Input**

1. User’s keystroke = left arrow and right arrow (to move left and right)
2. User’s keystroke = key space (to shoot the star)

**Output**

1. Score

**Solution Design**

1. First Page
2. Main Page
3. Play Again Page

**First Page**

The first page is pretty easy to understand. It only contains the title of the game “UFO WORLD DOMINATION “, and the enemies. There are 3 enemies which are skull, tako, and monster. There are also instructions on how to choose your enemy and how to play. By clicking keystroke on your keyboard, you can choose these enemies.

**Main Page**

The main page is the window where all of the game play in UFO WORLD DOMINATION occurs. The main page displayed the score on the top left of the screen to tell you how many skulls you have killed. The keystroke in your keyboard will be used to move the UFO and shoot the star bullet.

**Play Again Page**

This window tells you that you have lost this game. It will be shown if the skull has reached a certain y-point on the screen. There will be “GAME OVER” text, “try again” text, and “click anywhere on the screen” text. So if you want to play this game again, by clicking the screen you can play this game again.

**Implementation and explanation of the code**

Graphical user interface, text, application

Description automatically generated

For UFO World Domination, I only use these modules.

Import random = to put random position of an object.

Import math = for mathematical equation like square root, etc.

Import mixer = give the sound to the game (background sound, shooting sound, etc).

Text

Description automatically generatedThere are 5 classes that I used in this game, and it is saved in a file called screen.py. The classes are screen\_size class, MyColor class, myFPS class, p\_position class, and myFont class.

I set some of my function by default, therefore I won’t have to put in the number when the function is being called.

FPS is frame rate per seconds, and I set it to 60, therefore 60 FPS is the default of the FPS if nothing’s changed.

For the p\_position function, it is actually setting the position coordinate for the UFO or player by x and y.

**The first page (main menu) code**:

**Mainmenu.py**

**Graphical user interface, text

Description automatically generated**

These are the modules, I imported. I imported pygame, so I can use the built-in function in the pygame and I also imported other class from the screen file. Besides that, I import functions too from defaultgame file, hard file, and medium file. I also initialize the pygame and pygame.font.init.

Text

Description automatically generatedThe calling function, called the function on the screen file.

Then I created a function called main page. This function basically contains all the working function in this file.

I also created function called TTL\_text. It displays the tittle text in the first page.

In the main page, I assigned the button to be 0, therefore in this page, the button is 0. I did this in order to be able to change pages. Then I created the function called myButton. If the button is 0, it will stay in this page and in this page, there are texts of instructions about how to choose the enemy and how to play.

* If the button is 1, it will go to the main function in defaultgame.py file(VS skulls).
* Button 2 will go to the mainmedium function in the medium.py file(VS tako).
* Button 3 will go to the mainhard function in the hard.py file (VS monster).

Then, how do we change the button in this page?

While loop explanation

* Text

  Description automatically generatedIn pygame, we use while loop to make the screen stays on, if we don’t use while loop the screen will only be displayed in a few seconds. I assigned running to be True, therefore the while loop will loop only if the running equals to True. To quit the while loop, I assigned the running to be false if the event.type equals to the pygame.QUIT / the quit screen button.

So, the answer for how to assign the new button in the page is by pressing the arrow keystroke of left, down, and right. By pressing arrow left keystroke, the button will be assigned to 1, which will go to the main function in the defaultgame.py file and so on for the other.

In line 25 of main menu page, I already assigned clock to be pygame.time.Clock(), and I called the clock in line 25 by using clock.tick(Fps.getFPS()). I called the FPS function from the screen file too, to get the FPS. I put FPS in my code, so the FPS will be set to 60 fps in any device that opened this code.

Lastly, I called some of my functions that I mentioned before in my while loop, so it will be displayed on the screen. The myButton and the TTL\_text function. I used pygame.display.update() to update the screen display.

**The Main page:**

The main page actually contains 3 files which are:

* Defaultgame.py
* Medium.py
* Hard.py

I’m going to explain 1 of these files only, because there are only slight differences between these files. (I’ll show all the differences too)

**Defaultgame.py**

Text

Description automatically generated

These are the modules I imported. I imported pygame and initialize the pygame and pygame.font.init(). Besides that, I imported classes from screen.py file too. I also import math, so that I could use mathemathical equation in the code (i.e, math.pow,etc). Moreover, I imported random so that I could produce random positions which will be used for the enemy’s position in the code. The mixer is for putting the background sound of the game and sound effect when shooting and when the bullet hits the enemy.

Text

Description automatically generated

I called the function screen\_size by assigning SS equals to screen\_size(800,600). 800 is the width and 600 is the height in px. Then I called the function in line 20 by using the getter function of screen\_size.

A picture containing graphical user interface

Description automatically generatedFor the game title and icon, I already put some detailed comments there. I set the title of my game to UFO WORLD DOMINATION, and this will be displayed on the top left of the window pygame screen. Besides that, I assigned the logo to produce an image from file game3\_icon.jpg and then I called the logo inside pygame.display.set\_icon(logo). This is for displaying the icon on the top left of the window game.

There are also the calling functions, to call the class I imported from other files. The background is assigned to produce backgrund2.jpg file and the music background will be using the background2.wav. The music background will loop as I used (-1) for it.

Text

Description automatically generated

I also made some functions in this page. The first one is the display\_bg() function. This function actually shows the background in the window screen. The line 41 is actually filling the screen with a color that I get from calling the function cs.getColor(), which should be white. However, I put my background image above it by using myscreen.blit(background,(0,0)). The background is already assigned before in line 33 by using the background = pygame.image.load(“background2.jpg”). So the background image will be above the screen fill, therefore the screen fill can’t be seen unless the background image is removed.

 I also commented each of the object displayed on the screen. First, for the UFO. In this code I assigned ufoImg to produce ufo.png file, and I created a function to display the ufoImg in x and y position. As for the star image, I assigned the starImg to produce star1.png and star\_condition to be loaded. I also made a function to display the shooting star image by x and y position. However, to shoot the star from the middle of the object I added the x by 15.5 and y by 0.8. Secondly, I created the crashing function. Basically, this function calculates the distance between the enemy and the star. By using the modules, I imported from import math, I could use mathematical equation shown in the picture. This is actually counting distance by the distance formula:

Text

Description automatically generatedThis is the main function, where all of the code in the page runs.

Inside the main function, I created the UFO info where there is the position of the UFO x and y and the changes. I used getter function from the calling function I made in line 29. The ufoX\_ movement means that the ufo will move in the direction X by 0.5, on the other hand, it won’t move in the Y direction as it is equal to 0.

Besides that, there is also the skull info. I used a list for all the skull info, so that I could display multiple enemies on the screen. The NOE is the number of enemies, the NOE is 6, so there will be 6 skulls displayed on the screen. If you want to have more enemies, you could just increase the NOE. Then I used for loop and append the skulls info to the list.

The skullImg will display the skull.png file, while the skullX and skullY used random.randint which is for setting up random position where the skull will appear. The skull will move in X direction by 3 and Y direction by 40.I also created the skull function which is for displaying the image on the screen. By using the myscreen.blit(image,(position)).

Text

Description automatically generated\*(Still inside the main function)

I also created the star bullet info. I assigned the position and the changes; it will only move in the Y direction by 10. I also assigned the condition to be “loaded”.

Moreover, I created the My Score info. I assigned the position of myscore by 10 in both x and y direction. It will appear on the top left of the screen. I assigned font to pygame.font.Font(..), inside it I called the getter function from screen.py file and besides it contains the font size.

Then I created the function displaypoint(x,y), this function is used for displaying the score on the screen. I assigned the score to display Score: (the number) with white colour I called by the getter function by using the font that I assigned before.

Furthermore, I assigned the\_font and clickfont by calling the getter function and it has its own font size (64 and 16). In the lose\_text function, I assigned GO\_text to display GAME OVER and display white colour by using the getter function and use the\_font to set up the font. Then by using the myscreen.blit(..) to display it on the screen, inside it, it contains the (text,(position by x and y)). The same goes for the C\_text, but it displayed click anywhere on the screen text.

Text

Description automatically generated

I assigned the button to be 0 inside the main function and I also assigned TA\_font to have the font called by the getting function and has size of 32. Then I created the tryagain\_text() function . This function is basically used for playing the game again after you have lost the game or game over. By using the if function inside the function, I displayed if button equals to 0 it will display TRY AGAIN? By the font colour white and the font setting using the TA\_font I assigned before. Meanwhile, there is also the elif function, so if button equals to 1, the page will go to the main() function again.

Text

Description automatically generated

The while loop is basically used for the displaying the game continuously, if there is no while loop, the screen will only be displayed by a few seconds.

I assigned clock and it will be called in the while loop; it’s used to control the FPS of the game. So, in any device, the FPS will be set to 60 and it’s called by FPS.getFPS function.

In the while loop I also called the display\_bg function, so the background can be displayed in the screen continuously. I also created a for loop. Inside the for loop I used if functions. If event.type inside the pygame equals to pygame..QUIT or in other words clicking the quit on the screen of the pygame window, the while loop will stop, which mean the game will stop running. I also made other if function for which if a keystroke is pressed, there will be certain changes.

If event.type == pygame.KEYDOWN; it means if a key is pressed on the keyboard and inside this if, I put many other ifs again.

If event.key == pygame.K\_SPACE; it means if keystroke space on the keyboard is pressed and if the star\_condition equals to loaded, the starX will be assigned to be ufoX position, and it will emit the sound of shoot.wav. I used star\_Sound.play() to set up the sound. It doesn’t loop because I didn’t put -1 inside it unlike the one for the background music theme before. Then I called the shoot\_star function from the line 54 before to display the image and the star\_condition will be changed to shoot.

If event.key == pygame.K\_LEFT; it means if the left arrow keystroke is pressed, the ufoX\_ movement will be -5.5, this means the ufo will now move to the left position (in X) by 5.5 px and as for if event.key == pygame.K\_RIGHT; if the right arrow keystroke is pressed the ufo will move to the right direction (in X) by 5.5.

If event.type == pygame.MOUSEBUTTONDOWN; if the mousebutton is pressed, the button will be assigned to 1. So, it will replay the game again.

Text

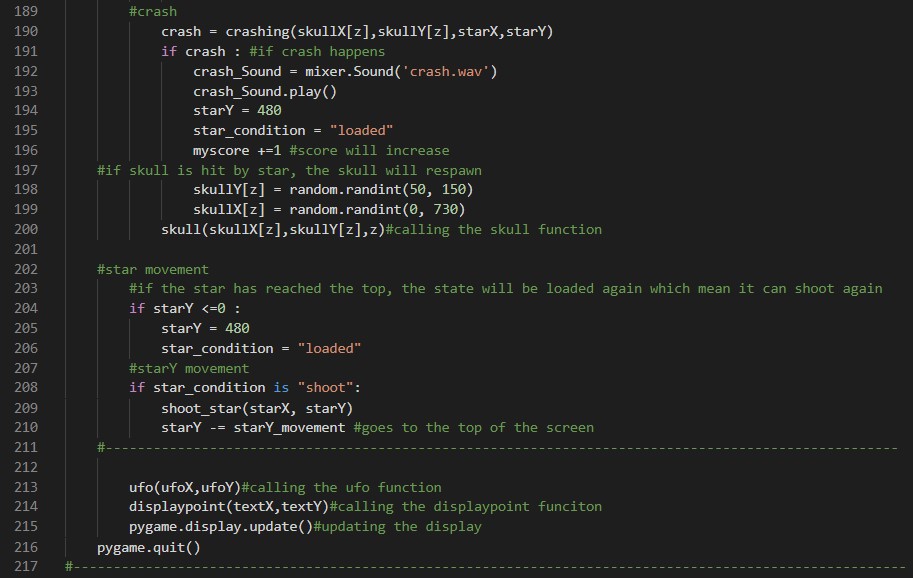
Description automatically generated(Still inside the while-loop but outside the for loop)

So, in this picture, I basically show the movement and boundaries of the ufo and the skull.

First of all the ufo. I used ufoX to be add and equal to the ufoX\_ movement for its movement and for the boundaries, I basically created it by using if function again. If the ufo has reached less than or equal 0, its positioned will be 0 and if it has reached more than or equal to 734 the position will be 734.

Secondly, for the skull. I used for loop again in range of NOE and if inside of it. If the skull has reached 365 in Y coordinate, the lose\_text() function and tryagain\_text() function will appear.

As for its boundaries, I assigned the skullX position[z] to be add and equal to the skullX\_ movement z]. As the skull is a list, that’s why we need to put the [z], so we know which skull we are talking about. If skullX\_ movement [z] equals or less than 0, the skullX\_ movement will be 3; it means, it will move in the X direction to the right by 3 and the skullY[z} += skullY\_movement[z], so each time it hits 0 or less than 0, it will change in Y direction by 40 px for 1 time. The same goes for the other boundary. If skullX[z] hits 734 or more, the change to the X direction will be to the left by 3 and it would also move in Y position by 40px one time as skullY[z] += skullY\_ movement [z].



(still inside the for loop in the skull movement) I also put the crash, and I assigned it to call the function crashing in line 58.

Inside if in for loop: If crash between star and the enemy happens, the crash\_Sound will play crash.wav. The star, then will be positioned in Y coordinate of 480 and X according to the ufo as starX equals to ufoX. The star\_condition will also be loaded and myscore will be added by 1 each time it crashes. The skull will respawn again in random position, and I called the skull function again outside the if.

(outside the for loop) As for the star movement, I used if function again. If the starY is less than or equal to 0; it means it has reached the top of the screen, it will be positioned in Y coordinate of 480 and the condition will be assigned to loaded. If the star condition is shoot, it will call the function of shoot\_star in line 54 and the starY -= starY\_ movement ;or in other words it will move to the top direction.

Text

Description automatically generated

Then I called again the function of ufo, displaypoint so it can be seen in the screen as it is now in the while loop. To update the screen I used pygame.display.update called at the end of the while loop.

Lastly, pygame.quit() to end the while loop.

**The differences:**

**Medium.py**

**main() -> mainmedium()**

In line 32 the background will show the image of ocean.jpg.

Text

Description automatically generated

In line 84, the NOE is changed to 8; before it is 6.

In line 87, the skullImg append the image of tako.png.

Text

Description automatically generated

In line 157 and 159, the movement of ufo in X direction is now faster as it becomes 5.5 and -5.5.

Text

Description automatically generated

In line 184 and line 187, the movement after the enemy hits the boundary changes from 3 and -3 to 3.5 and -3.5.

**Hard.py**

**main() -> mainhard()**



In line 32, the background will show the image of space.jpg

Text

Description automatically generated

In line 84, the NOE is now 10 and the skullImg will append monster.png image.

Application

Description automatically generated with low confidence

In line 157 and 159, the ufo movement in X direction is now faster as it becomes 6.5 and -6.5.

Text

Description automatically generated

In line 184 and 187, the movement to the X direction if it hits the boundaries will be 4 and -4 ; higher than before.

**The Play Again Page**

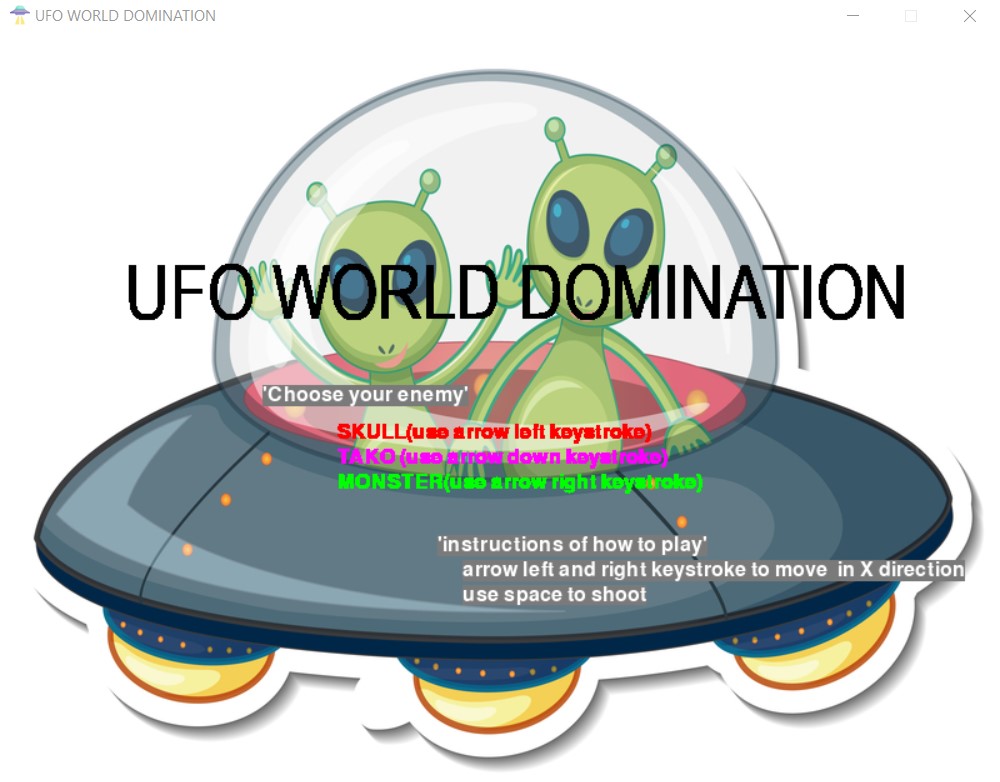
The play again page will be displayed if the game is lost and there will be game over text in the screen. The code of this page is already explained in the Main Page explanation.

Basically, if the enemy has reach Y coordinate of 360, the game will be over and there are two texts will be displayed which are:

* GAME OVER
* Click anywhere to play again

By clicking anywhere on the screen, the main()/ mainmedium()/ mainhard(), function will be played again

**PROOF OF WORKING PROGRAM**

The first page (main menu):

Graphical user interface

Description automatically generatedThe main page (defaultgame.py):

A screenshot of a video game

Description automatically generated with medium confidenceThe play again page (defaultgame.py):

A picture containing graphical user interface

Description automatically generatedThe main page (medium.py):

A picture containing graphical user interface

Description automatically generatedThe play again page (medium.py):

Graphical user interface, application

Description automatically generatedThe main page (hard.py):

Graphical user interface

Description automatically generatedThe play again page (hard.py):

**My reflection and experience**

I started doing all of my final projects in November and I tried my best finishing it before December, as it has been my habit to not procrastinate things that I can do. However, at first, I was uncertain of what project to be made and obviously my skill set as a programmer. When I first saw the project specification given by Mr. Jude, it took me about a month just to think about what project to be made and finally, I decided to make a shooting game.

I searched many tutorials on how to make simple games in You Tube and by observing those tutorials, I finally decided to use pygame to help me make my Final Project. I forgot why but it took me about 3 hours just to install the pygame inside my computer. It was quite hard and as I felt desperate, I asked for help from my upperclassman named RO. RO is very kind and patient with me, he is also smart and very clear in helping me not only in creating my FP but also includes materials in other classes.

When I first think about creating my game, I think about a simple game that requires almost no skill and very easy, therefore it can be enjoyed by anyone who plays it. A shooting game is what comes first to my mind. Meanwhile creating my game, one thing that I realised for sure is python is very sensitive. It’s very sensitive to capitalization, typos, indentation, etc. Those things bother me quite a bit as I did quite a few number of typos while creating it, therefore I have to check and recheck again and again as my code won’t run.

To sum up, at first, I was very uncertain whether I could do my Final Project or not. I was scared and desperate, yet I was quite thrilled because many ideas actually popped in my mind. There were lots of ideas but most of those ideas are way too complex for me and my programmer skill set to make it.However, I did not give up and by the help of people in my surroundings and surely Internet, I finally made one of my ideas to a realisation. I don’t have any basics of programming in my high school, but it doesn’t matter to me. One thing I always believe is if I’m willing to do my best, I believe one day I will be able to be a good programmer. Honestly, at the FYP I was quite scared to take Computer Science as it seems to be hard, stressful, and complex but as it goes on, it doesn’t turn out as scary as I expected. This is all thanks to Mr. Jude, who is very clear in explaining the materials and make programming fun for me. I am very thankful to Mr. Jude as he plays a big role for me to be able to enjoy programming world.

**References:**

For image of player, enemy, and icon:

<https://www.flaticon.com/search?word=ufo&type=icon>

For bullet sound:

<http://hsvensson.com/x/LJUDFILER/LJUDEFFEKTER/RIKOSCHT.WAV>

For crashing sound:

<https://mixkit.co/free-sound-effects/explosion/>

For background sound:

<https://www.chosic.com/download-audio/26012/>

For background image:

<https://www.kibrispdr.org/space-cartoon-background.html>

For the font:

<https://www.dafont.com/alpha.php?lettre=l>