

**Assignment Cover Letter****(Individual Work)**

Student Information:	Surname	Given Names	Student ID Number
1.	Weku	Brian	2101709995
Course Code	: COMP6502	Course Name	: Introduction to Programming
Class	: L1AC	Name of Lecturer(s)	: 1. Ida Bagus Kerthyayana 2. Tri Asih Budiono
Major	: CS		
Title of Assignment (if any)	: BIRDOP		
Type of Assignment	: Final Project		
Submission Pattern			
Due Date	: 6-11-2016	Submission Date	: 6-11-2016

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

(Name of Student)

“BIRDOP”

Name : Brian Moses Weku

ID : 2101709995

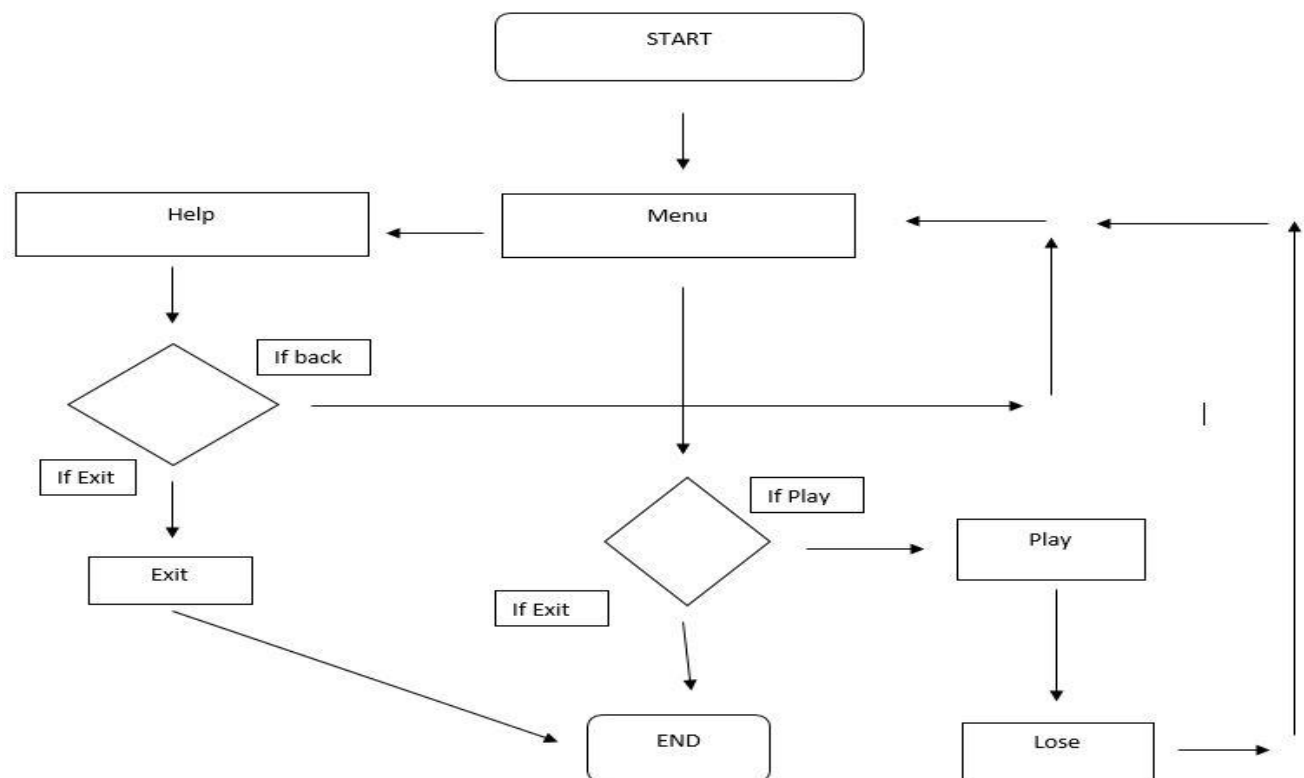
I. Description

The function of this program:

This program is built to entertain users. The game type is endless and if you die it resets from the beginning. With simple gameplay, the user can interact with the object in the screen and could have a fun moment and could be used as a competition to be played with others in search of higher scores

II. Design/Plan

Project's Hierarchy Chart



II. Explanation of Each function and Lessons that Have Been Learned

22nd of October,

I searched for ideas and decided to create a game through pygame. At first, I was trying to create an endless game that involves alien and obstacles.

```
1  from pygame import *
2  from pygame.mixer import *
3  from pygame.sprite import *
4  import random
5
6  class text_box(Sprite):
7      def __init__(self, message, x, y, font):
8          Sprite.__init__(self)
9          self.x=x
10         self.y=y
11         self.font=pygame.font.Font(None, font)
12         self.image=self.font.render(message,1,black,white)
13         self.rect=self.image.get_rect()
14         self.rect.center=(x, y)
15
16     class Alien(Sprite):
17         def __init__(self):
18             Sprite.__init__(self)
19             self.image=image.load('alien2.png')
20             self.rect=self.image.get_rect()
21             self.a=100
22             self.b=350
23             self.rect.center=((self.a, self.b))
24         def move(self, a, b):
25             self.a+=a
26             self.b+=b
27             self.rect.center=((self.a, self.b))
28
```

This day I learned plenty of basic things to create a game and the logic to it. Such as, if you were a human when you ride a car, when the car accelerates and goes forward the view from your car will go backwards while you gain distance, but for a programmer it's different, we manipulate the image background and let the main object still as if it was moving forward. As starters we need to import the required packages such as pygame. Next, I learned to create text boxes as sprites through class. The class involves position, the message, and color. Self.x and self.y serves as the position that will be input to locate the file position, message is the words that will be shown inside, and font is for how big is the size to create the text_box. Render means making so when you insert a message it will built the message, while self.rect is where it will get the rectangle size so if you will click on the rectangle it can create a command or do a command. For the Alien class, the definition of init() is initializing the program for starters so it's the basic of it and for the self. A and self. B it is similar to the text_box self.x and self.y and self.image is the loading

of the picture that I wanted. For def move it is a function where I can control the object and can change its location.

24th of October,

On Wednesday, I changed my mind into creating a bird but have the ability to become an alien. While bird will go unalterable, the alien is more flexible to control. Think of the alien as a power up.

```
7  #the bird
8  class Bird(Sprite):
9      def __init__(self,x,y):
10         Sprite.__init__(self)
11         self.image=image.load('bird.png')
12         self.rect=self.image.get_rect()
13         self.a=x
14         self.b=y
15         self.rect.center=((self.a, self.b))
16
17     def move(self, b):
18         self.b+=b
19         self.rect.center=((self.a, self.b))
20
```

This part is similar to the alien but the difference is the image that I want to load and the movement, because the bird movement is only allowed to go upwards and goes down through time. This class requires position.

```
#screen resolution and its title of game
screen=display.set_mode((1080,800))
pygame.display.set_caption("BIRDOP")
```

This is the name of the game and its resolution. The width of the screen is 1080 pixels and the height is 800 pixels. The name of this endless game is BIRDOP. It is a combination of bird that drops.

```
19  #colors
20  black=(0,0,0)
21  white=(255,255,255)
22  brown=(160,82,45)
23
```

RGB codes to be input in on the class text_box for colors.

```
31  menubg=image.load('bgground.png')
```

Image of the loading screen background.

```

36 def playon():
37     active1 = True
38     while active1:
39         screen.blit(menubg, (0,0))
40         first.draw(screen)
41         display.update()
42
43         for i in event.get():
44             if i.type==QUIT:
45                 pygame.quit()
46                 exit()
47

```

This is the main menu I insert it inside a function so it will be easy for me to call the function. I name the function playon() inside the playon there is active1=True this means while the program is still true it will run over and over until I quit the program. With the menubg as my preferred background picture I blit or insert it into my screen with the position 0,0 that means my top left corner and because of my image is as big as my screen resolution it will fit right in (the first.draw(screen) will be explained later. Display.update() means that everytime it passes this stage it will be updated. Then I created a loop, this loop will take every command there is and choose it and in this program when you pressed the close button (usually on top right if in windows) it will close and quit the program.

26th of October,

I made more sprites or buttons. The button will be displayed in different location and has different function

```

94 #back button
95 class Bbutton(Sprite):
96     def __init__(self):
97         Sprite.__init__(self)
98         self.x=100
99         self.y=700
100         self.image=image.load('backbutton.png')
101         self.rect=self.image.get_rect()
102         self.rect.center=(self.x, self.y)
103
104 #start button
105 class Sbutton(Sprite):
106     def __init__(self):
107         Sprite.__init__(self)
108         self.x=540
109         self.y=370
110         self.image=image.load('startbutton.png')
111         self.rect=self.image.get_rect()
112         self.rect.center=(self.x, self.y)
113
114 #help button
115 class Hbutton(Sprite):
116     def __init__(self):
117         Sprite.__init__(self)
118         self.x=540
119         self.y=450
120         self.image=image.load('Helpbutton.png')
121         self.rect=self.image.get_rect()
122         self.rect.center=(self.x, self.y)
123
124 #exit button
125 class Ebutton(Sprite):
126     def __init__(self):
127         Sprite.__init__(self)
128         self.x=900
129         self.y=700
130         self.image=image.load('exitbutton.png')
131         self.rect=self.image.get_rect()
132         self.rect.center=(self.x, self.y)
133

```

I learned that you can create buttons through pictures, so I decided to make buttons so that it will be different and look more game like. The first button that I create is back button this button will be used to get back to previous page, start button to start the game,

help button is to go to the help screen, and exit button same as the close button on your window but I will insert it in the main menu too for easier access. The self.x and self.y is similar to the class text_box this will determine its location.

```
1  #importing pygame, timer, music, random function, and other folder
2  from pygame import *
3  from pygame.sprite import *
4  import random
5  import burung_etc as b
6  import time
7
8  #def main():
```

This is the part that I imported again because basically I used 2 files, first is the classes named burung_etc and second is the main file named burung. From the first file I imported it to the second by typing import burung_etc as b so the b will tell you that I imported the function from the burung_etc file.

```
27  #sprites and pictures in the main menu
28  start=b.Sbutton()
29  inst=b.Hbutton()
30  bye=b.Ebutton()
```

In the second file I call the class from the burung_etc to burung. So I can summon it in my menu and others.

```
34  first=Group(start,inst,bye,sound,sound1)
```

```
def playon():
    active1 = True
    while active1:
        screen.blit(menubg, (0,0))
        first.draw(screen)
        display.update()
```

First, I group my sprites inside a variable and the first.draw(screen), the created button such as start, insst, bye will be inserted in my main menu.

```

245 #picture in the help menu
246 direction=Image.load('instructions.png')
247 back_button=b.Button()
248 third=Group(back_button)
249
250 def help():
251     active3=True
252     while active3:
253         screen.blit(direction, (0,0))
254         third.draw(screen)
255
256         display.update()
257
258         for i in event.get():
259             if i.type==QUIT:
260                 pygame.quit()
261                 exit()
262

```

This is the help menu as you can see it is similar with the main menu it uses active and uses direction as an image to load the background. The function to call this help menu is help().

```

back_button=b.Button()
third=Group(back_button)

def help():
    active3=True
    while active3:
        screen.blit(direction, (0,0))
        third.draw(screen)

        display.update()

```

While the back button is similar to the other buttons and inserting it

```

267
268         if i.type==MOUSEBUTTONDOWN:
269             if back_button.rect.collidepoint(mouse.get_pos()):
270                 active3=False

```

I.type==MOUSEBUTTONDOWN this means that if you click your mouse down and if you touched the rectangle or part of the image it will deactivate the while so it will return to the activated program which is the main menu.

2nd of November,

Figuring out the moving objects. The objects that is required to set the background moving in my program is the pipes and the background itself.

```
131 #moving background:
132 class Bg1(Sprite):
133     def __init__(self, x):
134         Sprite.__init__(self)
135         self.image=pygame.image.load('gamebg1.png')
136         self.rect=self.image.get_rect()
137         self.x=x
138         self.rect.top=0
139         self.rect.left=x
140     def move_left(self):
141         self.rect.left-=4
142 class Bg2(Sprite):
143     def __init__(self,x):
144         Sprite.__init__(self)
145         self.image=pygame.image.load('gamebg2.png')
146         self.rect=self.image.get_rect()
147         self.x=x
148         self.rect.top=0
149         self.rect.left=x
150     def move_left(self):
151         self.rect.left-=4
152 class Bg3(Sprite):
153     def __init__(self, x):
154         Sprite.__init__(self)
155         self.image=pygame.image.load('gamebg3.png')
156         self.rect=self.image.get_rect()
157         self.x=x
158         self.rect.top=0
159         self.rect.left=x
160     def move_left(self):
161         self.rect.left-=4
162 class Bg4(Sprite):
163     def __init__(self, x):
164         Sprite.__init__(self)
165         self.image=pygame.image.load('gamebg4.png')
166         self.rect=self.image.get_rect()
167         self.x=x
168         self.rect.top=0
169         self.rect.left=x
170     def move_left(self):
171         self.rect.left-=4
172
173
174 class Pipeup(Sprite):#first pair of pipes
175     def __init__(self):
176         Sprite.__init__(self)
177         self.image=pygame.image.load('pipeup.png')
178         self.rect=self.image.get_rect()
179         self.rect.top=700
180         self.rect.left=1000
181     def move_left(self):
182         self.rect.left-=4.5
183     def nextpos(self, top):
184         self.rect.left = 1000
185         self.rect.top = top
186 class Pipedown(Sprite): #first pair of pipes
187     def __init__(self):
188         Sprite.__init__(self)
189         self.image=pygame.image.load('pipedown.png')
190         self.rect=self.image.get_rect()
191         self.rect.bottom=1500
192         self.rect.left=1000
193     def move_left(self):
194         self.rect.left-=4.5
195     def nextpos(self, bottom):
196         self.rect.left = 1000
197         self.rect.bottom = bottom
198
199 class Pipeup1(Sprite): #second pair of pipes
200     def __init__(self):
201         Sprite.__init__(self)
202         self.image=pygame.image.load('pipeup.png')
203         self.rect=self.image.get_rect()
204         self.rect.top=500
205         self.rect.left=1600
206     def move_left(self):
207         self.rect.left-=4.5
208     def nextpos(self, top):
209         self.rect.left = 1000
210         self.rect.top = top
211 class Pipedown1(Sprite): #second pair of pipes
212     def __init__(self):
213         Sprite.__init__(self)
214         self.image=pygame.image.load('pipedown.png')
215         self.rect=self.image.get_rect()
216         self.rect.bottom=1300
217         self.rect.left=1600
218     def move_left(self):
219         self.rect.left-=4.5
220     def nextpos(self, bottom):
221         self.rect.left = 1000
222         self.rect.bottom = bottom
223
```

The pipes that is used is in this class Pipeup and Pipedown. Pipeup is the pipe on top and Pipedown is on bottom. The function move_left is so the pipe will decrease its x axis location and nextpos is the scramble of location after it touches a corner.

The background that is use is similar to the pipes it uses images and decreases its location's x-axis for the background I uses different colors of background to make it look better.

```
232
233     pipeup.move_left()
234     pipedown.move_left()
235     pipeup1.move_left()
236     pipedown1.move_left()
237     bg1.move_left()
238     bg2.move_left()
239     bg3.move_left()
240     bg4.move_left()
241
```

These function will move the backgrounds and pipes to the left.


```

73     pipeup=b.Pipeup()
74     pipeup1=b.Pipeup1()
75     pipedown=b.Pipedown()
76     pipedown1=b.Pipedown1()
77     bg1=b.Bg1(0)
78     bg2=b.Bg2(1080)
79     bg3=b.Bg3(2160)
80     bg4=b.Bg4(3240)

```

Inserted the classes in my main file. Bg1 too bg4 has inputs because the class requires their starting position

```

184         if bg1.rect.right<=0:
185             bg1=b.Bg1(3240)
186         if bg2.rect.right<=0:
187             bg2=b.Bg2(3240)
188         if bg3.rect.right<=0:
189             bg3=b.Bg3(3240)
190         if bg4.rect.right<=0:
191             bg4=b.Bg4(3240)

```

This is the function if the background reaches 0 it will have a new set position to start and move to the left.

```

157         if pipeup.rect.right<=0:
158             if pipedown.rect.bottom>=1350:
159                 newpos=random.randint(-125,0)
160             elif pipeup.rect.top<=-200:
161                 newpos=random.randint(0,125)
162             else:
163                 newpos=random.randint(-125,125)
164             posttop+=newpos
165             postbottom+=newpos
166             pipeup.nextpos(posttop)
167             pipedown.nextpos(postbottom)
168             score+=1
169             point.play()
170
171         if pipeup1.rect.right<=0:
172             if pipedown1.rect.bottom>=1350:
173                 newpos1=random.randint(-125,0)
174             elif pipeup1.rect.top<=-200:
175                 newpos1=random.randint(0,125)
176             else:
177                 newpos1=random.randint(-125,125)
178             posttop1+=newpos1
179             postbottom1+=newpos1
180             pipeup1.nextpos(posttop1)
181             pipedown1.nextpos(postbottom1)
182             score+=1
183             point.play()

```

For the pipes it is slightly different from the background. First I set the pipes as big as my screen resolution and so I just need to manipulate the space in between and how it will change its y axis. So after it reaches the left part of the screen the position will be either

increase or decrease, in order to have no misconduct I set a limit of how much it can decrease and how much it will increase. For example if you reach a top part of the position it will be forced to go low.

3th of November,

Creating the scoreboard that updates every time.

```
score=0
while active2:
    scoreboard=b.text_box("%d"%score, 540, 100, 35, white)
    second=Group(bg1, bg2, bg3, bg4, bird, pipedown, pipeup, pipedown1, pipeup1, scoreboard)
    second.draw(screen)

157     if pipeup.rect.right<=0:
158         if pipedown.rect.bottom>=1350:
159             newpos=random.randint(-125,0)
160         elif pipeup.rect.top<=-200:
161             newpos=random.randint(0,125)
162         else:
163             newpos=random.randint(-125,125)
164         postop+=newpos
165         postbottom+=newpos
166         pipeup.nextpos(postop)
167         pipedown.nextpos(postbottom)
168         score+=1
169         point.play()
170
171     if pipeup1.rect.right<=0:
172         if pipedown1.rect.bottom>=1350:
173             newpos1=random.randint(-125,0)
174         elif pipeup1.rect.top<=-200:
175             newpos1=random.randint(0,125)
176         else:
177             newpos1=random.randint(-125,125)
178         posttop1+=newpos1
179         postbottom1+=newpos1
180         pipeup1.nextpos(posttop1)
181         pipedown1.nextpos(postbottom1)
182         score+=1
```

First, I set my score=0 so every time it starts it will be 0. For scoreboard I insert the text_box class function. "%d"%score is the score replaces the d so for example the score is 5 it will show five. The position is set on 540 and 100, font size is 35 and the background color is white. Next is I inserted it inside the loop so every time it will be updated. The difference is when you put inside the loop the score will update over and over. Next, when the right part of the pipe touches the left screen the score will be added by one.

4th of October,

Creating my main object. The object will be a bird that can go up.

```
72     bird=b.Bird(100,350)     moveup=False
```

```
73     if i.key==K_SPACE:
74         start_time=pygame.time.get_ticks()
75         moveup=True
76         flap.play()
```

```
77
78     if isinstance(bird,b.Alien):
79         if moveup==True:
80             bird.move(0,-3)
81
82         if movedown==True:
83             bird.move(0,3)
84
85     if isinstance(bird,b.Bird):
86         if pygame.time.get_ticks()-start_time>=500:
87             moveup=False
88             bird.normal()
89         if pygame.time.get_ticks()-start_time>=700:
90             moveup=False
91             bird.down()
92         if moveup==True:
93             if bird.rect.top>=0:
94                 bird.move(-5.81)
95                 bird.up()
96         if moveup==False:
97             if bird.rect.bottom<=800:
98                 bird.move(4)
```

First I inserted the class and I set its first location and I set a variable named moveup to be False. If I pressed space start_time will start taking time and moveup which is before false becomes True. After that I specified that if it is true it will move up. If it will go up it will load different images so it will look better and when it will fall it will be different. For the time if the time reaches 500 millisecond it will stop and it will go down, I also inserted pictures to set if it is falling and if its at the highest point.

```
99     if bird.rect.colliderect(pipeup) or bird.rect.colliderect(pipedown) or bird.rect.colliderect(pipeup1) or bird.rect.colliderect(pipedown1):
100         hover.stop()
101         screen.fill(white)
102         second.draw(screen)
103         screen.blit(gover,(340, 300))
104         display.update()
105         crash.play()
106         time.sleep(3)
107         active2=False
```

In this part if the object collide with the obstacle it will stop all the screen action because I sleep it for 3 seconds and when it does active2 becomes False so it will go back to the menu.

5th of October,

I added the alien as power up with some music.

```
fourth=Group()  
fifth=(pipedown, p
```

```
if isinstance(bird, b.Bird):  
    if pygame.time.get_ticks()-spawn>=17000:  
        spawn=pygame.time.get_ticks()  
        tempalien=b.Alien(1080,random.randint(300,500))  
        fourth.add(tempalien)
```

Fourth is an empty group. Then for every 17 seconds it will spawn an alien icon by inserting it in the fourth group so it can be drawn

```
if spritecollideany(bird, fourth):  
    alientimer=pygame.time.get_ticks()  
    moveup=False  
    bird = b.Alien(bird.rect.centerx,bird.rect.centery)  
    hover.play()
```

When the bird collides with the alien it will become the alien by taking its last position.

```
if i.key==K_UP:  
    moveup=True  
  
if i.key==K_DOWN:  
    movedown=True
```

```
if i.type==KEYUP:  
    if i.key==K_UP:  
        moveup=False  
  
    if i.key==K_DOWN:  
        movedown=False
```

```
if moveup==True:  
    bird.move(0,-3)  
  
if movedown==True:  
    bird.move(0,3)
```

When the bird inherits the alien it will have different types of movement. So when you hold a button down it will be directed always and if you let go of the button it stops.

```
groupcollide(fifth,fourth,False,True)
```

To improve my code I will delete the alien if it touches the pipes, so the alien will not get inside the pipes.

```
if pygame.time.get_ticks()-alientimer>=7000:  
    bird=b.Bird(bird.rect.centerx,bird.rect.centery)  
    movedown = False  
    moveup = False
```

This is the timer for how long the bird can be an alien. After the timer reaches 7 seconds it will return the bird and its location.

```
if isinstance(bird, b.Bird): if isinstance(bird,b.Alien):
```


If instance means that if the bird is in the class b.Bird it will give a True. So, while it is an alien the if instance(bird,b.Bird) it will give a False so it will no use the program but if it is true it will run the program.

```
#the alien icon and when it became one
class Alien(Sprite):
    def __init__(self,x,y):
        Sprite.__init__(self)
        self.count=3
        self.image=image.load('images/alien3.png')

        self.rect=self.image.get_rect()
        self.a=x
        self.b=y
        self.rect.center=((self.a, self.b))
    def move(self, a, b):
        self.a+=a
        self.b+=b
        self.rect.center=((self.a, self.b))
    def move_left(self):
        self.rect.left-=4.5
    def Add(self):
        self.count += .25
    def checkcount(self):
        if self.count<=5:
            self.image=image.load('images/alien3.png')
        if self.count >= 5:
            self.image = image.load ('images/alien0.png')
        if self.count >= 10:
            self.image = image.load ('images/alien1.png')

        if self.count >=15:
            self.count=0
```

```
if isinstance(bird,b.Alien):

    bird.Add()
    bird.checkcount()
```

Self.add and self.checkcount will work together to make the alien animation. So when the count is less than 5 it will load the first image and if it is mre than 5 it will load the second image and this will go on until it reaches 15, if it is greater than 15 it will start from 0 again so this will go on and on.

Last but not least I included music to the game so the game will not be so silent.

```
10 pygame.mixer.init()
11 pygame.mixer.music.load('pixel.mp3')
12 pygame.mixer.music.play(-1)
13
14
```

The init will initialize the music while the music.load loads the document and play will play the music but it will be asked to play how many time because I wanted it to be played forever I run the program I set it too -1.

```
76 #play music button
77 class Playbutton(Sprite):
78     def __init__(self):
79         Sprite.__init__(self)
80         self.x=100
81         self.y=100
82         self.image=image.load('play.png')
83         self.rect=self.image.get_rect()
84         self.rect.center=(self.x, self.y)
85 #pause mmusic button
86 class Pausebutton(Sprite):
87     def __init__(self):
88         Sprite.__init__(self)
89         self.x=155
90         self.y=100
91         self.image=image.load('pause.png')
92         self.rect=self.image.get_rect()
93         self.rect.center=(self.x, self.y)
94
95 play()
96 if sound1.rect.collidepoint(mouse.get_pos()):
97     pygame.mixer.music.pause()
98 if sound.rect.collidepoint(mouse.get_pos()):
99     pygame.mixer.music.unpause()
```

Playbutton is the new sprite class that I will give a function if it's clicked it will play the music and if pausebutton is another sprite class that gives the pause function


```

flap=pygame.mixer.Sound('flap.ogg')
point=pygame.mixer.Sound('point.ogg')
crash=pygame.mixer.Sound('crash.ogg')
hover=pygame.mixer.Sound('hover.ogg')
active2=True

if i.key==K_SPACE:
    start_time=pygame.time.get_ticks()
    moveup=True
    flap.play()

```

```

if pipeup.rect.right<=0:
    if pipedown.rect.bottom>=1350:
        newpos=random.randint(-125,0)
    elif pipeup.rect.top<=-200:
        newpos=random.randint(0,125)
    else:
        newpos=random.randint(-125,125)
    postop+=newpos
    postbottom+=newpos
    pipeup.nextpos(postop)
    pipedown.nextpos(postbottom)
    score+=1
    point.play()

if bird.rect.colliderect(pipeup):
    hover.stop()

second.draw(screen)
screen.blit(gover,(340, 300))
display.update()
crash.play()
time.sleep(3)
active2=False

```

```

if pygame.sprite.groupcollide(birdgroup, fourth, False, True):
    alientimer=pygame.time.get_ticks()
    moveup=False
    bird = b.Alien(bird.rect.centerx,bird.rect.centery)
    hover.play()

if isinstance(bird, b.Bird):
    hover.stop()

```

These are some other files that will be played whenever I did a function whether if I became an alien, when I tap space to fly, when it crashes, or when you receive a score

III.b. Problem that Have Been Overcome

Problems that I overcome:

- score
- alien popping
- changing the bird form