Departamento de Ciencias de la Computación
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
CC-3501 Modelacion y Computacion Grafica para Ingenieros

TAREA N°2

Flappy Dunk

Integrante: José Miguel Pacheco
Profesor: Nancy Hitschfeld
Auxiliares: Mauricio Araneda
Pablo Pizarro
Pablo Polanco
Ayudantes: Iván Torres
Maria Josá Trujillo
Fecha de entrega: 8 de julio de 2018
Santiago, Chile

Índice de Contenidos

Lista de Códigos

1. Descripción del problema

Flappy Dunk es un juego en el cual el usuario controla una pelota la cual debe atravesar una serie de aros dispuestos en la pantalla, estos aros tienen ancho y alto aleatorios al igual que su posición, también la pelota no puede tocar el techo ni el suelo del juego, además de que se deben atravesar las aros de arriba hacia abajo y no debe saltarse ningún aro. Todo esto debe ocurrir en un escenario de al menos 3 colores y que no sean rectángulos planos y por ultimo la pantalla debe seguir siempre a la pelota para que esta se mantenga en el primer tercio de la pantalla.

2. Esquema de la solución

La solución para este juego consistió en la implementación de varias clases que luego se juntaron para crear la solución final, estas clases son 10 las cuales son: aro1, aro2, sol, fondo, nube, pelota, vista, text, fases y main. De estas, las primeras 6 corresponden a objetos que se visualizan en la pantalla, vista es para poder dibujar los objetos en la pantalla, main es la clase que junta todo y permite que todo funcione correctamente, esto quiere decir que la pelota no puede tocar el techo o el suelo, los aros solo se pueden cruzar de arriba hacia abajo, que los aros y las nubes tengan movimiento horizontal, además que no se puede saltar ningún aro y se encarga de contar el puntaje, text es la clase encargada de mostrar una imagen al momento de perder con el puntaje, el puntaje máximo desde que se abrió el juego y una opción de replay y por ultimo fases es una clase que contiene dos funciones que permiten saber si el balón paso por los aros limpiamente o tocando uno de los bordes, para así asignar el puntaje correspondiente. Además se utiliza la clase CC3501Utils que es de gran ayuda para facilitar la programación.

3. Solución detallada

Para lograr crear el juego, se creo una pelota que siempre estará en la misma posición horizontal de la pantalla, pero que al apretar la tecla espaciadora da un pequeño salto hacia arriba y luego de un momento comienza a caer, todo esto gracias a que se simula que esta bajo cierta gravedad y que la momento de apretar la barra espaciadora se le asigna una velocidad inicial vertical a la pelota (Todo el movimiento es regido bajo las leyes de la física y sus respectivas formulas). La clase Pelota() recibe una posición en la cual se crea la pelota y dentro de la clase se define el radio que es de 30 pixeles.

Código 1: Clase Pelota

```
class Pelota(Figura):
def ___init___(self, pos=Vector(0,0),rgb=(1.0, 1.0, 1.0)):
    self.radio=30
    super().___init___(pos,rgb)
```

Además para crear la pelota se creo la función figura() en la cual en base a OpenGl se crean círculos (con glBegin(GL_POLYGON)) de diferentes radios para hacer el contorno de la pelota y un diseño, también para el ala se utilizaron rectángulos y círculos que dan el aspecto de un ala.

Código 2: Clase Pelota, función figura

```
def figura(self):
    glBegin(GL_POLYGON)
    glColor3f(1.0, 0,0)
    i=0.0
    while (i<2*pi):
        x=self.radio*cos(i)
        y=self.radio*sin(i)
        glVertex(x,y)
        i=i+0.1
    glColor3f(0, 0, 0)
    i = 0.0
    glEnd()

    glBegin(GL_POLYGON)
    while (i < (2*pi)):
        x = 20 * cos(i)
        y = 20 * sin(i)
        glVertex(x, y)
        i = i + 0.1
    glEnd()

    glBegin(GL_POLYGON)
    glColor3f(1,1,1)
```

```
24    i = 0.0
25    while (i < (2*pi)):
26        x = 10 * cos(i)
27        y = 10 * sin(i)
28        glVertex(x, y)
29        i = i + 0.1
30    glEnd()
31
32    glBegin(GL_POLYGON)
33    glColor3f(1,1,0)
34    i=0.0
35    while (i < (2*pi)):
36        x = 25*cos(3*pi/4)+10 * cos(i)
37        y = 25*sin(3*pi/4)+10* sin(i)
38        glVertex(x, y)
39        i = i + 0.1
40    glEnd()
41    glBegin(GL_QUADS)
42    glColor3f(1,1,0)
43
44    glVertex2f(25*cos(3*pi/4)-20,25*sin(3*pi/4)+10)
45    glVertex2f(25*cos(3*pi/4)-20,25*sin(3*pi/4))
46    glVertex2f(25*cos(3*pi/4),25*sin(3*pi/4))
47    glVertex2f(25 * cos(3 * pi / 4), 25 * sin(3 * pi / 4) + 10)
48    glEnd()
49    glBegin(GL_POLYGON)
50    glColor3f(1, 1, 0)
51    i = 0.0
52    while (i < (2 * pi)):
53        x = 25*cos(3*pi/4)-20 + 5 * cos(i)
54        y = 25 * sin(3 * pi / 4) + 5 + 5 * sin(i)
55        glVertex(x, y)
56        i = i + 0.1
57    glEnd()
```

Y por ultimo la clase Pelota tiene unas ultimas 3 funciones, una llamada mover(), que recibe una variable numérica 'dt' en la cual se modifica la posición de la pelota en función de las leyes de la física (se define una aceleración de gravedad al comienzo) y el tiempo que se ha pasado sin presionar la barra espaciadora (que es lo que significa el dt), ya que al momento de presionar la barra espaciadora el dt es 0, y se simula que la pelota fue lanzada con una velocidad inicial vertical. Las otras dos funciones son chocandosuelo() y chocandotecho() que tal como dice su nombre, entregan un boolean dependiendo si la pelota toca o no toca el techo o suelo, esto se logra comparando la posición de la pelota mas o menos el radio con la posición del techo o el suelo, el centro de la pelota mas el radio si se compara con el techo y el centro de la pelota menos el radio si se compara con el suelo.

Código 3: Clase Pelota:función mover(), chocandosuelo() y chocandotecho()

```
global gver, vely
gver = Vector(0, -1)
vely = Vector(0, 4)

def mover(self,dt ):
      self.pos = sumar(self.pos, sumar(vely, ponderar(dt * dt / 2.0, gver)))

def chocandotecho(self):
   return self.pos.y-self.radio<=0

def chocandosuelo(self):
   return self.pos.y+self.radio>=600
```

Para crear un escenario en el cual se desarrollara el juego, se crearon 3 clases, la primera llamada Fondo() que simplemente coloca el fondo de la pantalla de un color azulado.

Código 4: Clase Fondo

```
class Fondo(Figura):
   def ___init___(self, pos=Vector(0, 0),rgb=(1.0, 1.0, 1.0)):
      super().___init___(pos,rgb)

   def figura(self):

      glBegin(GL_QUADS)
      glColor3f(103/255.0, 193/255.0,1)
      glVertex2f(2000, 0)
      glVertex2f(0, 0)
      glVertex2f(0,600)
      glVertex2f(2000,600)
      glEnd()
```

La segunda llamada Sol() que muestra un cuarto de sol en la esquina superior esquina de la pantalla.

Código 5: Clase Sol

```
class Sol(Figura):
   def ___init___(self, pos=Vector(0, 0), rgb=(1.0, 1.0, 1.0)):
      self.radio=100
      super().___init___(pos,rgb)

   def figura(self):
      glBegin(GL_POLYGON)
      glColor3f(1, 1,0)
      i=0
```

```
        while (i<=2*pi):
            x=self.radio*cos(i)
            y=self.radio*sin(i)
            glVertex(x,y)
            i=i+0.05
        glEnd()
```

Y por ultimo se creo la clase Nube() que para crearla se hace de la misma forma mostrada en el juego Ici Tower mostrado en un auxiliar (por lo que el código de como crearla no se enseñara), además tiene una función mover() la cual simula que las nubes se muevan hacia la pelota y al momento de salir de la pantalla se crea otra nube en el otro lado (con posición y tamaño aleatorio) y se repite el ciclo

Código 6: Clase Nube: función mover()

```
    def mover(self):
        self.pos=sumar(self.pos,ponderar(2, velx))
        if self.pos.x<-300:
            self.pos=Vector(randint(1200,1350),randint(50,450))
```

Las siguientes clases que se crearon fueron las clases Aro1() y Aro2(), estas clases permiten la creación del aro y que se haga el efecto de que la pelota traspasa el aro ya que Aro1 representa la parte superior (o de atrás) del aro, en cambio Aro2 representa la parte de abajo (o adelante) del aro. La clase Aro1 recibe la posición en la que se dibujara el aro y dentro de esta se definen de forma aleatoria el ancho y alto del aro además de una variable boolean 'c' que se le da el valor de False.

Código 7: Clase Aro1

```
class Aro1(Figura):
    def ___init___(self,pos=Vector(0,0),rgb=(1.0,1.0,1.0)):
        self.a=randint(50,100)
        self.b=randint(20,40)
        self.c=False
        super().___init___(pos,rgb)
```

Para crear la mitad superior del aro se utilizo la función figura() en la cual en base a OpenGl se crea la mitad de una elipse (con glBegin(GL_POINTS)) con el ancho y alto definidos anteriormente además de un grosor de 5 pixeles para cada puntos.

Código 8: Clase Aro1: funcion figura

```
def figura(self):
    glPointSize(5)
    glBegin(GL_POINTS)
    glColor3f(0, 0, 1)
    i=0.0
    while (i<pi):
        x=self.a*cos(i)
```

```
        y=self.b*sin(i)
        glVertex2f(x,y)
        i=i+0.001
    glEnd()
```

Por ultimo se definen las funciones mover() y cambiar() que reciben nada, la función mover hacer que el aro se mueva horizontalmente (con una velocidad definida antes) hacia la pelota para simular el movimiento de esta y cambia el valor de la variable definida anteriormente 'c' a True al momento de que el aro salga de la pantalla, y la función cambiar retorna un boolean indicando si el aro esta fuera de la pantalla o no, para así crear un nuevo ahora en el otro lado de la pantalla (explicado con detalle mas adelante en la clase main())

Código 9: Clase Aro1: funcion mover() y cambiar()

```
global velx
velx=Vector(-2,0)
def mover(self):
    self.pos = sumar(self.pos, ponderar(2, velx))
    if self.pos.x < -300:
        self.c = True

def cambiar(self):
    if self.c == True:
        return True
```

A diferencia de la clase anterior, Aro2 recibe otros parámetros como la posición, el ancho y el alto del aro1 respectivamente para que así quede una elipse completa que simule correctamente un aro. La función figura() es igual a la anterior salvo que este aro se crea con los ángulos complementarios a los anteriores, y la función mover() es igual a la anterior excepto que no tiene la condición para cambiar la variable çza que esta clase no tiene esa variable.

Código 10: Clase Aro2

```
global velx
velx=Vector(-2,0)

class Aro2(Figura):
    def ___init___(self, pos=Vector(0,0),c=0,v=0,rgb=(1.0,1.0,1.0)):
        self.a=c
        self.b=v
        super().___init___(pos,rgb)
    def figura(self):
        glPointSize(5)
        glBegin(GL_POINTS)
        glColor3f(0 , 0, 1)
        i=pi
        while (i<2*pi):
            x=self.a*cos(i)
```

```
            y=self.b*sin(i)
            glVertex2f(x,y)
            i=i+0.001
        glEnd()
    def mover(self):
        self.pos=sumar(self.pos,ponderar(2, velx))
```

Lo siguiente fue crear la clase Fases() la cual consiste solo de dos funciones, pasotoc() y pasolim(), las cuales indican si la pelota atravesó los aros tocando los bordes o no para así luego asignar el puntaje correspondiente.

Código 11: Clase Fases

```
class Fases:
    def pasotoc(self,pospe=Vector(0,0),posar=Vector(0,0),anchoar=0,altoar=0):
        if posar.x-anchoar<pospe.x<posar.x+anchoar and posar.y-altoar<pospe.y<posar.y:
            return True
    def pasolim(self,pospe=Vector(0,0),posar=Vector(0,0),anchoar=0,altoar=0):
        if posar.x-anchoar<pospe.x-30 and pospe.x+30<posar.x+anchoar and posar.y-altoar<
     pospe.y<posar.y:
            return True
```

Para poder mostrar los objetos en la pantalla se creo la clase Vista() que logra dibujar gracias a métodos definidos en la clase CC3501Utils, siempre y cuando lo que se quiera dibujar sea una Figura.

Código 12: Clase Vista

```
class Vista:
    def dibujar(self, pjs):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        for p in pjs:
            if p!=None:
                p.dibujar()
```

Además se añadió una característica extra la cual al momento de perder muestra un texto con el puntaje obtenido, el máximo puntaje que se a obtenido desde que se abrió el juego (gracias a una variable global que se explica en la siguiente clase) y además da la opción de repetir el juego presionando la ç.º de salir presionando la "q", todo esto se crea en la clase Text() la cual ocupa herramientas de pygame para crear la ventana e imprimir en texto en la pantalla.

Código 13: Clase Text

```
class Text:
    def ___init___(self, FontName=None, FontSize=100):
        pygame.font.init()
        self.font = pygame.font.Font(FontName, FontSize)
        self.size = FontSize
```

```
6
7   def render(self, surface, text, color, pos):
8       x, y = pos
9       for i in text.split("\r"):
10          surface.blit(self.font.render(i, 1, color), (x, y))
11          y += self.size
12  def texto(self,puntaje,bestpuntaje):
13      pygame.init()
14      white = (255, 255, 255)
15      size = width, height = 1200, 600
16      screen = pygame.display.set_mode(size)
17      pygame.display.set_caption("Flappy Dunk Chanta")
18      color = (255, 0, 0)
19      text = Text()
20      seguir=True
21      fuente = pygame.font.Font(None, 200)
22      gameover = fuente.render("GAME OVER",True, (0,0,0))
23      fuente2 = pygame.font.Font(None, 50)
24      continuar = fuente2.render("Para reintentar presione C y para salir presione Q", True,
    (0,0,0))
25      fuente3=pygame.font.Font(None,50)
26      mejor=fuente3.render("Mejor puntaje: "+str(bestpuntaje),True,(0,0,0))
27      while seguir:
28          for event in pygame.event.get():
29              if event.type == pygame.QUIT:
30                  exit()
31              if event.type == pygame.KEYDOWN:
32                  if event.key == pygame.K_q:
33                      exit()
34              if event.type == pygame.KEYDOWN:
35                  if event.key == pygame.K_c:
36                      seguir=False
37          screen.fill(color)
38          screen.blit(gameover, [200, 50])
39          screen.blit(continuar, [200, 500])
40          screen.blit(mejor,[450,350])
41          text.render(screen,'Puntuación: '+str(puntaje), white, (350, 250))
42          pygame.display.flip()
```

La ultima clase creada que es la que logra que todo funcione como corresponde es la clase Main(), en esta clase se crea todo lo necesario para el juego y en el orden correspondiente, lo primero de todo es definir una variable global 'bestpuntaje' la cual servira para comparar los puntajes obtenidos e ir mostrando en pantalla el mejor puntaje, luego se inicializa pygame para poder crear la ventan del tamaño y con el titulo deseado, una vez creada la ventana se llaman a todas las otras clases, ya sea Fases(), Vista(), Text(), Fondo(), se crea el sol y la pelota en la posición deseada y se crean 3 nubes y 3 aros, solo se necesitan 3 ya que siempre

van a haber dos mostrándose en pantalla mientras que el tercer aro o la tercera nube se cambia de un lado de la pantalla hacia al otro, es decir, al momento de salir de la pantalla alguno de los dos se vuelve a crear uno al otro lado fuera de la pantalla, y así cuando ya aparezca en pantalla otro desaparecerá y hará el mismo proceso. Luego se crea un lista vacía donde se van a insertar los elementos recién creado en el siguiente orden: fondo, sol, nube1, nube2, nube3, aro1, aro12, aro13, pelota, aro2, aro22 y aro23, es en ese orden ya que al momento de dibujar se dibujara en orden y los objetos que se dibujan después se superponen a los que ya están dibujados.

Código 14: Main parte inicial

```
bestpuntaje=0
def main() -> object:
    global bestpuntaje
    pygame.init()
    ancho = 1200
    alto = 600
    init(ancho, alto, "Flappy Dunk Chanta")
    fases=Fases()
    vista=Vista()
    text=Text()
    fondo = Fondo()
    sol=Sol(Vector(0,600))
    pelota=Pelota(Vector(150,300))
    nube1=Nube(Vector(100,randint(50,450)))
    nube2=Nube(Vector(600,randint(50,450)))
    nube3 =Nube(Vector(1100, randint(50, 450)))
    aro1=Aro1(Vector(400,randint(200,400)))
    aro2=Aro2(Vector(400,aro1.pos.y),aro1.a,aro1.b)
    aro12=Aro1(Vector(950,randint(200,400)))
    aro22=Aro2(Vector(950,aro12.pos.y),aro12.a,aro12.b)
    aro13=Aro1(Vector(1500,randint(200,400)))
    aro23=Aro2(Vector(1500,aro13.pos.y),aro13.a,aro13.b)
    pjs=[]
    pjs.append(fondo)
    pjs.append(sol)
    pjs.append(nube2)
    pjs.append(nube1)
    pjs.append(nube3)
    pjs.append(aro1)
    pjs.append(aro12)
    pjs.append(aro13)
    pjs.append(pelota)
    pjs.append(aro2)
    pjs.append(aro22)
    pjs.append(aro23)
```

Posterior a esto se carga un sonido para cuando la pelota pase limpiamente suene como una canasta de basquetbol y por ultimo se definen ciertas variables que serán de mucha ayuda, como el puntaje inicial, el puntaje bonus, un delta de tiempo 'dt' y tres variables boolean, 'hecho' (para entrar al while principal), 'esta' (con este se logra saber si la pelota paso por un aro) y 'empezar' (para salir de la escena inicial que se muestra y empezar el juego)

Código 15: Main continuación

```
dt=0
puntaje=0
hecho=True
esta=True
empezar=False
pygame.mixer.music.load("basket.mp3")
bonus=2
```

Con la ayuda de un while se muestra la escena inicial en la que se pueden apreciar todos los elementos del juego, la pelota, las nubes, dos aros, el fondo y el sol, la pelota se encontrara estática en la mitad de la pantalla hasta que el jugador presione la barra espaciadora, en ese momento se saldrá del while y continuara con el resto, además si el jugador quiere salir del juego puede presionar la 'x' de la ventana o presionar 'q' en cualquier momento.

Código 16: Main escena inicial

```
while not empezar:
    for evento in pygame.event.get():
        if evento.type == QUIT:
            exit()
        if evento.type==KEYDOWN:
            if evento.key==K_q:
                exit()
            if evento.key==K_SPACE:
                empezar=True
                pelota.mover(dt)

    vista.dibujar(pjs)
    pygame.display.flip()
```

Una vez presionada la barra espaciadora comienza el juego, para eso entra a otro while en el que al comienzo se crea una variable 'posición' con un valor aleatorio entre 100 y 450 que servirá para posicionar los nuevos aros cuando se creen, posterior a eso se comienzan a mover todos lo elementos que lo tengan que hacer, estos son los aros y las nubes horizontalmente y la pelota verticalmente. Posterior a eso entra muchos 'if' ya sea para volver el 'dt' 0 al presionar la barra espaciadora, analizar si la pelota toca el techo o el suelo con las funciones mencionadas anteriormente y además tiene dos para saber si la pelota se paso por un aro, esto quiere decir que paso por arriba o por abajo sin siquiera tocar el aro, esto se puede

lograr ya que al momento de ir avanzo la variable 'esta' tiene un valor de True, pero al momento de atravesar un aro esta cambia a False, entonces en los if se analiza si al momento de estar en el borde del aro la variable sigue siendo True significa que no atravesó el aro por lo que terminar el juego, en cambio si es False es porque si lo atravesó y cambia el valor de la variable a True nuevamente (este cambio de la variable también permite que al atravesar el aro el puntaje sume solamente una vez), este procedimiento se hace para los tres aros dependiendo cual es el que le tocaria atravesar a la pelota.

Código 17: Main primeros ifs

```
    for evento in pygame.event.get():
        if evento.type == QUIT:
            exit()
        if evento.type==KEYDOWN:
            if evento.key==K_q:
                exit()
            if evento.key==K_SPACE:
                dt=0
    if pelota.chocandosuelo() or pelota.chocandotecho():
        hecho=False
    if pelota.pos.x-(aro1.pos.x+aro1.a)>0:
        esta=True
    if pelota.pos.x - (aro12.pos.x + aro12.a) > 0:
        esta = True
    if pelota.pos.x - (aro13.pos.x + aro13.a) > 0:
        esta = True
    if -5<pelota.pos.x-(aro1.pos.x+aro1.a)<0 and esta:
        hecho=False
    if -5<pelota.pos.x - (aro12.pos.x + aro12.a) < 0 and esta:
        hecho = False
    if -5<pelota.pos.x - (aro13.pos.x + aro13.a) < 0 and esta:
        hecho = False
```

Luego de esos ifs, vienes aun mas ifs los cuales ocupan las funciones de la clases fases() para saber si la pelota atravesó el aro tocando los bordes o no, si la pelota atraviesa el aro limpiamente y además la pelota lo atravesó de arriba a abajo (esto se comprueba comparando la posición anterior de la pelota con la actual) el nuevo puntaje pasa a ser puntaje anterior mas el bonus, al bonus se le suma 1, la variable 'esta' pasa a ser False indicando que ya atravesó el aro, se imprime el puntaje en la consola, las posiciones en la lista donde se encontraba el aro (la parte de arriba y la de abajo) pasan a ser None para que así no los dibujen y simulen el efecto de desparecer y por ultimo se reproduce el sonido de una canasta de basquetbol. Y si la pelota lo atraviesa de abajo hacia arriba el juego termina. Este procedimiento se repite con los 3 aros dependiendo de a cual le toque atravesar la pelota. En cambio si la pelota atravesó tocando el aro de arriba hacia abajo, el nuevo puntaje es el puntaje anterior mas 1, el bonus vuelve a ser 2, el valor de la variable 'esta' cambia a False, se imprime el puntaje en la consola y por ultimo las posiciones de la lista donde se

encontraba el aro pasan a ser None para que desaparezca el aro. Y si la pelota lo atraviesa de abajo hacia arriba el juego termina y al igual que lo anterior este procedimiento se realiza en los tres aros.

Código 18: Main asignación de puntaje

```
    if fases.pasolim(pelota.pos,aro1.pos,aro1.a,aro1.b) and esta:
        if posant-pelota.pos.y>0:
            puntaje=puntaje+bonus
            bonus=bonus+1
            esta=False
            pygame.mixer.music.play()
            print(str(puntaje))
            pjs[5]=None
            pjs[9]=None
        else:
            hecho=False
    if fases.pasolim(pelota.pos,aro12.pos,aro12.a,aro12.b) and esta:
        if posant-pelota.pos.y>0:
            puntaje=puntaje+bonus
            bonus=bonus+1
            esta=False
            pygame.mixer.music.play()
            print(str(puntaje))
            pjs[6]=None
            pjs[10]=None
        else:
            hecho=False
    if fases.pasolim(pelota.pos,aro13.pos,aro13.a,aro13.b) and esta:
        if posant-pelota.pos.y>0:
            puntaje=puntaje+bonus
            bonus=bonus+1
            esta=False
            pygame.mixer.music.play()
            print(str(puntaje))
            pjs[7]=None
            pjs[11]=None
        else:
            hecho=False
    if fases.pasotoc(pelota.pos,aro1.pos,aro1.a,aro1.b) and esta :
        if posant-pelota.pos.y>0:
            puntaje=puntaje++1
            esta=False
            print(str(puntaje))
            bonus=2
            pjs[5]=None
            pjs[9]=None
```

```
42          else:
43              hecho=False
44      if fases.pasotoc(pelota.pos, aro12.pos, aro12.a,aro12.b) and esta:
45          if posant-pelota.pos.y>0:
46              puntaje=puntaje+1
47              esta=False
48              print(str(puntaje))
49              bonus=2
50              pjs[6]=None
51              pjs[10]=None
52          else:
53              hecho=False
54      if fases.pasotoc(pelota.pos, aro13.pos, aro13.a,aro13.b) and esta:
55          if posant-pelota.pos.y>0:
56              puntaje=puntaje+1
57              esta=False
58              print(str(puntaje))
59              bonus=2
60              pjs[7]=None
61              pjs[11]=None
62          else:
63              hecho=False
```

Por ultimo se analiza para cada aro si ya esta fuera de la pantalla, si es que lo esta, se crea un aro nuevo con dimensiones y posición diferente en el otro lado de la pantalla, luego se dibuja cada objeto del juego con sus nuevas posiciones, se actualiza la pantalla del juego, se guarda la posición de la pelota para después poder compararla, se aumenta en 0.1 el dt y se define un tiempo de espera para colocar en juego en 80fps. Al momento de perder se compara el puntaje obtenido con el mejor puntaje hasta el momento, si es mayor el nuevo mejor puntaje pasa a ser el puntaje actual y si no no se hace ningún cambio y se muestra el texto de GAME OVER con ambos puntajes en la pantalla y esperando a que el jugador seleccione si desea continuar jugando o quiere salir y para finalizar se llama a la función main() fuera de la función para que así al correr la clase main el juego se ejecute.

Código 19: Main final

```
1   if aro1.cambiar():
2       aro1 = Aro1(Vector(1350, posicion))
3       aro2 = Aro2(aro1.pos,aro1.a,aro1.b)
4       pjs[5]=aro1
5       pjs[9]=aro2
6   if aro12.cambiar():
7       aro12 = Aro1(Vector(1350, posicion))
8       aro22 = Aro2(aro12.pos,aro12.a,aro12.b)
9       pjs[6]=aro12
10      pjs[10]=aro22
11  if aro13.cambiar():
```

```
12          aro13 = Aro1(Vector(1350, posicion))
13          aro23 = Aro2(aro13.pos,aro13.a,aro13.b)
14          pjs[7]=aro13
15          pjs[11]=aro23
16
17      vista.dibujar(pjs)
18      pygame.display.flip()
19      posant=pelota.pos.y
20      dt=dt+0.1
21      pygame.time.wait(int(1000 / 80))
22      while (not hecho):
23          if bestpuntaje<puntaje:
24              bestpuntaje=puntaje
25          text.texto(puntaje,bestpuntaje)
26          main()
27  main()
```

4. Discusión sobre las dificultades

En un comienzo una de las mayores dificultades fue familiarizarse con todo lo que tenia que ver con Pygame y OpenGl, ya que sabia muy poco al respecto, para lo cual tuve que investigar mucho y ver muchos vídeos de como funcionaban hasta que supe como ocuparlos, también en un comienzo pensar en como abordar el problema fue un desafio ya que no sabia porque comenzar y como comenzar, luego que logre aclararme aparecieron dos grandes dificultades que no logre solucionar, la mas importante que es algo que era necesario hacer fue no poder mostrar el puntaje en la pantalla durante el juego, al comienzo utilizando pygame se generaba un conflicto puesto que opengl priorizaba sus dibujos y no los compatibilizaba con los de pygame por lo que arrojaba un error, luego intente con una función de opengl para mostrar caracteres en la pantalla lo cual tampoco resulto por otro error que no logre solucionar. El otro problema que se genero fue en algo extra que se implemento, ya que la idea era que al atravesar limpiamente por un aro en el juego sonara una canasta de basquetbol que es algo que hace, pero al momento de atravesar limpiamente por dos aros consecutivos en el segundo aro no se produce el sonido, en cambio si fueran 3 si se produciría en el tercero, esto es algo que nunca logre saber porque ocurria por lo que no pude arreglarlo.

5. Aprendizajes

Se logro obtener un gran aprendizaje para crear diferentes figuras con OpenGl además de como crear pantallas en python, dibujar en ellas e interactuar con el usuario que es algo muy útil al momento de crear cualquier otro tipo de juego o alguna aplicación para el uso de un tercero. Logre manejar bien la crear y dibujar objetos en una pantalla de python además como manejar el espacio tratando de ocupar lo menos posible pero manteniendo la funcionalidad del juego y lo atractivo, también luego de pensar un poco logre manejar bien la mecánica del juego y como solucionar los problemas que iban apareciendo a medida que avanzaba lo que es algo muy importante ya que me ayuda para saber actuar mejor en el futuro en alguna decision importante.