UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE INGENIERÍA Y CIENCIAS
CC3501 Modelación y Computación Gráfica para Ingernieros

TAREA 2B

PP - Block

Daniela Campos
danielacamposfischer@gmail.com

Profesora:
Nancy Hitschfeld K.

Auxiliares:
Pablo Pizarro R.
Pablo Polanco

Fecha:
14 de Mayo del 2017

Índice

1 Descripción del Problema

Se debe diseñar e implementar un juego similar al conocido BB - Tan, donde el usuario dispara pelotas con el objetivo de destruir bloques que van bajando por turnos, si es que algún bloque toca la base, el jugador pierde.

La pantalla se divide en 9 filas y 7 columnas, donde la fila superior está vacía y si la inferior no lo está el jugador pierde. Cada fila contiene bloques y PowerUps, y por cada celda se dan las siguientes probabilidades

1. 50 % de ser bloque
2. 25 % de ser Power Up
3. 25 % de ser vacío

Inicialmente el usuario comienza con una pelota y está centrado en el borde inferior de la ventana. La puntuación comienza en uno y aparece la primera fila de bloques por destruir. El jugador gracias a una guía de lanzamiento determina el ángulo con el cual lanzará las pelotas. Cuando cae la primera pelota, el jugador se cambia a esa posición y los bloques y Power Ups bajan un nivel. Para hacer un lanzamiento el usuario debe hacer click en con le mouse.

1.1 Objetos del Juego

A continuación se expondrán los objetos que deben estar presentes en el jugo y sus propiedades.

a) **Pelota:** Será lanzada por el usuario con el objetivo de destruir bloques o obterner Power Ups. Puede tener diferentes tamaños y colores, y dependiendo de estos tendrá diferentes comportamientos. En el caso de este juego, la pelota verde es mas grande y posee mayor poder destructivo y la pelota azul es mas pequeña y duplica la puntuación obtenida por destruir bloques.

b) **Bloque:** El objetivo de juego es destruir los bloques y para lograrlo es necesario golpearlo tantas veces como resistencia tenga, esta es igual a la puntuación al momento de crearlo y aparece en el centro del bloque. En caso de que la puntuación sea divisible por 10, la resistencia del bloque se duplica.

c) **Power Up:** Si entran en contacto con una pelota generan algún efecto, existen diferentes tipos. [1]

[1] El detalle de lo que hace cada Power Up se encontrará en Descripción de la Solución

El objetivo de la Tarea 2B es implementar un programa utilizando Python, Pygame o PyOpenGL, que sea una versión simplificada del juego, la cual considere física de choques, lanzamiento de pelotas, generación y destrucción de bloques y Power Ups. Además los bloques deberán cambiar de color de acuerdo a su resistencia y deberá existir una guía de lanzamiento de las pelotas.El juego deberá tener un menú principal donde el usuario pueda elegir la pelota con la que desea jugar.

2 Descripción de la Solución

El juego fue diseñado utilizando MVC o Modelo Vista Controlador, y a continuación se expondrán las clases y métodos presentes en cada parte.

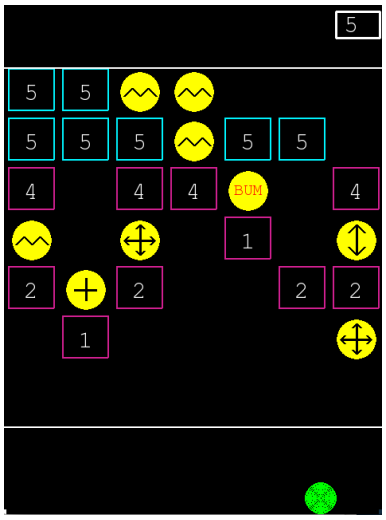

Figura 1: Ejemplo de el juego realizado

2.1 Modelo

2.1.1 Bloque

Se creó la clase Bloque, la cual contiene dos métodos. [2]

1. **Constructor:** Toma como argumentos la posición del bloque, su resistencia y la columna y fila donde se encuentra este. Además, le otorga al bloque una variable booleana llamada vida que es True si es que el bloque tiene resistencia mayor a cero. Dependiendo de su resistencia, le otorga diferentes colores.
2. **draw:** Toma como argumento la ventana. Dibuja el bloque con la función draw.lines de Pygame y agrega el número de resistencia que tiene el bloque en su centro.

2.1.2 Power Up

Los Power Ups presentes en el juego son los cuatro requeridos y otros 3 extras. A continuación se expondrán en detalle los diferentes tipos.[3]

[2] El código de la clase Bloque puede ser encontrado en el Anexo 1
[3] El código de la clase PowerUp se encuentra en el Anexo 2

1) **Extra-Ball:** Aumenta en uno la cantidad de pelotas del jugador y desaparece al contacto.

Figura 2: Extra-Ball

2) **Vertical-Laser:** Disminuye en uno la resistencia de los bloques que se encuentran en su misma columna. Desparece al contacto.

Figura 3: Vertical-Laser

3) **Horizontal-Laser:** Disminuye en uno la resistencia de los bloques que se encuntran en su misma fila. Desaparece al contacto.

Figura 4: Horizontal-Laser

4) **Change-Path:** Cambia el ángulo de movimiento de la pelota de manera aleatoria.

Figura 5: Change-Path

5) **Horizontal & Vetical Laser:** Disminuye en uno la resistencia de los bloques que se encuentran en su misma fila o en su misma columna. Desaparece al contacto.

Figura 6: Horizontal&Vertical-Laser

6) **Explosion:** Destruye los bloques que se encuentra alrededor de este. Desaparece al contacto.

Figura 7: Horizontal-Laser

7) **DoblePuntuacion:** Duplica la puntuación que se tiene al comento de que la pelota lo toque. Desaparece al contacto.

Figura 8: Horizontal-Laser

Posee dos métodos, los cuales serán explicados a continuación:

1. **Constructor:** Toma como argumento la posición, la columa y la fila del Power Up. Además de estos, le otorga a la pelota una variable booleana vida que determina si debe seguir siendo dibujado y una variable choice, la cual es elegida de manera aleatoria y determina que tipo de Power Up es.
2. **draw:** Toma como argumento la ventana. Dibuja el Power Up, lel dibujo correspondiente a cada uno es diferente y depende de la variable **choice**.

2.1.3 Pelota

Se creó la clase Ball que contiene cinco métodos.[4]

1. **Constructor:** Toma como argumento la posición de la pelota(tanto en el eje x como el y), la velocidad de esta(también en ambos ejes) y el radio. Además de sus argumentos, le otorga a la pelota tiempo y un boolean llamado viva que permite saber si se pasó del el borde inferior o no. Dependiendo del radio que esta tenga le otorga un color diferente.
2. **update:** No toma ningún argumento, mas que la pelota misma. Actualiza la posición de esta dependiendo de su velocidad y del tiempo transcurrido.
3. **draw:** Toma como argumento la ventana y dibuja la pelota utilizando la función draw.circle de pygame.
4. **crash:** Toma como argumento la ventana y un bloque y comprueba si es que la pelota se intersecta con el bloque gracias a la función **intesects** de la librería **Shapely**. De existir dicha intersección comprueba con que lado del bloque es y actualiza la velocidad según corresponda. Además, actualiza la resistencia del bloque.

[4]El código de la clase Ball puede ser encontrado en el Anexo 3

5. **crashPowerUp:** Toma como argumento la ventana y un PowerUp y comprueba si es que la pelota intersecta con el Power Up gracias a la función **intersects** de la librería **Shapely**. Si existe dicha intersección ejecuta lo que debería hacer el PowerUp.

2.2 Vista

2.2.1 Window

Se creó la clase Window, la cual se encarga de dibujar sobre la ventana. Posee 3 métodos.[5]

1. **Constructor:** Toma como argumento la ventana, una lista de pelotas, una de bloques y otra de Power Ups. Aparte de los argumento se otorga a si misma un color, la posición del jugador, la puntuación, la cantidad de pelotas que hay y donde está el mouse al momento de ser lanzada la pelota.
2. **clean:** No toma argumentos. Limpia la ventana llenandola de su color inicial.
3. **draw:** No toma argumentos. Dibuja las pelotas, los bloques y los Power Ups. Además Muestra las puntuación, dibuja los límites donde pueden existir bloques y Power Ups y mustra GAME OVER si es que los bloques atraviezan el límite inferior.

2.3 Controlador

El método main crea un controlador y lo actualiza hasta que se cierre la ventana.

2.3.1 Controller:

Se creó la clase Controller que recibe lo que hace el jugador y lo interpreta para que funcione el juego. Posee 2 métodos, los cuales serán explicados a continuación.[6]

1. **Constructor:** Toma como argumento un string el cual determina con que pelota se está jugando. Determina las dimensiones de la pantalla, crea la ventana y los objetos presentes en el juego.
2. **update:** No toma argumentos. Actualiza los valores de los objetos del juego dependiendo de lo que hace usuario.Chequea si es que la primera pelota sigue "con vida" es decir si no ha tocado el suelo y comprueba si es que las pelotas chocaron con bloques o PowerUps.

[5]El código de la clase Window se encuentra en el Anexo 4
[6]El código de la clase Controller se encuentra en el Anexo 5

2.4 Menu

El archivo Menu.py contienen la función game_intro que permite crear el menú del juego. Esta dibuja en la pantalla las pelotas con las que se puede jugar, si es que el usuario hace click sobre alguna de estas, retorna una variable de tipo string que le permite al controlador saber con que pelota está jugando.

Figura 9: Menu Principal del juego

3 Dificultades Encontradas

La mayor dificultad a la hora de realizar el juego fue lograr que los choques funcionaran bien, dado que, a pesar de ya haber deducido las ecuaciones necesarias para que los choques fueran realistas, la interacción entre los objetos era muy dificil de modelar. Esto se solucionó de manera parcial, utilizando la librería *Shapely*, con la cual se pueden modelar figuras y comprobar si estas se intersectan.

Otro problema que se presentó a la hora de realizar el juego fue la visualización, ya que en una primera instancia, la ventana mostraba los objetos de manera intermitente, esto se debía a la tasa de refresco de la pantalla. Esto se solucionó disminyendo la tasa de refresco a 1000/30 milisegundos, esta cantidad de tiempo fue alcanzada tras averiguar cual era la óptima.

Otra dificultad importante fue la cantidad de lineas de código con las que se estaba trabajando, ya que era muy dificil encontrar un error dado que se debía revisar muchas lineas de código. Se resolvió este problema, implementando el patrón MVC y separando en diferentes archivos las clases, ya que en un inicio todo se encontraba en el mismo archivo.

4 Conclusion

Tras utilizar el juego creado, se puede concluir que Pygame es una herramienta muy útil a la hora de crear juegos que presentan movimientos, ya que el comportamiento de los elementos del juego es bastante realista. También es importante señalar que utilizár el patrón de arquitectura MVC es sustancial para la computación gráfica.

Por supuesto existen cosas en el juego que pueden ser mejoradas, como la interacción de las pelotas entre ellas y con los bloques, pero estas podrían ser resueltas indagando más en las propiedades de pygame y obteniendo así mas experiencia.

5 Anexos

1. **Anexo 1** :

```
import pygame

class Bloque:
    pygame.font.init()

    #Constructor Bloque
    def __init__(self, x, y, vidas,columna, fila):
        self.x = x
        self.y = y
        self.fila = fila
        self.columna = columna
        if(vidas%10 ==0):
            vidas += vidas
        self.vidas = vidas
        self.vida = True
        if(self.vidas >= 20):
            self.color = (255,255,0)
        if((self.vidas >=10 ) & (self.vidas < 20)):
            self.color = (255, 69, 0)
        if((self.vidas >= 5) & (self.vidas < 10)):
            self.color =  (0,245,255)
        if(self.vidas < 5):
            self.color = (208,32,144)

    #Dibuja el bloque y escribe en el su resistencia
    def draw(self, surface):
        pygame.draw.lines(surface,self.color, False, [(self.x, self.y)
                ,(self.x + 57, self.y),(self.x + 57, self.y + 52),
                        (self.x,self.y + 52), (self.x, self.y) ], 2)
        myfont = pygame.font.SysFont("monospace", 30)
        label = myfont.render(str(self.vidas), 1, (255, 255, 255))
        surface.blit(label, (self.x + 20, self.y + 15))
```

2. **Anexo 2** :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from random import randint

import pygame
pygame.font.init()


class PowerUp:
    def __init__(self, x, y, columna, fila):
        self.x = x
        self.y = y
        self.columna = columna
        self.fila = fila
        self.choice = randint(1,7)
        self.vida = True

    #Dibuja los PowerUps
    def draw(self, screen):

        if(self.choice == 1):#ExtraBall
            pygame.draw.circle(screen, (255, 255, 0), (self.x + 30,
                self.y + 30), 25)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                30, self.y + 15), (self.x + 30, self.y + 43)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                15, self.y + 30), (self.x + 43, self.y + 30)], 3)

        if(self.choice == 2):#VerticalLaser
            pygame.draw.circle(screen, (255, 255, 0), (self.x + 30,
                self.y + 30), 25)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                30, self.y + 10), (self.x + 30, self.y + 48)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                20, self.y + 20), (self.x + 30, self.y + 10)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                40, self.y + 20), (self.x + 30, self.y + 10)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                20, self.y + 38), (self.x + 30, self.y + 48)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                40, self.y + 38), (self.x + 30, self.y + 48)], 3)
```

```
        if(self.choice == 3):#HorizontalLaser
            pygame.draw.circle(screen, (255, 255, 0), (self.x + 30,
                self.y + 30), 25)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                10, self.y + 30), (self.x + 48, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                20, self.y + 40), (self.x + 10, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                20, self.y + 20), (self.x + 10, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                38, self.y + 40), (self.x + 48, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                38, self.y + 20), (self.x + 48, self.y + 30)], 3)

        if(self.choice == 4):#ChangePath
            pygame.draw.circle(screen, (255, 255, 0), (self.x + 30,
                self.y + 30), 25)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                10, self.y + 35), (self.x + 20, self.y + 25)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                20, self.y + 25), (self.x + 30, self.y + 35)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                30, self.y + 35), (self.x + 40, self.y + 25) ], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                40, self.y + 25), (self.x + 50, self.y + 35)], 3)

        if (self.choice == 5):#Horizontal&VerticalLaser BONUS
            pygame.draw.circle(screen, (255, 255, 0), (self.x + 30,
                self.y + 30), 25)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                30, self.y + 10), (self.x + 30, self.y + 48)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                25, self.y + 15), (self.x + 30, self.y + 10)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                35, self.y + 15), (self.x + 30, self.y + 10)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                25, self.y + 43), (self.x + 30, self.y + 48)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                35, self.y + 43), (self.x + 30, self.y + 48)], 3)

            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                10, self.y + 30), (self.x + 48, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
```

```
                15, self.y + 35), (self.x + 10, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                15, self.y + 25), (self.x + 10, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                43, self.y + 35), (self.x + 48, self.y + 30)], 3)
            pygame.draw.lines(screen, (0, 0, 0), False, [(self.x +
                43, self.y + 25), (self.x + 48, self.y + 30)], 3)

        if (self.choice == 6):  #ExplosionBonus BONUS
            pygame.draw.circle(screen, (255, 255, 0), (self.x + 30,
                self.y + 30), 25)
            myfont = pygame.font.SysFont("monospace", 20)
            label = myfont.render("BUM", 1, (255,0, 0))
            screen.blit(label, (self.x + 12, self.y + 18))

        if (self.choice == 7): #DoublePuntuation BONUS
            pygame.draw.circle(screen,(0,245,255), (self.x + 30,
                self.y + 30), 25)
            myfont = pygame.font.SysFont("monospace", 30)
            label = myfont.render("$", 1, (0, 0, 0))
            screen.blit(label, (self.x + 21, self.y + 15))
```

3. **Anexo 3** :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import pygame
from random import randint
from math import ceil
from shapely.geometry import *
from shapely.geometry.geo import *

class Ball:

    #Constructor Pelota
    def __init__(self, initial_x, initial_y, speed_x, speed_y, radio):
        self.x = initial_x
        self.y = initial_y
        self.speed_x = speed_x
        self.speed_y = speed_y
        self.t = 0
        self.viva = True
        self.radio = radio
        if (self.radio == 20):
            self.color = (0,255,0)
        if (self.radio == 15):
            self.color = (0,0,255)

    #Updetea la posicion de la pelota
    def update(self):
        self.x = self.x + self.speed_x * self.t
        self.y = self.y - self.speed_y * self.t
        self.t += 0.1

    #Dibuja la pelota
    def draw(self,screen):
        pygame.draw.circle(screen, self.color, (int(self.x),
                int(ceil(self.y))) ,self.radio, self.radio)

    #Chequea si es que se choco con un bloque
    def crash(self,screen,bloque):
        b1 = Point(self.x,self.y).buffer(self.radio)
        p1 = box(bloque.x, bloque.y, bloque.x + 57,
                bloque.y+52, ccw=True)
        if p1.intersects(b1):
```

```
            if((self.y >= bloque.y + 52)):#choque por abajo
                self.speed_y = - self.speed_y
                bloque.vidas -=1
                if(self.radio == 20):
                    bloque.vidas -=1

            if((self.x <= bloque.x)): #choque por el lado izq
                self.speed_x = -self.speed_x
                bloque.vidas -= 1
                if (self.radio == 20):
                    bloque.vidas -= 1

            if(self.y <= bloque.y):
                self.speed_y = -self.speed_y
                bloque.vidas -= 1
                if (self.radio == 20):
                    bloque.vidas -= 1

            if(self.x >= bloque.x + 57):
                self.speed_x = -self.speed_x
                bloque.vidas -= 1
                if (self.radio == 20):
                    bloque.vidas -= 1

    #Chequea si se choco con un PowerUp
    def crashPowerUp(self,screen,PowerUp):
        b1 = Point(self.x,self.y).buffer(self.radio)
        b2 = Point(PowerUp.x + 30,PowerUp.y + 30).buffer(25)
        if(b1.intersects(b2)):

            if (PowerUp.choice == 1):
                PowerUp.vida = False
                screen.cant += 1

            if (PowerUp.choice == 2):
                PowerUp.vida = False
                for bloque in screen.bloques:
                    if (bloque.columna == PowerUp.columna):
                        bloque.vidas -= 1

            if (PowerUp.choice == 3):
                PowerUp.vida = False
                for bloque in screen.bloques:
```

```
                if (bloque.fila == PowerUp.fila):
                    bloque.vidas -= 1

        if (PowerUp.choice == 4):
            a = randint(0, 1)
            b = randint(0, 1)
            if (a == 1):
                self.speed_x = -self.speed_x
            if (b == 1):
                self.speed_y = -self.speed_y

        if (PowerUp.choice == 5):
            PowerUp.vida = False
            for bloque in screen.bloques:
                if (bloque.columna == PowerUp.columna):
                    bloque.vidas -= 1
                if (bloque.fila == PowerUp.fila):
                    bloque.vidas -= 1

        if(PowerUp.choice == 6):
            PowerUp.vida = False
            for bloque in screen.bloques:
                if(bloque.fila == PowerUp.fila - 1) &
                    ((bloque.columna == PowerUp.columna - 1) |
                        (bloque.columna == PowerUp.columna) |
                            (bloque.columna == PowerUp.columna +
                                1)):
                    bloque.vidas = 0
                if(bloque.fila == PowerUp.fila) & ((bloque.columna
                    == PowerUp.columna - 1) |
                    (bloque.columna ==PowerUp.columna + 1)):
                    bloque.vidas = 0
                if (bloque.fila == PowerUp.fila + 1) &
                    ((bloque.columna == PowerUp.columna - 1) |
                        (bloque.columna == PowerUp.columna) |
                            (bloque.columna == PowerUp.columna
                                + 1)):
                    bloque.vidas = 0

        if(PowerUp.choice == 7):
            PowerUp.vida = False
            screen.puntuacion += screen.puntuacion
```

4. **Anexo 4** :

```
import pygame

class Window:
    pygame.font.init()

    #Constructor Ventana
    def __init__(self,screen,balls,bloques,PowerUps):
        self.balls = balls
        self.bloques = bloques
        self.PowerUps = PowerUps
        self.window = screen
        self.color = (0,0,0)
        self.player = 217
        self.puntuacion = 1
        self.cant = 0
        self.pointerx = -1
        self.pointery = -1

    #Limpia la ventana
    def clean(self):
        self.window.fill(self.color)

    #Dibuja objetos y otros en la ventana
    def draw(self):

        #Dibuja la guia de lanzamiento si es que la pelota se
                encuentra en el borde inferior
        if((self.pointerx != -1) & (self.pointery != -1)):
            pygame.draw.lines(self.window, (255, 255, 255),
                False, [(self.balls[0].x, self.balls[0].y),
                        (self.pointerx, self.pointery)],2)

        #Dibuja todas las pelotas con vida
        for ball in self.balls:
            if(ball.viva == True):
                ball.draw(self.window)

        #Dibuja los bloques con vida
        for bloque in self.bloques:
            if((bloque.vida == True) & (bloque.vidas > 0)):
                bloque.draw(self.window)
```

```
        #Dibuja los PowerUps con vida
        for PowerUp in self.PowerUps:
            if(PowerUp.vida == True):
                PowerUp.draw(self.window)

        #Muestra la puntuacion
        myfont = pygame.font.SysFont("monospace", 30)
        label = myfont.render(str(self.puntuacion), 1,
                (255, 255, 255))
        self.window.blit(label, (430, 11))
        pygame.draw.lines(self.window, (255, 255, 255),
                False, [(420,10), (420,43), (475,43),
                        (475,10), (420,10)], 3)

        #Dibuja los limites donde pueden haber bloques
        pygame.draw.lines(self.window, (255,255,255),
                False, [(0,530),(480,530)], 2)
        pygame.draw.lines(self.window, (255, 255, 255),
                False, [(0, 80), (480, 80)], 2)

        #Si es que los bloques atraviezan el limite inferior
                muestra GAMEOVER
        for bloque in self.bloques:
            if(bloque.y > 510):
                myfont = pygame.font.SysFont("monospace", 80)
                label = myfont.render("GAME OVER", 1,
                        (255, 255, 255))
                self.window.blit(label, (20, 250))

        pygame.display.flip()
```

5. **Anexo 5** :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import os
from random import randint
import pygame
from pygame.locals import *
import sys
import math
from Ball import Ball
from Window import Window
from Bloque import Bloque
from PowerUp import PowerUp
from centered_figure import*

#Controlador del juego
class Controller:
    def __init__(self,pelota):

        #Dependiendo de la pelota elegida en el menu
                principal determina con cual se esta jugando
        if(pelota == "green"):
            self.radio = 20
        if(pelota == "blue"):
            self.radio = 15

        #Determina las dimensiones de la pantalla
        self.height = 640
        self.width = 480
        self.ticks_counter = 0

        #Crea la ventana
        pygame.init()
        self.window = pygame.display.set_mode([self.width,
                self.height])
        pygame.display.set_caption('PP - Block')

        #Crea las listas que contendran los objetos del juego
        self.balls = []
        self.bloques = []
        self.PowerUps = []
```

```
        #Le entrega estos objetos a la ventana para dibujarlos
        self.window = Window(self.window, self.balls,
                self.bloques, self.PowerUps)

        #Crea la pelota inicial
        self.balls.append(Ball(240, 620, 0, 0, self.radio))


        #Crea los Bloques y PowerUps iniciales
        for i in range(-1,7):
            #for j in range(0,4):
            a = randint(1,4)
            if((a == 1) | (a == 2)):
                self.bloques.append(Bloque(76 + i*68,
                        81,self.window.puntuacion,i,0))
            if(a == 3):
                self.PowerUps.append(PowerUp(76 + i*68, 81, i, 0))

    #Updetea el juego dependiendo de lo que hace el jugador
    def update(self):

        #Le pide a la ventana que se limpie
        self.window.clean()

        #Si la primera pelota vuelve al suelo la inicializa denuevo
        if(self.balls[0].viva == False):
            self.balls[0].x = self.window.player
            self.balls[0].y = 620
            self.balls[0].speed_x = 0
            self.balls[0].speed_y = 0
            self.balls[0].viva = True

            #Esto evita que a medida que avance el juego las
                 pelotas tomen mas velocidad
            for ball in self.balls:
                ball.t = 0

            #Si es que el PowerUp Extra Ball fue seleccionado
                 agrega una pelota a la lista
            for i in range(0,self.window.cant):
                self.balls.append(Ball(self.window.player, 620,
                        0,0, self.radio))
            self.window.cant = 0
```

```
            #Una vez que la pelota toca el suelo todos los
                bloques y PowerUps bajan un nivel
            for bloque in self.bloques:
                bloque.y += 62
                bloque.fila +=1

            for P  in self.PowerUps:
                P.y += 62
                P.fila += 1

            #Rellena la primera fila
            for i in range(-1,7):
                d = randint(1,4)
                if ((d == 1) | (d == 2)):
                    self.bloques.append(Bloque(76 + i * 68, 81,
                        self.window.puntuacion, i, 0))
                if (d == 3):
                    self.PowerUps.append(PowerUp(76 + i*68,81,i,0))


        #Le pide a la ventana que se dibuje
        self.window.draw()


        self.window.pointerx = -1
        self.window.pointery = -1


        #Interpreta lo que el usuario quiere hacer
        for event in pygame.event.get():
            if event.type == QUIT:
                exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                mouse_pos = pygame.mouse.get_pos()
                final_x = mouse_pos[0]
                final_y = mouse_pos[1]
                for ball in self.balls:
                    #Actualiza la velocidad de las pelotas dependiendo
                        de donde cliqueo el usuario
                    op = abs(final_x - ball.x)
                    ady = abs(final_y - ball.y)
                    hip = math.sqrt(op ** 2 + ady ** 2)
                    vx = float(op) / hip
                    vy = float(ady) / hip
```

```
                if (final_x < ball.x):
                    vx = -vx
                #Si se cliquea bajo los 440 pixeles se aumenta la
                     velocidad en un 10%
                if(final_y >= 440):
                    ball.speed_x = vx*1.1
                    ball.speed_y = vy*1.1
                #Si se cliquea bajo los 240 pero sobre los 440
                     pixeles se aumenta la velocidad en un 20%
                if(final_y <440) & (final_y >= 240):
                    ball.speed_x = vx * 1.2
                    ball.speed_y = vy * 1.2
                #Si se cliquea bajo los 240 pixeles se aumenta la
                     velocidad en un 50%
                if(final_y <240):
                    ball.speed_x = vx * 1.5
                    ball.speed_y = vy * 1.5

            #Crea una guia de lanzamiento si la pelota se
                encuentra en el piso
            if (event.type == pygame.MOUSEMOTION) &
                 (self.balls[0].y > 600):
                mouse_pos = pygame.mouse.get_pos()
                self.window.pointerx= mouse_pos[0]
                self.window.pointery = mouse_pos[1]


        #Actualiza la velocidad de la pelota si esta choca
             con los bordes
        for ball in self.balls:
            if (ball.x <= 0):
                ball.speed_x = -ball.speed_x
            if (ball.x >= 435):
                ball.speed_x = -ball.speed_x
            if (ball.y <= 0):
                ball.speed_y = -ball.speed_y
            if (ball.y > 620):
                player = ball.x
                ball.viva = False

        #Chequea si las pelotas chocan con los bloques,
            si los destruyen aumenta el puntaje
        for bloque in self.bloques:
            for ball in self.balls:
```

```
                if(bloque.vida == True):
                    ball.crash(self.window, bloque)
                    if(bloque.vidas <= 0):
                        bloque.vida = False
                        self.window.puntuacion += 1
                        if (self.radio == 15):
                            self.window.puntuacion += 1

        #Chequea si las pelotas chocan con los PowerUps y
                si es que los PowerUps deben desaparecer dada la altura
        for P in self.PowerUps:
            if(P.y > 500):
                P.vida = False
            for ball in self.balls:
                if(P.vida == True):
                    ball.crashPowerUp(self.window, P)

        #Lanza las pelotas de manera continua
        for i in range(len(self.balls)):
            for j in range(i + 1):
                if(self.balls[j].viva == True):
                    self.balls[j].update()

        #Si la pelota toca el suelo, guarda la posicion donde
                lo hizo para que la siguiente partidad comience ahi
        if(self.balls[0].viva == False):
            self.window.player = self.balls[0].x
        self.ticks_counter += 1
```

6. **Anexo 6** :

```
#!/usr/bin/python

import pygame

#Crea la ventana que alberga el menu
gameDisplay = pygame.display.set_mode((480,640))
clock = pygame.time.Clock()

#Sirve para escribir en la ventana
def text_objects(text, font):
    textSurface = font.render(text, True, (255,255,255))
    return textSurface, textSurface.get_rect()

#Funcion que crea el menu
def game_intro():

    intro = True
    while intro:

        #Llena la ventana de negro
        gameDisplay.fill((0, 0, 0))

        #Define los tipos de textos
        largeText = pygame.font.Font('freesansbold.ttf', 50)
        smallText = pygame.font.Font('freesansbold.ttf', 15)


        TextSurf, TextRect = text_objects("Elija una Pelota:",
                 largeText)
        TextRect.center = ((480 / 2), (640 / 3))
        gameDisplay.blit(TextSurf, TextRect)

        #Dibuja las pelotas y escribe sus cualidades
        pygame.draw.circle(gameDisplay, (0, 255, 0),
                (125, 400), 20, 20)
        TextSurfg, TextRectg = text_objects("x2 destruccion",
                 smallText)
        TextRectg.center = ((125), (450))
        gameDisplay.blit(TextSurfg, TextRectg)

        pygame.draw.circle(gameDisplay, (0, 100, 225),
                (355, 400), 15, 15)
```

```
        TextSurfb, TextRectb = text_objects("x2 puntaje",
                smallText)
        TextRectb.center = ((355), (450))
        gameDisplay.blit(TextSurfb, TextRectb)

        #Recibe lo que hace el usuario
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()

            if event.type == pygame.MOUSEBUTTONDOWN:
                mouse_pos = pygame.mouse.get_pos()
                x = mouse_pos[0]
                y = mouse_pos[1]
                #Distingue que pelota eligio dependiendo
                        de donde apreto el usuario
                if((x >= 105) & (x <= 145) & (y >= 380) &
                        (y <= 420)):
                    return "green"
                elif ((x >= 340) & (x <= 370) & (y >= 385) &
                        (y <= 415)):
                    return "blue"

        pygame.display.update()
        clock.tick(15)
```