Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Modelación y computación gráfica para la ingenieria

Tarea 2B

PP - Block

Autor: Pablo Ignacio Jesús Arancibia Barahona
Profesor: Nancy Hitschfeld K.
Auxiliares: Pablo Pizarro R.
Pablo Polanco
Ayudantes: Joaquín T. Paris
Rodrigo E. Ramos T.
Sergio Leiva
Fecha de entrega: 14 de mayo de 2017
Santiago, Chile

Índice de Contenidos

Lista de Figuras

Lista de Tablas

1. Introducción

Con este informe se intenta explicar la solución e implementación del juego PP-BLOCK (clonico de BB-TAN), el cual consiste en el lanzamiento de una serie de bolas capaces de destruir bloques dependiendo de la resistencia de este. Si algún bloque llega a la base del juego, el jugador entra en GAME-OVER:

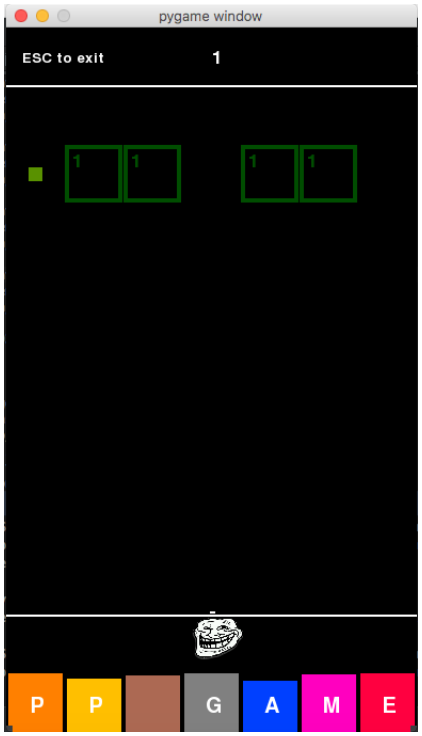

Figura 1.1: juego

2. Descripción del Juego

En el comienzo del juego, el jugador parte con una Pelota inicial. Al presionar la pantalla, la Pelota comienza una trayectoria recta con un angulo que el cursor genera en la posicion en la que se encontraba con respecto a la base del juego.
En caso de chocar con algun contorno de la pantalla, la Pelota rebota según su ángulo de incidencia.
En caso de chocar con algun bloque, rebota según el angulo de incidencia y ademas le resta uno a la resitencia del bloque.
Si la Pelota choca con un PowerUp, esta se comporta segun el tipo de PowerUp.
Por último, si la Pelota vuelve a llegar a la base del juego, todas las filas de bloques bajan una posicion, el jugador aumenta en 1 su puntaje y se genera una nueva fila, con bloques de resistencia igual al puntaje obtenido, siempre dejando un espacio vacio en la parte superior de la pantalla.
En caso, de que el angulo de incidencia sea igual a 0 o 180 grados, quedando rebotando continuamente de manera Horizontal, se crea un PowerUp que cambie el rumbo de esta.
Ademas, los powerUp una vez tocados, desaparecen de la pantalla dependiendo de casos especiales.

2.1. Elementos del juego

Como se explico anteriormente, el juego contiene diversos elementos que conforman el todo. Estas partes son:

a) Pelota: Objeto que será lanzado para destruir los bloques en pantalla. Puede tener diversas formas, tamaños y colores (con sus comportamientos respectivos). Al caer al borde inferior de la pantalla (excepto por la primera), esta se reune con el jugador según las condiciones explicadas más adelante.

b) Bloque: Elemento que será destruído por el jugador. Para destruir un bloque es necesario golpearlo tantas veces como su resistencia. La resistencia de los bloques es igual a la puntuación actual al momento de crearlo. En caso de que la puntuación corresponda a un número divisible por 10, la resistencia del bloque se duplica. Los bloques pueden ser cuadrados o triangulares.

c) Power-up: Ícono ubicado en una celda que al contacto con una pelota producen algún efecto.

   1. Extra-ball: Aumenta en uno la cantidad de pelotas del jugador. Desaparece al con- tacto.
   2. Vertical-laser: Crea un laser de manera vertical, que disminuye en uno la resistencia de los bloques presentes en esa línea. Desaparece al caer la última pelota y solo si se produjo contacto con alguna de éstas.
   3. Horizontal-laser: Mismo comportamiento de la anterior, pero creando un laser de manera horizontal.
   4. Change-path: Cambia el ángulo de movimiento de la pelota de manera aleatoria al contacto. Desaparece al caer la última pelota y solo si se produjo contacto con alguna de éstas.

d) Filas y columnas: La pantalla se divide en 9 filas y 7 columnas donde la fila superior siempre esa vacía y la inferior debe estarlo (de lo contrario, el jugador pierde). Cada fila está compuesta por bloques y power-ups ubicados en las columnas. Por cada celda de la fila se dan las siguiente probabilidades:

   1) 50 % de ser un bloque.
   2) 25 % de ser un power-up.
   3) 25 % de ser vacío.

   Una fila siempre debe contener al menos uno de cada caso nombrado anteriormente, donde el power-up corresponde a un extra-ball.

3. Solución

Para la construcción total del juego, se diseña un procedimiento ha seguir, el cual sirve de guia. Es por esta razón que la metodologia propuesta se conforma de 18 problemas, los cuales se describen a continuación:

1. Creación de la pantalla
2. Creación de la Clase lado, para la construcción del contorno y los bloques

3. Creación del contorno
4. Creación Bloques cuadrados
5. Creación Clase fila y funciones:
   - Generación aleatoria de una Fila
   - Creación Clase fila
   - Metodo Pintar y Borrar fila
   - Metodo mover fila
   - Función Mover filas
   - Función devolver cantidad de cuadrados y PowersUp
6. Creación de la Clase Pelota:
   - Creacion Clase Pelota
   - Función Mover Pelota
   - Mover hasta chocar
   - Movimiento de mas Pelotas al mismo tiempo
7. Creación de powerUp
8. Unificación del Modelo

Cada uno de estos pasos a seguir, son expuestos a continuación:

3.1. Creación de la pantalla

Para la construccion integra del juego se utiliza Pygame, el cual da versatilidad en el uso del lenguaje. Para la creacion de la pantalla se genera un archivo `PPGAME.py` el cual servira posteriormente como menu principal, en el cual se inicia el siguiente codigo.

```
1 import pygame
2 pygame.init()
3 pantalla = pygame.display.set_mode((385,660))
```

Lista de Códigos Fuente 1: Incializar pantalla

3.2. Creacion Clase lado

Dado que muchos objetos del juego necesitan bordes, se genera una clase lado. Esta clase tiene como finalidad generar una imagen tal como se muestra a continuación.

Figura 3.1: Lado

El cual tiene bordes en 45º para una mejor facilidad de manejo con los bloques. Para el desarrollo, se implementa la libreria `centered figure` (Proporcionada por el auxiliar Pablo Pizarro R.), y se crea una clase contorno en un archivo llamado `Contorno.py`

Luego la clase Contorno tiene como parametros, X e Y del centro de la figura, alto y ancho de la figura, posicion de la figura (Esta tiene como valores üp","down","left",right", que sirve para identifiacion posteriores), pantalla en la cual se trabaja, color del lado y finalmente el color de la pantalla.

Ademas, la clase tiene el metodo de dibujo y borrado de la pantalla.

```
1 from centered_figure import *
2
3 class Contorno:
4     def __init__(self, x,y, ancho,alto,posicion,pantalla,color,fondo):...
5     def draw(self):...
6     def borrar(self):...
```

Lista de Códigos Fuente 2: Contorno.py

Para ver como se genera los metodos, revisar el archivo Contorno.py

3.3. Creacion del contorno del juego

Como se explico antes, el archivo `PPGAME.py` posteriormente se convierte en el menu del juego, es por esta razon que se crea un nuevo archivo llamado `modelo.py` que sirve como nuevo ambiente al iniciar el juego.

En el archivo `modelo.py`, se crea una funcion `modelo` que posteriormente será la función principal del juego. Esta función recibe 3 parametros, el menú principal, los vertices de la forma de la Pelota inicial y la pantalla que se utiliza.

```
def modelo(menu, vertices, pantalla):
```

Lista de Códigos Fuente 3: Función modelo

Una vez creado esto, se genera el contorno limite de rebotes del juego, tal como se muestra a continuación:

```
def modelo(menu, vertices, pantalla):
    # CONTORNO
    Contorno1 = Contorno(-2, 54, 1, 496, "left", pantalla, BLANCO, NEGRO)
    Contorno2 = Contorno(-1, 54, 387, 1, "up", pantalla, BLANCO, NEGRO)
    Contorno3 = Contorno(386, 54, -1, 496, "right", pantalla, BLANCO,
    NEGRO)
    Contorno4 = Contorno(-1, 496 + 55, 387, -1, "down", pantalla, BLANCO,
    NEGRO)

    #OBJETOS
    contorno = [Contorno1, Contorno2, Contorno3]
```

Lista de Códigos Fuente 4: Creación del contorno

Ademas se genera una lista con el contorno superior, izquierdo y derecho, para tener de esta manera registro para las colisiones posteriores

3.4. Creacion Bloques cuadrados

Dado que se tiene una clase para crear lados, la forma de un bloque cuadrado es como sigue:

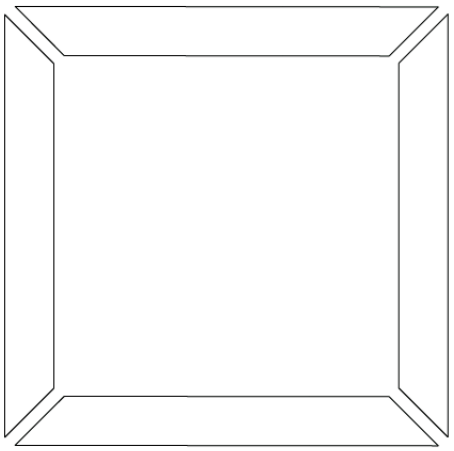

Figura 3.2: Bloque

Luego, la clase bloque tiene como parametros, el X e Y central, el tamaño del lado, color inicial, color de fondo, nombre (que para este caso corresponde a ÇUADRADO", las vidas o resistencia del bloque y finalmente la pantalla en la que se trabaja.

Ademas esta clase tiene como metodos, dibujar y borrar

```
from contorno import *

class bloque:
    def __init__(self, x, y, lado, color, fondo, nombre, vidas, pantalla)
    :...

    def draw(self):...

    def borrar(self):...
```

Lista de Códigos Fuente 5: Clase cuadrado

Para mayor informacion, ver archivo `Bloques.py`

3.5. Creación clase Fila

Una fila, es una lista que contiene los elementos que se muestran en pantalla de forma horizontal.

3.5.1. Generación aleatoria de una Fila

Como una fila tiene es a priori una lista, se crea un archivo propio, llamado `Fila.py` y una funcion independiente `generarFila` capaz de generar la lista.

Esta función tiene como parametros, la cantidad de celdas que contendra la fila, el lugar en el que se encuentra, que va de 0 a 8 correspondiente a la posicion de arriba a abajo en la que la fila se encuentra en la pantalla, el nivel que corresponde a la resistencia que tendran los bloques de la fila y finalmenta la pantalla en la que se trabaja.

```
def generarFila(celdas,lugar, nivel,pantalla):...
```

Lista de Códigos Fuente 6: Inicio función generador de fila

Al interior de la funcion, se inicializa la lista de retorno como vacia.
Ademas se crea una variable `powerMas` igual a False, que corresponde a la cantidad de powerUps extra-balls que contine la fila
Luego se itera según la cantidad de celdas que contiene la fila.
Al interior de la iteracion se escoge con probabilidad 1/2, si la celda será ocupada por un bloque o no.
Posteriormente se pregunta si la iteración se encuentra en la ultima celda. En caso de que se encuentre en la última celda y powerMas siga siendo False, en esa celda será ocupada por un Extra-ball (de esa forma se asegura que siempre haya una Extra-ball por nivel.
En caso contrario, se procede a crear un bloque en esa celda, el cual tiene ciertas condiciones:

- si el nivel es multiplo de 10, se escoge con probabilidad 1/2 si el bloque tendra el doble de resistencia o no
- si no, tan solo se crea un bloque normal

Ahora si no se escogio la creacion del bloque, se escoge con probabilidad 1/2 (dentro del Else) si es que se quiere un powerUp o un espacio vacio.

Luego se vuelve a preguntar si se encuentra en la última celda y si no hay Extra-Balls, en caso contrario si aun no existen Extra-Balls, se escoge con probabilidad 1/2 un Extra-ball y con probabilida 1/6 si es otro PowerUp.
En caso de que ya exista un Extra-ball, se escoge con probabilidad 1/3 los otros tres tipos de powerUp.

Finalmente si se llega al caso de escoger un espacio vacio, tambien se hace la pregunta de si se encuentra el la última celda y si existe o no un Extra-Ball

El codigo completo de esta función se encuentra en el archivo `Fila.py`

3.5.2. Creación de la clase Filas

La clase fila es una representacion mas ordenada de todos los parametros que contiene una fila. Esta clase, recibe 4 parametros iniciales.

1. Numero de celdas
2. Lugar en el que se encuentra la fila
3. Nivel de la fila (resistencia posterior de los bloques)
4. Pantalla de trabajo

Ademas en el iterior de la clase se genera un `self.lista` que produce con la funcion `generarLista()`

```
class fila:
    def __init__(self,celdas,lugar, nivel,pantalla):
        self.celdas = celdas
        self.lugar = lugar
        self.nivel = nivel
        self.lista = generarFila(self.celdas, self.lugar,self.nivel,
   pantalla)
```

Lista de Códigos Fuente 7: Clase fila

3.5.3. Metodo Pintar y Borrar una fila

Dado que se tiene `self.lista` que contine los elementos de la lista, se genera el metodo pintarFilaa que recorre lista de elemento y en caso de que sea distinto de cero se pinta el elemento que contiene la lista. En caso de ser una bloque con vidas >0 se pinta y si corresponde a un powerUp simplemente se llama a su función draw()
Analogo para el metodo de borrado

Ambos metodos se encuentra entre las lineas 53 - 65 del archivo `Fila.py`

3.5.4. Metodo Mover fila

Este motodo tiene dos funciones basicas, la primera función aumente en un determinado valor la coordenada Y de todos los elementos. Para esto se realiza una iteracion en los elementos de la

fila y si el elemento en cuestion es distinto de None, se aumenta su coordenada Y.

La segunda función es un pequeña algoritmo para que se observen los cambios en pantalla. Para esto se borrar la fila, luego se mueve en "55"todos los elementos (el numero 55 fue escogido para que toda la fila bajara completamente dejando espacio para las filas superiores) y posteriormente se vuelve a pintar la fila, finalmente a la variable lugar se le suma uno dado que deja el lugar anterior vacio.

Ambos metodos se encuentra entre las lineas 66 - 85 del archivo `Fila.py`

3.5.5. Función Mover Filas

Recapitulando, hasta el minuto se logra generar filas aleatoriamente, ademas se pueden pintar y borrar de la pantalla lo que hace que se logre mover hacia a abajo en pantalla.

Ahora, dado un conjunto de filas, se quiere lograr que todas bajen simultaneamente en pantalla.
Como sabemos, en pantalla solo caben 8 filas simutaneamente, por lo que se crea una función que recibe como paramentros una lista con todas las filas, la pantalla en la que se esta trabajando y el nivel actual del juego.

```
1 def moverFilas(filas,pantalla,nivel):
```

Lista de Códigos Fuente 8: Mover filas

Esta función itera sobra la lista de filas y mueve al iterior de la lista todas las filas a la siguiente posición.
Luego el la primera posicion de la lista genera una nueva fila.
Posteriormente mueve en pantalla todas las filas, pintando de esta manera la nueva fila.

Esta función se encuentra entre las lineas 101 y 110 del archivo `Fila.py`

3.5.6. Función Mover Filas

Ahora que se tiene todas las filas del juego, necesitamos saber todos los cuadros que existen en ella y todos los powerUp. Para esto se generan con funciones que devuelven una lista con todos los bloques y powerUp respectivamente. Tal como se muestra a continuación.

```
1 def cuadros_filas(filas):
2     cuadros = []
3     for i in filas:
4         if i != 0:
5             cuadros += i.cuadros()
6     return cuadros
```

Lista de Códigos Fuente 9: Función que devuelve bloques

```
1 def power_filas(filas):
2     powers = []
3     for i in filas:
4         if i != 0:
```

```
5               for elemento in i.lista:
6                   if elemento != 0 and elemento.nombre == "POWER":
7                       powers += [elemento]
8       return powers
```

Lista de Códigos Fuente 10: Funcion que devuelve powerUp

3.6. Creación de la clase Pelota

Esta es la clase mas importante dentro del juego, es por esto que para su desarrollo se dividio en 5 subproblemas que se explican a continuación.

3.6.1. Creación Clase Pelota

La clase pelota, recibe 4 parametros:

1. Vertices de la figura
2. Centro de la figura
3. Color de la Pelota
4. Pantalla de trabajo

Ademas esta clase contiene los metodos draw() y borrar() que se encuentran entre las lineas 10 - 23 del archivo `Pelota.py`

```
1  class Pelota:
2
3      def __init__(self,vertices,centro,color,pantalla):
4          self.ball = CenteredFigure(vertices,centro,color,0,pantalla)
5          self.centro = centro
6          self.color = color
7          self.vertices = vertices
8          self.pantalla = pantalla
9
10     def draw(self):...
11
12     def borrar(self):...
```

Lista de Códigos Fuente 11: Clase Pelota

3.6.2. Mover Pelota

Para mover una Pelota se debe seguir el siguiente algoritmo dado un angulo de movimiento.

1. Se borrar la Pelota en la posición actual
2. Al centro de la pelota se le suma `r*cos(angulo)` y `r*sin(angulo)` a la coordenada X e Y respectivamente
3. Se vuelve a dibujar la Pelota
4. Se actualiza la pantalla

Lineas 26 a 31 del archivo `Pelota.py`

3.6.3. Mover hasta chocar

Como logramos mover una pelota, ahora se debe ver si esta choca en algun punto. Para esto se crea un metodo `mover` el cual recibe como parametros:

1. Ángulo inicial
2. Lista de los lados del contorno
3. Lista de los lados de todos los cuadrados en pantalla
4. Lista de todos los PowerUps que existen
5. Velocidad de movimiento
6. Ymax que corresponde a la altura del contorno inferior del juego

Luego, al interior de esta funcion se mueve la pelota como se explico anteriormente, luego se pregunta si la nueva posicion es menor a la altura del contorno inferior, en caso de ser asi se borrar la pelota y se vuelve a centrar para un nuevo inicio, dibujandola posteriormente.

Si lo primero no ocurrio, se itera sobre la lista de contornos preguntandose si la Pelota a colisionado con alguno de estos, en caso de ser asi, se devuleve una lista con el angulo que llevaba y la posicion del contorno al interior de la lista de contornos.

Si lo anterior no ocurre, se itera en la lista de lados de los cuadrados, haciendo lo mismo que el paso anterior pero a la lista de retorno se le agraga una elemento True para diferencia si choca con un bloque o de un contorno.

Si lo anterior no ocurre, realiza lo mismo con la lista de los PowerUps
Finalmente si no choco con nada, tan solo se retorna None

Lineas 25 a la 51 del archivo `Pelota.py`

3.6.4. Movimiento de varias Pelotas al mismo tiempo

Esta función es una de las mas importantes del algoritmo, dado que tiene que realizar el efecto de multitarea de las Pelotas, para esto se crea una función `moverHasta`, la cual recibe como parametros

1. Una lista de las Pelotas
2. Un angulo inicial de movimiento
3. Una lista de los lados del contorno
4. Una lista de las filas que hay
5. Una lista de los PowerUps que hay
6. Ymax que corresponde a la altura del contorno inferior del juego

```
1 def moverHasta(bolas,angulo_inicial,contorno,filas,power,ymax):...
```

Lista de Códigos Fuente 12: Función moverHasta

Para hacer el efecto de multimovimiento, se realiza algo parecido a una cola de prioridad, en la cual se mueve en una unidad una Pelota, luego se mueve en una unidad la siguiente Pelota y asi sucesivamente.

Primero se crea una lista con los cuadros que hay en todas las filas, utilizando la funcion `cuadrosfilas(filas)` del archivo `Fila.py`

Luego se genera con esta lista, una lista de todos los lados que contienen los cuadros que hay en pantalla.

Ahora por cada Pelota que hay, se genera una copia de cada uno de estos elementos, ademas de generar una lista con todos los angulos.

Para el movimiento se realiza un `While True`, y al interior de este "mientras haya alguna Pelota que no haya llegado a la base del juego, se sigue iterando".

Luego se realiza una iteración en todas las Pelotas que existan y se pregunta si esa Pelota llego a la base del juego o no. Si aun no llega a la base del juego, se procede a mover.
Se sabe que la funcion mover puede deborver 5 resultados.

1. True, si la Pelota se encuentra en la base del juego
2. Lista de tamaño 2, si choco con un contorno
3. Lista de tamaño 3, si choco con un bloque
4. Lista de tamaño 4, si choco con un PowerUps
5. None, si no ocurrio ninguno de lo anterior

Ahora si al moverlo no devolvio None, entonces:

Preguntamos si la Pelota devolvio True, de ser asi, comprueba si otra pelota a caido antes que el, si no es asi se crea una variable posicion correspondiente a la coordenada X de esta. Si ya habia una Pelota que habia caido antes de el, se borra la pelota se cambia a la posicion de la primera pelota que cayo y se vuelve a dibujar.

Ahora si no ocurrio lo anterior, se pregunta si la pelota choco con algún contorno, en caso de chocar con este, se pregunta con cual contorno choco y cual fue su angulo de incidencia, poniendose en todos los caso.

a) Choca con el contorno izquierdo

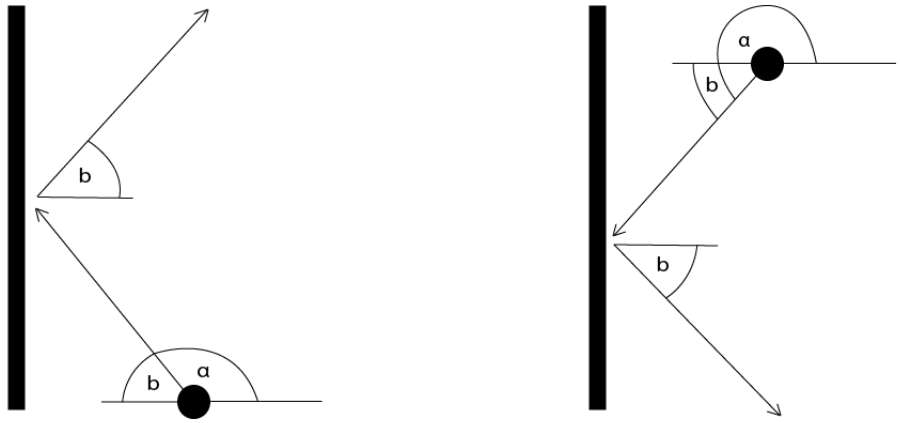

Figura 3.3: Choque izq

b) Choca con el contorno derecho

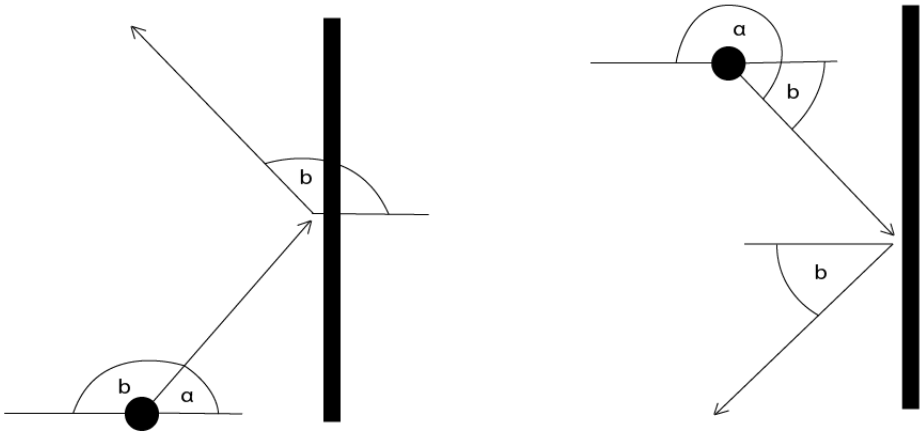

Figura 3.4: Choque der

c) Choca contorno superior

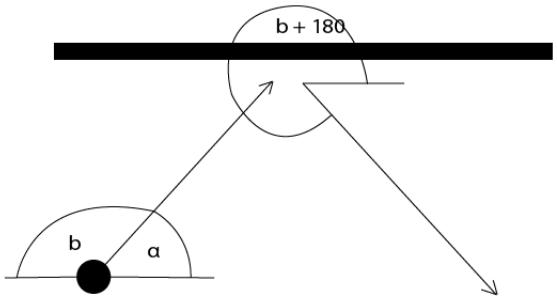

Figura 3.5: Choque up

d) Choca con el lado inferior (caso posterior para cuando choca con un bloque)

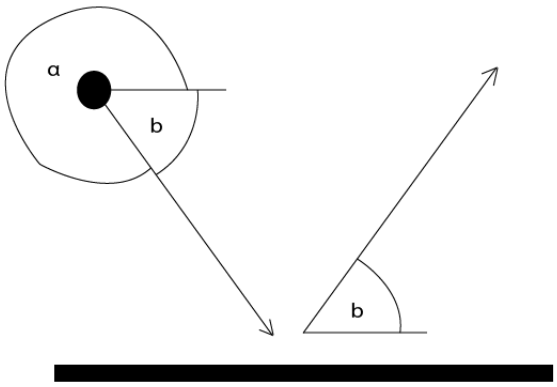

Figura 3.6: Choque down

Ahora en caso de haya chocado con un bloque, se utilizan los mismos angulos de incidencia. Pero en para este caso, se realiza una comprobación de las vidas que tiene el bloque y en caso de que las vidas de este llega a cero, tan solo se elimina de la pantalla y de la lista de los cuadrados y de los lados.

En caso de que haya chocado con un powerUp, se comprueba que tipo de powerUp es y tenemos dos casos. Si es un Extra-ball se suma uno a la variable powers que posteriormente se traduce en la creación de mas Pelotas. Si el powerUp es un Change-path, se modifica el angulo de incidencia aleatoriamente.

Finalmente se retorna una lista con la posicion de la primera Pelota, la cantidad de Extra-Ball y una lista con los cuadrados eliminados.

3.7. Creación de powerUps

La creacion de powerUp es igual de sencilla que la creación de bloques, tan solo va variando el tipo de powerUp. Es por esta razon que para mayor detalle de la creación de esta clase, se puede revisar el archivo `PowerUp.py`

3.8. Unificación del Modelo

Una vez con todos los elementos creados, se pasa a unificar todos esto. Para esto se crea una funcion `modelo` en el archivo `modelo.py` que recibe como parametros el menu, los vertices de las Pelotas y la pantalla a trabajar.

Primero, se crea el diseño completo de la pantalla del juego, el cual contiene los contornos, las filas y el resto de elementos que componen el todo de la pantalla.
Posteriormente, dentro del While de pygame, se crean los eventos, en caso de presionar el boton cerrar o el boton ESC, se sale de la pantalla y vuelve al menu.

En caso de presionar la pantalla con el mouse, comienza el juego. Primero se capta la posición y se genera el angulo de lanzamiento, luego se mueve hasta volver a la base del juego. Luego se crean las nuevas Pelotas en caso de ser necesario y se mueve hacia abajo las filas comprobando si exite algun cuadro que haya llegado a la base, de ser asi se entra en GAME-OVER, volviendo al menu.

4. Discución

Dentro de la construcción se susitaron varios problemas.

De partida, no se logro la creación efectiva de los bloques triangulares. Si bien es cierto, existe un archivo `BloqueTriangulo.py` el cual contiene las clases de los 4 triangulos, no se logro colocar en la generacion aleatoria de la fila. Por alguna razón esta no lograba dibujar los triangulos, por lo que se decide no colocar esa forma.

En segundo lugar, la creación de la función de movimiento multiple de Pelotas, se laggea demasiado al tener muchas Pelotas.

En tercer lugar, no se logra hacer efectivo la construccion de los powerUps del tipo laser.

Ademas en ciertas ocaciones se generan bugs, al rebotar en cuadrados ïnvisibles.º al ingresar a los cuadrados y matarlos por dentro.

Todos estos problemas son solucionables a corto plazo. Por ejemplo, la construcción de los triangulos, tan solo es necesario arreglar el problema de eleccion en la generación aleatoria de la fila. Tambien se encuentra una lentitud en la funcion de movimiento multiple al comparar Strings para saber el tipo de objeto por el cual choco, siendo posible arreglarlo con listas de booleanos para reducir el tiempo de comparación. Por otro lado, la creación del los powerUps laser, es de facil implementacion, en la cual al pasar por este tan solo se debe seleccionar los bloques que se encuentran en la fila de los laser para asi bajarles en 1 su resistencia y finalmente, se puede mejorar la eliminación de los bloques dado que las listas que contienen los lados de los bloques no eliminan de buena manera los bloques eliminados en pantalla.

5. Conclusiones

A partir del desarrollo de este proyecto se logran concluir 3 objetivos fundamentales. En primer lugar se logra un conocimiento basico-intermedio del uso de librerias pygame, las cuales sirven para el desarrollo optimo de videojuegos.
En segundo lugar, la aplicación de condiciones fisicas del juego fueron parte primordial dento del desarrollo, esto se traduce al rebote que tienen las pelotas del juego, a su vez el entender el funcionamiento de como se generan multiple movimientos simultaneamente, son de gran importancia para el desarrollo personal durante la carrera.
Finalmente, los problemas antes expuestos fueron simplemente por un no optima distribución del tiempo para la implementación de su solución es por esta razón que la organización es fundamental para el trabajo prolijo dentro de programas. De esta manera se propone una posterior mejora del juego pero a nivel personal para la mejora de conocimientos de estas tecnologias de información.

Por útlmo se espera que este informe haya sido una buena guia de como se concibio, organizo e implemento el juego anteriormente descrito.