Proyecto 2 Avatars Vs Rooks

Raquel Lizano y Michael Valverde Navarro

I. Introducción

En el siguiente trabajo se presentará la documentación del segundo proyecto del taller de programación realizado en parejas correspondiente al primer semestre del año presente, el cual consiste en la realización un juego de estilo tower defense, en donde existen diferentes tipos de avatars que buscan atacar al jugador. Por otra parte el jugador dispone de diferentes rooks con distintas características de ataque y defensa, esto con el motivo de detener el ataque de los avatars. Los avatars presentan diferentes atributos y habilidades.

Al ingresar al juego nos encontraremos con el menú principal, en donde el jugador deberá registrarse con el un nombre de usuario y configurar la frecuencia de ataque de las rooks, para posteriormente poder ingresar a un nivel de juego . En la pantalla de juego se encuentra un tablero de 9X5, el territorio de el jugador está del lado izquierdo y será atacado de el lado contrario. Para realizar este juego se utilizaron bibliotecas como pygame,math, random, entre otras, se implementa la programación orientada a objetos para realizar los avatars, rooks, matriz, entre otros. También se implementa una animación para cuando se gana la partida y otra para cuando se pierde el juego. Retornando los puntos obtenidos y guardándolos en un archivo .txt para poder mostrarlos en los posteriormente en la pantalla de puntuaciones.

II. Marco Teórico

II-A. Bibliotecas Utilizadas

Para el desarrollo del juego se utilizaron diferentes bibliotecas del lenguaje de programación Python, una biblioteca se puede definir como una colección de módulos o funciones que permiten realizar una acción para solventar un problema específico[3], en este caso para el desarrollo de este proyecto es necesario utilizar estos recursos ya que permiten que se cumplan los requerimientos asignados, además es necesario hacer uso de esto puesto que para este programa es necesario la implementación de interfaz gráfica la cual es la parte en el que el usuario interactúa directamente con el software.

Pygame fue una de las bibliotecas seleccionadas para desarrollar este juego, esto porque dicho módulo es creado específicamente para la creación de videojuegos, este recurso incluye múltiples funciones para desplegar y manipular imágenes con facilidad, revisar colisiones entre dos o más objetos, permite trabajar con grupos de sprites[4], permite crear gráficos y hace la manipulación de sonido más sencilla, tiene la posibilidad de poder controlar el mainloop de un juego directamente realizado con esta biblioteca.

Para las animaciones que se despliegan cuando el jugador pierde o gana el juego se utilizó la biblioteca Turtle[2], este recurso se describe como una biblioteca que está orientada al dibujo y creación de imágenes, se pueden crear juegos y animaciones sencillas, es por estas razones que se considera apropiado implementar la creación de estas animaciones en esta plataforma.

La biblioteca math[1] que contiene métodos para realizar cálculos matemáticos fue de utilidad en el proyecto puesto que se utilizan los métodos math.sqrt() y math.pow() para obtener la distancia entre dos coordenadas cuando los objetos no son sprites.

Random también se utiliza para la creación de nuevos avatars, haciendo uso del método randint().

II-B. Propuestas de diseño de solución

Para este proyecto es necesario implementar una matriz en la pantalla del juego, esta debe ser interactiva, es decir que cuando se le click responda a la instrucción que el usuario quiere que haga, para implementar esto se requiere de un mecanismo que dibuje una matriz, es necesario que la matriz quede dibujada en pantalla para que el usuario sepa en donde debe dar click para poner su personaje tipo rook, este mecanismo usa dos ciclos for anidados para dibujar las líneas de la matriz, sin embargo esto es solo una representación gráfica del espacio de juego, es necesario hacer la matriz interactiva, para esto se hace uso del paradigma de programación orientado a objetos, en el cual se declaran clases con atributos y métodos para representar objetos de manera abstracta.

Es por esto que se crea la clase de BackgroundTile() esta clase permite obtener una matriz invisible, esta clase utiliza una clase propia de Pygame llamada pygame.sprite.Sprite, por lo cual se concluye que se está haciendo uso de herencia para este juego. Haciendo uso de esta clase solamente se asegura que la matriz se le pueda dar click, cuando se hace uso de esta clase se crea una matriz de 6x11 que abarca toda la pantalla, por lo que quiere decir que se debe delimitar el área de juego, el área de juego debe ser una matriz de 9x5.

Para delimitar la matriz se van a necesitar dos clases más, una en la que queden inactivas los espacios de la matriz en la que no se puedan colocar objetos de tipo rook, esta clase corresponde a InactiveTiles(), en la cual cada vez que se le da click a una casilla creada por esta clase nos da valores como None, o simplemente en sus métodos se invoca un pass, por otra

parte se necesita obtener el espacio del juego en el que el usuario va a poder colocar elementos, esto se puede realizar con la implementación de otra clase llamada Playtile(), que permite establecer objetos en alguna de las casillas de la matriz.

Para seleccionar la rook que se quiere poner en una de las casillas del juego se deben crear botones para estas, esto es posible realizarlo con otra clase adicional llamada Buttontile(), esta clase se encarga de hacer las imágenes de rooks en botones para que el usuario usando el mouse seleccione la rook de su agrado y la aplique en el área de juego.

A continuación se demuestra como esta constituida la pantalla del juego usando las clases mencionadas anteriormente

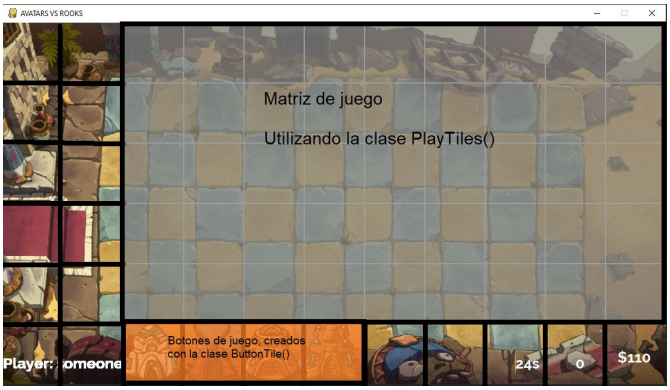


Figura 1. Matriz en la pantalla

Teniendo este ejemplo gráfico se puede ver que la pantalla esta constituida por una matriz con dimensiones de 6x11 de que cubre toda la pantalla y se delimita a conveniencia dependiendo de la zona de la matriz que se necesita utilizar, en la imagen la parte señalada con gris corresponde a la matriz por donde aparecen los avatares y se sitúan las rooks, y la parte señalada con naranja corresponden a las casillas creadas con la finalidad de servir como botones de juego. Las casillas que no se encuentran resaltadas con color corresponden a las creadas con la clase InactiveTiles() de esta manera se asegura que el usuario no tenga la posibilidad de poner una rook que no sea en el área de juego predeterminada.

Los personajes del juego se pueden dividir en 2, los avatares que son los encargados de hacer al usuario perder el juego, y los rooks que son los objetos encargados de defender y hacer que el jugador pueda ganar, son 4 personajes de tipo avatar con características o atributos diferentes, para las rooks también son 4 tipos con atributos diferentes, para poder modelar estos personajes y objetos se usan clases en las que se se asignan atributos como el tipo de avatar, la velocidad de desplazamiento en la pantalla, una coordenada en el eje X y en el eje Y para conocer la ubicación del personaje, un atributo que le indique cuándo detenerse, la potencia de ataque y la salud del personaje, entre otros.

```python
class Avatar(sprite.Sprite):
    """Create avatars with specific features, the object created will be the enemy of the game"""
    def __init__(self,type,image,attackPower,health,walkSeconds):
        super().__init__()
        self.type = type
        if self.type == "AVATAR_ARCHER":
            all_avatars_archers.add(self)
        elif self.type == "AVATAR_KNIGHT":
            all_avatars_knights.add(self)
        elif self.type == "AVATAR_CANNIBAL":
            all_avatars_cannibals.add(self)
        elif self.type == "AVATAR_LUMBERJACK":
            all_avatars_lumberjacks.add(self)

        self.speed = REG_SPEED
        self.lane = randint(0,4)
        all_avatars.add(self)
        self.image = image
        y = 50 + self.lane * 100
        self.health = health
        self.attackPower = attackPower
        self.walkSeconds = walkSeconds
        self.stop = False
        self.rect = self.image.get_rect(center=(1100,y))
        self.despawn_wait = None
        self.avatars_killed_count = 0

    def getX(self):
        return self.rect.x

    def getY(self):
        return self.rect.y
```

Figura 2. Clase Avatar

Los atributos de esta clase son self.type que indica el tipo de avatar que se crea, también se necesita un atributo para determinar si el avatar continua vivo o si ha sido derrotado, esto se realiza con el atributo self.health que corresponde a los puntos de vida que tiene mientras esta en el campo de juego, es necesario hacer que el avatar camine y se detenga para lograr esto se crea el atributo self.walkSeconds que recibe un integer que determina cuantos segundos debe esperar el avatar para poder caminar en la pantalla de nuevo, y el self.stop que es de tipo booleano que le indica al avatar cuando se debe detener estando en la pantalla. Al crearse un Avatar el objeto se añade a una clase contenedora, que permite manejar sprites, esta clase es sprite.Group(), esta clase también funciona como una estructura iterativa por lo cual es útil almacenar los objetos en esta estructura para posteriormente recorrerlos con un ciclo for y utilizar el método update de la clase avatar para hacer que los personajes se puedan mover y parar de acuerdo a sus atributos mencionados anteriormente.

```
def update(self,game_window,counters):
    """this method makes the avatar move"""
    game_window.blit(BACKGROUND,
                     (self.rect.x, self.rect.y), self.rect)

    timer = pygame.time.get_ticks()
    timerWalk = int(timer/1000)
    """Avatars move """
    if timerWalk%self.walkSeconds == 0 and not self.stop:
        self.rect.x -= self.speed*2.1
        self.stop = True
    else:
        self.stop = False


    """Check for collisions with bullets"""
    isCollidingWithFireBullet = sprite.spritecollide(self,all_fire_bullets,True)
    if isCollidingWithFireBullet is not None:
        for bullet in isCollidingWithFireBullet:
            self.health -= 8
            PlayHitSound(2)
```

Figura 3. Método update en clase Avatar

Para modelar los objetos que se encargan de defender es necesario crear otra clase nueva que nos permita obtener objetos con características especificas, para el caso de la creación de las rooks se crea una clase muy similar a la clase Avatar, esta nueva clase contiene atributos como self.type para determinar el tipo de rook que se esta creando, es muy importante tener un atributo para determinar el precio de las rooks por lo tanto se crea self.cost que toma un integer para determinar el valor monetario de este objeto, se crea un atributo de imagen para poder tener representación visual de como luce el objeto.

```
class Rook(sprite.Sprite):
    """Class to create a rook"""
    def __init__(self,type,cost,type_img):
        self.type = type
        if type == "WATER_ROOK":
            self.health = 16
        elif type == "FIRE_ROOK":
            self.health = 16
        elif type == "ROCK_ROOK":
            self.health = 14
        else:
            self.health = 2

        self.cost = cost
        self.type_img = type_img
        self.rect = self.type_img.get_rect()
        self.bullet_speed = 1
```

Figura 4. Clase Rook

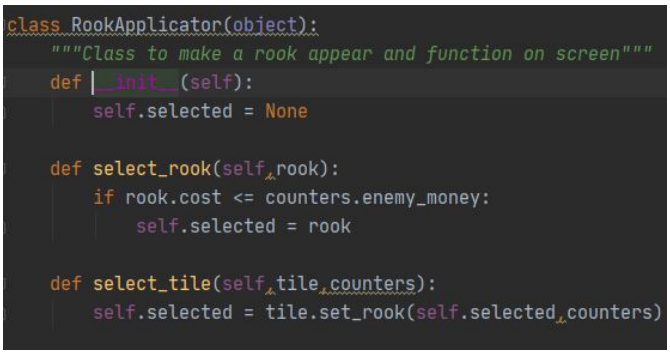
Para poder poner la rook en la matriz de juego se necesita crear un condicional if en el mainloop del juego, esto para indicarle a pygame que busque eventos cuando el mouse es clickeado, al ser clickeado se debe obtener la posición en la

que esta el mouse y contenerla en una lista, posteriormente se hace uso de otra clase llamada RookApplicator() para poder establecer la rook en la posición que el usuario desee.

```
while game_running:
    #Check for events
    for event in pygame.event.get():
        #Exit the game
        if event.type == QUIT:
            game_running = False
            program_running = False
        elif event.type == MOUSEBUTTONDOWN: #event for when the left button of the mouse is pressed
            if event.button == 1:
                coordinates = mouse.get_pos() #gets the coordinates of the mouse
                x = coordinates[0] #store the coordinates in a list
                y = coordinates[1]
                tile_y = y // 100 #get the column
                tile_x = x // 100 #get the row
                #print(tile_y,tile_x)
                print(x,y)
                rook_applicator.select_tile(tile_grid[tile_y][tile_x],counters) #Applies the rook to the game board
```

Figura 5. Aplicación de Rooks en el tablero

La clase RookApplicator() contiene dos métodos uno para cuando se selecciona la rook que quiere utilizar y otro para seleccionar la casilla en la que se desee posicionar



```
class RookApplicator(object):
    """Class to make a rook appear and function on screen"""
    def __init__(self):
        self.selected = None

    def select_rook(self,rook):
        if rook.cost <= counters.enemy_money:
            self.selected = rook

    def select_tile(self,tile,counters):
        self.selected = tile.set_rook(self.selected,counters)
```

Figura 6. Clase RookApplicator

Uno de los requerimientos para este proyecto es que los objetos tipo rook deben lanzar objetos a los enemigos, este mecanismo puede ser comparado a un objeto lanzando proyectiles para hacer daño a otro, para lograr cumplir con este requerimiento se determinó la necesidad de crear una clase llamada Bullets() para que cada rook tenga la posibilidad de lanzar agua, arena, rocas o fuego segun el tipo de rook. La clase Bullets() requiere de un atributo que corresponda a las coordenadas del objeto rook, ya que estas coordenadas le indican a este proyectil desde donde debe iniciar su salida, también es necesario indicar el tipo de proyectil que es, puesto que las rooks deben lanzar objetos diferentes dependiendo de su tipo, el atributo que permite hacer distinción de los proyectiles es self.bullet-type, se necesita que cada proyectil tenga una velocidad establecida para que pueda hacer el recorrido de moverse desde el lado izquierdo hasta el lado derecho de la pantalla. Al ser un objeto que debe estar apareciendo en pantalla debe tener una imagen asociada, esta imagen se asigna dependiendo del tipo de proyectil, las imágenes para los proyectiles corresponden a una llama de fuego para la fire rook, un montículo de arena para la sand rook, gotas de agua para la water rook, una ilustración de rocas para la rock rook.

Al declararse una instancia de esta clase se necesitan una característica para conocer el tipo de proyectil que se crea, para esto se cuenta con self.bullet-type, esto indica el tipo de proyectil y asigna la imagen correspondiente, es necesario conocer un punto en el eje x y en el eje y de la pantalla para conocer donde esta la rook que se va a encargar de disparar estos proyectiles, por esto es requerido un atributo para poder tener nocion desde que punto se dispara. Al obtener una instancia de esta clase, se obtiene un grupo de sprites dado por sprite.Group(), en el cual se recorre la lista de coordenadas de las rooks para actualizar los proyectiles. En esta clase se tiene un método llamado update que permite hacer que los proyectiles se muevan en dirección a los enemigos, el proyectil al tocar un avatar le resta puntos de su vida, y al llegar a un nivel de vida igual o menor que cero, el avatar muere y el proyectil desaparece de la pantalla, aparece una imagen de explosión para representar la ausencia del avatar en el campo de juego.

II-C. Demostración de Funcionamiento

Pantalla Principal:

Al iniciar el juego el usuario se encuentra con la pantalla principal, esta incluye el titulo del juego, y dos espacios, el primero será para que el usuario se registre con su nombre de jugador, el segundo espacio es para determinar cada cuantos

Figura 7. Rooks lanzando proyectiles de arena, roca y fuego

Figura 8. Avatar al ser derrotado por una Water Rook utilizando un proyectil

segundos se quiere que las rooks dispare a los avatars. También se cuenta con botones de los niveles disponibles en el juego, las puntuaciones de los jugadores anteriores, un apartado de instrucciones para que el jugador entienda la dinámica del juego, y los créditos.

Luego de que el usuario ingrese su nombre de jugador e ingrese los segundos de disparo de las rooks se podrá elegir el nivel de juego. El usuario tiene la disponibilidad de elegir uno de los 3 niveles. En la parte inferior de la pantalla aparece el nombre del jugador, las rooks disponibles para ser utilizadas y defender el campo de juego, en la esquina inferior derecha aparece el timer en segundos, la cantidad de avatars que han logrado cruzar hacia el otro lado del tablero, y la cantidad de

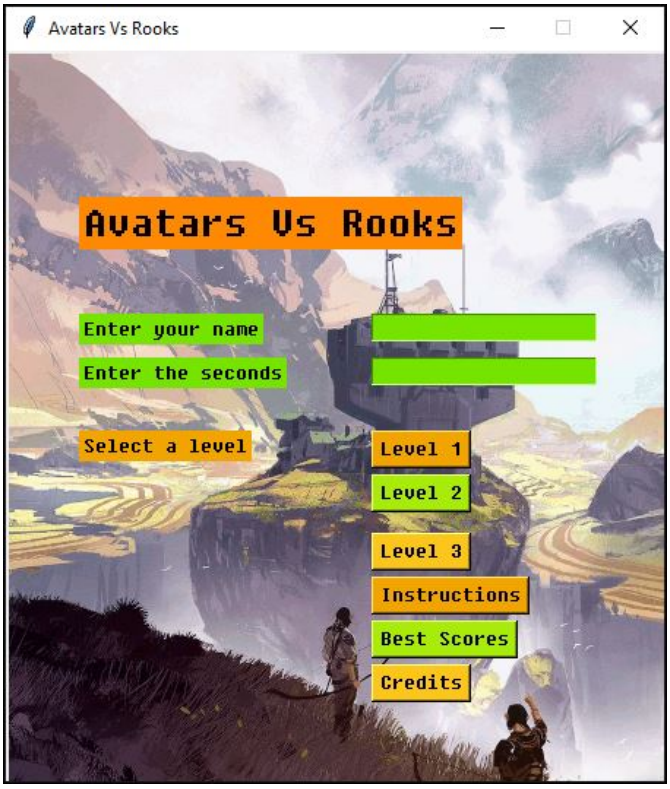


Figura 9. Menu principal

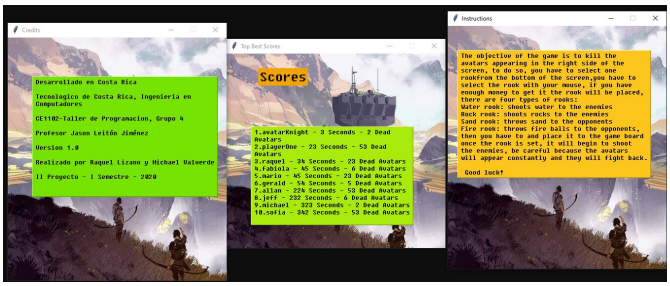


Figura 10. Ventana de créditos, instrucciones y puntuaciones

dinero con la que cuenta el jugador para poder comprar más rooks.

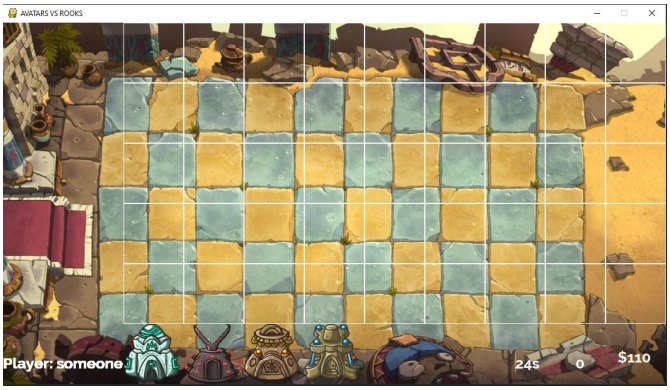


Figura 11. Nivel 1

Para cada nivel se cuenta con diferente escenario, esto para evitar que el usuario sienta que el juego es interminable al tener un solo fondo de juego.

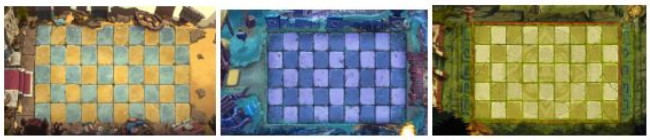


Figura 12. Escenarios de los niveles

Para poner rooks en la matriz dibujada, se le debe hacer click a las rooks que se quieren obtener y posicionarlas en cualquier parte del espacio definido, las rooks empezaran a lanzar proyectiles a sus adversarios automáticamente, los rooks avanzaran poco a poco y el objetivo es matar la mayor cantidad de avatares y evitar que lleguen al otro lado del tablero. El dinero incrementa automáticamente durante el tiempo de juego, sin embargo se puede obtener una cantidad de dinero mas alta al poner sand rooks. El jugador gana el juego cuando haya pasado los 3 niveles, para poder ganar se debe eliminar 10 avatars en el nivel 1, 20 avatars en el 2 y 30 avatars en el nivel 3, se debe procurar que los avatares no pasen hacia el otro lado, o que la cantidad de avatares que pasan sea mínima, cuando se gana o pierda el juego despliega una animación.

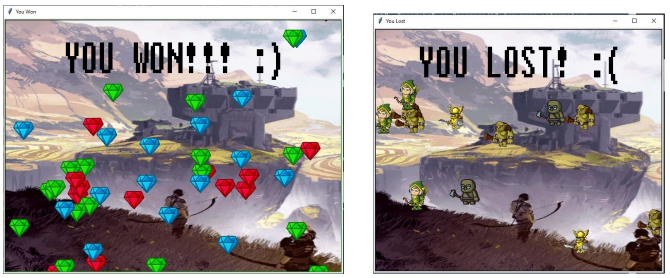


Figura 13. Animaciones cuando se gana o pierde

III. Conclusiones

En este proyecto podemos ver reflejado una de las tantas utilidades que tiene las listas a la hora de poder obtener datos ya sea de manera aleatoria, guardar imágenes,parámetros ya que se puede acceder a estos datos mediante un índice

La iteración es parte fundamental del juego, ya sea para actualizar la pantalla de juego como para seleccionar datos de una lista o muchas más funciones.

La programación orientada a objetos es un paradigma de programación en el que los programas son vistos como formados por entidades llamadas objetos que recuerdan su propio estado interno y que se comunican entre sí mediante el paso de mensajes que se intercambian con la finalidad de cambiar sus estados internos, compartir información y solicitar a otros objetos el procesamiento de dicha información, permitiendo analizar y diseñar con más facilidad programas con un enfoque realista, también la creación de programas extensos gracias a la reutilización y polimorfismo, la facilidad de identificar errores, de entender y modificar el código fuente debido al encapsulamiento. pero los objetos no sólo transmiten mensajes, sino que también comparten el comportamiento entre otros objetos del mismo tipo y heredan características de otros tipos relacionados.

Así de esta manera que representan un modelo de la interacción de las cosas en el mundo real, interesa qué hacen los objetos más que cómo se hace.

IV. Recomendaciones

Es recomendable encapsular las variables de las clases e ingresar a a ellas solo por medio de funciones set o get. ya que gran parte del proyecto incluye listas, se recomienda recorrerlas utilizando ciclos for, para obtener un código eficiente, ya que es más común utilizar este ciclo para recorrer estructuras iterables. Mantener separados los elementos gráficos de la parte lógica en el código.

Referencias

[1] Varios autores. The python standard library » numeric and mathematical modules. *URL: https://docs.python.org/3/library/math.html*, 2019.
[2] Varios autores. The beginner's guide to python turtle. *URL:https://realpython.com/beginners-guide-python-turtle/*, 2020.
[3] Will McGugan. *Beginning game development with Python and Pygame: from novice to professional*. Apress, 2007.
[4] Pete Shinners. Sprite module introduction. *URL:https://www.pygame.org/docs/tut/SpriteIntro.html*, 2020.