UNIVERSIDAD CATÓLICA DE TEMUCO

INGENIERÍA CIVIL EN INFORMÁTICA
FACULTAD DE INGENIERÍA

Proyecto Semestral Programación I

Red Bounce Ball

Revillod Jerez Luciano Ignacio

Programación I, Sección III, 2022

Enrique Nicolas Ketterer

Grupo VII

Temuco, 22 de junio de 2022

Índice

I) Introducción.....Página 3.

II) Resumen.....Página 3.

III) Manual de usuario.....Página 4.

IV) Diagrama de casos de uso.....Página 5.

V) Diagrama de flujo.....Página 6.

VI) Librerías.....Página 7.

A) Pygame

B) Ctypes

Introducción

En sus inicios los videojuegos fueron ideados para los niños y como un método de distracción casual, en la actualidad forman parte del día a día de las personas, pero ¿cómo se llegó a esto?.

Programar un videojuego no es una tarea fácil, para ello se requieren múltiples conocimientos y herramientas, así como años de intentos fallidos y limitaciones para crear los primeros prototipos funcionales. ¿Cuáles son las etapas en el desarrollo de estas aplicaciones? ¿Existe acaso una fórmula especial? ¿Es todo un simple código de texto?.

El Informe a presentar corresponde a una investigación y desarrollo del videojuego basado en el lenguaje de programación Python "Red Bounce Ball", con él se busca identificar y exponer los distintos procedimientos que conlleva la programación de un videojuego de nivel básico.

Resumen

El presente proyecto tiene como objetivo exponer y explicar la utilidad y funcionalidad del videojuego Red Bounce Ball, los contenidos que principalmente se pueden encontrar en el proyecto son Condicionales, Ciclos, Funciones o Subprogramas, Clases, Manejo de archivos y librerías externas (Pygame y Ctypes). En cuanto a la idea y diseño del videojuego, es importante mencionar que se realizó pensando en un aspecto clásico, tal como lo es "Bounce" originalmente.

Los resultados del proyecto son los esperados, un juego funcional de dos etapas como base, por supuesto actualizable a más.

Manual de usuario

Requisitos para ejecutar el videojuego:

a. Un editor de texto capaz de ejecutar códigos de Python.
   Se recomienda Visual Studio Code.
b. Una versión actualizada de Python (3.x).
c. Las librerías pygame y ctypes, en versiones compatibles con python 3.
d. Un teclado que posea las teclas direccionales:
   arriba, izquierda y derecha.

Objetivo y características:

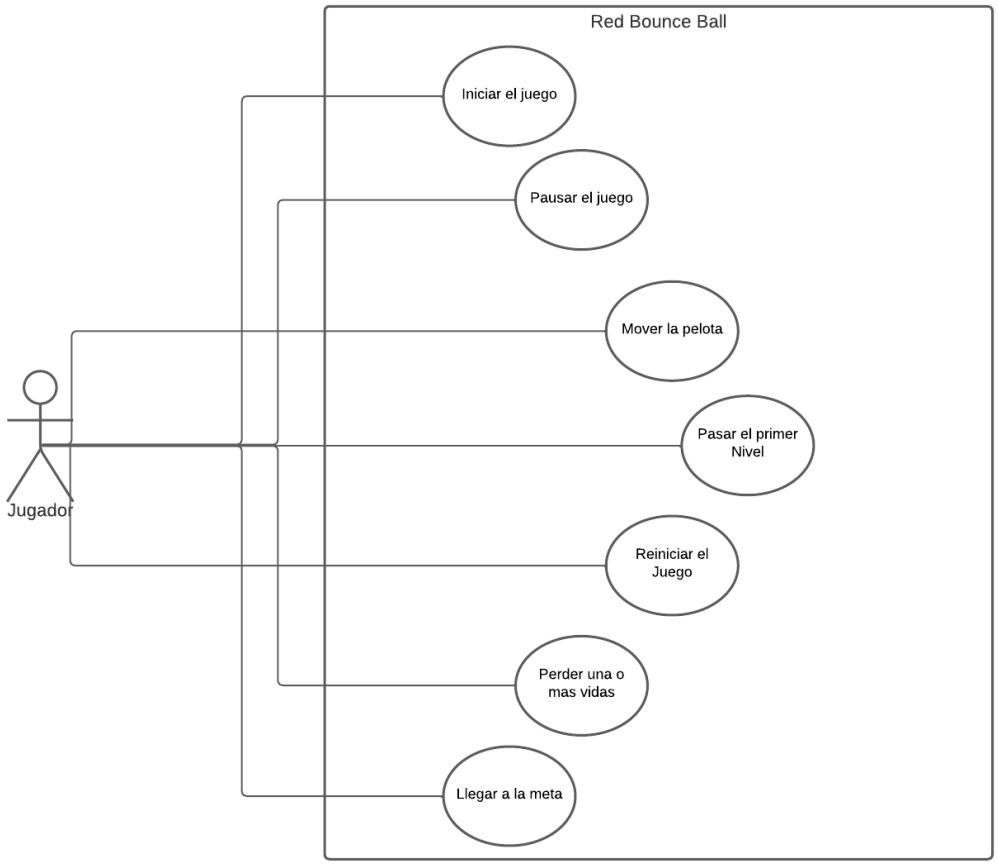
a. El código consiste en un juego de dos dimensiones
   basado en plataformas, un jugador y colisiones.
b. El objetivo principal del jugador es llegar al final del mapa.
c. El jugador posee 3 vidas (intentos).

Movilidad:

a. El jugador tiene las mismas capacidades de movimiento que los clásicos juegos en dos dimensiones (2D).
b. Dentro del mapa existen múltiples obstáculos, estos buscarán impedir
   el avance del jugador, al tocar uno de estos obstáculos el jugador
   perderá una vida equivalente a un intento.
c. Los movimientos tienen asignados las siguientes teclas:
   1. Saltar: Tecla “ K_UP ” (Flecha arriba).
   2. Derecha: Tecla “ K_RIGHT ” (Flecha derecha).
   3. Izquierda: Tecla “ K_LEFT ” (Flecha izquierda).
   4. Salto a la derecha: Combinación de teclas:
      “K_UP ” + “ K_RIGHT ”.
   5. Salto a la izquierda: Combinación de teclas:
      “K_UP ” + “ K_LEFT ”

Diagrama de casos de uso Red Bounce Ball

Un diagrama de casos de usos muestra de manera gráfica las distintas funciones que puede realizar un usuario en un sistema de información, en este caso un programa.



Para construir este diagrama se utilizó la herramienta de modelado en línea “Lucidchart''.

Con el diagrama se busca visualizar los casos de uso que posee quien ejecute el programa (Jugador).

Diagrama de flujo

¿Qué es un diagrama de flujo?

Un diagrama de flujo es un diagrama que describe un proceso, sistema o algoritmo informático. Se usan ampliamente en numerosos campos para documentar, estudiar, planificar, mejorar y comunicar procesos que suelen ser complejos en diagramas claros y fáciles de comprender.

K.S. (2008, diciembre). LucidChart. LucidChart. https://www.lucidchart.com/

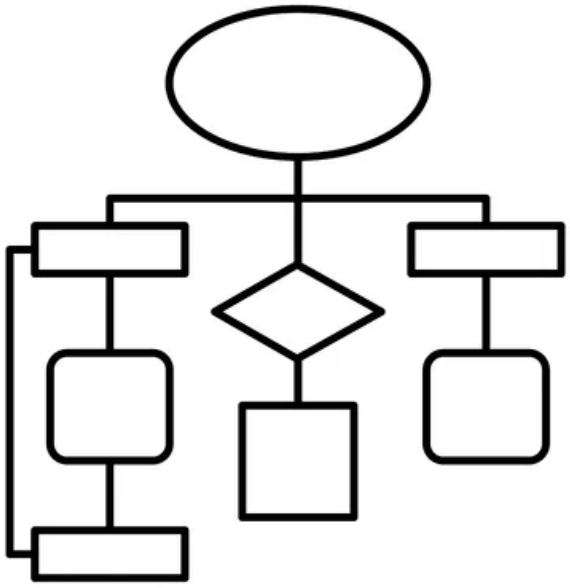


Imagen referencial de la estructura de un diagrama de flujo.

Para “Red Bounce Ball” el diagrama consiste en los distintos casos que se pueden presentar en la ejecución del programa, por ejemplo: presionar una tecla, esta acción provocará un cambio en el programa.

Archivo .psc original adjuntado como “red bounce ball.psc”

Librerías utilizadas en el proyecto

Pygame

El proyecto “Red Bounce Ball” consiste en un código basado en Python y su librería Pygame.

“Pygame es un conjunto de módulos de Python diseñados para escribir videojuegos. Pygame añade funcionalidad a la excelente biblioteca de SDL. Esto le permite crear juegos con todas las funciones y programas multimedia en el lenguaje Python”.

Pygame. (s/f). Pygame.org. http://pygame.org

Para utilizar esta librería y todas sus funciones se debe seguir el siguiente procedimiento:

```
Import pygame
From pygame.locals import *
```

Este texto debe ir, de preferencia al inicio del código. Es importante mencionar que, al importar cualquier otra librería se recomienda seguir el mismo procedimiento.

Ctypes

“Ctypes es una biblioteca de funciones foráneas para Python. Proporciona tipos de datos compatibles con C y permite llamar a funciones en archivos DLL o bibliotecas compartidas. Se puede utilizar para envolver estas bibliotecas en Python puro.”

Python,org. www.docs.python.org/es/3.10/library/ctypes.html

Funciones:

En el desarrollo de Red bounce ball se utilizaron las siguientes funciones incorporadas en pygame y ctypes.

- **pygame.init ( )**: Función que se encarga de iniciar todos los módulos de la librería pygame
- **pygame.mixer.init ( )**: Inicia el reproductor de audio de pygame.
- **pygame.display.set_caption ( )**: Esta función recibe un str de entrada.
  El string ingresado será el nombre de la ventana del programa.
- **pygame.mouse.set_visible( )**: Visibilidad del mouse en la ventana principal.
- **pygame.display.set_mode(resolución)**: Función que recibe la resolución de la ventana a crear.
- **pygame.Surface:** Función que crea una superficie en la ventana, por defecto es de color negra.

```
#=============================================================================================#
#                                    Inicializa PyGames.-                                     #
#                                Inicializa el Engine de PyGame.                              #
#=============================================================================================#

def pygame_init():
    pygame.init()
    pygame.mixer.init()
    pygame.display.set_caption('Red Bounce Ball - Luciano Revillod Jerez')
    pygame.mouse.set_visible(False)
    return pygame.display.set_mode(res)

#=============================================================================================#
#                                    Funcion MapInit                                          #
#                      Crea y retorna una surface donde se imprimira el mapa                  #
#         Entradas: (AnchoX,AnchoY en Pixeles)   Salidas: Surface de las medidas indicadas    #
#=============================================================================================#

def MapInit(AnchoX,AnchoY):
    return pygame.Surface((AnchoX,AnchoY))
```

- **pygame.event.get ( )**: Identificador de eventos dentro del ciclo, ejemplo:
  clickear la X (QUIT).
- **pygame.event.wait ( ):** Función que detiene todo evento en ejecución, puede ser utilizada para crear una función de pausa.
- **pygame.key.get_pressed ( )**: Función que verifica si se presiona alguna tecla.
  En el código se usó para mover al jugador.

- **pygame.quit ( ):** Función que frena la ejecución de pygame o sea que, cierra el programa.
- **pygame.time.delay ( )**: Esta función se utilizó para solucionar un problema de velocidad y tiempo, básicamente ralentiza las acciones en milisegundos, en el caso del códigos son 25 ms, de esta forma el jugador se mueve un poco menos fluido, pero es más controlable.
- **pygame.display.flip ( )**: Actualiza la pantalla.

```
#=============================================================================================#
#                                        Funcion Pausa                                        #
#=============================================================================================#

def Pausa():
    while 1:
        e = pygame.event.wait()
        if e.type in (pygame.QUIT, pygame.KEYDOWN):
            return

ckey = pygame.key.get_pressed()#.................. Verificador de pulsaciones

while ToF:

    if ckey[pygame.K_q]: #........................ Atajo para cerrar el programa
        ToF = False

    ev = pygame.event.get() #..................... Capturador de eventos
    for e in ev:
        if e.type == pygame.QUIT:
            ToF = False
        if e.type == pygame.MOUSEMOTION : mouX,mouY = e.pos #Event Mouse.

    if menu == True: #............................ Si la variable menu es True:
        pygame.mouse.set_visible(True)#........... Mouse Visible.
        ball.Updateball(initball) #............... Comienza el update del bot de inicio.
        Menu() #.................................. Blit de background del menu.
        ball.Pintaball(initball,Screen,sprt) #.... Se pinta el sprt sobre la posicion del bot (Update).

        if boton_manual.draw(Screen): #........... Si se presiona el boton manual:
            Manual()#............................. Blit de background del manual.
            pygame.display.flip()
            Pausa()#.............................. Pausa la img por tiempo indefinido.

        if boton_iniciar.draw(Screen):#........... Si se presiona el boton manual:
            pygame.mouse.set_visible(False)#...... Mouse se vuelve invisible
            menu = False #........................ Variable menu toma valor False.

    else:    #................................... Por lo tanto, si menu es falto se ejecutan
        #..................................... el resto de las acciones update y da inicio al level01.
        Background() #........................... Blit de background del mapa.
        mundo.draw(Screen) #..................... mundo corresponde a la clase mundo, por lo tanto
        mundo.deadpointdraw(Screen) #............ las 4 funciones rellenan/dibujan el mapa con los
        mundo.flagdraw(Screen) #................. distintos sprites/rect correspondiente.
        mundo.finaldraw(Screen) #................
        pelota.update(Screen,mundo,sprt) #....... Update de la posicion e imagen de la pelota (player)

    pygame.time.delay(22)
    pygame.display.flip()
pygame.quit
```

- **pygame.transform.scale ( )**: Función que reescala imágenes, recibe dos parámetros de entrada, una imagen/sprite y el tamaño en píxeles a reescalar.
- **get_rect ( )**: Función de pygame que captura el espacio que ocupa nuestro sprite en el mapa.
- **get_width:** Obtiene el ancho del sprite cargado.
- **get_height:** Obtiene el alto del sprite cargado.

```
#=============================================================================================#
#                              Clase Redball e Iniciacion de parametros                       #
# Funcion __init__():  Entradas: self, x, y  ;  Salidas: update visual (blits en pantalla)   #
#=============================================================================================#

class Redball():
    def __init__(self, x, y): #........................... Se inicializan los parametros de la pelota.
        img = pygame.image.load("./sprt/S07.png") #........ Carga de imagen asignada a la pelota.
        self.rescale = pygame.transform.scale(img, (20, 20)) #... Rescala la imagen cargada.
        self.rect = self.rescale.get_rect() #.............. Se obtiene el "rect" de la img reescalada
        self.ancho = self.rescale.get_width() #............ Se obtiene el ANCHO del "rect".
        self.alto = self.rescale.get_height() #............ Se obtiene el ALTO del "rect".
        self.rect.x = x #.................................. Posicion de la pelota en eje X.
        self.rect.y = y #.................................. Posicion de la pelota en eje Y.
        self.gravity = 0 #................................. Gravedad Inicial de la pelota.
        self.salto = False #............................... Valor False, se usa para checkar si se salto.
        self.vida = 3  #................................... Vida inicial del jugador.
```

- **collliderect:** Función de pygame que identifica la colisión entre objetos con rect.

```
#=============================================================================================#
#                                          DEADPOINTS                                         #
#=============================================================================================#

            for i in mundo.deadpoint: #............ Se recorren los elementos de la lista deadpoint.
                if i[1].colliderect(self.rect.x + dirX, self.rect.y, self.ancho, self.alto):
                    self.vida -= 1 #............... Se resta una vida al contador.
                    dirX = 0 #..................... Frena el movimiento en eje X
                    dirY = 0 #..................... Frena el movimiento en eje Y
                    self.gravity = 0  #............ Frena la "Gravedad"
                    self.rect.x = self.rect.x #.... Retorna posicion de inicio en X
                    self.rect.y = 775 #............ Retorna posicion de inicio en Y

                if i[1].colliderect(self.rect.x, self.rect.y + dirY, self.ancho, self.alto):
                    self.vida -= 1 #............... Se resta una vida al contador.
                    dirX = 0 #..................... Frena el movimiento en eje X
                    dirY = 0 #..................... Frena el movimiento en eje Y
                    self.gravity = 0  #............ Frena la "Gravedad"
                    self.rect.x = self.rect.x #.... Retorna posicion de inicio en X
                    self.rect.y = 775 #............ Retorna posicion de inicio en Y
```

- **collidepoint:** Función de pygame que identifica la colisión entre objetos con rect y un punto, en este caso el punto del mouse.

```
#=============================================================================================================#
#                                          Clase Botones                                                      #
#Funcion __init__(self, x, y): Entradas: self, Screen, sprt, img  ; Salidas:  inicializacion de parametros#
#       Funcion Draw(self,Screen): Entradas: self, Screen ; Salidas: blit del boton en pantalla              #
#=============================================================================================================#
class botones():
    def __init__(self, x, y, img): #.......... Inicializacion de parametros.
        self.img = img #...................... Imagen (la funcion la recibe como entrada)
        self.rect = self.img.get_rect() #..... Se obtiene el rect de la imagen cargada.
        self.rect.x = x #..................... Coordenada en X donde se posicionara la img.
        self.rect.y = y #..................... Coordenada en Y donde se posicionara la img.
        self.click = False #.................. Check de clickeo. (True or False)

    def draw(self,Screen): #.................. Se dibujan los botones en la pantalla (Screen).
        button = False  #..................... Representa si se realiza lo que esta indicando el boton.
        pos = pygame.mouse.get_pos() #........ Check de la posicion del mouse en la pantalla

        if self.rect.collidepoint(pos): #..... Check de colision entre el mouse y el rect del boton.
            if pygame.mouse.get_pressed()[0] == True and self.click == False: #.... [0] corresponde al click derecho
                button = True #............... Se realiza la accion.
                self.click = True #........... Clickeo del mouse = True

        if pygame.mouse.get_pressed()[0] == False:
            self.click = False
        Screen.blit(self.img, self.rect) #.... Se dibuja por pantalla el boton, recibe una imageny un rect.
        return button
```

- **ctypes.Structure**: Estructura de una clase en ctypes, contiene una función llamada _fields_ donde se ubican los parámetros de inicio junto a su respectivo de dato, ejemplo: (‘VelocidadX’, ctypes. ct_cshort)

```
import pygame
import ctypes

class ball(ctypes.Structure):
    _fields_ = [('SdX',ctypes.c_short),('SdY',ctypes.c_short),
    ('DirX',ctypes.c_short),('SdVel',ctypes.c_short)]

#=============================================================================================================#
#                                       Funcion "ballInit"                                                    #
#                             Asignamos valores a la estructura de la initball.                               #
#                  Entrada: ball() (_fields_) ; Salidas: Inicializacion de parametros                         #
#=============================================================================================================#

    def ballInit(initball):
        initball.SdX = 150  # Posicion (px) inicial de la initball en eje X.
        initball.SdY = 510  # Posicion (px) inicial de la initball en eje y.
        initball.DirX = 1   # Direccion inicial/predeterminada en el eje X.(1 o -1)
        initball.SdVel = 3  # Velocidad Constante de la initball.
        return
```

Conclusión

Python es un lenguaje muy versátil, ya que mediante su librería pygame nos permite crear programas con interfaz gráfica e interacciones con el usuario, si bien no es la forma más óptima de crear un videojuego, pygame nos ayuda a conocer y entender a grandes rasgos cómo funcionan los distintos elementos un videojuego.

En cuanto al desarrollo del proyecto se puede concluir que la metodología de idear, investigar, diseñar, experimentar y desarrollar es una muy buena práctica, ya que siguiendo estos pasos se pueden obtener ideas claras para posteriormente ejecutarlas de manera correcta.

En el proyecto se utilizaron distintas funciones, con ellas se buscó optimizar y realizar de manera óptima las características del juego. Aunque también es importante mencionar que durante la codificación surgieron bastantes imprevistos, ante estos casos es recomendable analizar, identificar el error y buscar una solución, por esta razón código se encuentra ampliamente comentado, de esta forma se busca que cualquier persona con conocimientos en programación pueda entender cada sección del código.

Bibliografía

- Lindstrom, L., Dudfield, R., Shinners, P., Dudfield, N., & Kluyver, T. (2000, 28 octubre). Documentación oficial de pygame. pygame. https://www.pygame.org/docs
- K.S. (2008, diciembre). LucidChart. LucidChart. https://www.lucidchart.com/
- Heller, T. (2009). CtypesLib. Ctypes Library. https://docs.python.org/es/3.10/library/ctypes.html

Repositorio GIT

- https://github.com/MrRevillod/Proyect_Try50k/tree/master