https://www.overleaf.com/project/60c40d24b11f98cedece740b

Instituto Tecnológico de Costa Rica

Taller de Programación

Grupo 5

Documentación Proyecto Brick Car Game

Profesor:

Jason Leiton Jimenez

Estudiantes:

Randall Bryan Bolaños López 2019043784

I Semestre

2022

Introducción:

A lo largo del tiempo la humanidad ha tenido la necesidad de buscar entretenimiento, debido a esto se crearon los videojuegos, una gran forma de entretenerse desde la comodidad de su hogar o con sus amigos. Con la creación de los videojuegos llegaron muchos tipos de los mismos, juegos de estrategia, juegos de disparos e incluso juegos online competitivos.

Uno de estos géneros son los juegos de carreras, hoy en día estos juegos de carreras son simuladores muy parecidos a la vida real, tanto que se puede conectar instrumentos para así mejorar esta simulación, tales como pedales o volantes, de tal manera que la inmersión sea más realista y divertida sin la necesidad de gastar miles de dolares para comprarse un auto ultimo modelo, por solo unos cuantos dolares puedes tener esta simulación y sentirlo como si fueras un conductor de carreras profesional.

Desafortunadamente, este tipo de simulaciones no siempre han sido así, este tipo de juegos de carreras donde tienes que evitar obstaculos ya sean de la pista o los mismos vehiculos que intentan ir más rapido que vos y ganarte en la competición en su momento fueron muy revolucionarios, pero hoy en día se quedan atrás con lo que se espera para un videojuego de carreras. Pero es importante recordar que así empezó este subgenero de las simulaciones en tiempo real, empezó como un juego de carreras con el unico proposito de no colisionar contra objetos.

El objetivo del jugador era simple, el carro siempre está avanzando, mientras que otros carros están en la misma pista, el jugador debe de esquivarlos para poder avanzar en la carrera, de tal manera que el jugador debe decidir en cual carril desea incorporarse en un momento indicado, y en el caso de que no lo haga a tiempo, perderá la partida.

Este proyecto está inspirado en esos videojuegos arcade de carreras, más especificamente en los juegos de formula 1 de la consola Brick Game, este juego utiliza la interfaz de tetris, es decir, los coches están formados por bloques grises y negros, los cuales estan en movimiento constante simulando una competición.

En este proyecto se le hizo un homenaje a dicho juego junto con algunas adiciones diferentes, tales como el hecho de que se posee "marchas"donde puedes cambiar de velocidad, de igual manera se posee un contador de tiempo, puntos y aparece tu nombre ingresado de usuario.

Al igual que los juegos de ese entonces, este juego tiene como objetivo terminar la carrera, para eso tiene que superar varios niveles los cuales cada vez va a ir aumentando la dificultad de esquivar los obstaculos, el jugador deberá de ir mejorando cada vez su tiempo de reacción, de tal manera que será capaz de reaccionar más rapido y efectivo cada vez que aparezca un nuevo obstaculo.

En dado caso de que pierda la carrera, el puntaje y el nombre de usuario se va a guardar, y si es de los mejores 7 puntajes entonces va a aparecer en la ventana de mejores puntajes, esta ventana solamente podrá verla una vez termine la carrera, dandole click a un boton el cual lo lleva a los scores. En esta misma ventana puede reiniciar el juego sin necesidad de salirse, y tratar de mejorar su score, igualmente, aunque reinicie el juego el score anterior estará guardado, no lo va a perder por solamente reiniciar.

El juego posee 4 ventanas, una ventana principal, una ventana de créditos, una ventana donde el juego

tiene funcionamiento y la ventana donde aparecen los mejores puntajes. En la ventana principal se tiene un fondo con la imagen relacionada a las carreras, un cuadro en blanco donde se puede guardar el usuario y varios botones, uno para poder escoger la dificultad del juego, otro para poder acceder a la pantalla de créditos y por ultimo uno donde lo va a llevar a la pantalla del juego.

La ventana de creditos va a aparecer toda la información respecto al proyecto e información respecto al autor del mismo. La ventana del juego va a tener varios cuadros, los cuales son los del cronometro, puntaje y donde aparece el usuario, de igual manera estos cuadros se actualizan de ser necesario mientras que los obstaculos van a apareciendo en la carrera, simulando una sensación de avance en la carretera. Por ultimo, al finalizar la carrera va a aparecer un cuadro donde le muestra el puntaje y varios botones, uno para reiniciar la carrera y otro para ir a la pantalla de puntuaciones.

La pantalla de puntuaciones va a actualizar cada vez que termina una carrera, y va a aparecer los mejores siete nombres con los mejores puntajes correspondientes.

Conclusiones:

- Se pueden colocar imágenes con el label, pero los archivos .png pierden su transparencia.
- Es mas útil colocar imágenes con canvas debido a que mantiene la transparencia de las imágenes
- La librería playsound es bastante útil para reproducir sonidos, pero no funcional para otro tipo de acciones.
- Es mejor utilizar la librería vlc ya que tiene mejores opciones para su uso
- La creación de un botón es útil para el llamado de diferente cambio de ventana.
- La función de .after es buena para llamar la recursividad de una función, es una buena opción para usar si no desea utilizar hilos.
- La función random es muy útil cuando se desea provocar un "aleatorio" en determinada función.
- La ejecución de todos los hilos al mismo tiempo es un consumo ineficiente y totalmente innecesario.
- Al utilizar recursividad de pila siempre hay que tener presente que en algún momento esta recursividad va a llegar a su fin.
- Se debe asignar como global cualquier variable que vaya a ser utilizada en otra función distinta a la cual se creó en primera instancia.
- La forma de crear el "impacto" de los proyectiles es comparando las coordenadas "x" y "y" de los proyectiles y los objetivos.
- Utilizar la menor cantidad de variables globales es lo mejor, para así no afectar la legibilidad del código.
- Si se delimita los tamaños de la ventana para que no se puedan modificar, las coordenadas "x" y "y" siempre van a ser las mismas no importa el usuario.
- Se pueden crear botones a los cuales se les asigne una función en específica, de tal forma se puede llegar a debugear de una forma más sencilla.
- El uso de "print" es útil para saber donde está fallando el código
- Utilizar archivos de imágenes o música de menor tamaño ayuda a que el programa sea menos pesado en relación de megabytes.

Recomendaciones:

- Se recomienda el uso del código del enseñado en clase para la implementación de imágenes en el código, la cual facilita su uso de muchas maneras.
- La función de movimiento es complicada, debido a que varios objetos deben de mover al mismo tiempo, por lo que se recomienda utilizar .after para que esto llegue a servir.
- La utilización de programas externos para crear los sprites de las imágenes es muy útil, en este trabajo en concreto se utilizó Photoshop CS6 para la creación de los png.
- Se recomienda Paint.net para el mapeo del tablero, ya que esta aplicación te dice las coordenadas "x" y "y" de un punto específico en el que tienes el puntero del mouse.
- El uso de archivos .mp3 para la música es bastante útil ya que no pesan demasiado y tampoco producen un gran consumo de recursos.
- Se recomienda usar la librería vlc para las funciones de música, la cual permite utilizar archivos .mp3.
- Para la utilización de la librería vlc en Python es necesario tener instalado vlc en el ordenador, ya que vlc es en realidad un reproductor de música de Windows.
- Utilizar la función destroy() es una buena alternativa para el cambio de ventana principal a ventana de juego.
- Se recomienda el tener un cronograma donde se tenga un objetivo claro de lo que se va a hacer por día, de tal forma para que no se desperdicie tiempo, ya que los problemas son muy comunes y eso provoca atrasos.
- Se recomienda el uso de variables para aquellos aspectos donde van a ir cambiando a lo largo del programa, tales como la posición, el nivel de dificultad y el tipo de obstaculo
- Se recomienda el modulo usado para guardar los puntajes del juego, de tal forma que siempre van a estar los mejores puntajes.
- Aspectos sencillos como la incorporación de botones con cuadros de texto es mejor realizarlos en los primeros días de proyecto para no perder tiempo después en aspectos sencillos.
- Se recomienda la creación de clases para tener un mayor orden de código.
- Si se desea utilizar una librería de tal forma que solo quiera reproducir la música, también se puede utilizar playsound, el problema con esta librería es que solo posee una función y es la de reproducir.
- La utilización de "bind" para un evento es bastante recomendable
- Siempre poner la información del autor, para así no tener problemas de un plagio para tiempo futuro.

Analisis de resultados:

Los resultados producidos por el proyecto son bastante favorables. El código compila de una forma adecuada y procede a hacer funcionar las funciones de la misma.

El movimiento de los personajes y el spawn obstáculos fue complicado de lograr, pero se logró mediante la creación de clases y eventos. Se creó una ventana de login en la cual si el jugador no tiene una cuenta, el sistema deja el espacio en blanco pero sigue guardando las puntuaciones, y si ya posee una cuenta, entonces se crea de nuevo el usuario, y guarda el puntaje nuevo.

La creación de botones donde muestra los créditos del autor. De igual manera cuando se logra iniciar sesión en el juego, con presionar el botón play empieza el juego. El movimiento de los obstaculo parece ser constante en su mayoría, y cada uno tiene su spawn en el eje "x"distinto, de tal forma que spawnean en dos carriles al mismo tiempo.

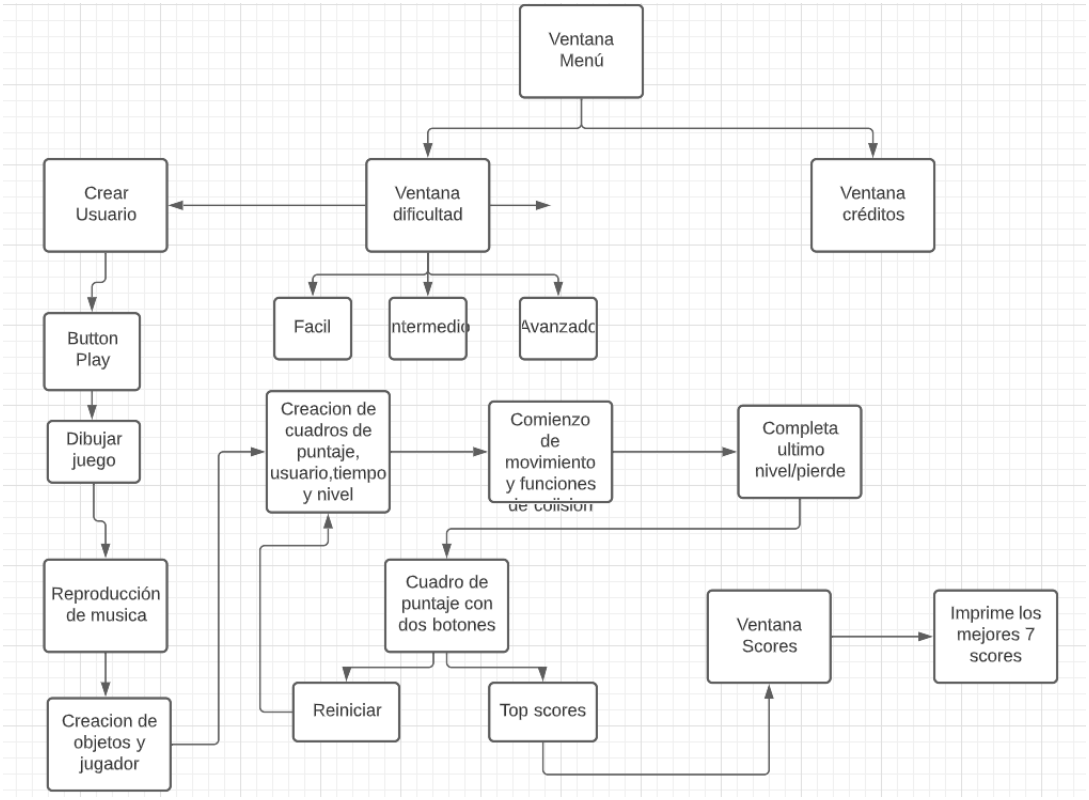
También se logró compilar el código del para que el puntaje se actualice cada vez que los obstáculos desaparecen de la pista, sumandose constantemente.

Se logró el sonido de la música en bucle desde el principio de la ventana principal, y se incorporó una funcion para que cuando empiece el juego, sea una musica diferente mientras se está jugando.

También se logró hacer un botón para seleccionar la dificultad del juego, cambiando así la velocidad de movimiento de los obstáculos.

De igual manera, se muestra en la pantalla de juego el nivel actual en el que se encuentra y el cronometro en el momento que empieza el juego, dando así una muestra de en que nivel se encuentra y cuanto lleva de tiempo la carrera.

Diagrama de módulos:



Plan de pruebas:

- En primer lugar, se realiza la debida implementación de las librerías que serán necesarias para el proyecto. Después de esto se pasó a hacer la primera ventana del juego, la ventana que va a desembocar en el menú y que va a contener varios botones.

```
ventana=Tk()
ventana.title("Brick Car racing game")
ventana.geometry("1200x900+390+40")
ventana.resizable(width=NO, height=NO)
#Crear imagen de fondo
C_menu=Canvas(ventana, width=1200, height=900, bg='black')
C_menu.place(x=0, y=0)
C_menu.fondo=cargarImg('menu.png')
imgCanvas= C_menu.create_image(0,0, anchor=NW, image= C_menu.fondo)
#Ventana usuario
data=StringVar()
textUsuario= Entry(C_menu,textvariable=data)
textUsuario.place(x=500,y=500)
saveUsuario=Button(C_menu, text= "Guardar Usuario", command=UsuarioGod)
saveUsuario.place (x=550, y=550)
```

- Después de esto se procedió a crear los botones que van a desembocar en los créditos, de igual manera los botones para la dificultad.

```
##############Ventana Ayuda###################
ventanabout=Toplevel(ventana)#Ventanda de creditos
ventanabout.title("About")
ventanabout.geometry("1200x900+390+40")
ventanabout.resizable(width= NO, height=NO)
C_about=Canvas(ventanabout,width=1200, height=900,bg="Pink")
C_about.place(x=0, y=0)
C_about.fondo=cargarImg("About.png")
imgAbout=C_about.create_image(0,0, anchor=NW, image=C_about.fondo)
ventanabout.withdraw()
```

```
############VentanaDificultad#############
ventanaDificultad=Toplevel(ventana)
ventanaDificultad.title("Seleccione Dificultad")
ventanaDificultad.geometry("1200x900+390+40")
ventanaDificultad.resizable(width=NO,height=NO)
C_dificultad=Canvas(ventanaDificultad,width=1200,height=900, bg="Pink")
C_dificultad.place(x=0,y=0)
ventanaDificultad.withdraw()

selecdificultad=Button(C_menu,width=8,height=4,text="dificultad",command=FunDificultad).place(x=800,y=
facil=Button(ventanaDificultad,width=10, height=6,text="Facil",command=Dificultadprin).place(x=200,y=1
medio=Button(ventanaDificultad,width=10, height=6,text="medio",command=Dificultadmedia).place(x=200,y=
dificil=Button(ventanaDificultad,width=10, height=6,text="avanzado",command=Dificultadavanzada).place(

PlayAbout=Button(C_menu,width=6,height=2,bg="Pink",text="Creditos",command=about).place(x=1100, y=800)
Jugar=Button(C_menu, width=6,height=2,bg="Red",text="Empezar",command=lambda:prejuego()).place(x=800,y
PlayAtras=Button(C_about,width=6,height=2,bg="Pink",text="Volver atrás", command=lambda: Atras()).plac
```

- Después de haber comprobado que los botones sirven de tal forma que muestran las ventanas de información correspondientes, se procedió a programar las primeras funciones del juego, las cuales serían la música y el login que deberá usar el usuario para ingresar al juego.

```python
def musicaInicial():
    "Funcion que permite la reproducción de musica en la pantalla principal"
    global player
    if HiloVivo== False:
        player.play()
        time.sleep(3)
        player.stop()
        return musicaInicial()
    if HiloVivo==True:
        return MusicaJuego()
def HiloInicialMusica():
    "Hilo para que la musica siga repitiendose constantemente"
    hiloFondo = threading.Thread(target=musicaInicial)
    hiloFondo.start()


def MusicaJuego():
    "Funcion de la musica en la pantalla de juego"
    global playermusicaJuego
    playermusicaJuego.play()
    time.sleep(10)
    playermusicaJuego.stop()
    return MusicaJuego()

def HiloJuego():
    "Hilo de la musica en la pantalla de juego y que detiene la musica de la pantalla principal"
    global HiloVivo
    HiloVivo=True
    if HiloVivo==True:
        hiloJuego=threading.Thread(target=musicaInicial)
        hiloJuego.start()
```

- Una vez listo el login del juego y los botones de la ventana del menú, se procede a crear la clase de la pantalla de juego, en esta clase se van a definir la función de cargar imagen, variables que se necesitarán para videojuego.

```python
def start(self):
    global Dificultad
    self.lose_label.place_forget()
    self.lose_button.place_forget()
    self.score_button.place_forget()
    self.status = True
    self.player_position = 1
    self.car_y = 0
    self.roca_y = 0
    self.score = 0
    self.NivelActual=0
    self.DificultadInicial= Dificultad
    self.DificultadActual=0
    self.VelocidadNivel=0
    self.MarchaActual=1
    self.y = 0
    self.background.place(x=0, y=0, relwidth=1, relheight=1)
    self.player.place(x=171, y=603)
    self.posx=171
    self.posy=603
    self.score_label.place(x=347, y=50, width=113)
    self.Nivel_label.place(x=225, y=50)
    self.spawn('car')
    self.spawn('roca')
    self.cronometro()
    self.loop()
```

- Después de haber creado todo lo relacionado con la ventana de juego, se procede a crear las funciones encargadas de la generación de los obstáculos y jugadores.

```python
class Game:
    "Clase donde está la mayor parte del juego programado"
    def __init__(self):
        "Creacion de la ventana del juego"
        HiloJuego()
        self.width, self.height = 600, 592
        self.game = Tk()
        self.game.title('BrickCarGameProyecto')
        self.game.geometry("600x900+590+40")  # Tamaño de ventana
        self.game.resizable(False, False)  # No se cambia el tamaño de la ventana
        self.game.attributes('-topmost', True)  # Ventana siempre al frente

        "Funciones donde se indica las teclas para las cuales va a ocurrir el movimiento del carro"

        self.game.bind('<KeyRelease-a>', lambda event: self.left())
        self.game.bind('<KeyRelease-A>', lambda event: self.left())
        self.game.bind('<KeyRelease-w>', lambda event: self.Arriba())
        self.game.bind('<KeyRelease-W>', lambda event: self.Arriba())
        self.game.bind('<KeyRelease-s>', lambda event: self.abajo())
        self.game.bind('<KeyRelease-S>', lambda event: self.abajo())

        self.game.bind('<KeyRelease-d>', lambda event: self.derecha())
        self.game.bind('<KeyRelease-D>', lambda event: self.derecha())
```

- Una vez el juego funciona correctamente, las funciones de score y tiempo se actualizan hasta que haya una colision o se termine la carrera, de tal manera que se crea una ventana para ir a la ventana de puntaje o se reinicia la carrera

```python
    "funcion que va a crear la ventana de derrota o victoria, elimina los objetos y al jugador"
    def lose(self):
        self.lose_label.place(x=24, y=200, width=310, height=120)
        self.score_button.place(x=71, y=295, width=100)
        self.lose_button.place(x=171, y=295, width=100)
        self.car.place_forget()
        self.roca.place_forget()
        self.player.place_forget()
        self.status = False
    "Creacion de todo lo relacionado con los obstaculos"
```

- De esta manera, si se decide que uno quiere ver los mejores scores, se procede a crear una ventana donde se va a tomar todos los 7 mejores usuarios con mejores puntuaciones y van a aparecer en pantalla con sus puntuaciones correspondientes.

```python
"La funcion donde va a comparar los resultados anteriores y el actual para imprimirlo en una nueva
def highscore(self):  # funcion que comprueba si se deben guardar
    with open('puntajes.json') as file:  # abre el doc
        puntajes = json.load(file)
        name=data.get()

    if self.score > puntajes['Scores'][0]:  # si el puntaje es mayor al mas alto lo guarda y corre

        puntajes['Scores'] = [self.score] + puntajes['Scores'][:-1]
        puntajes["Nombres"] = [name] + puntajes["Nombres"][:-1]
        with open('puntajes.json', 'w') as file:
            json.dump(puntajes, file)
    elif self.score == puntajes['Scores'][0]:  # si es igual no lo guarda
        pass

    elif self.score > puntajes['Scores'][1]:  # si es mayor al segundo deja al primero y corre los

        puntajes['Scores'] = puntajes['Scores'][0:1] + [self.score] + puntajes['Scores'][1:-1]
        puntajes["Nombres"] = puntajes["Nombres"][0:1] + [name] + puntajes["Nombres"][1:-1]
        with open('puntajes.json', 'w') as file:
            json.dump(puntajes, file)
    elif self.score == puntajes['Scores'][1]:  # si es igual no lo guarda
        pass
```

Bitácora:

- 10/04/2022 Se realizó el recorte y la toma de datos multimedia necesarios para el proyecto, de igual forma se realizó el cambio de formato necesario para que funcione en el código.
- 12/04/2022 Se realizó la incorporación de la ventana principal junto con el tablero como fondo, además de la implementación de la música de fondo de manera permanente.
- 15/04/2022 Implementación de los objetos y el jugador en la pantalla de juego, y el mapeo del tableo por coordenadas (x,y).
- 18/04/2022 Se hizo la ventana del menú principal y se incorporó un botón de tal manera que se pueda cambiar a la ventana del juego, la implementación de una función de movimiento para que los obstáculos se movieran se habilitó el sonido.
- 22/04/2022 Incorporación de los botones de ayuda y créditos. Incorporación de movimiento de jugador.
- 25/04/2022 Se presentó un problema a la hora de programar la generación de obstáculos aleatorio
- 27/04/2022 Intento de llegar a la solución del problema del día anterior sin ningún éxito.
- 30/04/2022 Se logró resolver el error después de varios días intentando solucionarlo, ahora los obstáculos que se crean en carriles distintos.
- 01/05/2022 Los obstáculos se implementan y se mueven automáticamente en el momento que empieza el nivel.
- 04/05/2022 Se hacen varias funciones para poder movilizar al jugador con las teclas WASD y así manejarlo con esas teclas
- 06/05/2022 Se procede a hacer las funciones de las marchas, para que le jugador avance más o menos según la marcha
- 08/05/2022 Se hacen las funciones de colision, comparando el carril y el eje z"donde se encuentra los objetos y el jugador, provocando así que aparezca la pantalla de "perdiste"
- 10/05/2022 Se crea la ventana de scores, en el cual se muestra los mayores scores historicos del juego, junto con la implementación de funciones con respecto a la musica y el cronometro.

Fuentes consultadas:

- Ïs there any way to kill a Thread?"Stack Overflow. https://stackoverflow.com/questions/323972/is-there-any-way-to-kill-a-thread (accedido el 5 de mayo de 2022).
- cctmexico. TkInter para Python: ¿Cómo activar una segunda ventana, con un botón y una condición? (Básico). (27 de mayo de 2017). [Video en línea].https://www.youtube.com/watch?v=hky6aQw65lw
- "save a csv from dataframe using tkinter Code Example". Grepper — The Query Answer System for the Coder Community. https://www.codegrepper.com/code-examples/python/frameworks/-file-path-href/save+a+csv+from+dataframe+using+tkinter
- "VLC module in Python - An Introduction - GeeksforGeeks". GeeksforGeeks. https://www.geeksforgeeks.org/vlc-module-in-python-an-introduction/
- "Python Tkinter — Mover objetos usando el método Canvas.move() – Acervo Lima". Acervo Lima – La mayor colección de tutoriales. https://es.acervolima.com/python-tkinter-mover-objetos-usando-el-metodo-canvas-move/