**INSTITUTO TECNOLOGICO DE COSTA RICA**

**INTRODUCCIÓN A LA PROGRAMACIÓN**

**PROYECTO I**

**STEVEN CASTRO**

**2018225404**

**I SEMESTRE 2018**

Contenido

[Introducción: 3](#_Toc514111540)

[Descripción del problema: 3](#_Toc514111541)

[Análisis de resultados: 4](#_Toc514111542)

[Bitácora de actividades: 11](#_Toc514111543)

[Estadística de tiempos: 12](#_Toc514111544)

[Conclusión: 13](#_Toc514111545)

[Referencias: 14](#_Toc514111546)

# Introducción:

**PyDeathRace** es un juego de carreras en el que el jugador controla a un carro a través de un circuito, este carro puede conseguir puntos si completa una vuelta a través del circuito o si destruye a los otros carros que circulan por la pista, el objetivo del juego es que el jugador consiga la mayor cantidad de puntos en un plazo de 3 minutos de juego.

# Descripción del problema:

Para crear este juego, se necesita hacer uso de librerías en el lenguaje de programación Python, como “Pygame” o “Tkinter” al mismo tiempo que para guardar los puntajes necesitamos hacer uso de un archivo JSON al que le estaremos dando las estadísticas de los jugadores.

Es también necesario un conocimiento básico de cómo funciona el lenguaje de programación Python, ya que todo lo que haremos será basado en ese entorno, así como el uso de operadores matemáticos, lógicos y funciones propias tanto del lenguaje como de los módulos a utilizar.

El carro, o, “sprite”, del jugador, debe ser capaz de poder controlarse recibiendo un “input” del teclado, en la gama de acciones que puede realizar el carro están: moverse hacia delante (acelerar), hacia atrás (frenar y dar marcha atrás), voltear a la izquierda o derecha y disparar.

Los otros carros, o “dummies”, deben ser capaces de seguir una trayectoria predefinida a través de la pista a una velocidad constante, los dummies asimismo deben “desaparecer” si el disparo del jugador los alcanza, cuando esto sucede los dummies suman 100 puntos a la puntuación del jugador.

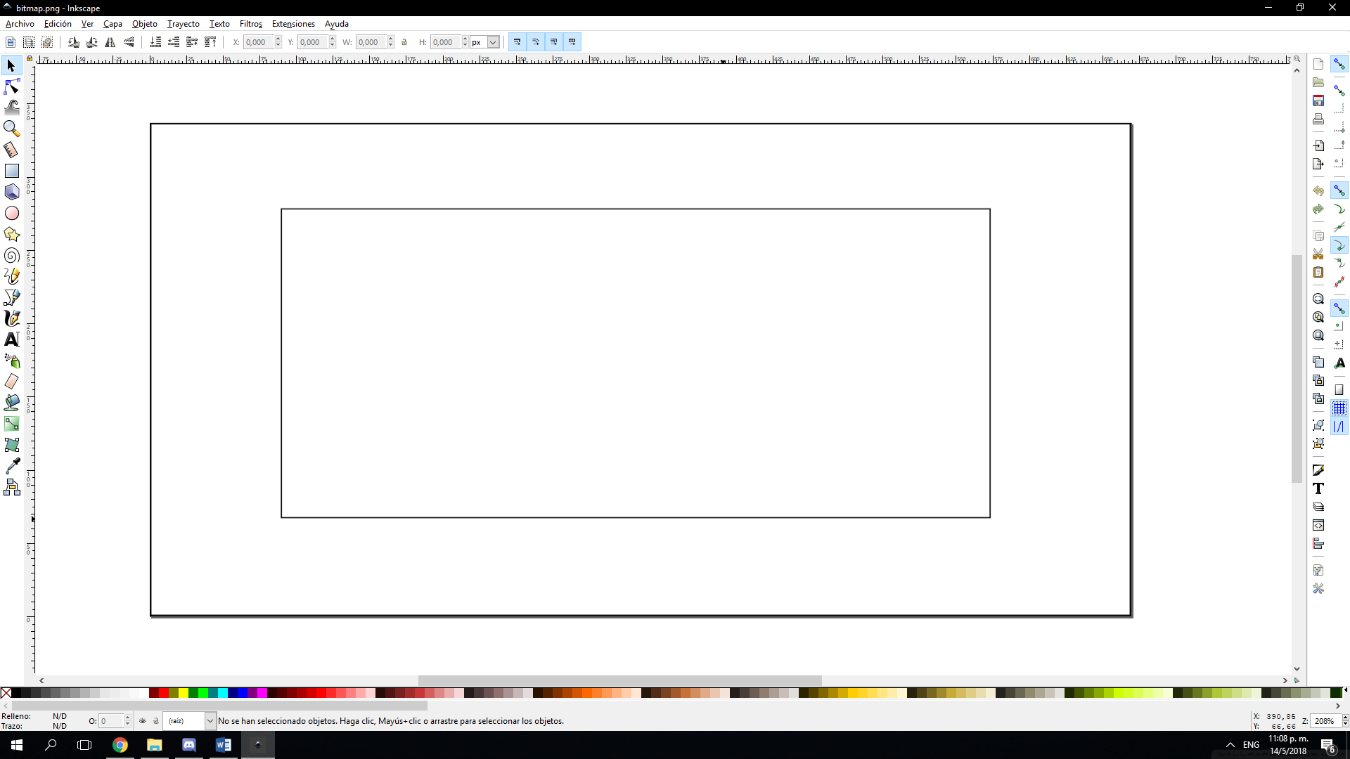
La puntuación del jugador debe ser una variable que aumente cada vez que el jugador hace algo benéfico, como por ejemplo: matar dummies o completar una vuelta; y debe bajar cada vez que el jugador haga algo que le perjudique, como por ejemplo: salirse de la pista o chocar contra un obstáculo.

El juego debe de acabar cuando hayan pasado 3 minutos, o lo que es lo mismo: 180000 ticks. Luego de acabar, el nombre del jugador y su puntuación se guardan en el archivo JSON.

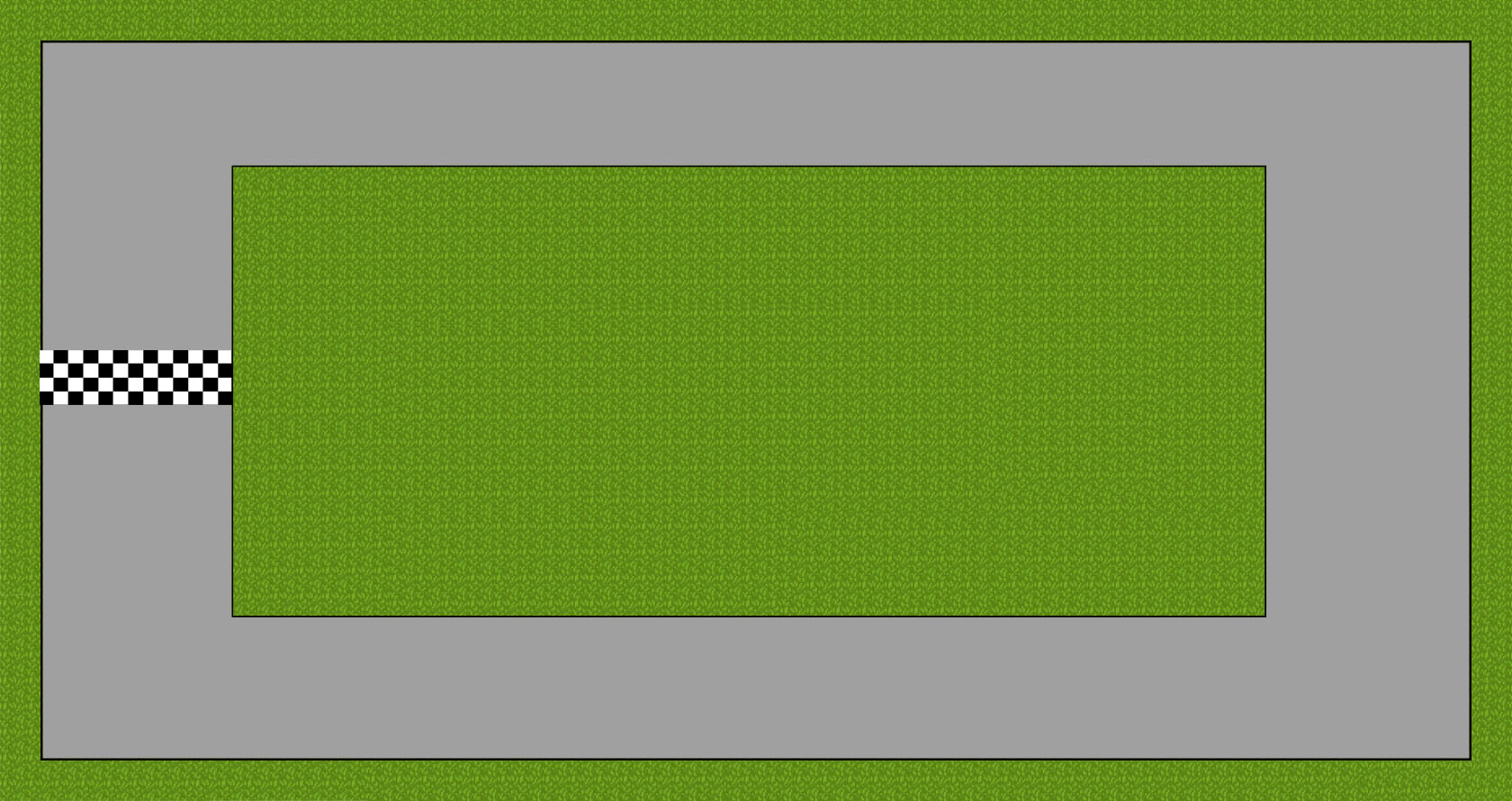
# Análisis de resultados:

No todos los requerimientos del proyecto fueron alcanzados satisfactoriamente, algunos se lograron mientras que otros se alcanzaron en cierto sentido.

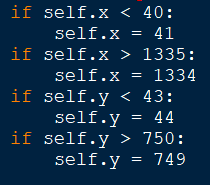
**Funciones del programa:**

1. **“Layout” de la pista:** La pista del juego es una pista rectangular relativamente básica, los bordes fueron hechos en InkScape:

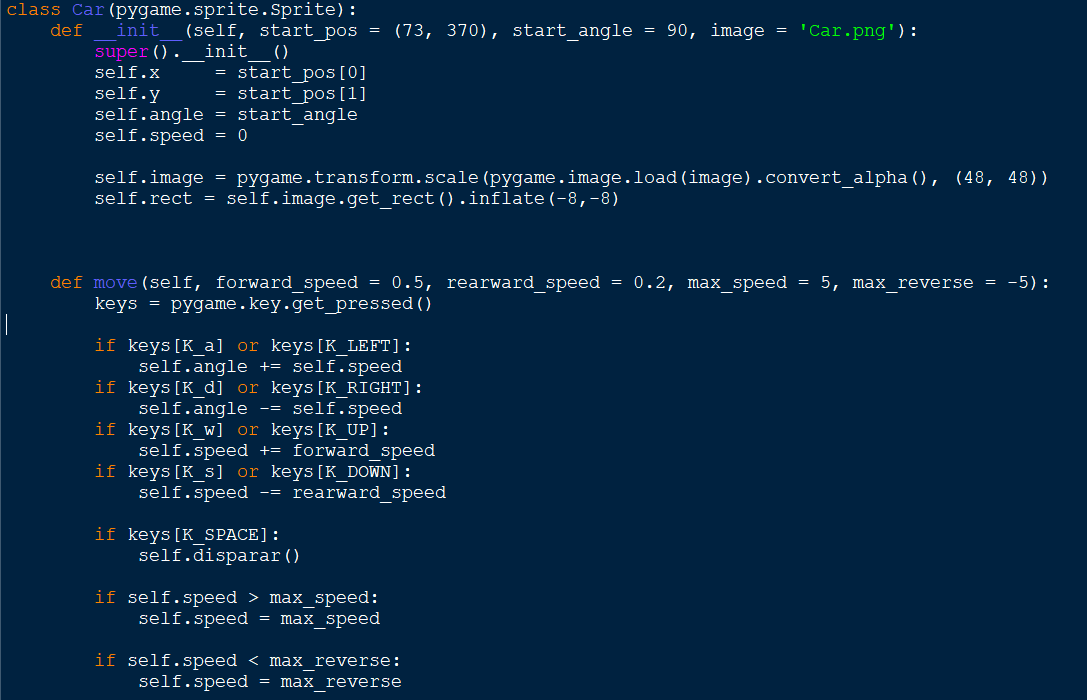
mientras que se coloreó y se añadieron sus texturas en Photoshop, el resultado final es este:



La imagen de la pista se llama con esta línea:

Y se delimitan sus bordes así:

De esta manera el carro no se puede salir de la pista.

1. **Controles del carro:** El carro es capaz de moverse y rotar según la tecla que se le dé como input, esto está dado por este código:

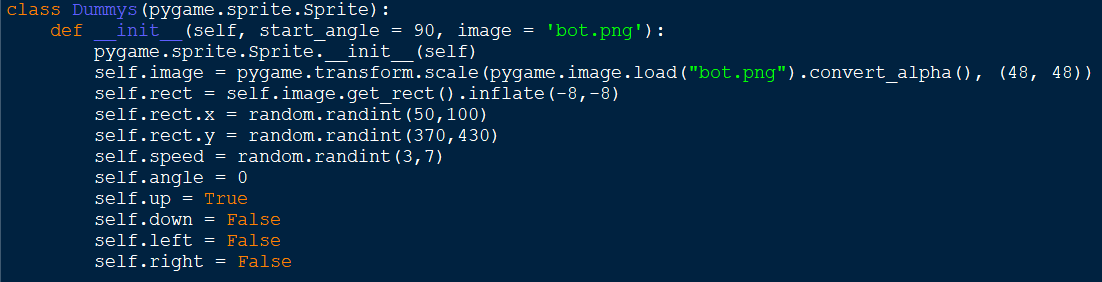
**Controles**

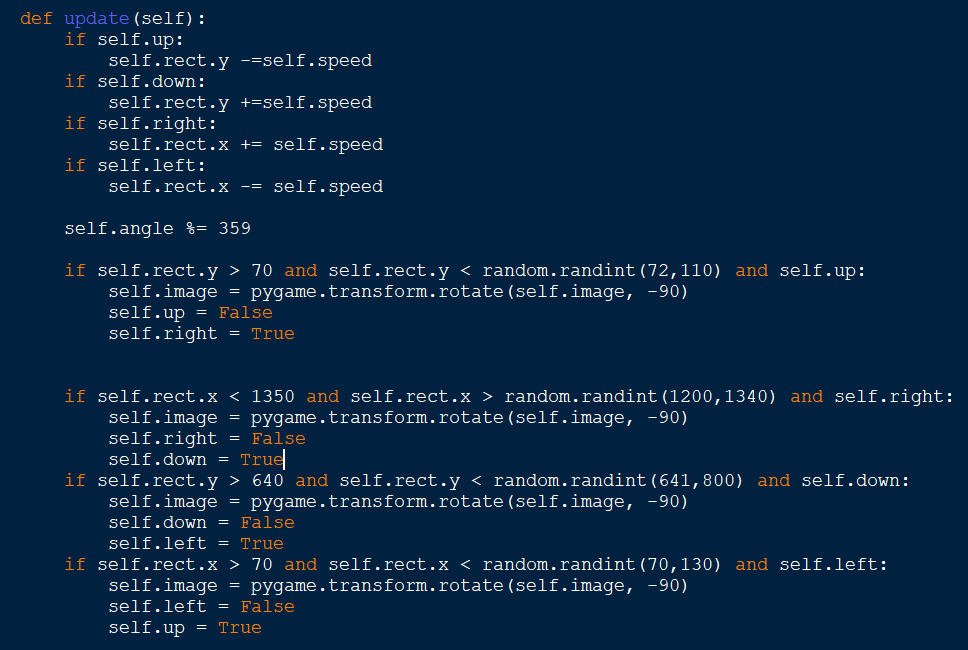
****

**Rotación**

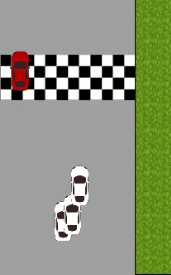
Cabe destacar que el carro también tiene incluido un delimitador de velocidad para que este no acelere infinitamente.

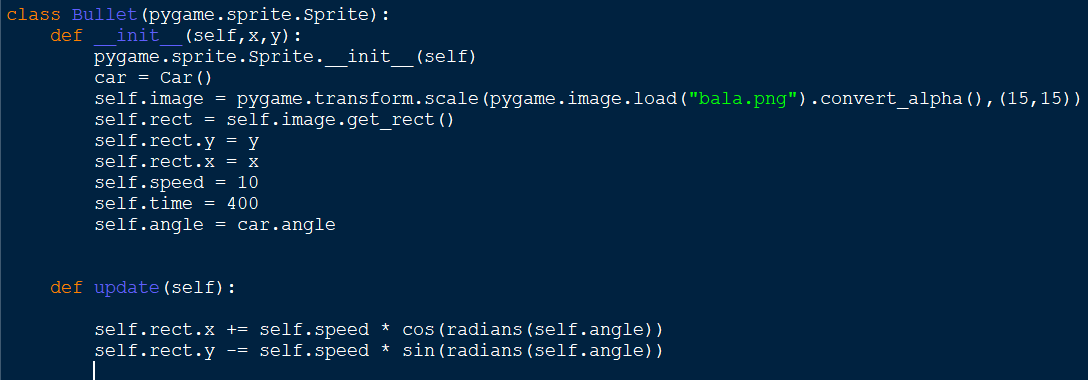


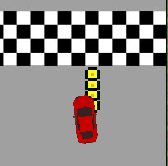
1. **Los vehículos Dummies:** Los vehículos controlados por el juego poseen su propia clase:

Asimismo, estos vehículos poseen una inteligencia artificial básica que define sus patrones de movimiento:

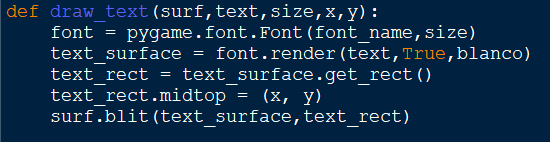
En la práctica, los dummies se verían así:

Siguiendo la trayectoria de la pista

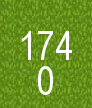
1. **Disparar balas:** Las balas del carro aparecen justo donde el carro está ubicado y su colisión con los dummies es satisfactoria



El único inconveniente es que las balas solo se mueven hacia arriba:

1. **Sistema de puntuación:** El sistema de puntuación está dado por el siguiente código:

Y se muestra en el juego así:

Donde el número de arriba es una cuenta atrás del tiempo que queda en el juego y el número de abajo es la puntuación.

La puntuación es completamente funcional y no posee inconvenientes.

# Bitácora de actividades:

1. **Comprensión de los módulos a utilizar:**

Luego de recibir el proyecto, me dispuse a investigar en internet información sobre los módulos que utilicé para el proyecto, para esto, visité principalmente páginas como <https://www.pygame.org/docs/> y <https://stackoverflow.com/> que contenían dudas muy similares a las que me fueron generadas mientras comprendía el código, lo cual me fue muy útil a la hora de aclarar mis dudas. Esto me tomó aproximadamente **5 horas**.

1. **Búsqueda de Tutoriales:**

Antes de disponerme a crear mi código, busqué tutoriales en youtube acerca del módulo pygame para poder ubicarme más en la materia, los que más me fueron de ayuda principalmente fueron <https://www.youtube.com/playlist?list=PLsk-HSGFjnaH5yghzu7PcOzm9NhsW0Urw> y <https://www.youtube.com/playlist?list=PL46E99FE946C1C946> Esto me tomó aproximadamente **3 horas.**

1. **Programación y solución de errores:** Luego de investigar sobre pygame, me dispuse a crear el código para mi juego, creo que esta fue la parte más extensa ya que a como escribía surgían errores que me confundían por lo que duré bastante tiempo tratando de solucionarlos ya que tenía que devolverme a ver tutoriales si no entendía algún concepto o el porqué de un error. Esto me tomó aproximadamente **20 horas**.
2. **Documentación interna y externa:** Luego de haber concretado ya gran parte del código, me dispuse a crear la documentación tanto interna como externa, esta fue la parte que más disfruté ya que implicaba explicar mi código, esto me tomó aproximadamente **3 horas**.

# Estadística de tiempos:

|  |  |
| --- | --- |
| Análisis de requerimientos | 2 Horas |
| Diseño de la aplicación y diagrama de clases | 4 Horas |
| Investigación de funciones | 5 Horas |
| Programación | 15 Horas |
| Documentación interna | 1 Hora |
| Pruebas | 5 Horas |
| Elaboración documento | 2 Horas |
| **TOTAL** | **34 Horas** |

# Conclusión:

Mientras hay algunas cosas que logré hacer con mi escueto conocimiento acerca de pygame, hay otras funciones que no logré hacer, pero planeo revisitarlas en una futura revisión de este código. Sin embargo, gracias a este proyecto he aprendido las bases de pygame y siento que para la siguiente iteración del proyecto tengo la experiencia necesaria para que no se me dificulte tanto llevarla a cabo, además de que con la lectura del código he profundizado mi habilidad para programar y mi rango de comprensión en esta área ha aumentado considerablemente.

# Referencias:

1 – KidsCanCode(2016).”Game Development with Pygame” [Archivo de video], recuperado de <https://www.youtube.com/playlist?list=PLsk-HSGFjnaH5yghzu7PcOzm9NhsW0Urw>.

2- ChelinTutorials(2014).”Programación de Juegos Pygame” [Archivo de video], recuperado de <https://www.youtube.com/playlist?list=PL46E99FE946C1C946>.