Taller de programación

Proyecto programado #2

Profesor: Antonio Gonzalez Torres

Estudiante: Adrián López Vásquez

Carné: 2016108981

Grupo 04

Fecha entrega: 27 de mayo de 2018

I Semestre 2018

Índice:

1) Introducción:

En este proyecto del curso de taller de programación se continuó el juego desarrollado como primer proyecto del curso. Para esta parte se debió corregir algunos errores presentes en la primera parte, así como el desarrollo de nuevas funciones que permitan que el juego contenga más elementos, se parezca más a un juego normal y permita el desarrollo de habilidades a la hora de la programación.

Para esta parte se enfrentó a diversos desafíos que requieren la búsqueda de información para el desarrollo de este. A diferencia del proyecto anterior se debió enfatizar en los sonidos, colisiones, desarrollo de objetos y la utilización del json para guardar la partida jugada anteriormente. Para esta segunda parte se tuvo menos complejidad que en la primera parte no obstante la inversión de horas fue similar. En este caso la mayor complejidad estuvo en guardar la partida ya que se tuvo que almacenar una gran cantidad de variables para ser usadas nuevamente. Se tuvo que formar una lista con ellas y colocarlas adecuadamente para que el juego funcionara.

Para esta versión del juego se tienen 3 niveles distintos en los cuales se tienen obstáculos que se encuentra en el mapa de manera aleatoria. La puntuación del juego se obtiene mediante vueltas, kills. Una vuelta equivale a 50 puntos, mientras que un kill a un dummy mediante disparo suma 10 puntos.

Para delimitar el mapa se tiene que si se sale de la pista se procede a un rebajo automático de puntos por ende se debe manejar de la mejor manera y rápidamente. Para pasar de nivel se debe jugar 3 minutos sin colisiones, si se da la colisión cuando se terminen los 3 minutos se vuelve al menú principal. Al igual que en la versión anterior se cuenta con un top puntajes y se puede ingresar el nombre del jugador, además en esta versión se incluyen sonidos de aceleración, frenada, música principal y vueltas. Todo esto permite el desarrollo del juego y el aprendizaje de pygame, Python, json y el manejo de objetos

2) Descripción del problema

Este juego se basa en los juegos antiguos de carreras en los que se tiene una vista aérea de una pista, una carrera de varios vehículos con choques, sonidos, menus y que fuera multijugador. Este juego como se menciono anteriormente es la continuación del primer proyecto.

En esta parte del proyecto se debió corregir algunos errores presentes en la etapa anterior. Primeramente, se debe corregir errores en los dummy y disparos, ya que los dummy no poseen una trayectoria definida constate y realizan giros no deseados. Por su parte los disparos no existen un contador que permita la detección de si la bala ha impacto al dummy. Se debe desarrollar un método que permita la actualización de los dummy, el conteo de choques de dummy con balas y que estos desaparezcan si colisionan.

Como segunda parte se debe implementar el uso de sonidos en el juego. Para ello se debe investigar el uso de sonido en pygame, buscar sonidos acordes con el juego, que permitan que este tenga naturalidad. Se debe implementar un sonido de aceleración, freno, sonido ambiental en el menú, y un sonido que permita indicar si el carro ha pasado por la meta.

Siguiendo con esta lógica se debe implementar obstáculos que permitan aumentar la dificultad del juego y con ello se deben introducir tiempos que permitan que si se pasan los 3 minutos se pasa al siguiente nivel del juego. Si un jugador colisiona con un obstáculo va perdiendo vidas, así si el vehículo choca 3 veces se destruye. Para que los jugadores avancen de nivel se deben contar las colisiones y no se deben permitir estas. Como método en el proyecto anterior se desarrolló la detección de colores que permite que en este proyecto si el carro colisiona fuera de la pista se destruya automáticamente.

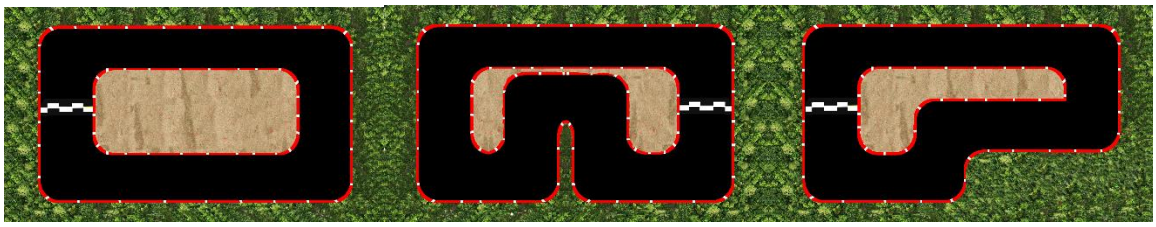
Por ultimo para cumplir con las condiciones mínimas del proyecto se debe permitir guardar la partida, para ellos se debe desarrollar un archivo json que permita guardar las variables de la partida instantaneas y al ser abierto este json se pueda cargar la partida, utilizando las variables guardadas anteriormente permitiendo la continuación del juego. Para este json se deben guardar muchas variables como posiciones, ángulos, tiempos y otras variables de actualización. Se debe

guardar los tipos de obstáculos, sus posiciones, la posición de los jugadores y dummy, puntajes y choques.

Para esta parte se utiliza el módulo mixer de pygame, el random para la generación de números aleatorios, El uso de sprites y groups para el manejo de objetos, el modulo time que permite obtener los clicks del juego y así obtener un reloj. El módulo draw para las actualizaciones y dibujo de rectángulos, el modulo key para la detección de presión en una tecla, el manejo de eventos, el uso de screen.blit para realizar actualizaciones en pantalla, el módulo mouse para detección de presión en las teclas del mouse y la posición directa del mouse en todo momento.

3) Análisis de resultados

Primeramente, se determinaron los errores presentes en la primera parte del juego. Se encontraron deficiencias en los dummy y en la clase de disparos por lo que se desarrollo esta parte ampliamente. Se realizaron tres tipos de pista:



Para estos tres tipos de pista se desarrollo una trayectoria definida de los dummy. Estos avanzan hasta que se encuentra con una posición de la pista lo que les permite girar posteriormente son rotados 90 grado o -90 grados. Con ello los dummy siguen un ciclo. De estas pistas se definió el color negro como su borde natural, así si el carro no se encuentra en este color se realiza un rebajo de puntos y se disminuye su velocidad.

Posteriormente se mejoro la clase de disparos lo que permitió que las colisiones fueran efectivas y cuando la bala detecta el dummy se produce la colisión, el dummy se elimina y se contabilizan las kills.

Se crearon varios vehículos que vienen a ser las imágenes de los dummy y de los jugadores:

Posteriormente se realizó la implementación de sonidos. Como se ve en la documentación de pygame que se muestra en la bibliografía. El mixer permite la manipulación de archivos de sonido.

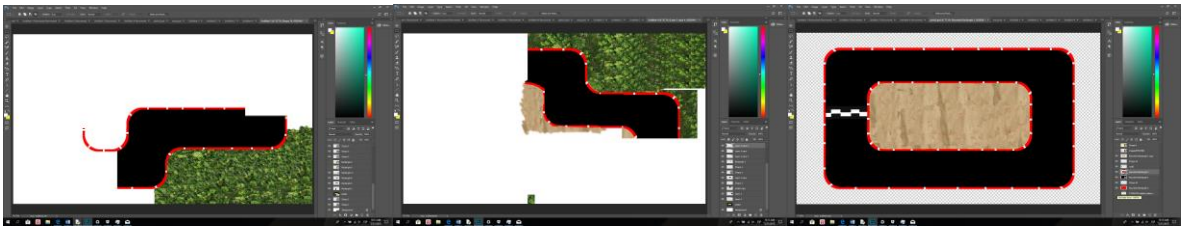
Para este proyecto se decidió utilizar la extensión png para imágenes y la extensión wav para archivos de sonido ya que son fáciles de usar y convertir. Como se muestra en la siguiente imagen se uso un programa para obtener .wav de los videos de youtube.

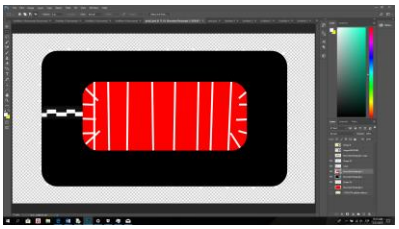
Así se obtuvieron 5 archivos distintos de audio, uno de aceleración, uno de choque, uno de colisión, otro de frenado y uno de meta. Todos fueron convertidos a wav y colocados en la carpeta del proyecto. Como se vio en esta parte los sonidos son reproducidos correctamente, pero para hacer énfasis en los sonidos y evitar la mezcla de estos se debe pausar el sonido anterior para así evitar ruido.

Se utiliza el código:

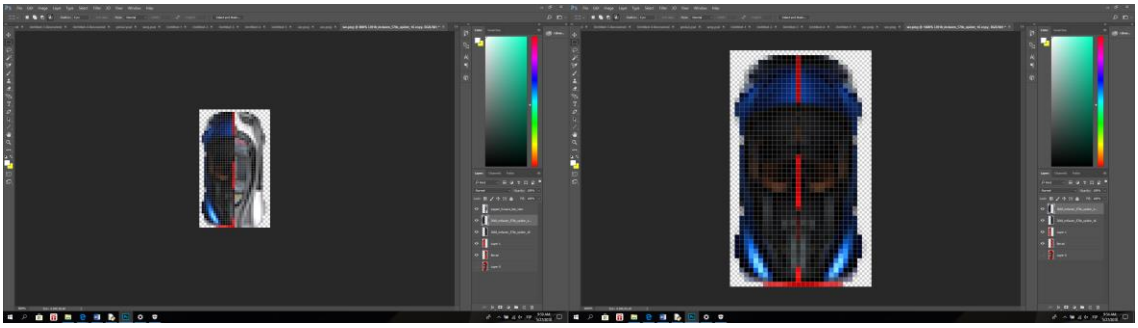
```
sound=pygame.mixer.Sound("name.wav")

sound.set_volume(volumen)
```

Se muestra seguidamente el desarrollo de las pistas usando el programa Photoshop:



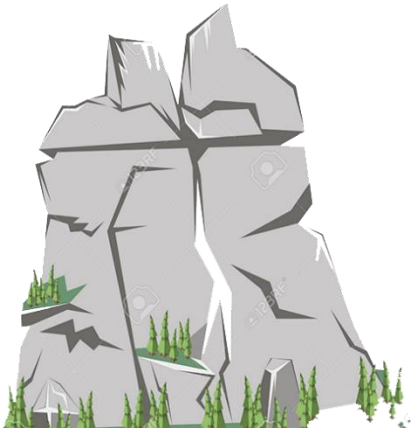


Se muestra seguidamente el desarrollo de los carros:



Estos carros como se ve el nivel de detalle es muy bajo debido a la cantidad de pixeles ya que son muy pequeños comparado con la pista.

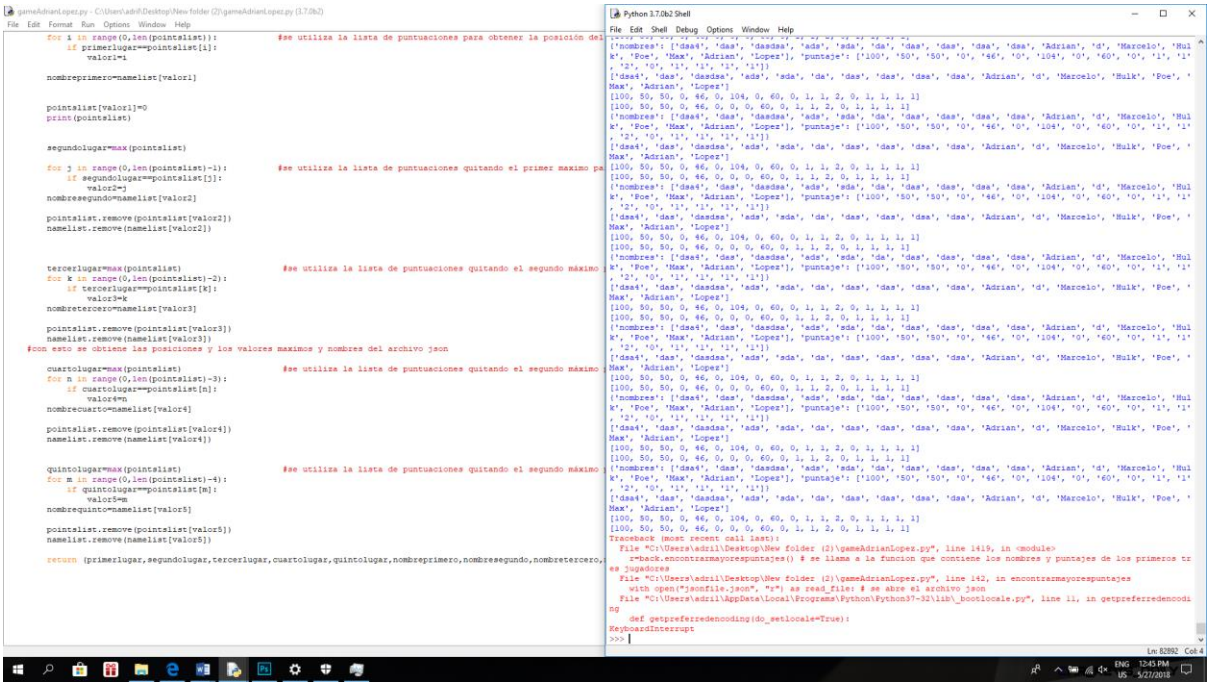
Seguidamente se da el desarrollo de los obstáculos, para ellos se crean varios obstáculos y se editan usando Photoshop. Se definen dos tipos de roca y un tipo de árbol.



Se utilizan números aleatorios para decidir que tipo de imagen se tiene y se generan otros dos números aleatorios para definir su posición en x y en y. Se realiza un rectángulo de estos y mediante el método collide se permite ver si este obstáculo a colisionado con otro objeto lo que permitió la contabilización de choques y la generación de sonido de choque. Al colisionar el

vehículo con el obstáculo fuera de la pista negra, fuera del color (0,0,0) el vehículo desaparece. Si el vehículo colisiona tres veces también desparece esto se logró mediante la implementación de if y un contador de colisiones para cada jugador. Esta función corre correctamente y como se ve en el juego cada vez que se da la colisión se tiene un sonido y menos vidas.

Posteriormente mediante un json se crea un archivo .json que guarda las posiciones de los jugadores en x y y, las posiciones de los dummy, las posiciones de los obstáculos, tipo obstáculos, y los valores de puntaje, kills y vueltas. Como se ve en la siguiente imagen:

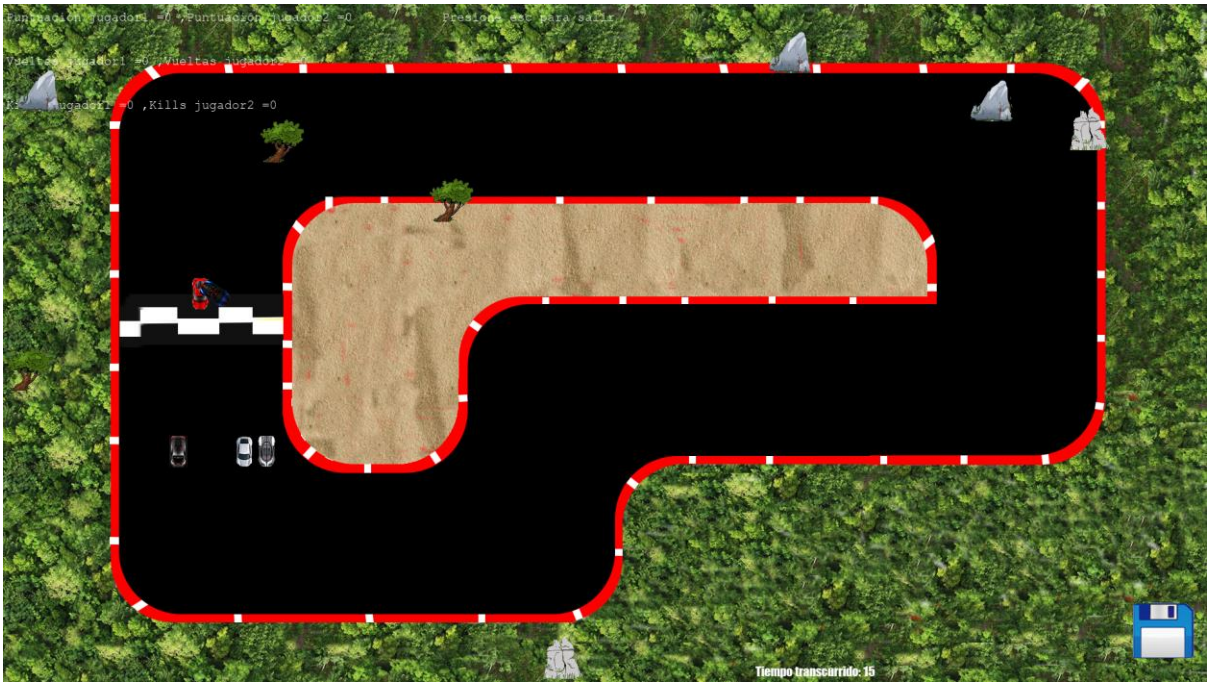


Estos valores guardados son cargados y usados posteriormente continuando así la partida. Con tener las posiciones se puede replicar el escenario anterior. Se debe implementar el tiempo para permitir controlar este y pasar si se cumplen los 3 minutos. AL cargar las partida se realiza una lista con el json y se inicializan las variables.

Por último se debe actualizar el menú añadiendo una opción para cargar la ultima partida guardada. Esta se ve de la siguiente manera:

Se anadio un botón de guardado en el juego que permite guardar la partida si este se pulsa.

Así el juego toma el siguiente look:



Como se ve se tienen las distintas pistas, tiempos, vidas, obstáculos, el botón de guardado, tiempos, la meta y los distintos vehículos, en las imágenes se ven los desplazamientos que realizan los dummy.

4) Bitácora de actividades

Análisis del problema: Se investigó sobre el uso de sonidos en pygame y se indago en el uso de json para la resolución de problemas. Se analizó lo que faltaba en el proyecto anterior y se planearon los métodos para poder solucionar el problema. Duración 1 hora

Investigación: Se investigo la documentación de pygame, se estudiaron ejemplos en los cuales se implementaban sonidos y objeto. Duración=30 min

Creación fondos de las pistas: Mediante Photoshop se crearon las distintas pistas usando triangulos redondeados y distintas imagenes. Duración: 1 hora

Movilidad dummy: Al tener las pistas se realizó un ciclo de los dummy a seguir, cuando estos pasaban por un punto específico de la pantalla cambiaban de dirección y se rotaban completando un ciclo de movimiento. Para ello se tuvo que obtener los pixeles correspondientes a esos cambios de dirección. Duración 2 horas

Disparos: Se modificó la clase disparos y se permitió que se detectara la colisión de los dummy con las balas. Duración 30 min

Búsqueda sonidos e imagenes: Se buscaron los sonidos de aceleración, frenar, colisiones, meta y sonido de fondo más adecuados para el juego y se buscaron imágenes de carros para usar en el juego, además se buscaron las imágenes de los obstáculos. Duración 30 min

Conversión sonidos a wav e implementación de sonidos: Se convirtieron los sonidos a .wav y se implementaron mediante el .play . Duración 30 min

Obstáculos de la pista: Se creó una clase obstáculos y se definió un método aleatorio para la selección de estos y un método aleatorio para las posiciones de este y se definieron funciones para la detección de colisiones. Duración 1 horas

Cambio reglas juego: Se definieron reglas en el juego como cantidad de choques, como se definen los puntos, reducción de puntos si se sale de la pista y como pasar de nivel. Duración 30 min

Guardar partida: Mediante el uso de json se realizo un archivo .json que guardaba las posiciones y puntajes y otras variables del juego para ser posteriormente guardadas. Posteriormente se definio el cargar partida así, si se pulsa un botón en el menú se cargan las variables anteriormente guardadas. Duración 1 hora

Mejoras código: Se realizaron una serie de mejoras al código acortando pasos y usando menos variables. Duración 30 min

Documentación interna: Se comento cada una de las funciones en el código y que realiza el código en cada sección y cual era la intención buscada. Duración 1 hora .

Elaboración documento: Se realizó un análisis de lo realizado en todo el proyecto y se recopilo en este documento. Duración 1 hora y media.

5) Estadística de tiempos

Tabla 1. Tiempo utilizado en el desarrollo de las actividades

Actividad	**Tiempo**
Análisis del problema	1 hora
Investigación	30 min
Creación fondos de las pistas	1 hora
Movilidad dummy	2 horas
Disparos	30 min
Busqueda sonidos e imagenes	30 min
Conversión sonidos a wav e implementación sonidos	30 min

Obstaculos de la pista	1 horas
Cambio de reglas juego	30 min
Guardar partida	1 horas
Mejoras codigo	30 min
Documentación interna	1 hora
Elaboración documento	1 hora y media
Total	**11 horas 30 min**

6) Conclusiones

Como se puede ver se cumplió con los objetivos del proyecto y así se logró guardar la partida, cambiar las reglas del juego, la implementación de 3 niveles y la aparición de varios obstáculos. AL guardar la partida se logró volver a las condiciones anteriores tanto en posiciones como en puntuación y demás constantes. El juego se observa más completo y se tiene sonido de aceleración, freno, choque. Se realizaron las correciones anteriores se logro el movimiento de los dummy y corrección de los disparos. Se crearon nuevos niveles con Photoshop. Se familiarizó con json, Python, lista y demás elementos de Python y pygame lo que permitió no solo crear un juego si no desarrollar habilidades en programación. Se hizo manejo de los métodos de pygame, tiempos, sprites, grupos entre

otros. Como se ve el juego presenta buena estética, movilidad y se cumple con las condiciones mínimas del juego presentada.

7) Bibliografía

(s.f.). Recuperado el 11 de Mayo de 2018, de https://pythonprogramming.net/making-interactive-pygame-buttons/

(s.f.). Recuperado el 7 de Mayo de 2018, de http://www.pygame.org/docs/ref/transform.html

Craven, P. V. (s.f.). Recuperado el 10 de Mayo de 2018, de http://programarcadegames.com/index.php?chapter=introduction_to_sprites&lang=en

Idris, I. (2013). *Instant Pygame for Python Game Development How-To.* Packt Publishing.

Jens, H. (s.f.). Recuperado el 10 de Mayo de 2018, de http://thepythongamebook.com/start

realpython. (s.f.). Recuperado el 10 de Mayo de 2018, de https://realpython.com/python-json/

Robinson, S. (17 de Agosto de 2016). *stackabuse*. Obtenido de Reading and Writing JSON to a File in Python: http://stackabuse.com

Sridhar, J. (11 de Agosto de 2017). *Makeuseof*. Obtenido de https://www.makeuseof.com/tag/json-python-parsing-simple-guide/