Instituto Tecnológico de Costa Rica

Sede Cartago

Primer Proyecto de programación

Monkey Kong

Estudiante: José Fabián Mendoza Mata

Profesor: Jeff Schdmit

Fecha de entrega: 7/04/2019

Grupo: 1

Indice:

Introducción:

Donkey Kong es un juego de video creado por Nintendo en el año 1981. Es un muy básico juego en 2D del género plataformas/niveles que se centra en controlar al personaje sobre una serie de plataformas, mientras evita obstáculos que son lanzados desde la parte superior por un aterrador y gigante gorila. El juego original consiste en que Mario (en el juego, Jumpman) debe rescatar a una dama que había sido capturada por un enorme mono llamado Donkey Kong.

El jugador controla en el juego el movimiento de Mario, que puede moverse a la derecha, a la izquierda, arriba y abajo. El objetivo consiste en llegar a rescatar a la dama sin ser derribado por los obstáculos. Se obtienen puntos por: saltar obstáculos, subir de plataforma y rescatar a la dama.

Descripción del problema

Al inicio del juego debe mostrarse una pantalla de presentación (splash animado) del juego y del desarrollador por cinco segundos.

Las imágenes de la dama, el mono, el personaje que trata de rescatar a la dama, los obstáculos y las plataformas con sus respectivas escaleras, deben ser definidos por el programador y serán un aspecto importante dentro de la calificación.
El programa debe manejar 3 niveles, reutilizando la mayor parte de la interfaz gráfica y de las funciones. Cada nivel debe tener mayor cantidad de plataformas.

El Salón de la Fama va a contener los nombres de los jugadores que han obtenido la mayor cantidad de puntos. En caso de existir empates, se van a mantener los 3 últimos nombres. Los nombres y puntajes deben almacenarse por medio de archivos de texto o planos (investigar).

El juego debe contener una configuración, que va a mostrar al menos los siguientes parámetros que regirán el juego, deben configurarse por medio de alguna opción:

- Cantidad de vidas del personaje (Mario).
- Cantidad de obstáculo por cada nivel.
- Puntos por: subir de plataforma, obstáculo, paso de nivel.

Los parámetros deben almacenarse en un archivo de texto.

Análisis de resultados

1. El punto de partida para la resolución del problema fue la pantalla de inicio, la cual se hizo con la librería de Python tkinter, la cual proporciona elementos utilizados en esta interfaz como:

- Ventana de la interfaz: Esta se construyó a partir del método Tk(), el cual crea una pantalla por defecto de la librería tkinter, esta se puede modificar con los métodos .geometry(para cambiar las dimensiones de ancho y largo de la ventana) o .title(para cambiar el título de la pantalla), esta ventana será únicamente para interacción con el usuario, no se efectuarán animaciones en esta.

- Botones: Se ocuparon para pasar de ventana en ventana durante la ejecución del programa, estos se hicieron con el objeto Button() de la librería tkinter, a este boton se puede modificar el texto que muestra, el color de fondo, el color de letra y se le puede asignar un comando, el cual es ejecutado cuando se presiona, esto para lograr pasar de la ventana de inicio a la de configuración y viceversa.

2. Después de lograr crear la ventana principal se procedió a la elaboración de la ventana de juego, la cual se hizo con la librería externa para Python pygame, esta librería es muy útil para la creación de videojuegos ya que provee métodos para facilitar la tarea de crearlos, algunas de las características que utilizaron fueron:

- pygame.display.set_mode((ancho, largo)): Crea una ventana de pygame, esta se utilizará para mostrar la ejecución del juego al jugador, también se harán validaciones de interrupción del teclado para que el usuario pueda interactuar con el personaje, se

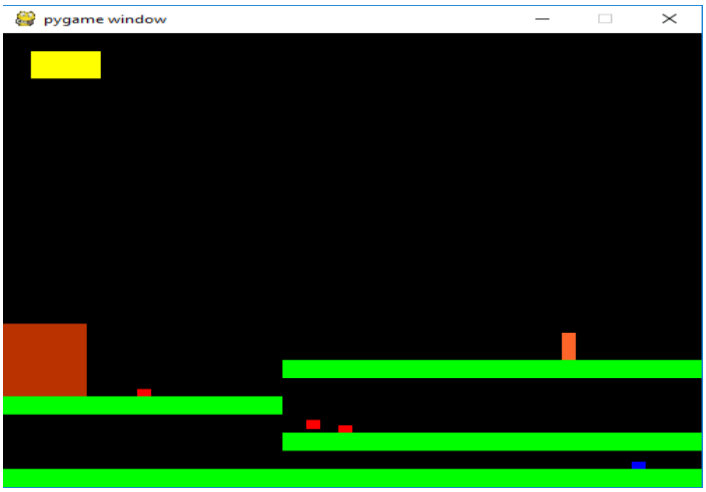
crearán las plataformas y por ende, se revisará los objetos que colisionan con estas, se crearán los objetos de forma aleatoria y demás.

- pygame.Rect(x,y, width, height): Este método crea un rectángulo de la clase pygame, este tiene métodos los cuales nos permite detectar colisiones con otros rectángulos en el juego, todos los obstáculos, plataformas, el jugador y la princesa tienen este atributo para poder interactuar entre sí.

- *ventana*.blit(imagen, rectángulo_ventana): Este método permite actualizar una imagen en una ventana de pygame, esto se hace para dar la impresión de movimiento a la ventana, este se complementa con el método pygame.time.Clock() que permite medir tiempo en pygame y *ventana*.tick(milisegundos) la cual espera una cantidad de milisegundos para actualizar la ventana, para que el jugador pueda realizar acciones sobre ella.

3. Después de crear la ventana se creó la clase Player, la cual tiene atributos y métodos los cuales se cambian dependiendo de los eventos del teclado, este se movía a la derecha e izquierda y tenía gravedad, además se crearon clases Plataforma para probar elementos de la librería pygame como colisiones entre rectángulos.

4. Se mejoró el boceto del programa para implementar la clase Obstacle, la cual crea un obstáculo que al colisionar con el jugador lo matará, además se hicieron rectángulos para representar al mono y la princesa y sus papeles en el juego, el mono da la impresión de tirar los “barriles” aunque esto es sólo una impresión, la princesa es la meta del juego, por lo cual si se colisiona con ella, el juego terminará y el jugador gana.



5. Se creó la ventana de configuración, permite al usuario elegir la dificultad, la cantidad de vidas y el sonido, la dificultad se configuró para cambiar la cantidad de plataformas a subir y la cantidad de barriles que hacen spawn, aunque este parámetro no se indica.en la configuración.

6. Esta es la etapa final del juego, se mejoró el aspecto del juego con el uso de Sprites, esto lo provee la librería pygame y se actualizan dependiendo de las acciones del usuario, ahora los personajes están bien definidos con sus imágenes, así como los obstáculos que son barriles, se logró implementar un sistema de puntos, los cuales da 150 pts por subir de plataforma y 100 pts por cada obstáculo evadido, se implementó el uso de escaleras para poder trasladarse de plataforma en plataforma.

Dificultades encontradas:

En la elaboración del programa se presentaron situaciones que cuales fueron difíciles de arreglar o de modelar, algunas de ellas se presentan a continuación:

- Obtener colisiones entre objetos: Esto fue una dificultad al empezar a modelar el problema, ya que no encontraba un método para poder detectar colisiones entre elementos en la pantalla, esto se arregló rápidamente al encontrar los métodos .colliderect() y .collidelist() que provee Python, los cuales otorgaron comodidad para tratar con colisiones en el juego.

- Visualizar movimiento en la pantalla: No lograba hacer que la pantalla mostrara diferentes imágenes a lo largo de su ejecución, al revisar la documentación concluí utilizar los métodos .flip() y .blit(), estos permiten actualizar la ventana y actualizar una imagen respectivamente, con ello se solucionó el problema.

- Saltos en plataformas e interacción con escaleras: Esto fue un gran problema, ya que en un momento las colisiones se hacían un trabajo tedioso, opté por hacer listas de rectángulos, uno para plataformas y otro para las escaleras, para poder obtener datos de manera mas cómoda.

- Guardar y cargar configuración: Se optó por utilizar archivos de texto(.txt) para almacenar información útil para el programa, como lo es la configuración, además se optó por una configuración por defecto, la cual es:

Con sonido, dificultad en medio y 3 vidas.

Bitácora de actividades:

- Se empezó por crear una ventana de inicio, la cual podía abrir la ventana de juego, esto se logró utilizando la librería tkinter de Python.

- Se puso la meta de crear una ventana de juego con el jugador respectivo como una clase, luego se pusieron acciones que se activaban con las teclas del teclado.

- Se creó la clase plataforma, con la cual se pretendía entender la colisión entre elementos de la pantalla de juego.

- Se crearon varios objetos plataforma, además empezó la creación de escaleras para poder movilizarse entre plataformas, se crearon rectángulos para representar al mono y a la princesa, además de sus respectivos eventos.

- Se creó la ventana de configuración, la cual pretende darle libertad al usuario de escoger la dificultad, número de vidas y sonido del juego, estos datos se guardan en configuracion.txt.

- Se añadieron sprites al juego, dándole mejor aspecto, además se arreglaron algunos errores pequeños que afectaban la jugabilidad, los obstáculos aparecen de forma aleatoria y a una frecuencia que depende del nivel que escogió el jugador.

- Se creó una ventana de pygame para poder mostrar el splash de animación, el cual es Mario moviéndose por su mundo de Nintendo, mostrando también el nombre del creador del juego.

Cuadro de horas:

Función	Total
Requerimiento/diseño	8 horas
Investigación de funciones	7 horas
Programación	20 horas
Documentación Interna	1 hora
Pruebas	3 horas
Elaboración Documento	2.5 horas
Total	41.5 horas

Conclusión

El proyecto se realizó de buena manera, logrando implementar utilidades de pygame y tkinter para poder cumplir con lo pedido, se utilizaron varios aspectos de programación, como la utilización de funciones, clases, librerías, archivos de texto, entre otros de los cuales se adquirió conocimiento para futuros proyectos, además se logró un desarrollo lógico para poder resolver problemas relacionados con la creación de videojuegos.