Universidad De Oriente Campus Poza Rica

Dirección de posgrados, educación continua e investigación Proyecto de Licenciatura

Ingenieria en sistemas computacionales, 29/03/2023

Survibox puedes sobrevivir?

Abstract

En conclusion al momento de programar vemos que es mas que ceros y unos pues lleva detras una sintaxis, operaciones, funciones y procesos conectados y podemos aprender de eso para solucionar problemas como el proyecto que nos toco realizar.

Nombre: Martínez Cristóbal Pedro Jesús

---

1 Introduction

1.1 Ursina

Ursina es una biblioteca de Python para el desarrollo de videojuegos y aplicaciones interactivas en 3D. Es una herramienta de código abierto y gratuita, que utiliza el motor gráfico Panda3D para renderizar gráficos en tiempo real y proporciona una API fácil de usar para la creación de entornos y objetos en 3D, animaciones, física y detección de colisiones. **Ursina se centra en simplificar el proceso de creación de videojuegos y aplicaciones interactivas en 3D, lo que la hace ideal para principiantes y estudiantes que quieran aprender sobre programación de videojuegos.**

Por eso usamos el motor gráfico de Ursina en este proyecto pues al haber a documentación facilita el desarrollo de programas con base en las necesidades que uno busca(`https://docs.github.com/es/pages/`), **En el link se encuentra la documentación utilizada en el proyecto pues gracias a ella pudimos solucionar algunos problemas que fueron surgiendo en el proceso.**

1.2 Random

La biblioteca random de Python es una biblioteca incorporada en el lenguaje Python que proporciona funciones para generar números aleatorios, elegir elementos aleatorios de secuencias y mezclar elementos aleatoriamente.

La biblioteca random también proporciona varias funciones para generar números aleatorios con distribuciones específicas.

Es importante tener en cuenta que los números aleatorios generados por la biblioteca random no son verdaderamente aleatorios, ya que se generan utilizando un algoritmo determinista.

1.3 Math

La biblioteca math de Python es una biblioteca incorporada en el lenguaje Python que proporciona funciones matemáticas comunes y constantes matemáticas. Algunas de las funciones más comunes en la biblioteca math son:

- La función str() en Python es una función incorporada que se utiliza para convertir un objeto en una cadena.
- La función sqrt() devuelve el resultado como un número de punto flotante.
- En Python, las variables son espacios de memoria reservados para almacenar valores, mientras que las constantes son valores que no pueden cambiar durante la ejecución del programa.

1.4 Coordenadas

Las coordenadas en Python se refieren a un par de valores que indican una ubicación en un plano bidimensional, también conocido como sistema de coordenadas cartesianas. El primer valor representa la posición horizontal, también conocido como el eje X, mientras que el segundo valor representa la posición vertical, también conocido como el eje Y.

En Python, se puede utilizar la biblioteca Ursina para trabajar con coordenadas en un plano tridimensional, para crear objetos en un espacio 3D y manipular sus coordenadas X, Y y Z.

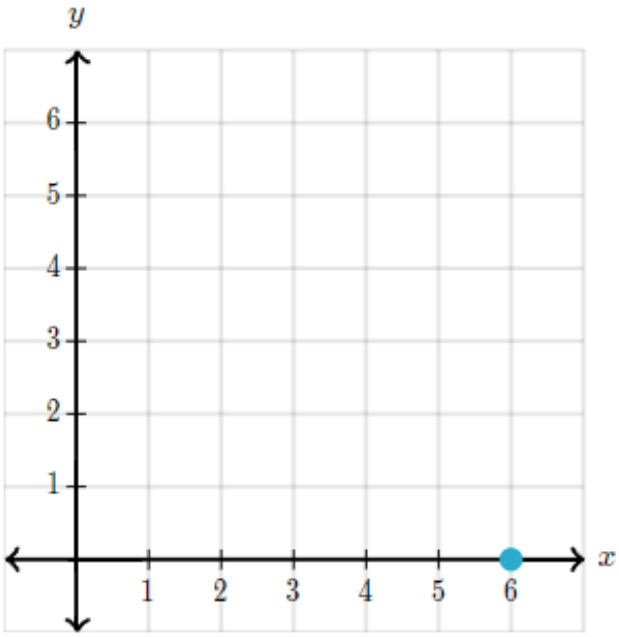

Figure 1: ejemplo

Para trabajar con coordenadas en Python implica utilizar una combinación de bibliotecas y funciones específicas para representar y manipular posiciones en un plano bidimensional o tridimensional.

2 Beta

El juego en su fase beta era crear una entidad, jugador igual Entity model igual cube, color igual color.red, collider=box

que se moviera de arriba, abajo de izquierda a derecha para ello se implementaron held keys que controla el movimiento del objeto en cuestión que seria jugador el código implementado controla el movimiento del objeto en cuatro direcciones diferentes basado en las teclas de las flecha que el usuario presiona.

```
if held_keys["left arrow"]:
    jugador.x -= velocidad_jugador * time.dt
if held_keys["right arrow"]:
    jugador.x += velocidad_jugador * time.dt
if held_keys["up arrow"]:
    jugador.y += velocidad_jugador * time.dt
if held_keys["down arrow"]:
    jugador.y -= velocidad_jugador * time.dt
```

Figure 2: ejemplo

2.1 Funcionamiento

En la función update(), se actualiza el estado del juego en cada fotograma. Se comprueba si el mensaje final está habilitado, lo que indica que el juego ha terminado y no se debe actualizar más. Si el tiempo de inicio es None, se establece en el tiempo actual. Si no se ha iniciado el juego, el código sale de la función sin hacer nada más. Si ha pasado suficiente tiempo desde el inicio del juego, se crea un nuevo enemigo de forma aleatoria. Se comprueba si las teclas de dirección se están presionando para mover al jugador. También se comprueba si el jugador y los enemigos se están intersectando para ver si el jugador ha perdido el juego.

```
def update():
    global enemigos, tiempo_de_inicio, juego_iniciado, mensaje_final, puntos
#Si el mensaje final está habilitado, no se actualiza el juego
    if mensaje_final.enabled:
        return

    if tiempo_de_inicio is None:
        tiempo_de_inicio = time.time()
        ultimo_punto = tiempo_de_inicio
        return
```

Figure 3: ejemplo

El código que se muestra es un bucle for que itera a través de una lista de enemigos. Para cada enemigo, se calcula la distancia entre el enemigo y el jugador, y se determina la velocidad requerida para que el enemigo se mueva hacia el jugador con una velocidad constante velocidad enemigo.

Luego, se actualiza la posición del enemigo multiplicando la velocidad en X e Y por la cantidad de tiempo transcurrido desde la última actualización "timedt" y agregando el resultado a la posición actual del enemigo.

```
for enemigo in enemigos:
    dx, dy = jugador.x - enemigo.x, jugador.y - enemigo.y
    distancia = math.sqrt(dx*dx + dy*dy)
    velocidad_x = velocidad_enemigo * dx / distancia
    velocidad_y = velocidad_enemigo * dy / distancia
    enemigo.x += velocidad_x * time.dt
    enemigo.y += velocidad_y * time.dt
```

Figure 4: ejemplo

En el código, se definen algunas constantes como la velocidad del jugador y de los enemigos, así como el tiempo de espera inicial antes de que aparezcan los enemigos.

2.2 General

El código es un juego simple en el que el jugador debe esquivar los enemigos que aparecen aleatoriamente en la pantalla. El jugador se mueve con las teclas de flecha y el objetivo es evitar colisionar con los enemigos, ya que esto resulta en el fin del juego.

El juego comienza cuando el usuario hace clic en el botón "Iniciar Juego", lo que establece una variable booleana llamada "juego iniciado" en True. Después de un tiempo de espera inicial, se generan aleatoriamente nuevos enemigos en la pantalla.

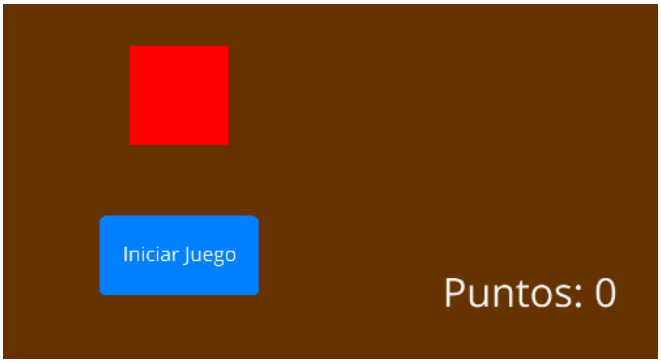

Figure 5: ejemplo

Si el jugador colisiona con un enemigo, aparece un mensaje de texto que indica que el juego ha terminado. El juego se detiene en este punto y se deshabilita la creación de nuevos enemigos. Los puntos se calculan en función del tiempo que el jugador ha estado jugando y se muestran en la pantalla.

Figure 6: ejemplo

3 Entorno/Desarollo

Se podrían haber utilizado otros motores de juego para crear un juego similar al que se muestra en el código. Algunos ejemplos populares incluyen:

Unity: es un motor de juego muy popular que se utiliza para crear juegos en 2D y 3D para una variedad de plataformas, incluyendo PC, consolas y dispositivos móviles.

Unreal Engine: es otro motor de juego popular que se utiliza para crear juegos en 2D y 3D. Al igual que Unity, también es compatible con una variedad de plataformas.

Godot Engine: es un motor de juego gratuito y de código abierto que es conocido por su facilidad de uso y flexibilidad.

Hay muchos otros motores y frameworks de juego disponibles, y la elección de uno depende en gran medida de las necesidades específicas del proyecto y la experiencia y preferencias del desarrollador.

3.1 Pygame

Pygame: es una biblioteca de Python diseñada para facilitar la creación de videojuegos. Proporciona herramientas para la gestión de gráficos, sonido, eventos, colisiones, y otras características necesarias en la creación de juegos. Pygame es una opción popular para la creación de juegos en 2D y ha sido utilizada en muchos juegos comerciales.

3.2 Turtle

Turtle: es un módulo de Python que permite la creación de gráficos en 2D mediante el uso de un cursor (conocido como "turtle") que puede moverse por la pantalla y dibujar líneas, círculos y otras formas geométricas. Aunque originalmente fue diseñado como una herramienta educativa para enseñar programación a niños, también puede ser utilizado para crear juegos sencillos en 2D.

En resumen, Pygame es una biblioteca completa y potente diseñada para la creación de videojuegos en 2D, mientras que Turtle es una herramienta más simple y orientada a la educación que permite la creación de gráficos en 2D, pero no proporciona características avanzadas para la creación de juegos completos.

3.3 Unity

Unity: es un motor de videojuegos multiplataforma que permite la creación de juegos en 2D y 3D para diversas plataformas, incluyendo PC, consolas, dispositivos móviles y realidad virtual. Unity proporciona un editor visual, un motor de física, herramientas de animación, efectos especiales y un lenguaje de programación C para la creación de juegos.

3.4 UnrealEngine

Unreal Engine: es otro motor de videojuegos multiplataforma que también permite la creación de juegos en 2D y 3D para diversas plataformas, incluyendo PC, consolas, dispositivos móviles y realidad virtual. Unreal Engine proporciona un editor visual, herramientas de programación visual (Blueprints) y un lenguaje de programación C++ para la creación de juegos. Además, cuenta con un sistema de física avanzado, herramientas de animación y efectos especiales de alta calidad.

Ambos motores son herramientas muy potentes y populares para la creación de videojuegos profesionales. Unity es más fácil de aprender y es una buena opción para principiantes, mientras que Unreal Engine ofrece más características avanzadas y herramientas de alta calidad para la creación de juegos de alta calidad. En general, la elección entre ambos depende de las necesidades específicas del proyecto y de la experiencia y preferencias del desarrollador.

3.4.1 aplicaciones

Este código implementa un juego básico en 2D, por lo que podría ser utilizado como punto de partida para el desarrollo de juegos más avanzados en diferentes plataformas. También podría ser utilizado como base para desarrollar simulaciones interactivas, por ejemplo, en el campo de la educación o la formación.

Además, el uso de las matemáticas y la física en la simulación del movimiento de los enemigos y del jugador podría ser aplicado en áreas como la robótica y la animación. También se podría utilizar para el desarrollo de herramientas de visualización en tiempo real para el análisis de datos en diversas áreas, como la ciencia de los materiales, la ingeniería.

3.4.2 Algoritmo

términos generales:

- Se importan las librerías necesarias y se definen las constantes y variables que se van a utilizar en el juego.
- Se establece la configuración inicial de la ventana y se crea el jugador.
- Se define una lista vacía para los enemigos, y se establece el tiempo de inicio en nulo.

- En el bucle de actualización, se comprueba si el mensaje final está habilitado. Si lo está, el juego no se actualiza.
- Si el tiempo de inicio es nulo, se establece el tiempo de inicio y el último punto. Si no es nulo, se comprueba el tiempo transcurrido.
- Si el juego no ha sido iniciado, no se crean nuevos enemigos.
- Se comprueba si ha pasado el tiempo de espera inicial, y si es así, se comienza a crear enemigos aleatorios.
- Se actualiza la posición del jugador según las teclas presionadas.
- Se comprueba si el jugador ha colisionado con algún enemigo. Si es así, se habilita el mensaje final.
- Se comprueba si ha pasado un segundo desde el último punto obtenido, y si es así, se aumenta el número de puntos.
- Para cada enemigo creado, se actualiza su posición y se comprueba si ha colisionado con el jugador.
- Se define la función para iniciar el juego, que establece los valores iniciales y comienza el juego.
- Se crea el botón de inicio y se inicia la aplicación.

4 Testeos/Limitantes

Como en todo en la vida lleva una estructura al igual que un reloj te da la hora esta lleva un mecanismo que hace que todo funcione en orden en programación cualquier error por pequeño que sea puede arruinar tu codigo si no fuera por la documentación y ChatGPT para ver que errores había el programa en cuestión estaría lleno de bug pues comenzando con lo básico la creación de enemigos aleatorios y que estos te siguieran era una tarea complicada de hacer considero la parte de hacer el botón de inicio y darle los parámetros para que aparezcan los enemigos después de darle clic a iniciar la parte más difícil pues debes poner muchas condicionales de acuerdo a los lapsos de tiempo las limitantes a considerar sería la falta de imaginación y que uno debe de empezar a trabajar con programas pues como con todo lenguaje es repetición hasta aprenderlo de memoria y una vez conociendo como funciona aplicarlo a otros programas