VIDEOJUEGOS
INTRODUCCIÓN A PYTHON

CLASE 09

Contenido

“

Este espacio curricular permite fortalecer el espíritu crítico y la actitud creativa del futuro graduado. Lo alienta a integrar los conocimientos previos, recreando la práctica en el campo real, al desarrollar un proyecto propio.

”

PROGRAMACIÓN I

En un juego endless, el enfoque principal es mantener al jugador comprometido y desafiado a medida que avanza en el juego. Esto se logra mediante la creación de una experiencia de juego repetitiva pero cada vez más desafiante, donde los obstáculos, la dificultad o la velocidad aumentan a medida que el jugador avanza.

Este tipo de juegos a menudo se basan en mecánicas simples y adictivas que son fáciles de entender y jugar, lo que permite a los jugadores participar en sesiones de juego rápidas y repetidas. Ejemplos populares de juegos endless incluyen títulos como "Flappy Bird", "Temple Run" y "Crossy Road", donde el objetivo principal es superar la puntuación más alta o durar el mayor tiempo posible.

Introducción juegos endless

Un juego endless (también conocido como juego infinito o sin fin) es un tipo de juego en el que no hay un final definido o un objetivo final que se alcance. En cambio, el juego continúa de manera indefinida o hasta que el jugador decida dejar de jugar.

En un juego endless, el enfoque principal es mantener al jugador comprometido y desafiado a medida que avanza en el juego. Esto se logra mediante la creación de una experiencia de juego repetitiva pero cada vez más desafiante, donde los obstáculos, la dificultad o la velocidad aumentan a medida que el jugador avanza.

Este tipo de juegos a menudo se basan en mecánicas simples y adictivas que son fáciles de entender y jugar, lo que permite a los jugadores participar en sesiones de juego rápidas y repetidas. Ejemplos populares de juegos endless incluyen títulos como "Flappy Bird", "Temple Run" y "Crossy Road", donde el objetivo principal es superar la puntuación más alta o durar el mayor tiempo posible.

Los juegos endless suelen ser populares en dispositivos móviles, ya que se adaptan bien a sesiones de juego cortas y rápidas. También ofrecen la posibilidad de competir con amigos o jugadores de todo el mundo a través de tablas de clasificación en línea, lo que fomenta la competitividad y el deseo de superar los récords personales o de otros jugadores.

características

Las características comunes de un juego endless (infinito o sin fin) incluyen:

1. Repetición infinita: El juego no tiene un final definido y puede continuar de manera indefinida o hasta que el jugador decida dejar de jugar.

2. Mecánicas sencillas: Los juegos endless suelen basarse en mecánicas simples y fáciles de entender. Esto permite a los jugadores participar rápidamente y sin mucha curva de aprendizaje.

3. Dificultad progresiva: A medida que el jugador avanza en el juego, la dificultad aumenta gradualmente. Puede ser a través de la aparición de obstáculos más difíciles, incremento de la velocidad del juego u otras formas de desafío.

4. Puntuación y récords: La mayoría de los juegos endless tienen un sistema de puntuación para motivar a los jugadores a superar su propio récord o competir con otros jugadores. Esto fomenta la rejugabilidad y el deseo de mejorar.

5. Sesiones de juego cortas: Los juegos endless suelen ser diseñados para sesiones de juego rápidas y casuales. Son ideales para jugar en momentos de espera o cuando se dispone de poco tiempo.

6. Gráficos y sonidos atractivos: Para mantener el interés del jugador, los juegos endless suelen tener gráficos coloridos y llamativos, así como efectos de sonido envolventes.

7. Recompensas y desbloqueables: Algunos juegos endless ofrecen recompensas o desbloqueables a medida que el jugador avanza o alcanza ciertos hitos. Esto proporciona una sensación de progresión y satisfacción.

8. Elementos de competencia: Muchos juegos endless incorporan tablas de clasificación en línea, donde los jugadores pueden comparar sus puntuaciones con otros jugadores y competir por los primeros puestos.

Parallax

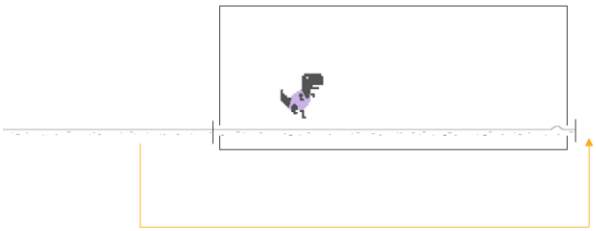
El efecto parallax en el fondo de los videojuegos 2D es una técnica visual que crea la ilusión de profundidad y movimiento en un entorno de juego. Consiste en tener múltiples capas de fondos que se mueven a diferentes velocidades en relación con la cámara principal del juego.

Cada capa de fondo tiene su propia velocidad de desplazamiento, donde las capas más cercanas a la cámara se mueven más rápido y las capas más lejanas se mueven más lentamente. Esto crea una sensación de profundidad y da la impresión de que los objetos o elementos en primer plano se mueven más rápido que los elementos en el fondo.

El efecto parallax se logra desplazando las capas de fondo en proporciones diferentes según su distancia relativa a la cámara. Por ejemplo, las capas de fondo más cercanas se desplazarán más rápidamente, mientras que las capas de fondo más lejanas se desplazarán más lentamente.

Este efecto se utiliza comúnmente en juegos 2D para agregar una sensación de inmersión y mejorar la apariencia visual del entorno del juego. Puede ayudar a crear una mayor sensación de profundidad y dinamismo, dándole vida al mundo del juego y mejorando la experiencia visual del jugador.

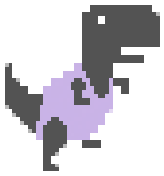
En nuestro ejemplo del juego Dino, creamos un efecto de movimiento infinito con el desplazamiento sobre el eje x. En este caso, se le resta 1 (dependiendo de la velocidad que quieras general) al valor X del rectángulo de la figura. Para crear un efecto continuo se instancias dos imágenes. Cuando la imagen 1 sale del viewport se la reubica del otro lado de la pantalla.



```python
class Piso(pygame.sprite.Sprite):
    def __init__(self, paramX, paramY):
        super().__init__()
        self.image=pygame.Surface([2403,25])
        self.rect=self.image.get_rect()
        self.rect.x=paramX
        self.rect.y=paramY
        self.image=pygame.image.load(".\\Sprites\\piso.png")
        self.image.set_colorkey(COLOR.BLANCO)
    def update(self):

        self.rect.x-=Const._velocidad
        #controlamos cuando sale de la pantalla lo ponemos del otro lado
        #calculando el punto cero mas el ancho
        if self.rect.x + self.rect.width<=0:
            self.rect.x=self.rect.width -10
```

El personaje principal



Los detalles técnicos del personaje principal en un juego endless pueden variar según la plataforma y el motor de juego utilizado. Sin embargo, aquí hay algunos aspectos técnicos comunes a considerar:

Sprites y animaciones: El personaje principal se representa mediante una serie de sprites o imágenes que representan sus diferentes estados y acciones, como caminar, saltar, atacar, etc. Estas imágenes se alternan rápidamente para crear animaciones fluidas y dar vida al personaje.

Colisiones: El personaje principal necesita tener una detección de colisiones adecuada para interactuar con los elementos del juego, como obstáculos, enemigos o recolección de objetos. Se utilizan técnicas como bounding boxes o formas más complejas para detectar colisiones con precisión.

Física y movimiento: El personaje principal debe tener una física y un movimiento realistas. Esto puede incluir gravedad, aceleración, velocidad

máxima, salto, deslizamiento, entre otros parámetros. Estos aspectos se implementan utilizando algoritmos de movimiento y simulación física.

Controles del jugador: Se deben implementar los controles del jugador para que el personaje principal responda a las acciones del jugador, como presionar teclas o tocar la pantalla en dispositivos táctiles. Esto implica capturar la entrada del jugador y traducirla en acciones del personaje, como saltar, moverse hacia la derecha o atacar.

Lógica de interacción: El personaje principal puede interactuar con otros elementos del juego, como recolectar objetos, combatir enemigos o activar eventos. Esto requiere la implementación de la lógica de interacción correspondiente, como verificar la colisión con objetos, administrar puntos o salud, y realizar acciones específicas según la interacción.

Estados y control de animación: El personaje principal puede tener diferentes estados, como correr, saltar, agacharse, atacar, etc. Estos estados se gestionan mediante una máquina de estados o un sistema de control de animación para cambiar y reproducir las animaciones adecuadas según el estado actual del personaje.

Sonidos y efectos visuales: El personaje principal puede tener efectos de sonido asociados a sus acciones, como pasos, golpes o voces. También puede haber efectos visuales, como partículas o destellos, que se activan al realizar ciertas acciones.

En este ejemplo Dino, es nuestro personaje principal. Se encontrará siempre en las misma posición x,y. el efecto de movimiento es provocado por el movimiento de los obstáculos y el piso.

```
class Dino(pygame.sprite.Sprite):
    _lista_Animaciones=[]
    _pos_animacion=0
    _estado="corriendo"
    _contador_frame=0
    _velocidad_salto=-20
    def __init__(self,paramX,paramY):
        super().__init__()
        self.image=pygame.Surface([46,47])
        self.rect=self.image.get_rect()
        self.rect.x=paramX
        self.rect.y=paramY
        self._lista_Animaciones.append(
          pygame.image.load(
           ".\\Sprites\\dino1.png"))
        self._lista_Animaciones.append(
          pygame.image.load(
          ".\\Sprites\\dino2.png"))
        self.image=self._lista_Animaciones
           [self._pos_animacion]
        self.image.set_colorkey(COLOR.BLANCO)

    def saltar(self):
        if self._estado!="saltando":
            self._estado="saltando"
            self._velocidad_salto=-20

    def update(self):
        if self._estado=="corriendo":
            self._pos_animacion+=1
            if self._pos_animacion>1:
                self._pos_animacion=0
            self.image=self._lista_Animaciones
             [self._pos_animacion]
        elif self._estado=="saltando":
            self.rect.y +=self.
              _velocidad_salto
            self._velocidad_salto+=1
            if self.rect.y<=200:
                self._velocidad_salto=1
                self._estado="bajando"
        elif self._estado=="bajando":
            self.rect.y +=self.
              _velocidad_salto
            self._velocidad_salto+=1
            if self.rect.y>=271:
                self.rect.y=271
                self._estado="corriendo"
```

Los enemigos

El movimiento de los enemigos en un juego endless puede ser crucial para el desafío y la diversión del juego. Aquí hay algunas formas comunes de implementar el movimiento de los enemigos en un juego endless:

Movimiento horizontal: Los enemigos pueden moverse de izquierda a derecha o viceversa en el nivel, creando obstáculos para el jugador. Pueden seguir una trayectoria lineal o tener movimientos más complejos, como zigzaguear o cambiar de dirección en puntos específicos.

Movimiento vertical: Los enemigos también pueden moverse verticalmente, ascendiendo o descendiendo en el nivel. Esto puede añadir una dimensión adicional de desafío para el jugador, obligándolo a ajustar su posición vertical mientras evita los enemigos.

Movimiento siguiendo al jugador: Algunos enemigos pueden tener un comportamiento de persecución, tratando de acercarse al jugador. Esto puede requerir algoritmos de seguimiento o detección de posición del jugador para que los enemigos se muevan hacia él.

Movimiento aleatorio: Los enemigos pueden moverse de forma aleatoria por el nivel, lo que crea un elemento impredecible. Esto puede hacer que el jugador deba estar atento y reaccionar rápidamente para evitarlos.

Movimiento en patrones predefinidos: Algunos enemigos pueden tener patrones de movimiento predefinidos, donde siguen rutas específicas en el nivel. Estos patrones pueden ser simples o complejos, dependiendo del diseño del juego.

Movimiento basado en la interacción con el jugador: Los enemigos pueden reaccionar a las acciones del jugador, como acercarse cuando el jugador se acerca demasiado o alejarse cuando el jugador ataca. Esto puede requerir lógica adicional para que los enemigos tomen decisiones basadas en el estado del jugador.

Movimiento acelerado: A medida que el juego avanza y aumenta la dificultad, los enemigos pueden moverse más rápido o tener patrones de movimiento más desafiantes. Esto crea un sentido de progresión y dificultad creciente a medida que el jugador avanza en el juego.

```
class Planta(pygame.sprite.Sprite):
    def __init__(self, paramX, paramY):
        super().__init__()
        self.image=pygame.Surface([31,70])
        self.rect=self.image.get_rect()
        self.rect.x=paramX
        self.rect.y=paramY
        self.image=pygame.image.load(
          ".\\Sprites\\planta.png")
        self.image.set_colorkey(COLOR.BLANCO)

def update(self):

        self.rect.x-=Const._velocidad
        #controlamos cuando sale de la pantalla
        lo ponemos del otro lado
        #calculando el punto cero mas el ancho
        if self.rect.x + self.rect.width<=0:
            self.rect.x=650
```

posiciones relativas y absolutas

En Pygame, el término "path relativo" y "path absoluto" se refiere a cómo especificas la ubicación de un archivo, como una imagen o un sonido, en relación con la estructura de directorios en tu proyecto.

Path relativo: Un path relativo especifica la ubicación de un archivo en relación con el directorio de trabajo actual. En lugar de proporcionar la ruta completa desde la raíz del sistema de archivos, utilizas una ruta que comienza desde el directorio actual.

Por ejemplo, si tienes una imagen llamada "imagen.png" en un subdirectorio llamado "recursos", y tu script de Pygame se encuentra en el directorio principal del proyecto, puedes especificar el path relativo como "./recursos/imagen.png". El "./" representa el directorio actual.

Path absoluto: Un path absoluto especifica la ubicación completa de un archivo desde la raíz del sistema de archivos. Proporcionas la ruta completa que incluye todas las carpetas y subcarpetas necesarias para llegar al archivo.

Por ejemplo, si tienes la misma imagen "imagen.png" en la ubicación "C:\Proyectos\Juego\recursos\imagen.png", puedes especificar el path absoluto como "C:\Proyectos\Juego\recursos\imagen.png".

En resumen, el path relativo se basa en la estructura de directorios actual y utiliza una ruta que comienza desde el directorio actual, mientras que el path absoluto proporciona la ruta completa desde la raíz del sistema de archivos. La elección entre usar un path relativo o absoluto depende de cómo esté organizado tu proyecto y de cómo desees especificar la ubicación de los archivos dentro de ese contexto.

Path relativo

```
self.image=pygame.image.load(".\\Sprites\\piso.
png")
```

path absoluto

```
self.image=pygame.image.load("D:\Trabajo\sala_P
ython\planta.png")
```

Detección de eventos y presión de teclas

En Pygame, los "events" (eventos) son acciones o sucesos que ocurren durante la ejecución de un juego, como la pulsación de una tecla, el movimiento del ratón o el clic en un botón. Pygame proporciona una forma de detectar y responder a estos eventos para permitir la interacción del jugador con el juego.

Para detectar la presión de una tecla en Pygame, se puede utilizar el bucle principal del juego junto con el manejo de eventos. Aquí hay un ejemplo de cómo detectar la pulsación de una tecla específica:

```
for evento in pygame.event.get():
            if evento.type==pygame.QUIT:
                salir=True
            if evento.type==pygame.KEYDOWN:
                if evento.key==pygame.K_SPACE:
                    _audio_salto.play()
                    dino.saltar()
```

En este ejemplo, se utiliza el bucle principal del juego para verificar constantemente los eventos que ocurren. Luego, se itera sobre la lista de eventos y se verifica si el tipo de evento es `KEYDOWN` (pulsación de tecla) y si la tecla específica es `K_SPACE` (tecla de espacio). Si se cumple esta condición, se ejecuta el código correspondiente para responder a la pulsación de tecla.

Mecánica de movimiento

Una mecánica de movimiento en un videojuego se refiere a cómo se desplaza y se controla el personaje, objeto o elemento principal del juego dentro del mundo virtual. Es el conjunto de reglas y acciones que determinan cómo el jugador interactúa y se mueve en el entorno del juego.

Las mecánicas de movimiento pueden variar ampliamente según el tipo de juego y el género. Algunos ejemplos comunes de mecánicas de movimiento incluyen:

- Movimiento básico: Desplazarse hacia la izquierda, derecha, arriba y abajo dentro de un espacio bidimensional o tridimensional.
- Salto: Permitir al personaje saltar o elevarse en el aire para superar obstáculos o alcanzar plataformas más altas.
- Correr: Aumentar la velocidad de movimiento del personaje para desplazarse rápidamente por el entorno del juego.
- Volar: Permitir al personaje volar o desplazarse libremente en el aire.
- Deslizamiento: Permitir al personaje deslizarse por superficies inclinadas o resbaladizas.
- Entre otras.

Conceptos elementales de diseño

El diseño de niveles implica la creación de entornos jugables que sean atractivos, desafiantes y satisfactorios para los jugadores. Para lograrlo, es importante considerar una serie de elementos clave en el diseño de niveles. A continuación, exploraremos estos elementos en detalle:

a) Terreno y ambiente:

El uso efectivo del terreno y la creación de una ambientación adecuada son elementos fundamentales en el diseño de niveles. Estos aspectos ayudan a establecer la atmósfera del juego y a crear un mundo inmersivo para los jugadores. Algunos puntos a tener en cuenta son:

- Utiliza diferentes tipos de terreno, como colinas, cuevas o cuerpos de agua, para crear variedad y oportunidades tácticas en el nivel.
- Elige una ambientación que se alinee con la temática del juego y genere una atmósfera coherente y cautivadora.
- Aprovecha la iluminación y los efectos visuales para resaltar áreas importantes, crear contrastes y transmitir emociones en el nivel.

b) Objetos y obstáculos:

La colocación estratégica de objetos y obstáculos en el nivel es esencial para proporcionar desafíos interesantes y opciones tácticas al jugador. Estos elementos influyen en la jugabilidad y en la toma de decisiones del jugador. Aquí tienes algunas pautas para su inclusión:

- Distribuye objetos y obstáculos de manera equilibrada a lo largo del nivel, asegurando que haya desafíos y recompensas en momentos adecuados.
- Incluye power-ups y mejoras que otorguen habilidades especiales o mejoras temporales para recompensar al jugador y aumentar su satisfacción.
- Introduce trampas y peligros que añadan riesgo y tensión al juego, requiriendo que los jugadores tomen decisiones rápidas y estratégicas.

c) Enemigos y AI (Inteligencia Artificial):

Los enemigos y la inteligencia artificial desempeñan un papel crucial en el diseño de niveles, ya que son responsables de proporcionar desafíos y crear encuentros emocionantes para los jugadores. Aquí tienes algunos aspectos a considerar:

- Diseña encuentros con enemigos que sean desafiantes y emocionantes, utilizando la ubicación, el número y las habilidades de los enemigos de manera efectiva.
- Desarrolla comportamientos de la inteligencia artificial de los enemigos que sean realistas, inteligentes y respondan de manera creíble a las acciones del jugador.
- Considera la inclusión de jefes y mini-jefes para agregar momentos de mayor dificultad y marcar hitos importantes en la progresión del juego.

d) Puzzles y secretos:

La integración de acertijos y elementos ocultos en el diseño de niveles fomenta la exploración, estimula la mente del jugador y añade variedad al juego. Aquí tienes algunas sugerencias:

- Incorpora puzzles lógicos, físicos o basados en interacciones para desafiar al jugador y agregar una capa de profundidad al juego.

Otros aspectos que se pueden considerar

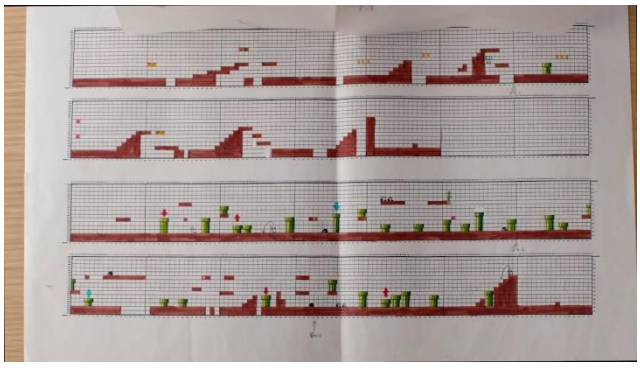
- Niveles temáticos: Explora la idea de diseñar niveles que se basen en diferentes temas o ambientaciones para brindar variedad y mantener el interés del jugador a lo largo del juego. Por ejemplo, un nivel de jungla, un nivel de nieve o un nivel futurista.

- Progresión de la dificultad: Diseñar niveles que presenten una progresión gradual en la dificultad, permitiendo que los jugadores adquieran nuevas habilidades y se enfrenten a desafíos cada vez más complejos a medida que avanzan en el juego. Pero siempre debes enseñarle al jugador las mecánicas para avanzar en el juego antes de que este jugador las necesite

- Diseño de niveles lineales vs. no lineales: El jugador sigue un camino predeterminado, y niveles que pueden ser lineales o no lineales. Puedes ofrecen múltiples rutas y opciones para que el jugador explore.

- Consideraciones de accesibilidad: Es importancia diseñar niveles accesibles para diferentes tipos de jugadores, teniendo en cuenta aspectos como la legibilidad visual, la claridad de las señales y la posibilidad de ajustar la dificultad para adaptarse a las necesidades individuales.

Ejemplo



"La cueva misteriosa"

Descripción:
El objetivo de este nivel es guiar al jugador a través de una cueva oscura y peligrosa en busca de un tesoro escondido. El nivel consta de varios elementos y desafíos para

mantener el interés del jugador y proporcionar una experiencia satisfactoria.

Elementos del nivel:

1. Terreno y ambiente:
   - La cueva: Utiliza terreno rocoso con paredes irregulares y estalactitas colgando del techo para crear una sensación de misterio y peligro.
   - Iluminación tenue: Implementa una iluminación ambiental tenue con antorchas parpadeantes que arrojan sombras en las paredes, generando una atmósfera intrigante.

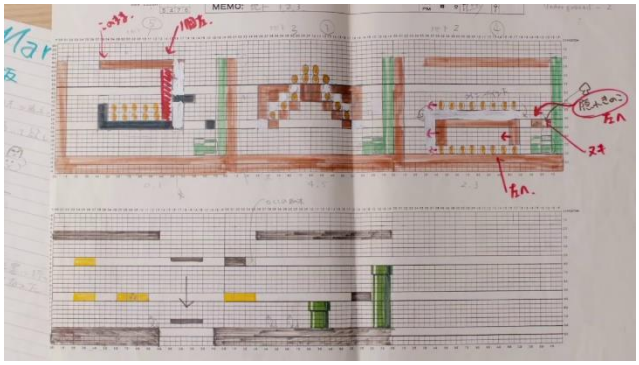
2. Objetos y obstáculos:
   - Plataformas móviles: Introduce plataformas móviles que se desplazan de un lado a otro, requiriendo que el jugador calcule bien el momento para saltar y evitar caer al abismo.
   - Rocas rodantes: Coloca rocas que ruedan desde las alturas, creando obstáculos que el jugador debe evitar para no ser golpeado.

3. Enemigos y AI:
   - Murciélagos hostiles: Incluye murciélagos voladores que atacan al jugador al acercarse, lo cual exige reflejos rápidos y habilidades de esquiva.
   - Arañas venenosas: Agrega arañas venenosas que se mueven sigilosamente por las paredes y lanzan telarañas para atrapar al jugador, obligándolo a liberarse rápidamente.

4. Puzzles y secretos:
   - Interruptores y puertas: Incorpora interruptores que deben ser activados para abrir puertas bloqueadas y permitir al jugador avanzar en el nivel.
   - Tesoros ocultos: Incluye áreas secretas en las que se pueden encontrar tesoros adicionales, recompensando a los jugadores exploradores.

5. Progresión de la dificultad:
   - A medida que el jugador avanza en el nivel, los desafíos se vuelven más difíciles y requieren mayor habilidad y precisión.
   - Los enemigos y obstáculos están ubicados estratégicamente para aumentar la tensión y crear momentos de acción emocionantes.

Este es solo un ejemplo sencillo, el diseño de niveles puede ser mucho más complejo y detallado según las necesidades del juego y los objetivos establecidos.



Bibliografía

Sweigart, A. (2012). Making Games with Python & Pygame: A Guide to Programming with Graphics, Animation, and Sound. Recuperado de https://inventwithpython.com/makinggames.pdf

Pygame. (s.f.). [Página oficial de pygame]. Recuperado 2 julio, 2019, de https://www.pygame.org/news