Generador de Pasillo Animado en Blender

Descripción General

Este script de Python para Blender genera automáticamente un pasillo curvo con una cámara animada que recorre toda la estructura. Es perfecto para crear escenas cinemáticas, visualizaciones arquitectónicas o fondos para animaciones.

El pasillo consta de bloques alternados en colores oscuro y naranja a ambos lados de un suelo blanco, siguiendo una trayectoria sinusoidal suave. La cámara sigue automáticamente este camino a lo largo de 300 frames.

Características Principales

Generación automática de pasillo con curvas suaves.

Cámara animada que sigue el recorrido completo.

Colores alternados en los bloques (patrón ajedrezado).

Iluminación profesional con luz solar y luces puntuales.

Totalmente parametrizable (ancho, longitud, curvas, velocidad).

Listo para renderizar directamente en Blender.

Cómo Usar

Paso 1: Preparar Blender

- Abre Blender (versión 2.8 o superior recomendada).
- Ve a la pestaña `Scripting` en la parte superior.
- Crea un nuevo archivo de texto o abre uno existente.

Paso 2: Ejecutar el Script

- Copia y pega el código completo en el editor de texto.
- Presiona `Alt + P` o haz clic en `Run Script`.
- ¡El pasillo se generará automáticamente!

Paso 3: Ver la Animación

- Presiona `NUMPAD 0` para ver desde la cámara.
- Presiona `Z` y selecciona `Material Preview` o `Rendered`.
- Presiona `ESPACIO` para reproducir la animación.
- La cámara recorrerá todo el pasillo automáticamente.

Parámetros Configurables

Puedes personalizar el pasillo modificando estas variables en la sección `PASO 3` del script:

```
ancho_pasillo = 3.5          # Ancho entre bloques (m s grande =
   pasillo m s amplio)
num_bloques = 40             # Cantidad de bloques a lo largo del
   camino
longitud_total = 50          # Distancia total del recorrido
amplitud_curva = 8           # Qu  tan pronunciadas son las curvas
   (0 = recto)
frecuencia_curva = 2         # N mero de ondulaciones en el
   recorrido
```

Para ajustar la velocidad de la cámara, modifica en `PASO 8`:

```
num_frames = 300  # M s frames = c mara m s lenta
```

Explicación Detallada del Código

A continuación desglosamos cada parte del script para que entiendas qué hace y por qué.

PASO 1: Función de Creación de Materiales

```
def crear_material(nombre, color_rgb):
    mat = bpy.data.materials.new(name=nombre)
    mat.diffuse_color = (*color_rgb, 1.0)
    return mat
```

¿Qué hace? Crea materiales con un color específico en formato RGB (valores de 0 a 1). Ejemplo: `(1.0, 0.0, 0.0)` = rojo puro.

PASO 2: Limpieza del Entorno

```
bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete()
```

Elimina todos los objetos existentes para empezar con un lienzo limpio.

PASO 3: Definición de Materiales

```
mat_pared_oscura = crear_material("ParedOscura", (0.1, 0.1, 0.1))
mat_pared_naranja = crear_material("ParedNaranja", (0.8, 0.3, 0.1))
mat_suelo_blanco = crear_material("SueloBlanco", (1.0, 1.0, 1.0))
```

Crea tres materiales: pared oscura, pared naranja y suelo blanco.

PASO 4: Configuración de Parámetros

```
ancho_pasillo = 3.5
num_bloques = 40
longitud_total = 50
amplitud_curva = 8
frecuencia_curva = 2
```

Define las dimensiones y características del pasillo.

PASO 5: Generar Puntos del Recorrido

```
for i in range(num_bloques):
    t = i / (num_bloques - 1)
    y = t * longitud_total
    x = amplitud_curva * math.sin(frecuencia_curva * math.pi * t)
```

```
    dx_dt = amplitud_curva * frecuencia_curva * math.pi * math.cos(
       frecuencia_curva * math.pi * t)
    dy_dt = longitud_total
    angulo_rotacion = math.atan2(dx_dt, dy_dt)
```

Calcula las posiciones de cada bloque usando una función sinusoidal para crear curvas suaves. `t` es el progreso del 0 al 1, `y` avanza linealmente, `x` se mueve lateralmente siguiendo una onda. El ángulo se calcula con la tangente.

PASO 6: Crear Bloques del Pasillo

```
for i, pos in enumerate(posiciones_camino):
    # Calcular vector perpendicular
    dx = math.cos(angulo)
    dy = -math.sin(angulo)

    # BLOQUE IZQUIERDO
    x_izq = x_centro - dx * (ancho_pasillo / 2)
    y_izq = y_centro - dy * (ancho_pasillo / 2)
    bpy.ops.mesh.primitive_cube_add(location=(x_izq, y_izq, 0.5))
    # ... rotaci n, escala y material
```

Para cada punto, coloca cubos a ambos lados del centro, los rota para seguir la curva y alterna colores.

PASO 7: Crear Curva Bezier para la Cámara

```
bpy.ops.curve.primitive_bezier_curve_add(location=(0, 0, 0))
path_curve = bpy.context.active_object
# ... configurar puntos Bezier con handles autom ticos
```

Crea una curva invisible que actúa como "riel" para la cámara, con puntos más densos para movimiento suave.

PASO 8: Configurar Cámara con Follow Path

```
follow_path = camera.constraints.new(type='FOLLOW_PATH')
follow_path.target = path_curve
follow_path.use_curve_follow = True
follow_path.forward_axis = 'FORWARD_Y'
follow_path.up_axis = 'UP_Z'
```

Agrega una restricción a la cámara para que siga la curva y mire hacia adelante.

PASO 9: Animar el Movimiento

```
num_frames = 300
follow_path.offset = -0
follow_path.keyframe_insert(data_path="offset", frame=1)
follow_path.offset = -100
follow_path.keyframe_insert(data_path="offset", frame=num_frames)
```

Crea keyframes para que la cámara recorra la curva del inicio al final en 300 frames.

PASO 10: Iluminación

```
bpy.ops.object.light_add(type='SUN', location=(0, 25, 30))
# Luces puntuales cada 5 bloques
```

Añade luz solar general y luces puntuales para crear atmósfera.

PASO 11: Ocultar la Curva

```
path_curve.hide_render = True
path_curve.hide_viewport = True
```

Hace invisible la curva guía.

Conceptos Clave Explicados

Curva Bezier: Tipo de curva matemática que crea transiciones suaves. Blender ajusta automáticamente los "handles".

Constraint (Restricción): Instrucción que obliga a un objeto a comportarse de cierta manera, como seguir un camino.

Funciones trigonométricas: `math.sin()` y `math.cos()` generan patrones de onda; el seno da la posición lateral y el coseno la dirección de la tangente.

atan2(): Calcula el ángulo de rotación para que un objeto apunte en una dirección específica.

Personalizaciones Avanzadas

- **Cambiar colores:** Modifica los valores RGB en `crear_material`.
- **Pasillo recto:** `amplitud_curva = 0`
- **Recorrido más rápido:** `num_frames = 150`
- **Más ondulaciones:** `frecuencia_curva = 4`

Resolución de Problemas

"La cámara no se mueve" Asegúrate de presionar ESPACIO y estar en vista de cámara (NUMPAD 0).

"Los bloques se ven raros" Reduce `amplitud_curva` o aumenta `num_bloques`.

"No veo los colores" Cambia a vista `Material Preview` (Z) o renderiza (F12).

Requisitos

- Blender versión 2.8 o superior.
- Python incluido con Blender.
- Sistema operativo: Windows, macOS o Linux.

Licencia

Este proyecto es de código abierto. Siéntete libre de usarlo, modificarlo y compartirlo.

Contribuciones

¿Tienes ideas para mejorar el script? ¡Las contribuciones son bienvenidas!

1. Haz un fork del repositorio.
2. Crea una rama para tu mejora.
3. Envía un pull request.

Recursos Adicionales

- Documentación oficial de Blender Python API
- Tutoriales de Blender Scripting en YouTube
- Comunidad de Blender en Stack Exchange

Código Completo

```
import bpy
import math

def crear_material(nombre, color_rgb):
    """
    Crea un material b sico con un color espec fico usando el
       modelo RGB
    """
    mat = bpy.data.materials.new(name=nombre)
    mat.diffuse_color = (*color_rgb, 1.0)
    return mat


def generar_pasillo_curva_suave():
    """
    Genera un pasillo con curvas suaves continuas y c mara que
       recorre todo el trayecto
    """

    # ========== PASO 1: Limpieza del Entorno ==========
    bpy.ops.object.select_all(action='SELECT')
    bpy.ops.object.delete()


    # ========== PASO 2: Definici n de Materiales ==========
    mat_pared_oscura = crear_material("ParedOscura", (0.1, 0.1, 0.1)
       )
    mat_pared_naranja = crear_material("ParedNaranja", (0.8, 0.3,
       0.1))
    mat_suelo_blanco = crear_material("SueloBlanco", (1.0, 1.0, 1.0)
       )


    # ========== PASO 3: Par metros del Pasillo ==========
    ancho_pasillo = 3.5          # Ancho entre las dos l neas de
       bloques
    num_bloques = 40             # Total de bloques a lo largo del
       camino
    longitud_total = 50          # Longitud total del recorrido
    amplitud_curva = 8           # Cu nto se curva (mayor = m s
       ondulado)
    frecuencia_curva = 2         # N mero de ondulaciones
```

```
    # ========== PASO 4: Generar Puntos del Recorrido (Curva Suave)
       ==========
    posiciones_camino = []

    for i in range(num_bloques):
        # Progreso a lo largo del camino (0 a 1)
        t = i / (num_bloques - 1)

        # Posici n en Y (avanza a lo largo del camino)
        y = t * longitud_total

        # Posici n en X (crea la curva suave usando funci n seno)
        x = amplitud_curva * math.sin(frecuencia_curva * math.pi * t
           )

        # Calcular  ngulo  de rotaci n basado en la tangente de la
           curva
        dx_dt = amplitud_curva * frecuencia_curva * math.pi * math.
           cos(frecuencia_curva * math.pi * t)
        dy_dt = longitud_total
        angulo_rotacion = math.atan2(dx_dt, dy_dt)

        posiciones_camino.append({
            'x': x,
            'y': y,
            'angulo': angulo_rotacion
        })


    # ========== PASO 5: Crear Bloques del Pasillo ==========
    for i, pos in enumerate(posiciones_camino):
        x_centro = pos['x']
        y_centro = pos['y']
        angulo = pos['angulo']

        # Calcular vector perpendicular para posicionar bloques a
           los lados
        dx = math.cos(angulo)
        dy = -math.sin(angulo)

        # ===== BLOQUE IZQUIERDO =====
        x_izq = x_centro - dx * (ancho_pasillo / 2)
        y_izq = y_centro - dy * (ancho_pasillo / 2)

        bpy.ops.mesh.primitive_cube_add(location=(x_izq, y_izq, 0.5)
           )
        bloque_izq = bpy.context.active_object
        bloque_izq.rotation_euler.z = angulo
```

```
        bloque_izq.scale = (0.8, 0.8, 1.0)

        # Alternar colores
        if i % 2 == 0:
            bloque_izq.data.materials.append(mat_pared_oscura)
        else:
            bloque_izq.data.materials.append(mat_pared_naranja)


        # ===== BLOQUE DERECHO =====
        x_der = x_centro + dx * (ancho_pasillo / 2)
        y_der = y_centro + dy * (ancho_pasillo / 2)

        bpy.ops.mesh.primitive_cube_add(location=(x_der, y_der, 0.5)
           )
        bloque_der = bpy.context.active_object
        bloque_der.rotation_euler.z = angulo
        bloque_der.scale = (0.8, 0.8, 1.0)

        # Alternar colores (opuesto al izquierdo)
        if i % 2 == 0:
            bloque_der.data.materials.append(mat_pared_naranja)
        else:
            bloque_der.data.materials.append(mat_pared_oscura)


        # ===== SUELO =====
        bpy.ops.mesh.primitive_plane_add(size=1.5, location=(
           x_centro, y_centro, 0))
        suelo = bpy.context.active_object
        suelo.rotation_euler.z = angulo
        suelo.scale.x = ancho_pasillo / 1.5 + 0.3
        suelo.scale.y = 1.0
        suelo.data.materials.append(mat_suelo_blanco)


    # ========== PASO 6: Crear Curva Bezier Suave para la C mara
       ==========
    # Crear m s puntos para una curva m s suave
    num_puntos_camara = num_bloques * 2  # M s puntos = movimiento
       m s suave

    bpy.ops.curve.primitive_bezier_curve_add(location=(0, 0, 0))
    path_curve = bpy.context.active_object
    path_curve.name = "CameraPath"

    # Limpiar puntos por defecto
    curve_data = path_curve.data
```

```
    curve_data.splines.clear()

    # Crear nueva spline tipo BEZIER (m s suave que POLY)
    spline = curve_data.splines.new('BEZIER')
    spline.bezier_points.add(num_puntos_camara - 1)

    # Generar puntos de la c mara con m s densidad
    for i in range(num_puntos_camara):
        t = i / (num_puntos_camara - 1)
        y = t * longitud_total
        x = amplitud_curva * math.sin(frecuencia_curva * math.pi * t
            )

        # Altura de la c mara
        altura_camara = 2.0

        # Configurar punto bezier
        point = spline.bezier_points[i]
        point.co = (x, y, altura_camara)
        point.handle_left_type = 'AUTO'
        point.handle_right_type = 'AUTO'

    # Configurar la curva para que sea suave
    curve_data.dimensions = '3D'
    curve_data.resolution_u = 24  # Alta resoluci n para suavidad


    # ========== PASO 7: Crear C mara con Follow Path SIMPLIFICADO
        ==========
    pos_inicial = posiciones_camino[0]

    # Crear c mara
    bpy.ops.object.camera_add(location=(pos_inicial['x'],
        pos_inicial['y'], 2.0))
    camera = bpy.context.active_object
    camera.name = "CameraSeguimiento"

    # Configurar c mara
    camera.data.lens = 35
    camera.data.clip_start = 0.1
    camera.data.clip_end = 1000

    # Hacer que esta sea la c mara activa
    bpy.context.scene.camera = camera

    # Agregar constraint "Follow Path" - ESTE ES EL CLAVE
    follow_path = camera.constraints.new(type='FOLLOW_PATH')
    follow_path.target = path_curve
```

```
    follow_path.use_curve_follow = True  # Esto hace que siga la
       orientaci n de la curva
    follow_path.use_fixed_location = False
    follow_path.forward_axis = 'FORWARD_Y'  # La c mara apunta
       hacia adelante (eje Y)
    follow_path.up_axis = 'UP_Z'  # El eje Z apunta hacia arriba


    # ========== PASO 8: Animar la C mara a lo Largo de la Curva
       ==========
    num_frames = 300  # Duraci n de la animaci n (300 frames =
       12.5 segundos a 24fps)
    bpy.context.scene.frame_start = 1
    bpy.context.scene.frame_end = num_frames

    # Configurar FPS para mejor control
    bpy.context.scene.render.fps = 24

    # Keyframe al inicio (offset = 0 significa inicio de la curva)
    follow_path.offset = -0
    follow_path.keyframe_insert(data_path="offset", frame=1)

    # Keyframe al final (offset = 100 significa final de la curva)
    follow_path.offset = -100
    follow_path.keyframe_insert(data_path="offset", frame=num_frames
       )

    # Hacer la interpolaci n lineal (velocidad constante)
    if camera.animation_data and camera.animation_data.action:
        for fcurve in camera.animation_data.action.fcurves:
            for keyframe in fcurve.keyframe_points:
                keyframe.interpolation = 'LINEAR'


    # ========== PASO 9: Iluminaci n ==========
    # Luz solar general
    bpy.ops.object.light_add(type='SUN', location=(0, 25, 30))
    sol = bpy.context.active_object
    sol.data.energy = 1.5
    sol.rotation_euler = (math.radians(50), 0, math.radians(20))

    # Luces puntuales a lo largo del recorrido
    for i in range(0, len(posiciones_camino), 5):
        pos = posiciones_camino[i]
        bpy.ops.object.light_add(type='POINT', location=(pos['x'],
           pos['y'], 4))
        luz = bpy.context.active_object
        luz.data.energy = 500
```

```
        luz.data.color = (1.0, 0.95, 0.85)


    # ========== PASO 10: Ocultar la Curva Path del Render ==========
    path_curve.hide_render = True
    path_curve.hide_viewport = True  # Tambi n ocultarla en el viewport


    print("=" * 70)
    print("    PASILLO CON CURVAS SUAVES Y C MARA EN SEGUIMIENTO GENERADO!")
    print("=" * 70)
    print(f"       Bloques por lado: {num_bloques}")
    print(f"       Frames de animaci n: {num_frames} ({num_frames/24:.1f} segundos)")
    print(f"       Longitud del recorrido: {longitud_total} unidades")
    print(f"       Amplitud de curva: {amplitud_curva}")
    print(f"       Puntos de c mara: {num_puntos_camara}")
    print()
    print("       C MO VER LA ANIMACI N:")
    print("  1. Presiona NUMPAD 0 (vista desde c mara)")
    print("  2. Presiona Z     Material Preview o Rendered")
    print("  3. Presiona ESPACIO (reproducir animaci n)")
    print("  4. La c mara recorrer  TODA la curva autom ticamente")
    print()
    print("       TIP: Si la c mara va muy r pida o lenta, cambia 'num_frames'")
    print("=" * 70)


# ========== EJECUTAR ==========
if __name__ == "__main__":
    generar_pasillo_curva_suave()
```

Listing 1: Script completo del generador de pasillo