



garba



1.	UN VISTAZO RÁPIDO	3
1.1	ESCENA 1 – AVDA. DE LA PAZ.....	4
1.2	ESCENA 2 - EL AYUNTAMIENTO	5
1.3	ESCENA 4 – HABITACIÓN OSCURA	6
1.4	USO DEL MAPA	6
1.5	ESCENA 3 – PANORÁMICA	7
1.6	ESCENA 5 - PARALLAX DEMO	11
2.	PANTALLA DE TÍTULO.....	13
3.	¿CÓMO EMPEZAMOS?	14
4.	ESTRUCTURA DE FICHEROS.....	15
5.	EL ENTORNO.....	16
6.	CHECKLIST DEL DESARROLLADOR	17
7.	VARIABLES GLOBALES	18
8.	FICHEROS DE TRADUCCIÓN	19
9.	FICHERO CONFIG.....	20
9.1	DIMENSIONES Y RESOLUCIÓN.....	25
9.2	SISTEMA DE PATHFINDING (BÚSQUEDA DE RUTAS).....	27
9.3	ESTILO NARRATIVO Y VISUAL.....	28
9.4	ESTILO DEL CURSOR.....	29
9.5	HERRAMIENTAS DE DESARROLLO	29
9.6	CONFIGURACIÓN DEL JUGADOR:	31
9.7	DEFINICIÓN DE PERSONAJES:	31
9.8	PARÁMETROS DE PERSONAJE:	31
10.	ESCENA DE INTRO	36
11.	ESCENA DE ENDING.....	39
12.	ESCENA 1 - AVENIDA DE LA PAZ	42
13.	ESCENA 2 – AYUNTAMIENTO.....	48
14.	ESCENA 4 – HABITACIÓN OSCURA	54
15.	ESCENA 3 – CALLE PANORÁMICA.....	58
16.	ESCENA 5 – DEMO PARALLAX	69



1. UN VISTAZO RÁPIDO

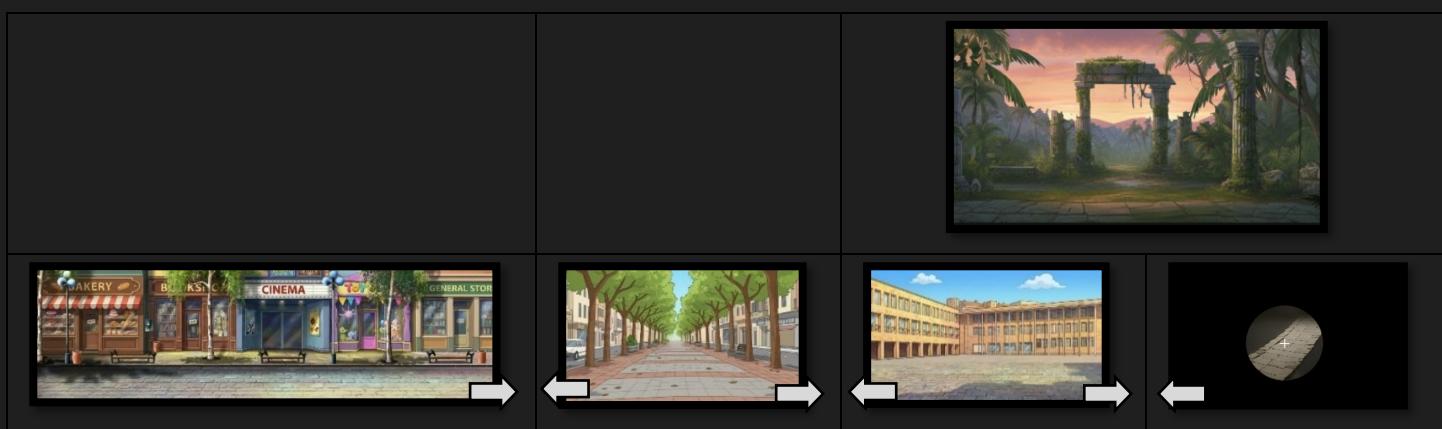
Se va a realizar un paseo rápido por la miniaventura donde en unas pocas pantallas el Engine muestra todas las combinaciones y características que permite configurar.

Se trata de 5 pantallas con diferentes características: sencilla, scroll, efecto parallax, zona oscura, etc. y en cada una de ellas estarán las acciones típicas de las aventuras gráficas: coger objetos, usarlos, combinarlos, hablar con NPC (Non-Player Character), uso de mapas, ejemplo de condicionales y variables, intercambio de personajes, uso de máquinas y objetos animados automáticos o usables, diálogos, cutscenes (escenas automáticas), etc. De esta forma el programador ya podrá partir de un ejemplo con estas acciones y crear las suyas propias con relativa facilidad.

En este manual se explicará paso a paso cómo hacer uso de cada una de ellas.

- El nivel requerido de programación es: **BÁSICO**
- El nivel requerido de diseño gráfico es: **MEDIO**

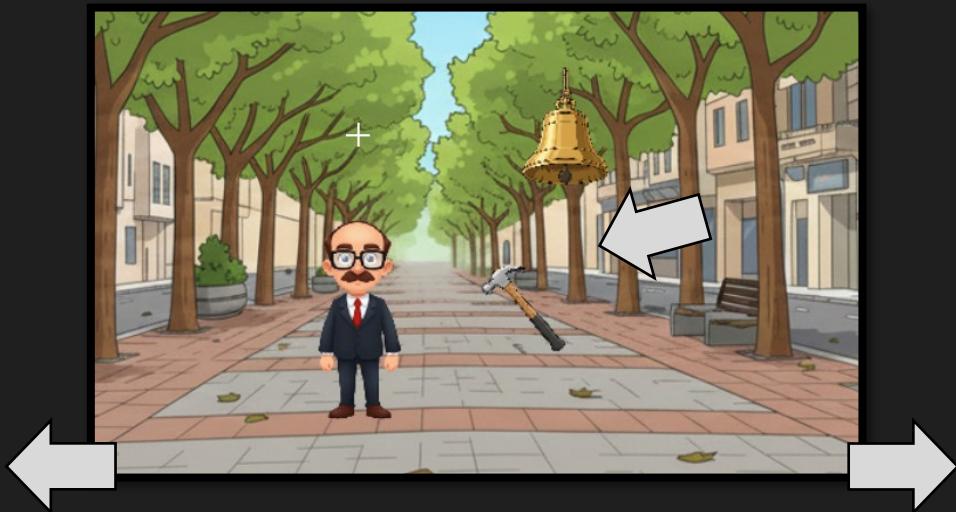
MAPA DE LA DEMO. Cuatro pantallas accesibles caminando. Una solo con el mapa.



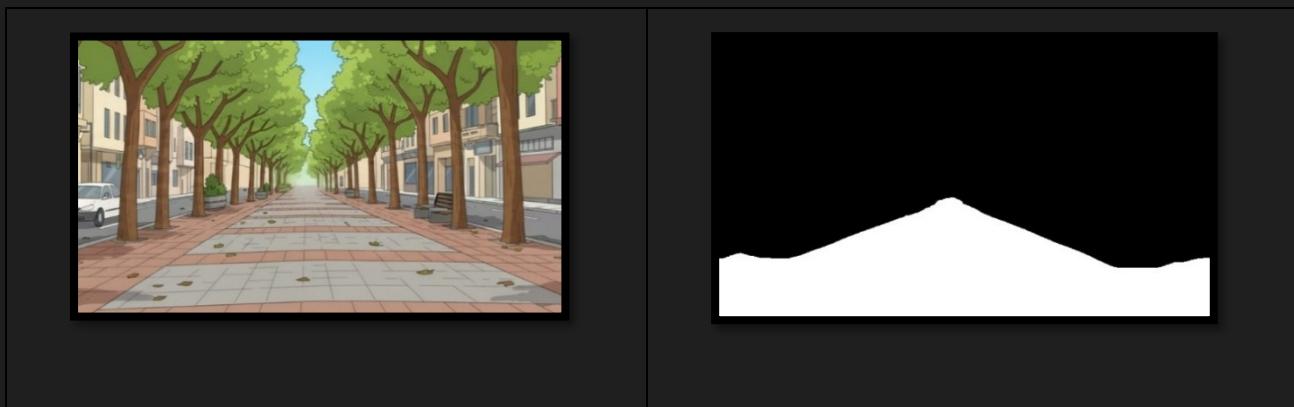


1.1 ESCENA 1 – AVDA. DE LA PAZ

Se trata de la pantalla de inicio, una pantalla simple, donde aparece un objeto – Hotspot - que cogemos y lo usamos con la campana. Se oirá el sonido de la campana, y el sonido de primer puzzle completado.



Todos los fondos del juego se componen por lo menos de dos imágenes. La imagen que se ve en pantalla y una imagen en blanco y negro que indica las zonas caminables seguras.





1.2 ESCENA 2 - EL AYUNTAMIENTO

En esta escena hay varios elementos importantes:

1.- Al comenzar la escena el NPC viaja automáticamente hasta el centro. Realizando una CutScene (se puede saltar con ESC y solo se realiza una vez). En ella se indica que ya estuvo aquí hace 25 años: que es un guiño a otra aventura gráfica que hice en 2001 y comenzaba en este sitio.

2.-Arriba a la izquierda hay un Hotspot no cargado si no generado por coordenadas.

3.-Deabajo, está la prensa grande que solo funciona cuando el personaje le dice “usar” creando su animación.

4.- A la derecha existe una prensa pequeña que está en animación continua.

5.- El siguiente Hotspot es un farol que cogeremos para ver la habitación oscura.

6.- El objeto siguiente es un objeto animado, que no podemos coger, pero si pasamos cerca de él, el NPC cambia de color. Se puede usar para hogueras, pasillos oscuros, etc.



Composición. Principal, Caminable, Escena de cambio de paleta del NPC por Hoguera.

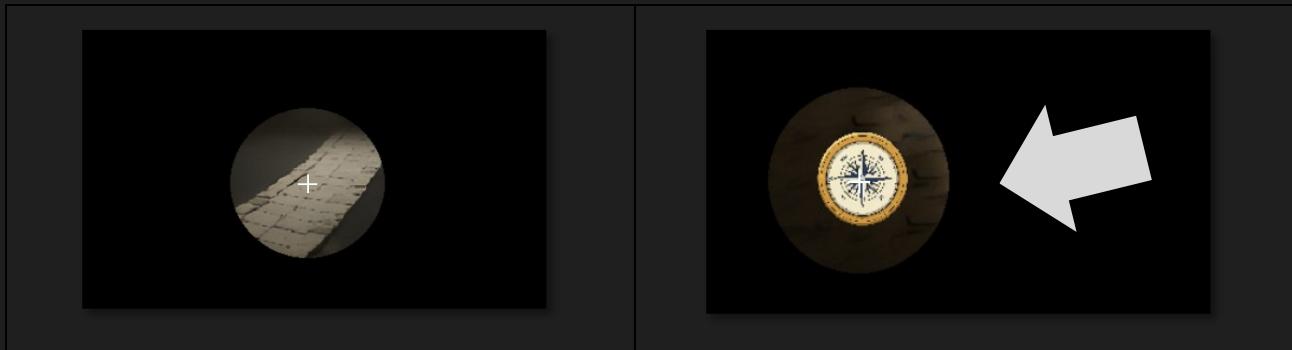




1.3 ESCENA 4 – HABITACIÓN OSCURA

Si avanzamos a la derecha, y entramos sin el farol, tendremos una habitación totalmente negra. Cogemos el farol y entramos de nuevo.

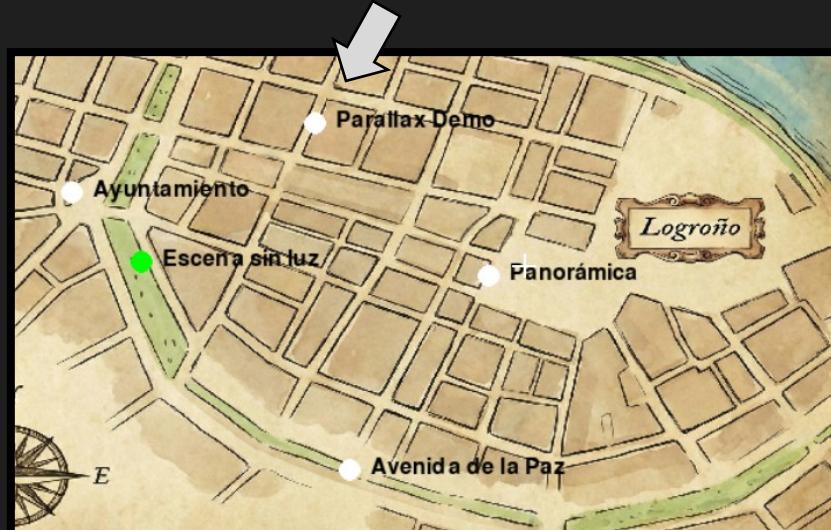
1.- con el farol podemos ver la habitación. Tendremos un foco de luz modificable.



2.- Cogemos el mapa y lo usamos: El mapa está pensado para ser útil de verdad. Aparte de llevarnos a sitios recónditos, ir de forma inmediata a ciertas escenas.

1.4 USO DEL MAPA

1.- Usamos el mapa para ir a la Escena Parallax que es la única que no es accesible andando.





1.5 ESCENA 3 – PANORÁMICA

La primera peculiaridad de esta escena es que tiene scroll horizontal. Se puede recorrer la calle sin saltar bruscamente a otra imagen. Y como es una escena larga, aprovecharemos para poner en ellas varios elementos.

1.- El NPC conocido como “Garba”. Podemos hablar con él, y dependiendo de si tenemos algún objeto que necesite o si hemos hablado otras veces nos dirá una cosa u otra. En este caso, nos pide, una linterna con pilas cargadas.



Composición de la escena:

El sistema detecta el scroll automáticamente y se puede modificar la velocidad.



2.- Objeto compuesto – Crafting: “Cogeré la linterna sin pilas + las pilas” y usándolas en el inventario, crearé una linterna con pilas que daré al NPC “Garba”. La conversación será diferente si tengo la linterna con pilas que si no la tengo.



HABLAR a Garba

¡Mira! Ya he conseguido ponerle pilas a la linterna.

Hola Garba, ¿qué haces aquí?

¿Has visto algo sospechoso?





3.- El siguiente objeto que tenemos es un Hotspot con diferente animación y estado asignado. Aquí se trata de una papelera, pero se puede usar a cualquier cosa, puertas, palancas, etc. Tiene el estado de abrir y cerrar. Es un Sprite Sheet de dos imágenes.



4.- Objeto animado con recorrido propio. En este caso, un bonito cuervo negro recorre la pantalla volando en contra del scroll. Con este formato puedes poner nubes, ríos, personajes moviéndose desesperadamente de un lado a otro, vehículos, etc. Se trata de Sprite Sheets con la cantidad de imágenes que quieras, y donde definir el tiempo de cada sprite en pantalla así como diferentes recorridos.



5.- Existe otro objeto que nuestro personaje principal no puede coger porque pesa bastante, así que tendrá que intercambiarse con otro NPC del que tendremos el control para coger la pala e intercambiarla. Así usamos esta escena, para explicar el intercambio de personajes.



Nos acercamos al otro personaje y pulsamos en este caso el verbo USAR. En ese momento podremos controlar a “Bart”, que en este caso es solo nuestro personaje con un precioso traje rojo de franela. Con Bart cogemos el objeto PALA y pasa a nuestro inventario.

Se pueden crear tantos NPC como se quieran, y se les puede dar la libertad de moverse por todos los escenarios o restringirlos.

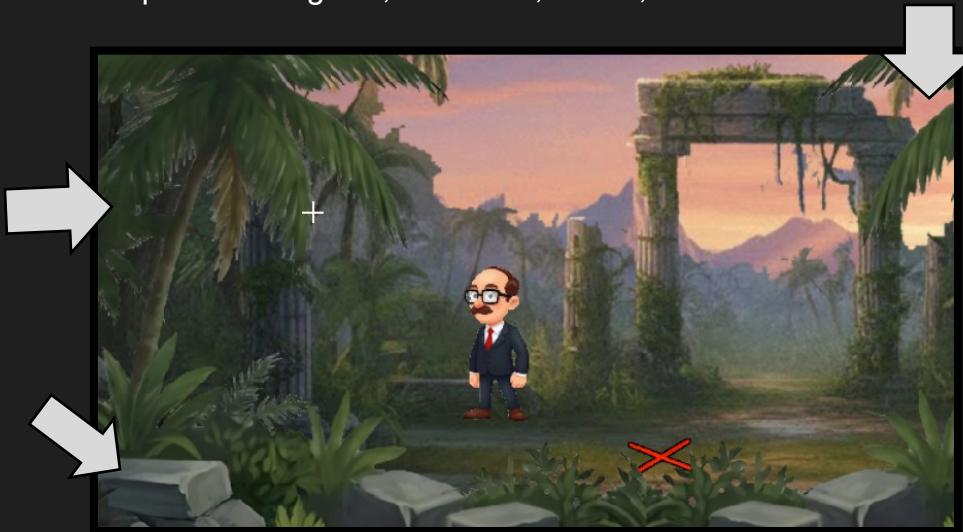


1.6 ESCENA 5 - PARALLAX DEMO

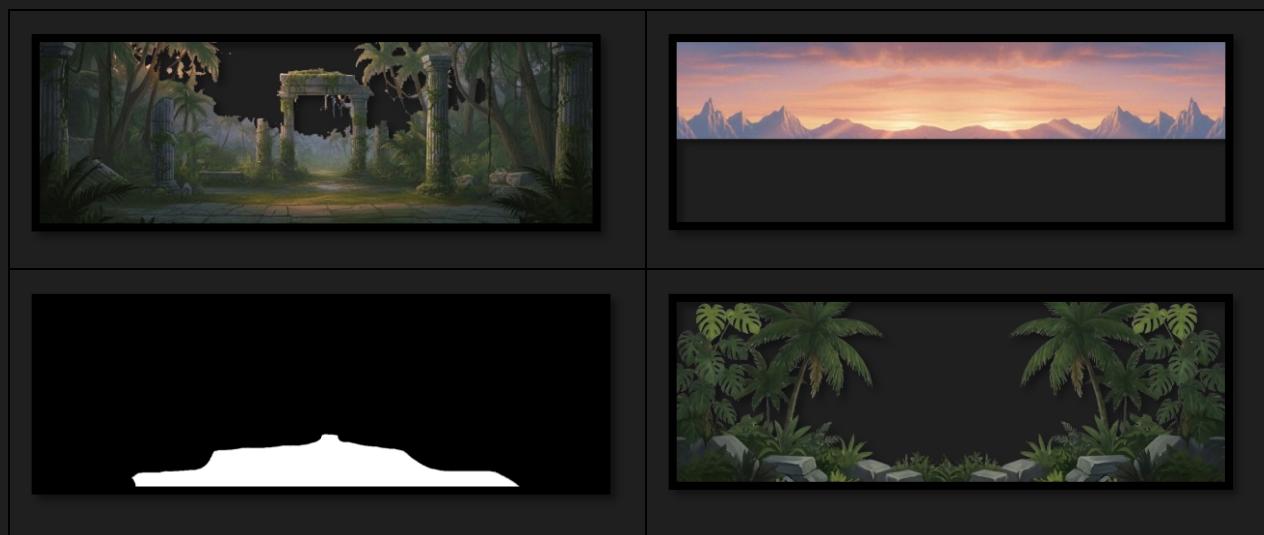
Esta escena está configurada para usar el efecto Parallax y dar más profundidad. Por si la necesitas usar en diferentes escenas. Además puede funcionar con scroll automático del cielo o capa trasera. Se compone de 4 imágenes. Que son:

- imagen de cielo o trasera (automática o manual al caminar). FAR
- Imagen media o imagen principal. MIDDLE
- Imagen que indica la zona caminable. WALKABLE
- Imagen más cercana. Automática con ajuste de velocidad. NEAR

En todas ellas se puede configurar, velocidad, zonas, etc

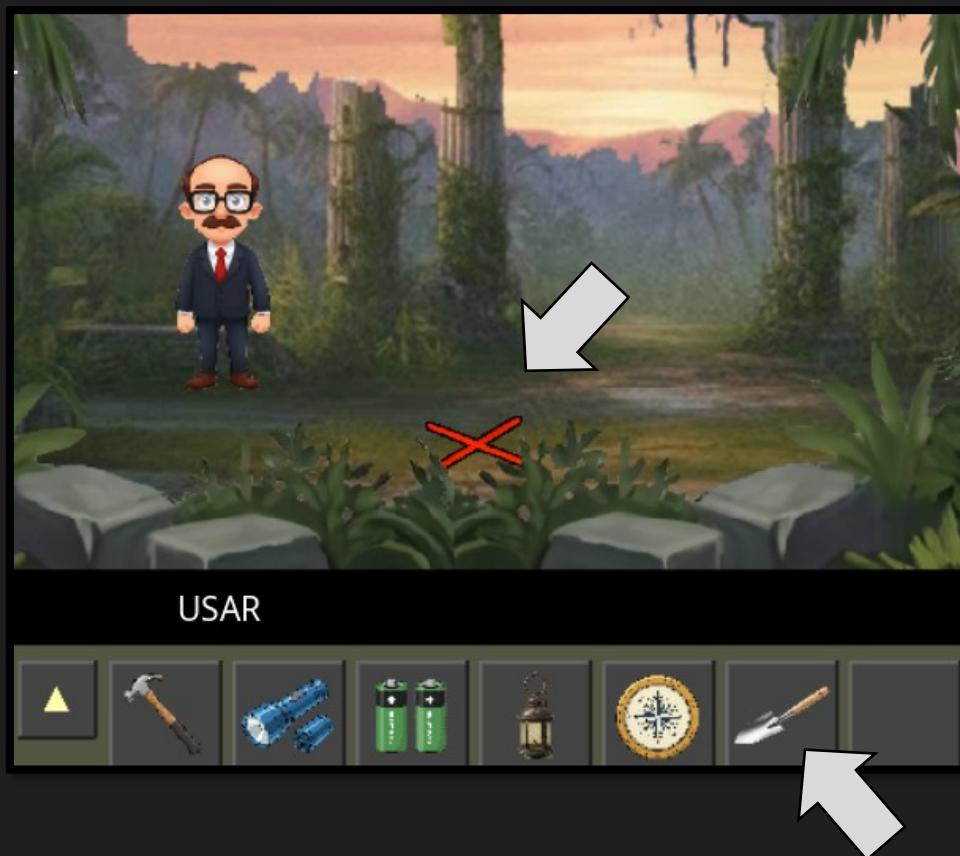


La composición se hace con cuatro imágenes.





En esta escena, está también el final de la demo. Usando la pala en la “X” pirata, nos lleva a la animación “ending”.





2. PANTALLA DE TÍTULO

Pantalla de inicio, previamente hará una carga rápida de las primeras imágenes, sonidos, que irá liberando de la RAM cuando se dejan de usar.

- NUEVA PARTIDA
- CARGAR PARTIDA
- IDIOMA
- CREDITOS
- SALIR

Lo primero será elegir idioma, y si no está el tuyo, será tan fácil, como duplicar cualquier fichero de la carpeta /languages y traducirlo.



Partimos de 57 idiomas disponibles en esta versión. Y se ha sido muy cuidadosos en este tema porque era importante para nosotros que el Engine estuviera disponible para todo el mundo. Son los 50 idiomas más hablados de este planeta tan diverso y molón.

Para ello el sistema usa una “fuente libre” con todos los caracteres posibles, y no necesita cargar más; haciéndolo independiente y funcional en cualquier plataforma y país.

El sistema además está programado para ser traducido “al vuelo”, y poder cargar partidas realizadas en otros idiomas, así como para adaptar todos los textos, verbos, funciones de uso, a las diferentes tipografías, calculando tamaños y reduciendo hasta que caben en el lugar adecuado.

Esto llevó una semana de trabajo XD.



PS. Sí, antes que lo preguntes, está traducido a Klingon obviamente.



3. ¿CÓMO EMPEZAMOS?

Antes de programar nada, se recomienda cacharrear un poco con el script por defecto, para que como programador, y con la ayuda en diseños y textos que vayas a disponer te hagas una idea y mapa mental de lo que vas a necesitar y lo que puedes hacer.

Ten en cuenta que cuando tomes ya ciertas decisiones de tamaños, configuraciones, luego volver atrás requeriría un extra de trabajo. Nos referimos a tamaños de pantallas, etc.

El sistema está optimizado y realiza diferentes escalados hasta el full screen, pero para optimizar resultados, saber qué quieres para empezar es importante. Ten en cuenta que tendrás que posicionar objetos, NPCs, animaciones, y todo escalado a la resolución de tu juego.

Nuestra recomendación es comenzar con un sistema de pantalla grande aunque luego muestres al usuario una diferente.

Las fuentes, en este caso están programadas para pintarse dos veces, una en el tamaño definido por el usuario, y luego hace un repintado en HD para que los textos se vean siempre nítidos a cualquier resolución. Pero las imágenes, una vez definidas las escalamos con modelos matemáticos que son muy buenos, pero no perfectos. Por eso, mejor comenzar con altas resoluciones, y luego ir bajando.

Una vez que tengas tu historia, entra la labor de los diseñadores gráficos. Realmente llevará más trabajo que cualquier otra cosa. La gestión de Sprites “todavía” no lo hace la IA, así que toca tirar de edición gráfica.

Para los textos, recomendamos trabajar en tu idioma nativo, y no traducir hasta que hayas terminado y estés seguro del resultado. Los ficheros de traducción son ficheros yaml, más cómodos de traducir que ficheros json. Mas visuales.



4. ESTRUCTURA DE FICHEROS

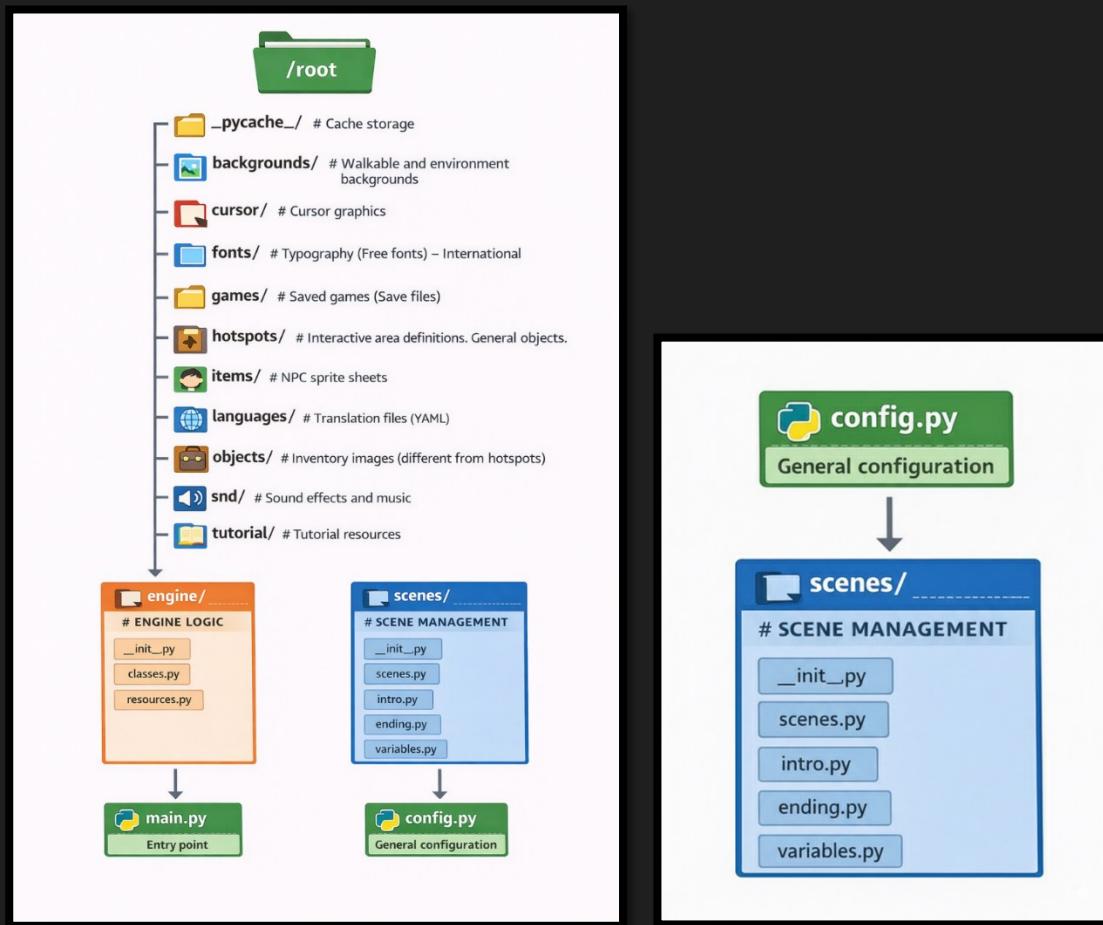
El sistema está estructurado a prueba de errores, separando el código en directorios y carpetas todos los scripts y sistemas gráficos. Para que cada apartado sea independiente.

No hay que tocar nada del Engine en la programación, ya que las configuraciones, y escenas son independientes. Aun así, lógicamente este es un software “open source” total, y el programador avanzado podrá ajustar a su gusto alguna característica que eche en falta.

Si el avance es bueno para la comunidad, que proponga el ajuste ya que PyCAPGE, estará en GIT para su distribución y actualizaciones.

Tenemos:

- __pycache__ → Caché de objetos.
- \backgrounds → Solo fondos de las escenas. Zonas caminables, etc
- \cursor → Para poder usar diferentes tipos de cursor.
- \engine → El código que no debes tocar si no sabes XD
- \fonts → Fichero de fuente con todos los caracteres del mundo.
- \games → Partidas guardadas. Formato json.
- \hotspots → Objetos que irán en pantalla para interactuar.
- \items → Los NPC y personajes principales.
- \languages → Archivos de idioma, (yaml). Automático.
- \objects → Las imágenes que irán a inventario. (No Hotspots)
- \scenes → Tus escenas, intro, final, y variables de tu juego.
- \snd → Ficheros de audio.
- \tutorial → Esto que estás leyendo.
- config.py → Fichero de configuración. El que puedes editar.
- main.py → Fichero de carga principal. No editar si no estás seguro.
- readme.txt → Lo de siempre.



En teoría para realizar tu aventura gráfica a nivel de código solo tendrás que trabajar con estos ficheros: la configuración general, la intro, el final, las variables que uses a nivel global de acciones-puzzles completados y las escenas de tu juego.

5. EL ENTORNO

Instalación del Entorno (El gran olvidado):

- "Instala Python 3.x"
- "Instala las dependencias: Ejecuta pip install pygame pyyaml en tu terminal".
- "Cómo ejecutar el juego: Doble clic en main.py o ejecutar: python main.py desde la consola".



6. CHECKLIST DEL DESARROLLADOR

PASOS PARA CREAR UNA NUEVA ESCENA:

1. **Gráficos:** Crea el fondo (bg.png) y su máscara de caminar (bg_bm.png) en la carpeta /Backgrounds.
2. **Config:** Añade el nombre de la escena
3. **Código:**
 - o Instancia la clase Scene en scenes.py.
 - o Define on_enter (música) y on_exit.
 - o Añade las exits (salidas a otras escenas).
 - o Añade los hotspots (objetos interactivos).
4. **Registro:** Añade scene_manager.add_scene(mi_nueva_escena) al final del archivo.
5. **Prueba:** Usa el modo Debug para ajustar las coordenadas de los hotspots.



7. VARIABLES GLOBALES

```
# =====
# MANUAL DE DESARROLLADOR: GESTIÓN DE ESTADO GLOBAL (variables.py)
# =====

# El diccionario GAME_STATE es el objeto que el motor guarda y carga
# cuando el jugador salva la partida. Todo lo que esté aquí dentro es persistente.

GAME_STATE = {
    # -----
    # 1. ESTADO DE OBJETOS (INVENTARIO Y MUNDO)
    # -----
    # Estas variables controlan si un objeto ha sido recogido del suelo.
    # Regla: Si es 'False', el objeto aparece en la escena.
    # Si es 'True', el objeto desaparece del mundo (porque ya está en el
    # inventario).

    "campana_recogida": False,
    "martillo_recogido": False,
    "farol_recogido": False,

    # Ejemplo de estados combinados:
    "pilas_linterna_recogidas" : False,      # Las pilas por separado
    "linterna_sin_pilas_recogida" : False,    # La linterna vacía
    "linterna_con_pilas_recogida": False,     # El resultado de combinarlas (Crafting)

    # -----
    # 2. PROGRESO DE LA HISTORIA (FLAGS)
    # -----
    # Se usan para saber si el jugador ya ha visto un evento o ha activado algo.
    "intro_ayuntamiento_vista": False,
    "pala_recogida": False,

    # -----
    # 3. CONTROL DE PERSONAJES (MULTIPLAYER / NPCs)
    # -----
    # Estas variables definen quién es el personaje "jugable" en este momento.
    # El motor usa esto para intercambiar el control entre personajes.

    "controlando_gilo": True,   # Si es True, el jugador mueve a Gilo.
    "controlando_bart": False,  # Si es True, el jugador mueve a Bart.

    # Nota para el programador:
    # Al cambiar estas variables en tiempo de ejecución, el motor ocultará
    # la versión "NPC" del personaje y activará la versión "PLAYER".
}
```



8. FICHEROS DE TRADUCCIÓN

Se ha elegido el formato yaml vs json.

```
=====
# TRANSLATION: ENGLISH (en.yaml)
=====

language_name: "English"

=====
# VERB SYSTEM - VERB BOX
=====

verbs:
    WALK: "GO"           # <--- KEY IN UPPERCASE
    OPEN: "OPEN"
    CLOSE: "CLOSE"
    PUSH: "PUSH"
    PULL: "PULL"
    PICK UP: "PICK UP"
    USE: "USE"
    TALK TO: "TALK TO"   # <--- KEY IN UPPERCASE
    GIVE: "GIVE"
    LOOK AT: "LOOK AT"
    WITH: "with"

=====
# MAIN MENU - ACTIVE WITH F2
=====

menus:
    FILE_TITLE: "FILE"
    SAVE_CMD: "SAVE"
    LOAD_CMD: "LOAD"
    HELP_TITLE: "HELP"
    NO_OPT: "NO"
    GAME_HELP_OPT: "GAME HELP"
```

9. FICHERO CONFIG

```
# =====
# CONFIGURACIÓN PRINCIPAL - ¡EDITA AQUÍ!
# =====
CONFIG = {
    # --- DIMENSIONES INTERNAS (Lienzo Virtual) ---
    # Esta es la resolución "real" de tus gráficos. Si haces pixel art, usa valores bajos (ej. 320x240).
    # Si usas gráficos HD, usa valores altos (1920x1080).
    "GAME_WIDTH": 800,
    "GAME_HEIGHT": 638, # Altura total incluyendo la interfaz (Verbos + Inventario).

    # --- DIMENSIONES DE LA VENTANA (Lo que ve el usuario) ---
    # El motor estirará el juego para llenar esta ventana manteniendo el aspecto.
    # Recomendación: Empieza con una resolución cómoda para trabajar, como 1280x720.
    "WINDOW_WIDTH": 960,
    "WINDOW_HEIGHT": 766,

    # --- JUGABILIDAD ---
    "PLAYER_SPEED": 3.5,          # Velocidad de movimiento del personaje (Píxeles por frame).
    "TEXTBOX_HEIGHT": 40,         # Altura de la barra negra donde aparece el texto "Mirar Campana".
    "VERB_MENU_HEIGHT": 128,       # Altura del panel de verbos e inventario (parte inferior).
    "BOTTOM_MARGIN": 20,          # Un pequeño margen extra al final de la ventana.

    # --- CÁMARA (Scroll) ---
    # Controla qué tan suave sigue la cámara al jugador.
    # Valores bajos (1.0) = Cámara lenta/pesada. Valores altos (10.0) = Cámara instantánea.
    "CAMERA_SMOOTHING": 5.0,

    # --- PATHFINDING (Inteligencia de movimiento) ---
    # Define cómo calcula el personaje la ruta para esquivar obstáculos.
    # "EUCLIDEAN": El más preciso y natural (permite cualquier ángulo).
    # "MANHATTAN": Estilo retro cuadriculado (solo se mueve en cruz).
```



```
# "DIAGONAL": Permite diagonales pero prefiere líneas rectas.
"PATHFINDING_TYPE": "EUCLIDEAN",

# Tamaño de la cuadricula invisible de navegación.
# 5 = Muy preciso (el personaje pasa por huecos pequeños).
# 20 = Menos preciso, mejor rendimiento (estilo juegos muy antiguos).
"PATHFINDING_GRID_SIZE": 10,

# --- ESTILO NARRATIVO ---
# Define dónde aparecen los textos cuando alguien habla.
# "LUCAS": Texto flotante de colores sobre la cabeza del personaje (Monkey Island).
# "SIERRA": Caja de texto centrada con retrato (King's Quest / Larry).
# "SUBTITLE": Texto fijo en la parte inferior de la pantalla (Cine).
"NARRATION_STYLE": "LUCAS",

# --- SISTEMA ---
"ENABLE_SOUND": True,           # True = Sonido activado. False = Mudo (útil para desarrollar rápido).

# --- MODOS DE DESARROLLO (DEBUG) ---
"DEBUG_MODE": False,            # Si es True, muestra cajas de colisión y consola de errores.
"SHOW_HINTS_ONLY": False,       # Si es True, solo muestra nombres de objetos (como ayuda al jugador).
"SHOW_WALKABLE_MASK": False,   # (Tecla F4) Muestra en rojo por dónde se puede caminar. Útil para artistas.

# --- CURSOR ---
# "MODERN": Cambia el icono según la acción (Ojo, Mano, Pies...). Requiere imágenes en /cursor.
# "CLASSIC": Usa una cruz simple (+) para todo, estilo SCUMM clásico.
"CURSOR_STYLE": "CLASSIC",

"DOUBLE_CLICK_MS": 500,         # Tiempo en milisegundos para detectar doble clic (salida rápida).

# --- ANIMACIÓN "IDLE" ---
```



```
# Tiempo en segundos que el personaje debe estar quieto para hacer su animación "especial" (ej. rascarse, mirar reloj).
"IDLE_COOL_THRESHOLD": 10.0,

# --- INICIO RÁPIDO (Solo para desarrolladores) ---
# ID de la escena para saltar el menú y la intro. Útil para probar una sala concreta.
# Poner "None" para jugar normal desde el título.
"DEV_START_SCENE": "None", # Ejemplo: "DARK_ROOM"
}

# =====
# CONFIGURACIÓN DEL JUGADOR
# =====

PLAYER_CONFIG = {
    "NAME": "Gilo",                      # Nombre interno (para logs y debug).
    "ASSET_PREFIX": "gilo",                # ¡IMPORTANTE! Prefijo de los archivos de imagen.
                                            # Si pones "gilo", el motor buscará "gilo_wd.gif", "gilo_talk.gif", etc.
    "TEXT_COLOR": (255, 255, 255),        # Color del texto cuando este personaje habla (RGB).
    "CHAR_ID": "Gilo"                     # ID que conecta con la definición de abajo (CHAR_DEFS).
}

# =====
# DEFINICIÓN DE PERSONAJES (Spritesheets)
# =====

# Aquí registras CADA personaje que tenga animaciones complejas (Player o NPCs).
CHAR_DEFS = {
    "Gilo": {
        "prefix": "gilo",    # Prefijo del archivo.
        "width": 163,        # Ancho de CADA frame individual dentro de la hoja de sprites.
        "height": 300,       # Alto de CADA frame.
        "base_scale": 0.3,   # Escala inicial (0.3 significa que se verá al 30% de su tamaño original).
    }
}
```



```
# Mapa de animaciones: Define cuántos frames tiene cada acción.  
# Si tu archivo "gilo_wd.gif" (walk down) tiene 6 dibujos, pon 6 aquí.  
"frames": {  
    "walk_down": 6, "walk_left": 6, "walk_right": 6, "walk_up": 6,  
    "talk_down": 6, "talk_left": 6, "talk_right": 6, "give": 6,  
    "idle": 1,      "push": 1,      "pull": 1,      "pick": 1,  
    "open": 1,      "close": 1,     "cool": 6,  
}  
},  
# Puedes añadir tantos personajes como quieras copiando el bloque anterior:  
"Bart": {  
    "prefix": "bart",  
    "width": 163,  
    "height": 300,  
    "base_scale": 0.3,  
    "frames": {  
        "walk_down": 6, "walk_left": 6, "walk_right": 6, "walk_up": 6,  
        # ... resto de animaciones ...  
    }  
},  
}  
  
# ======  
# CRÉDITOS DEL JUEGO  
# ======  
# Texto que aparece al pulsar "Créditos" en el menú principal.  
CREDITS_TEXT = """  
======  
TU JUEGO DE AVENTURA  
======  
Creado por: Tu Nombre
```



```
=====
MOTOR PyCAPGE
=====
Código base: Garba
...
"""

# =====
# CONFIGURACIÓN DE TEXTOS (Fuentes)
# =====
# Ruta a tu archivo de fuente .ttf. Si no existe, usará Arial por defecto.
UI_FONT_PATH = os.path.join("fonts", "ui_font.ttf")

TEXT_CONFIG = {
    # Velocidades de habla (segundos por letra).
    # Menor número = Más rápido.
    "SPEED_SLOW": 0.15,
    "SPEED_MEDIUM": 0.08, # Velocidad por defecto.
    "SPEED_FAST": 0.04,

    # Tamaños de fuente para los diálogos.
    "SIZE_SMALL": 18,
    "SIZE_MEDIUM": 20,
    "SIZE_LARGE": 32,

    # Configuración activa inicial
    "CURRENT_SPEED": "SPEED_MEDIUM",
    "CURRENT_SIZE": "SIZE_MEDIUM",
    "FONT_NAME": UI_FONT_PATH,
    "OUTLINE_WIDTH": 2      # Grosor del borde negro alrededor de las letras (para legibilidad).
```



Referencia para config.py del motor PyCAPGE.

El archivo config.py es el corazón estático de PyCAPGE. Aquí se definen las constantes globales que determinan el comportamiento del motor,

la resolución, la física del movimiento, la estética de la interfaz y la definición de los *sprites* de los personajes.

Regla de Oro: Este archivo está diseñado para ser modificado. Sin embargo, se debe respetar estrictamente la sintaxis de Python (diccionarios,

listas, tipos de datos booleanos/enteros/floatantes).

9.1 Dimensiones y Resolución

PyCAPGE separa la "Resolución Lógica" (el lienzo de pixel art) de la "Resolución Física" (la ventana de Windows).

GAME_WIDTH / GAME_HEIGHT

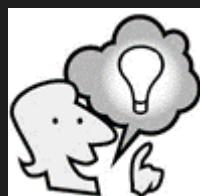
- *Descripción:* Define la resolución interna del juego. Es el tamaño del lienzo ("canvas") donde se dibujan los fondos y personajes antes de ser escalados.
- *Recomendación:* Para un estilo "Retro HD" o pixel art nítido, usa **800x450** (16:9) o **640x360**. Si buscas un estilo VGA clásico (SCUMM), usa **320x200** o **320x240**.
- *Impacto:* Afecta directamente al rendimiento. Resoluciones más bajas consumen menos recursos y facilitan el pixel art.

WINDOW_WIDTH / WINDOW_HEIGHT



- *Descripción*: Define el tamaño inicial de la ventana que abre el Sistema Operativo.
- *Funcionamiento*: El motor escala automáticamente lo que ocurre en GAME_WIDTH para llenar WINDOW_WIDTH, aplicando filtros de suavizado (*smoothscale*) o pixel-perfect según la configuración interna.
- *Uso*: Se recomienda poner una resolución estándar moderna (ej. 1280x720 o 1920x1080) para que el juego se vea bien en monitores actuales al iniciarse.

Para la demo he usado una resolución de 800 * 638. Con esto consigo que la pantalla visible se dibuje con una resolución de 800*450, que es un 16:9 estándar. El resto lo ocupan la zona de texto y los verbos. Posteriormente para visualizarlo lo he aumentado un 20%. Mi consejo es hacerlo al revés comenzar con muy altas resoluciones, y usar luego una más pequeña en ventana para hacer el juego. Así, en el escalado, si alguien tiene un pantalla grande se verá más nítido.



comenzar con muy altas
resoluciones, y usar luego una más
pequeña en ventana para hacer el juego .



9.2 Sistema de Pathfinding (Búsqueda de Rutas)

```
# --- PATHFINDING ---
# "EUCLIDEAN": Más preciso, movimiento natural (usa raíz cuadrada).
# "MANHATTAN": Más rápido, ideal para rejillas tipo ciudad (sin diagonales).
# "DIAGONAL": (Chebyshev) Rápido, permite diagonales pero menos preciso que Euclidean.
"PATHFINDING_TYPE": "EUCLIDEAN",
"PATHFINDING_GRID_SIZE": 10, # 5 para precisión alta, 20 para rendimiento/retro
```

Controla cómo los personajes navegan por el escenario evitando obstáculos.

- "*EUCLIDEAN*": Movimiento más natural y orgánico. Calcula distancias reales (línea recta). Es lo recomendado para aventuras modernas.
- "*MANHATTAN*": Movimiento tipo "cuadrícula" (solo arriba, abajo, izquierda, derecha). Útil para juegos retro muy estrictos o ciudades con calles rectas.
- "*DIAGONAL*" (*Chebyshev*): Permite diagonales pero calcula el coste de forma simplificada. Un punto medio en rendimiento.



9.3 Estilo Narrativo y Visual

```
# SISTEMA DE NARRACIÓN
# "LUCAS": Texto flotante sobre la cabeza del personaje.
# "SIERRA": Texto en una caja centrada (como si fuera un cómic/narrador).
# "SUBTITLE": Texto en la parte inferior de la pantalla (fijo).
"NARRATION_STYLE": "LUCAS",
```

- "LUCAS": Texto flotante sobre la cabeza del personaje. El color del texto depende de la configuración del personaje. (Estilo *Monkey Island*).
- "SIERRA": Texto en una caja centrada o con retrato (aunque la implementación actual centra el texto).
- "SUBTITLE": Texto fijo en la parte inferior de la pantalla, tipo cine.

Lucas	Subtitle	Sierra



9.4 Estilo del Cursor

- *Opciones:*
 - "CLASSIC": Dibuja una cruz simple (+) mediante código. Ideal para prototipar o estilo minimalista.
 - "MODERN": Usa gráficos .gif o .png animados que cambian según la acción (ojos para mirar, mano para coger, etc.). Requiere que los archivos existan en la carpeta cursor/.



9.5 Herramientas de Desarrollo

```
"ENABLE_SOUND": True,  
"DEBUG_MODE": False,  
"SHOW_HINTS_ONLY": False,  
"SHOW_WALKABLE_MASK": False,      # VARIABLE PARA MOSTRAR MÁSCARA DE CAMINAR F4--
```



F3 – DEBUG MODE (coord x,y)



F1. GAME HELP



F4 - WALKABLE



9.6 Configuración del Jugador:

Define quién es el protagonista al iniciar el juego.

```
# =====
# CONFIGURACIÓN DEL JUGADOR POR DEFECTO
# =====
PLAYER_CONFIG = {
    "NAME": "Gilo",          # Nombre para mostrar en textos/diálogos Bart Gilo Indy, Garba
    "ASSET_PREFIX": "gilo",   # Prefijo de los archivos (ej: "player_walk.gif") bart , indy, garba
    "TEXT_COLOR": (255, 255, 255), #blanco
    "CHAR_ID": "Gilo"
}
```

9.7 Definición de Personajes:

Esta es la sección más compleja y potente para los artistas. Aquí se define cómo el motor debe "recortar" y animar las hojas de sprites (*spritesheets*). Cada personaje es una clave en el diccionario (ej: "Gilo", "Bart").

9.8 Parámetros de Personaje:

prefix: La cadena base del nombre de archivo. El motor buscará archivos con el patrón: prefix_accion.gif.

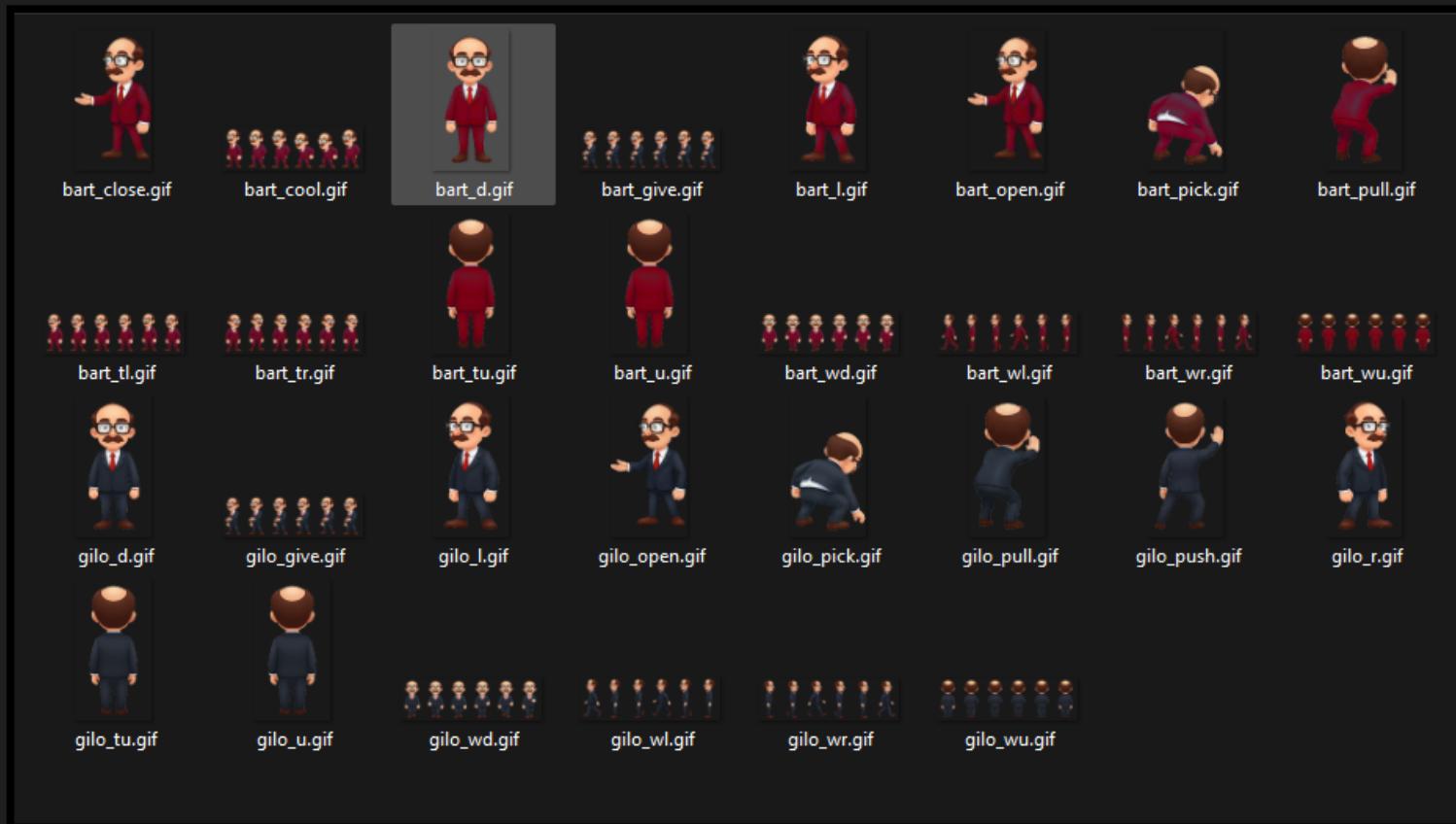
- *Ejemplo:* Si prefix es "indy", buscará indy_wd.gif (walk down), indy_talk_left.gif, etc.

Clave en frames	Sufijo de Archivo	Acción
"walk_down"	_wd.gif	Caminar hacia abajo (frente)
"walk_left"	_wl.gif	Caminar izquierda
"walk_right"	_wr.gif	Caminar derecha
"walk_up"	_wu.gif	Caminar arriba (espalda)
"talk_down"	_td.gif	Hablar de frente
"talk_left"	_tl.gif	Hablar perfil izquierdo
"talk_right"	_tr.gif	Hablar perfil derecho
"give"	_give.gif	Acción de dar objeto
"idle"	_d.gif / _l.gif...	Estado de reposo (suele ser 1 frame)
"push"	_push.gif	Empujar
		etc





- Dimensiones de **un solo frame** (fotograma) dentro de la hoja de sprites. Todos los frames de una animación deben tener el mismo tamaño.
- **base_scale**: Escala inicial del sprite. 1.0 – original-. 0.3 reduce al 30%. Esto se multiplica luego por la escala de profundidad de la escena.
- **frames**: Diccionario que indica cuántos fotogramas tiene cada animación específica.
- **Se ve mejor con unos ejemplos:** En el directorio ITEMS tenemos los NPC





- Tenemos este fichero : gilo_wd.gif



```
"Gilo": {  
    "prefix": "gilo",  
    "width": 163,      # Ancho del frame  
    "height": 300,     # Alto del frame  
    "base_scale": 0.3, # escala de partida.  
    "frames": {  
        "walk_down": 6, "walk_left": 6, "walk_right": 6, "walk_up": 6,  
        "talk_down": 6, "talk_left": 6, "talk_right": 6, "give": 6,  
        "idle": 1,       "push": 1,      "pull": 1,       "pick": 1,  
        "open": 1,       "close": 1,     "cool": 6,  
    }  
},
```

Vemos que lo hemos definido con una anchura de 163 pixels, por 300 de alto cada uno, y que la animación son 6 items.



Luego para insertarlo en el juego por defecto le hemos asignado un escalado muy alto: 0.3.

Esto es que hicimos el dibujo muy grande para la resolución final.

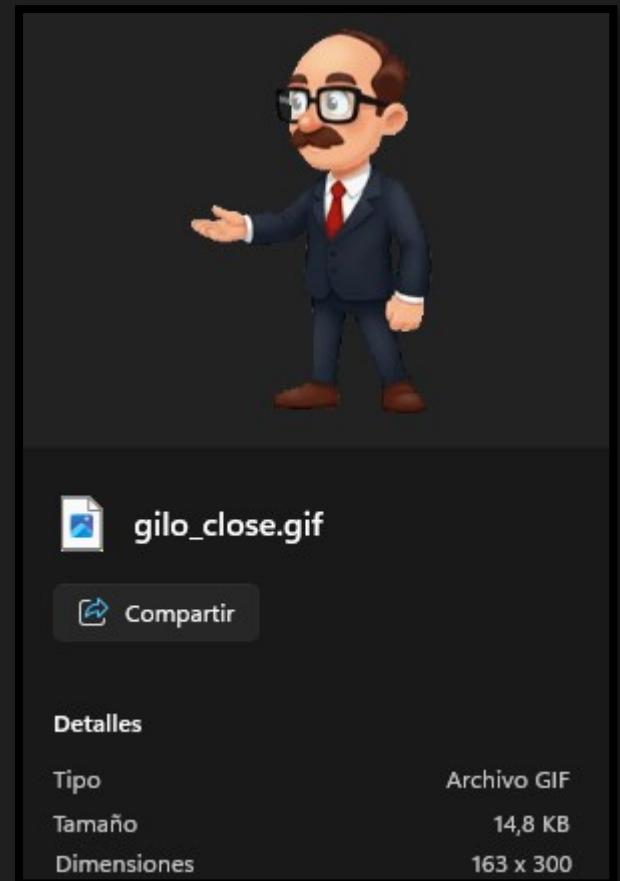
Por eso, como se ha indicado, mejor comenzar con resoluciones altas

De esta forma definimos más npc.

Teniendo en cuenta el nombre asignado, y el prefijo para los gráficos.

Otro ejemplo, en este caso, de un item sin animación

Con el nombre : gilo_close.gif → Una imagen. 163 x 300 → Close":1





10. ESCENA DE INTRO

Se ha creado una escena especial de inicio o presentación que el usuario puede usar o crear las suyas propias con una cutsecene o como prefiera.

Pero para facilitar las cosas, esta intro con diapositivas puede ser de utilidad.

La variable `self.playlist` es una **lista de diapositivas**. Cada elemento dentro de los corchetes [...] es un diccionario { ... } que representa una pantalla

o "slide" de la secuencia.

```
# =====
# CONFIGURACIÓN DE DIAPOSITIVAS (INTRO)
# =====

def start_intro(self):
    # 1. CARGA DE TEXTOS
    # Recupera el diccionario de textos cinematográficos cargado desde el YAML actual.
    # Esto permite que si cambias de idioma, 'cine_texts' contenga las traducciones correctas.
    cine_texts = self.get_texts()

    # 2. DEFINICIÓN DE LA PLAYLIST (SECUENCIA DE SLIDES)
    # Se define una lista de diccionarios. Cada diccionario es una "diapositiva" o escena.
    # El gestor (IntroManager) las mostrará en orden secuencial.
    self.playlist = [
        # --- DIAPOSITIVA 1: Inicio ---
        {
            "image": "avda_paz.jpg",           # Imagen de fondo (carpeta /backgrounds).
```



```
# Texto narrativo a mostrar.  
# (YAML es.yaml -> cinematics -> INTRO_1)  
# -> "Año 2001. La ciudad parecía tranquila..."  
"text": cine_texts["INTRO_1"],  
  
    "effect": "none"                      # Sin efecto de zoom, imagen estática.  
,  
  
# --- DIAPOSITIVA 2: El Nudo ---  
{  
    "image": "ayun.jpg",                  # Cambiamos a la imagen del Ayuntamiento.  
  
    # (YAML es.yaml -> cinematics -> INTRO_2)  
    # -> "Pero en el Ayuntamiento se tramaba algo oscuro."  
    "text": cine_texts["INTRO_2"],  
  
    "effect": "none"                      # Imagen estática.  
,  
  
# --- DIAPOSITIVA 3: La Llamada ---  
{  
    "image": "darkness-room.jpg",        # Imagen oscura.  
  
    # (YAML es.yaml -> cinematics -> INTRO_3)  
    # -> "Solo un héroe (o dos) podrían detenerlo."  
    "text": cine_texts["INTRO_3"],  
  
    "effect": "none"  
,  
  
# --- DIAPOSITIVA 4: Logo y Música ---
```



```
{  
    "image": "logo_pycapge.png",      # Logo del motor o juego.  
    "text": "",                      # Cadena vacía: No muestra texto, solo imagen.  
  
    # Inicio de música sincronizada con esta diapositiva.  
    # Se reproduce "sintonia3.ogg" desde la carpeta /snd.  
    "music": "sintonia3.ogg",  
    "music_loops": 0,                 # 0 = Suena una sola vez (sin bucle).  
  
    # --- EFECTO VISUAL (ZOOM) ---  
    "effect": "zoom_out",            # La imagen empieza grande y se aleja.  
    "zoom_intensity": 1.0,             # Intensidad alta: Empieza al doble de tamaño (200% -> 100%).  
  
    # Duración personalizada para esta slide (sobrescribe el default de 6.0s).  
    "duration": 10.0,                # Se mantiene 10 segundos en pantalla.  
}  
]  
  
# 3. INICIALIZACIÓN  
self.current_index = 0              # Puntero al inicio de la lista.  
self.active = True                  # Activa el bucle de renderizado de la intro.  
self.load_current_slide()           # Carga la primera imagen en memoria.  
  
# Cambia el estado global del juego para que el main.py sepa que debe dibujar la Intro.  
self.set_state(GameState.INTRO)
```



11. ESCENA DE ENDING

```
# =====
# CONFIGURACIÓN DE DIAPOSITIVAS (ENDING)
# =====

def start_ending(self):
    # 1. CARGA DE TEXTOS
    # Igual que en la intro, recuperamos los textos traducidos.
    cine_texts = self.get_texts()

    # 2. DEFINICIÓN DE LA PLAYLIST FINAL
    self.playlist = [
        # --- DIAPOSITIVA 1: Resolución ---
        {
            "image": "darkness-room.jpg",      # Reutilizamos fondo oscuro.

            # (YAML es.yaml -> cinematics -> ENDING_1)
            # -> "Y así, tras desenterrar el secreto..."
            "text": cine_texts["ENDING_1"],

            "effect": "none",

            # Música triunfal o de créditos que empieza aquí.
            "music": "sintonia1.ogg"
        },
        # --- DIAPOSITIVA 2: La Calma ---
        {
            "image": "avda_paz.jpg",          # Vuelta a la escena inicial (cierre circular).
        }
    ]
```



```
# (YAML es.yaml -> cinematics -> ENDING_2)
# -> "La ciudad volvió a la normalidad."
"text": cine_texts["ENDING_2"],

# Efecto sutil de alejamiento.
"effect": "zoom_out",
"zoom_intensity": 0.2           # Intensidad baja: Empieza al 120% y reduce al 100%.
},

# --- DIAPOSITIVA 3: Créditos / Agradecimiento ---
{
  "image": "logo_pycapge.png",    # Logo final.

  # (YAML es.yaml -> cinematics -> THANKS)
  # -> "GRACIAS POR JUGAR"
  "text": cine_texts["THANKS"],

  "effect": "zoom_out",          # Zoom medio.
  "zoom_intensity": 0.5,          # Zoom medio.
  "duration": 6.0,               # Tiempo mínimo de visualización.

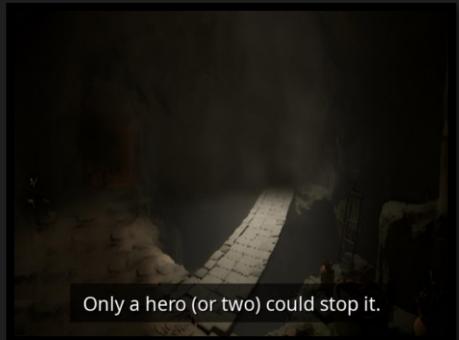
  # --- ESPERA DE INPUT ---
  # Esta propiedad "wait_for_input": True es crítica.
  # Significa que el juego NO volverá al menú principal automáticamente
  # cuando acabe el tiempo. Se quedará congelado en esta imagen
  # hasta que el jugador pulse una tecla o haga clic.
  "wait_for_input": True
}
```



```
# 3. INICIALIZACIÓN
self.current_index = 0
self.active = True

# Cambia el estado global a ENDING. El main.py dejará de procesar lógica de juego
# y pasará a llamar a ending_manager.draw() y ending_manager.update().
self.set_state(GameState.ENDING)
self.load_current_slide()
```

SLIDES SAMPLES FOR INTRO AND ENDING





12. ESCENA 1 - AVENIDA DE LA PAZ

```
# =====
# ESCENA 1: Avenida de la Paz
# =====

# 1. CREACIÓN DEL OBJETO ESCENA
# -----
# Scene(ID_INTERNO, NOMBRE_VISIBLE, FONDO, MASCARA, ESCALADO, RANGO_Y, TRANSICION)
s1 = Scene(
    "AVDA_PAZ",                      # ID único para referenciar esta escena en el código. Grabar partidas . etc
    SCENE_NAMES["AVDA_PAZ"],           # Nombre traducido [YAML: "Avenida de la Paz"].
    "avda_paz.jpg",                  # Imagen de fondo (carpeta backgrounds/).
    "avda_paz_bm.jpg",                # Máscara de caminar (blanco = se pisa, negro = pared).
    scale_range=(0.4, 2.2),            # Escalado del PJ: 0.4 al fondo (lejos), 2.2 al frente (cerca).
    y_range=(230, 400),                # Límites verticales donde se aplica ese escalado (pixels Y).
    transition_type=TRANSITION_FADE # Efecto visual al entrar (Fundido a negro).
)

# 2. EVENTO AL ENTRAR (ON ENTER)
# -----
# Se ejecuta automáticamente cada vez que el jugador entra en esta habitación. Siempre con lambdas.
# Aquí lanzamos un texto flotante de bienvenida.
s1.on_enter = lambda: game_play_event(
    texto=OBJ_DESCS["SCENE1_INTRO_MSG"], # [YAML: "ESTO ES UN TEXTO DEMO PARA EL USUARIO"]
    text_time=8,                      # El texto durará 8 segundos en pantalla.
    pos=(100, 100)                   # Posición (X, Y) donde aparecerá el texto flotante.
)

# 3. HOTSPOT 1: LA CAMPANA (Objeto interactivo fijo)
# -----
```



```
s1.add_hotspot_data(
    name="campana",           # ID interno del objeto.
    image_file="campana.png", # Imagen del objeto (carpeta objects/).
    x=480, y=200,            # Posición en pantalla (coordenada de los pies/base).
    walk_to=(480, 330),      # Dónde se parará el personaje para interactuar con esto. Siempre que esté lejos usarlo.
    label_id="BELL",          # Nombre visible al pasar el ratón [YAML: "Campana"].
    hint_message="BELL_HINT", # Pista si se pulsa la ayuda [YAML: "¡Usa el martillo aquí!"].
    scale=0.1,                # Tamaño de la imagen de la campana (10% del original).
    primary_verb="LOOK AT",   # Verbo por defecto al hacer clic derecho.

    # Diccionario de reacciones a los verbos:
    actions={

        "LOOK AT": "BELL_LOOK", # [YAML: "Es una campana realmente grande."]
        "OPEN":     "BELL_OPEN", # [YAML: "Es de bronce macizo."]
        "PUSH":     "BELL_PUSH", # [YAML: "¡Ding!"]
        "PULL":     "BELL_PULL", # [YAML: "¡Dooooooonng! Menudo Badajo."]

    # --- INTERACCIÓN COMPLEJA: USAR OBJETO CON OBJETO ---
    # La clave se forma así: USE + ID_ITEM_INVENTARIO + _ON_ + ID_HOTSPOT
    "USE_MARTILLO_ON_CAMPANA": lambda: game_play_event(
        texto=OBJ_DESCS["BELL_MAGIC_USE"], # [YAML: "¡El sonido resuena el doble con la magia!"]

        # Lista de sonidos a reproducir en secuencia o mezcla.
        play_sound=["church-bell.ogg", "medal"],

        # Activa una variable global para recordar que esto ya se hizo.
        flag="puzzle_campana_resuelto",

        # Borra el martillo del inventario tras usarlo.
        delete_item="martillo"
    ),
}
```



```
        }

    )

# 4. HOTSPOT 2: EL MARTILLO (Objeto recolectable)
# -----
s1.add_hotspot_data(
    name="martillo",
    image_file="martillo.png",
    x=450, y=315,
    label_id="HAMMER",           # [YAML: "Martillo"]
    scale=0.08,                  # Escalado de la imagen.

    # 'flag_name' es CRUCIAL para objetos que se cogen. Variable global de tu juego.
    # Si 'martillo_recogido' es True en variables.py, este objeto NO se dibujará.
    flag_name="martillo_recogido",

    description="HAMMER_LOOK",    # Descripción genérica.
    primary_verb="LOOK AT",       # Clic derecho = Mirar.

    actions={
        "LOOK AT": "HAMMER_LOOK", # [YAML: "Martillo de carpintero."]

        # Al tener 'flag_name' y verbo "PICK UP", el motor sabe automáticamente:
        # 1. Ponerlo en el inventario.
        # 2. Marcar la flag como True.
        # 3. Hacer desaparecer el objeto de la escena.
        # 4. Mostrar el texto asociado: [YAML: "Vale, me lo llevo."] En el inventario pongo otra imagen o la misma, como prefiera.
        "PICK UP": "HAMMER_PICK"
    }
)
```



```
# 5. SALIDAS (EXITS)
# -----
# Zona invisible que cambia de escena.
# add_exit(x, y, ancho, alto, ID_ESCENA_DESTINO, spawn_x, spawn_y)

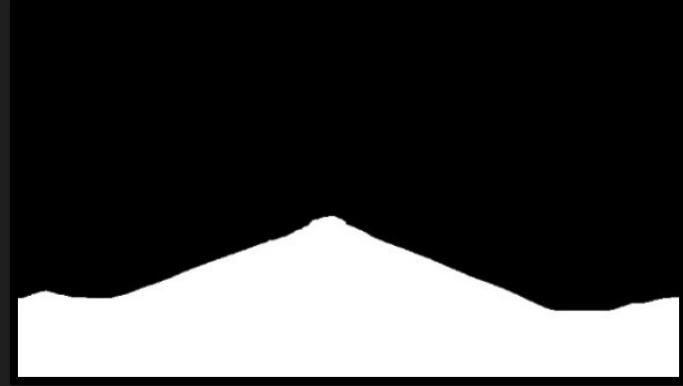
# Salida derecha hacia el Ayuntamiento
s1.add_exit(
    x=750, y=0, w=50, h=GAME_AREA_HEIGHT, # Rectángulo invisible a la derecha de la pantalla.
    target_scene="TOWN_HALL",               # ID de la escena a cargar (definida en s2).
    spawn_x=50, spawn_y=400                # Dónde aparecerá el personaje en la nueva escena.
)

# Salida izquierda hacia la Panorámica
s1.add_exit(
    x=0, y=0, w=50, h=GAME_AREA_HEIGHT, # Rectángulo invisible a la izquierda.
    target_scene="PANORAMIC",            # ID de la escena a cargar (definida en s3).
    spawn_x=1500, spawn_y=400           # Coordenadas de aparición en la Panorámica.
)

# 6. REGISTRO FINAL
# -----
# Añadimos la escena configurada al gestor del juego para que funcione.
scene_manager.add_scene(s1)
```



Principal Image avda_paz.jpg (png, webp, etc)



Walkable zone avda_paz_bm.jpg (png, webp, etc)



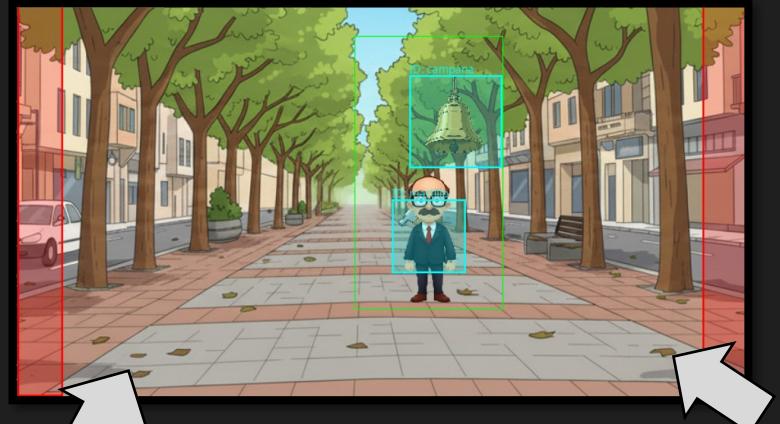
Perspective – Scale Range



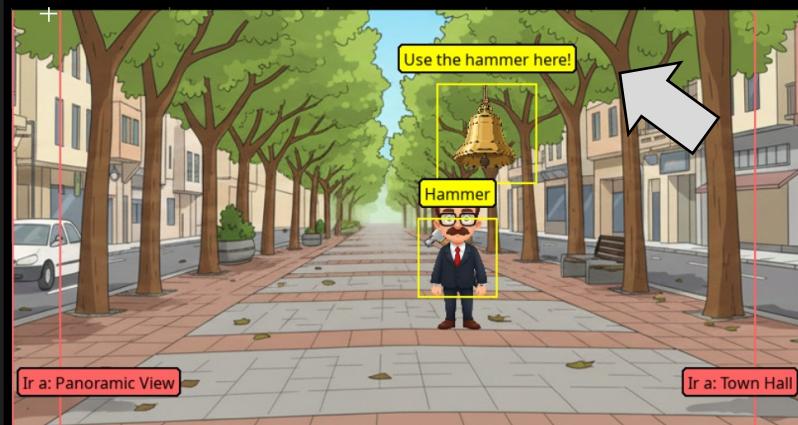
Perspective - Scale Range



Demo text



Doble Click to Run



Game Help and Game Clues



LOOK AT Hammer

Default Verb



13. ESCENA 2 – AYUNTAMIENTO

```
# =====
# ESCENA 2: Ayuntamiento
# =====

# 1. DEFINICIÓN DE LA ESCENA CON AMBIENTE
# -----
s2 = Scene(
    "TOWN_HALL",           # ID Interno.
    SCENE_NAMES["TOWN_HALL"], # Nombre [YAML: "Ayuntamiento"].
    "ayun.jpg",             # Imagen de fondo.
    "ayun_bm.jpg",          # Máscara de caminar.
    scale_range=(0.1, 1.8),  # Escalado muy agresivo (muy pequeño al fondo, grande cerca).
    y_range=(230, 400),      # Rango vertical del escalado.

    # Efecto visual: La escena entra deslizándose desde la izquierda.
    transition_type=TRANSITION_SLIDE_LEFT,

    # Sonido de pasos específico para esta sala (suelo de madera).
    step_sound_key="step_wood",

    # Archivo de iluminación: Una imagen que define qué color/tinte
    # tiene el personaje según donde pise (ej: naranja cerca del fuego, gris para un callejón oscuro etc).
    lightmap_file="ayun_light.jpg"
)

# 2. CINEMÁTICA DE INTRODUCCIÓN (CUTSCENE)
```



```
# -----
# Definimos una función local para la intro automática.
def intro_ayuntamiento():
    # Verificamos una variable (flag) para que esto SOLO ocurra la primera vez.
    if not GAME_STATE.get("intro_ayuntamiento_vista", False):

        # Lista de instrucciones para el "Director de Cine" del motor.
        acciones = [
            {"type": "WAIT", "seconds": 1.0},                      # Espera 1 seg.
            {"type": "MOVE", "x": 400, "y": 450},                  # Mueve al PJ a estas coordenadas.
            {"type": "FACE", "dir": "camera"},                     # Haz que mire a la pantalla.
            {"type": "WAIT", "seconds": 0.5},                       # Pausa dramática.

            # Diálogos automáticos (SAY). El jugador no puede saltárselos (salvo tecla especial).
            # CINE_TEXTS viene de 'cinematics' en el YAML.
            {"type": "SAY", "text": CINE_TEXTS["AYUN_1"], "time": 2.5}, # [YAML: "Estuve aquí hace 25 años..."]
            {"type": "SAY", "text": CINE_TEXTS["AYUN_2"], "time": 2.0}, # [YAML: "...nada ha cambiado."]

            {"type": "WAIT", "seconds": 0.5},

            # IMPORTANTE: Al final, marcamos la flag como True para no repetir esto nunca más.
            {"type": "FUNC", "func": lambda: GAME_STATE.update({"intro_ayuntamiento_vista": True})}
        ]
        # Ejecutamos la secuencia.
        cutscene_manager.start_cutscene(acciones)

    # Asignamos esta función al evento de entrada.
    s2.on_enter = intro_ayuntamiento

# 3. SALIDAS
# -----
```



```
# Izquierda -> Vuelve a la Avenida.  
s2.add_exit(x=0, y=0, w=50, h=GAME_AREA_HEIGHT, target_scene="AVDA_PAZ", spawn_x=750, spawn_y=400)  
# Derecha -> Va a la Habitación Oscura.  
s2.add_exit(x=750, y=0, w=50, h=GAME_AREA_HEIGHT, target_scene="DARK_ROOM", spawn_x=50, spawn_y=400)  
  
# 4. HOTSPOT AVANZADO: PUZZLE DE COMBINACIÓN  
# -----  
# Truco PRO: Construimos la clave del diccionario dinámicamente.  
# El motor busca: USE_[NOMBRE_ITEM]_[ON]_[NOMBRE_HOTSPOT]  
# Usando ITEM_NAMES['STONE'] nos aseguramos de usar el nombre correcto del objeto piedra.  
key_use_stone = f"USE_{ITEM_NAMES['STONE'].upper()}ON_VENTANA_VECINA"  
  
s2.add_hotspot_data(  
    name="ventana_vecina",  
    x=60, y=90, width=100, height=100, # Zona invisible (sin imagen, solo coordenadas).  
    label_id="WINDOW_OLD",           # [YAML: "Ventana Vieja"]  
    walk_to=(110, 380),             # Donde se coloca el PJ para mirar.  
    primary_verb="LOOK AT",  
    actions={  
        "LOOK AT": "WINDOW_LOOK",      # [YAML: "Es la ventana de la Sra. Fletcher."]  
        "OPEN":     "WINDOW_OPEN",       # [YAML: "Está atascada, lleva años pintada."]  
  
        # Reacción al usar la PIEDRA en la VENTANA.  
        key_use_stone: OBJ_DESCS["WINDOW_STONE"] # [YAML: "Mejor no rompo nada hoy."]  
    }  
)  
  
# 5. HOTSPOT ESTÁNDAR: RECOGER OBJETO  
# -----  
s2.add_hotspot_data(  
    name="farol",
```



```
image_file="farol.png",
x=600, y=420, scale=0.1,
label_id="LANTERN_OLD",      # [YAML: "Un farol antiguo"]
flag_name="farol_recogido",   # Si es True, no se dibuja (ya lo cogiste).
primary_verb="PICK UP",
actions={
    "LOOK AT": "LANTERN_LOOK", # [YAML: "Un farol de Aliexpress."]
    "PICK UP": "LANTERN_PICK", # [YAML: "Me lo llevo."]
}
)

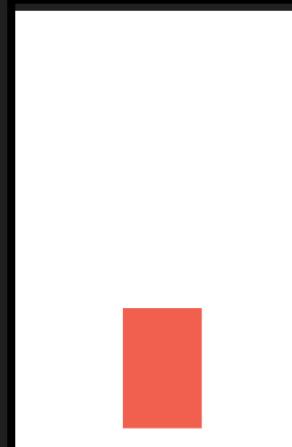
# 6. HOTSPOT ANIMADO: MÁQUINA
# -----
# Este objeto tiene gráficos animados y colisión (es sólido).
s2.add_hotspot_data(
    name="maquina_demo",
    image_file="animation_demo.png",
    num_frames=5, anim_speed=200,    # Configuración del Sprite Sheet.
    x=280, y=420, scale=0.65,
    solid=True,                   # El pathfinding lo esquivará.
    label_id="MACHINE_STRANGE",   # [YAML: "Máquina Extraña"]
    primary_verb="USE",
    actions={
        "LOOK AT": "MACHINE_LOOK",   # [YAML: "Parece una maquinaria compleja."]

        # Al USAR, llama a una función helper que reproduce la animación una vez.
        "USE": lambda: play_object_animation("maquina_demo", OBJ_DESCS["MACHINE_USE"]), # [YAML: "¡Funciona!"]
        "PUSH": lambda: play_object_animation("maquina_demo", OBJ_DESCS["MACHINE_PUSH"]) # [YAML: "Le doy un empujoncito..."]
    }
)
```



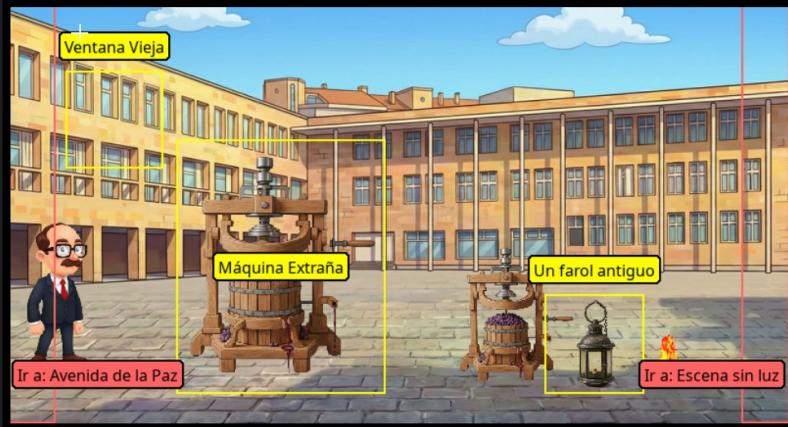
```
# 7. ELEMENTO DE AMBIENTE (DECORACIÓN): HOGUERA
# -----
# 'add_ambient' añade elementos visuales que no necesariamente son interactivos (clicables),
# aunque aquí le hemos puesto acciones para que sirva de ejemplo híbrido.
s2.add_ambient(
    image_file="hoguera.gif",
    x=675, y=420,
    num_frames=50, # Animación larga.
    anim_speed=50, # Velocidad rápida (fuego).
    scale=1.0,
    layer="back", # "back" = Detrás del personaje / "front" = Delante (tapando al PJ).
    solid=True, # No puedes caminar sobre el fuego.
    label_id="BONFIRE", # [YAML: "Hoguera"]
    actions={
        "LOOK AT": "BONFIRE_LOOK", # [YAML: "Me pongo colorado si me acerco mucho..."]
    }
)

# Añadimos la escena al motor
scene_manager.add_scene(s2)
```

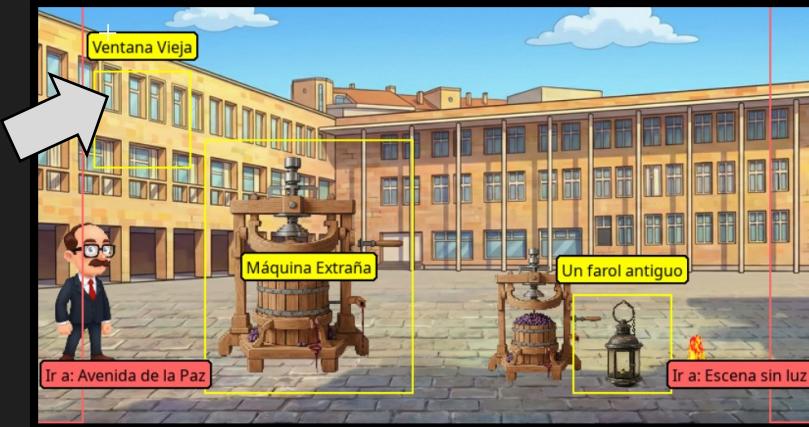


changing the palette with a background

CutScene -Automatic – ESC to exit



Standars Hotspots , and animated objects



Special hotspot



14. ESCENA 4 – HABITACIÓN OSCURA

```
# =====
# # --- ESCENA 4: PANTALLA OSCURA ---
# =====

# 1. DEFINICIÓN DE LA ESCENA
# Aquí instanciamos la clase Scene con sus propiedades básicas.
s4 = Scene(
    "DARK_ROOM",                      # ID INTERNO: Debe ser único. Se usa para viajar aquí (target_scene).
    SCENE_NAMES["DARK_ROOM"],           # NOMBRE VISIBLE: Viene del YAML ("Escena sin luz").
    "darkness-room.jpg",               # IMAGEN DE FONDO: Archivo en la carpeta /backgrounds.
    "darkness-room_bm.jpg",            # MÁSCARA DE CAMINAR: Define por dónde se puede pisar (blanco/negro).

    scale_range=(1.75, 2.2),            # ESCALA DEL PERSONAJE: (Mínima, Máxima).
                                         # El personaje se verá 1.75x al fondo y 2.2x al frente.
    y_range=(325, 400),                # RANGO Y: Píxeles verticales donde se aplica el escalado.
                                         # (Desde Y=325 hasta Y=400).

    # --- CONFIGURACIÓN DE LUZ (Especial de esta escena) ---
    is_dark=True,                     # ACTIVA OSCURIDAD: La pantalla se cubrirá de negro.
    light_flag="farol_recogido",       # CONDICIÓN DE LUZ: Nombre de la variable en GAME_STATE.
                                         # Si GAME_STATE["farol_recogido"] es True, habrá luz.
    light_radius=70,                  # RADIO DE LUZ: Tamaño del círculo transparente alrededor del ratón.

    transition_type=TRANSITION_FADE # TRANSICIÓN: Efecto visual al entrar (Fundido a negro).
)

# 2. DEFINICIÓN DE EVENTO AL ENTRAR (on_enter)
```



```
# Esta función se ejecutará AUTOMÁTICAMENTE cada vez que el jugador entre a esta sala.
def enter_dark_room():
    # Verificamos si NO tenemos el farol recogido
    if not GAME_STATE.get("farol_recogido", False):
        # Si no hay luz, lanzamos un mensaje de queja del personaje.
        # YAML: DARK_ROOM_ENTER: "¡No veo un pimiento! Necesito luz."
        game_play_event(texto=OBJ_DESCS["DARK_ROOM_ENTER"], text_time=3.0)

# Asignamos la función creada al evento on_enter de la escena
s4.on_enter = enter_dark_room

# 3. CONFIGURACIÓN DEL SISTEMA DE MAPA (Viaje Rápido)
# Definimos una lista de destinos. Cada línea es un punto en el mapa.
# Formato: (ID_ESCENA, X_MAPA, Y_MAPA, SPAWN_X, SPAWN_Y, ICONO)
map_destinations = [
    # El propio cuarto oscuro (se marca en el mapa en x=250, y=350). Al viajar apareces en 400, 390.
    ("DARK_ROOM", 250, 350, 400, 390, "pin.png"),
    # La calle principal.
    ("AVDA_PAZ", 400, 500, 400, 390, "pin.png"),
    # El ayuntamiento.
    ("TOWN_HALL", 200, 300, 435, 420, "pin.png"),
    # La panorámica.
    ("PANORAMIC", 500, 360, 975, 400, "pin.png"),
    # La escena Parallax (Demo técnica).
    ("PARALLAX", 375, 250, 590, 370, "pin.png")
]

# 4. OBJETO INTERACTIVO: EL MAPA (Tirado en el suelo)
s4.add_hotspot_data(
    name="mapa1",                      # ID único del objeto en esta escena.
    image_file="mapa1.png",              # Imagen del objeto (en carpeta /items u /objects).
```



```
x=250, y=400, scale=0.1,          # Posición y tamaño.

label_id="MAP",                  # ETIQUETA TRADUCIBLE: Busca "MAP" en el YAML.
                                # YAML items -> MAP: "Mapa Viejo"

flag_name="tengo_mapa",          # FLAG DE EXISTENCIA:
                                # Si "tengo_mapa" es True, este objeto NO SE DIBUJA en la escena
                                # (porque ya lo recogimos). Si es False, aparece en el suelo.

primary_verb="PICK UP",          # VERBO POR DEFECTO: Al hacer clic derecho o hover.
solid=True,                      # SÓLIDO: El jugador tendrá que rodearlo al caminar, no lo atraviesa.

actions={
    # Acción MIRAR
    # YAML descriptions -> MAP_LOOK: "Un mapa turístico."
    "LOOK AT": "MAP_LOOK",

    # Acción COGER
    # YAML descriptions -> MAP_PICK: "Me lo llevo."
    # Al cogerlo, automáticamente pone "tengo_mapa"=True y añade el item al inventario.
    "PICK UP": "MAP_PICK",

    # Acción USAR (Lógica condicional avanzada con lambda)
    # Aquí preguntamos: ¿Tengo el mapa en mi poder?
    "USE": lambda: load_and_open_map(map_destinations, "mapa1.jpg")
        if GAME_STATE.get("tengo_mapa") # SI LO TENGO: Abre la interfaz de mapa.
        else game_play_event(texto=OBJ_DESCS["MAP_FAIL"]), # SI NO LO TENGO (está en el suelo):
                                                # YAML: "No puedo usarlo ahí tirado..."

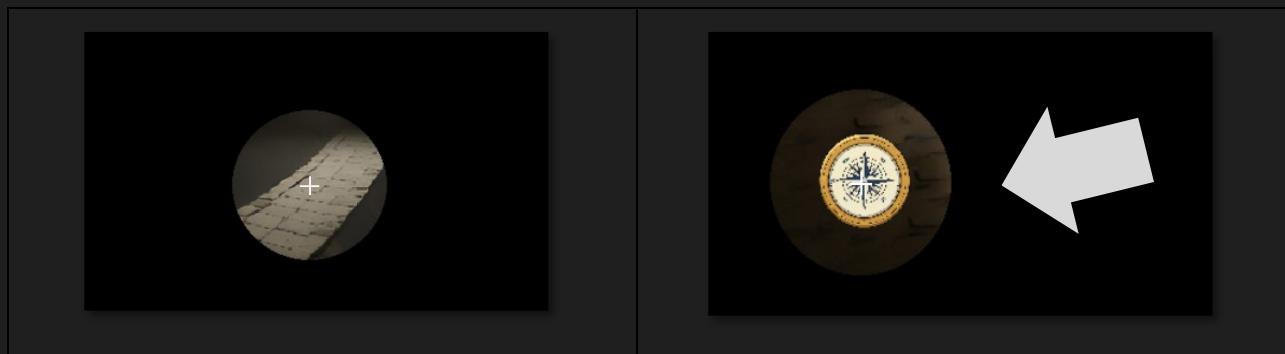
    # Acción ABRIR (Misma lógica que usar)
    "OPEN": lambda: load_and_open_map(map_destinations, "mapa1.jpg")
```



```
    if GAME_STATE.get("tengo_mapa")
        else game_play_event(texto=OBJ_DESCS["MAP_FAIL"])
    }
)

# 5. SALIDAS (EXITS)
# Definimos una zona invisible que nos lleva a otra escena.
s4.add_exit(
    x=0, y=0, w=50, h=GAME_AREA_HEIGHT, # Rectángulo de activación (Borde izquierdo de la pantalla).
    target_scene="TOWN_HALL",           # Destino: ID de la Escena 2 (Ayuntamiento).
    spawn_x=750, spawn_y=400           # Dónde aparece el personaje en la nueva escena.
)

# 6. REGISTRO FINAL
# Añadimos la escena configurada al gestor del juego.
scene_manager.add_scene(s4)
```





15. ESCENA 3 – CALLE PANORÁMICA

```
# =====
# ESCENA 3: Panorámica (Scroll y Mecánicas Avanzadas)
# =====

# 1. DEFINICIÓN BÁSICA DE LA ESCENA
# -----
# Scene(ID_INTERNO, NOMBRE_VISIBLE, IMAGEN_FONDO, MASCARA_SUELO, ...)
s3 = Scene(
    "PANORAMIC",                                # ID único usado por el código para referirse a esta escena.
    SCENE_NAMES["PANORAMIC"],                     # Nombre visible (se ve al guardar partida). [YAML: "Panorámica"]
    "panoramica.jpg",                            # Imagen de fondo (debe ser ancha para permitir scroll).
    "panoramica_bm.jpg",                         # Máscara de caminabilidad (blanco = se pisa, negro = pared).

    # scale_range: (Escala arriba del todo, Escala abajo del todo).
    # Aquí (1.9, 2.1) significa que el personaje se ve muy grande (zoom),
    # ideal para primeros planos. 1.0 es el tamaño original.
    scale_range=(1.9, 2.1),

    # y_range: Límites verticales (píxeles) por donde puede caminar el personaje.
    y_range=(325, 400),

    # Sonido de pasos específico para esta escena (ej. alfombra o tierra).
    step_sound_key="step_rug",

    # Tipo de transición al entrar (deslizar hacia arriba).
    transition_type=TRANSITION_SLIDE_UP
)

# 2. EVENTO "AL ENTRAR" (ON ENTER)
```



```
# -----
# Esta función lambda se ejecuta automáticamente nada más cargar la escena.
s3.on_enter = lambda:
    # Reproduce música de fondo, con fundido de 5 segundos y volumen al 50%.
    play_scene_music("sintonia2.ogg", duration_s=5, volume=0.5),

    # Muestra un texto flotante introductorio.
    # [YAML: "Esto es un ejemplo de pantalla con SCROLL horizontal"]
    game_play_event(texto=OBJ_DESCS["SCENE3_INTRO_MSG"], pos=(400, 150), text_time=4.0)
)

# 3. OBJETO INTERACTIVO CON ESTADOS: CUBO DE BASURA
# -----
# Este objeto tiene dos estados visuales (cerrado/abierto) controlados por frames.
s3.add_hotspot_data(
    name="basura",                      # ID interno del objeto.
    image_file="trashcan.png",           # Sprite sheet (debe contener los frames lado a lado).
    num_frames=2,                       # El sprite tiene 2 imágenes (Frame 0: Cerrado, Frame 1: Abierto).
    anim_speed=0,                       # Velocidad 0 para que no se anime solo (lo controlamos nosotros).
    x=800, y=355,                      # Posición inicial en el mundo.
    label_id="TRASHCAN",                # Nombre al pasar el ratón. [YAML: "Cubo de Basura"]
    primary_verb="LOOK AT",             # Verbo por defecto (clic derecho).
    scale=0.15,                         # Tamaño del objeto respecto a su imagen original.
    solid=True,                         # El personaje no puede atravesarlo (choca).

    # walk_to: Dónde se para el personaje para interactuar (X, Y).
    walk_to=(875, 365),

    # facing: Hacia dónde mira el personaje al interactuar ("left" = mira a la izquierda).
    facing="left",
```



```
actions={  
    # Acción MIRAR: Muestra descripción.  
    # [YAML: "Está lleno de desperdicios."]  
    "LOOK AT": "TRASH_LOOK",  
  
    # Acción ABRIR: Ejecuta una función compleja.  
    "OPEN": lambda: (  
        # Cambia el gráfico del objeto al frame 1 (tapa abierta).  
        change_state_object("basura", 1),  
        # Muestra texto y el personaje hace animación de "abrir".  
        # [YAML: "¡Puaj! Qué peste."]  
        game_play_event(texto=OBJ_DESCS["TRASH_OPEN"], anim="open")  
    ),  
  
    # Acción CERRAR: Vuelve al frame 0 (tapa cerrada).  
    "CLOSE": lambda: (  
        change_state_object("basura", 0),  
        # [YAML: "Mejor cerrado."]  
        game_play_event(texto=OBJ_DESCS["TRASH_CLOSE"], anim="close")  
    ),  
}  
)  
  
# 4. SISTEMA DE DIÁLOGO (NPC GARBA)  
# -----  
# Definimos el árbol de conversación en una función para cargar los textos al momento.  
def get_garba_dialogue_tree():  
    return {  
        "start": { # Nodo inicial  
            "options": [  
                {
```



```
# Opción condicional: Solo aparece si tienes la linterna con pilas.
# [YAML: "¡Mira! Ya he conseguido ponerle pilas a la linterna."]
"text": DIALOGUE_TEXTS.get("GARBA_START_OPT1", "GARBA_START_OPT1"),

# Respuesta del NPC.
# [YAML: "¡Genial! Sabía que eras un manitas."]
"response": DIALOGUE_TEXTS.get("GARBA_START_ANS1", "GARBA_START_ANS1"),

"condition": "linterna_con_pilas_recogida", # Flag requerida en GAME_STATE.
"next": "tengo_la_linterna_con_pilas" # Salta a otro nodo.

},
{
    # Opción estándar.
    # [YAML: "Hola Garba, ¿qué haces aquí?"]
    "text": DIALOGUE_TEXTS.get("GARBA_START_OPT2", "GARBA_START_OPT2"),
    "response": DIALOGUE_TEXTS.get("GARBA_START_ANS2", "GARBA_START_ANS2"),
    "next": "quiero_una_linterna_con_pilas"
},
{
    # Opción de salida ("EXIT" cierra el diálogo).
    "text": DIALOGUE_TEXTS.get("GARBA_START_OPT4", "GARBA_START_OPT4"),
    "response": DIALOGUE_TEXTS.get("GARBA_START_ANS4", "GARBA_START_ANS4"),
    "next": "EXIT"
}
],
},
# ... (Definición de otros nodos como "quiero_una_linterna..." etc)
"quiero_una_linterna_con_pilas": {
    # ... opciones del nodo ...
},
"tengo_la_linterna_con_pilas": {
```



```
# ... opciones del nodo ...
}

}

# Definición del NPC en la escena
s3.add_hotspot_data(
    name="garba",
    image_file="garba_tr.gif",      # GIF animado (idle).
    num_frames=6, anim_speed=300,   # Configuración de la animación.
    x=1250, y=360,
    scale=0.65,
    text_color=(255, 255, 0),     # Color del texto cuando habla este NPC (Amarillo).
    label_id="NPC_GARBA",          # [YAML: "Garba"]
    primary_verb="TALK TO",
    actions={
        # Al HABLAR: Inicia el sistema de diálogo con el árbol definido arriba.
        "TALK TO": lambda: dialogue_system.start_dialogue(
            get_garba_dialogue_tree(),
            "start",
            npc_ref=s3.hotspots.get_hotspot_by_name("garba") # Referencia para que mueva la boca.
        ),
        "LOOK AT": "NPC_GARBA_LOOK", # [YAML: "¿Qué miras? ¿Tengo monos en la cara?"]

        # Acción Especial: DAR OBJETO (Give)
        # La clave se forma automáticamente: GIVE + ID_OBJETO + ON + ID_HOTSPOT
        "GIVE_LINTERNA_CON_PILAS_ON_GARBA":
            lambda: game_play_event(
                texto=OBJ_DESCS["GARBA_GIVE_FLASHLIGHT"], # [YAML: "Toma Garba, quédate tú."]
                delete_item="linterna_con_pilas",           # Elimina el objeto del inventario.
                play_sound="medal"                         # Sonido de éxito.
            )
    }
}
```



```
    }

)

# 5. CRAFTING: LINTERNA Y PILAS
# -----
# Objeto A: Linterna sin pilas
s3.add_hotspot_data(
    name="linterna_sin_pilas",
    image_file="linterna_sin_pilas.png",
    x=950, y=370, scale=0.085,
    label_id="FLASHLIGHT_EMPTY",           # [YAML: "Una linterna sin pilas"]
    flag_name="linterna_sin_pilas_recogida", # Variable global para saber si ya la cogimos.
    primary_verb="PICK UP",
    actions={
        "PICK UP": "FLASHLIGHT_EMPTY_PICK", # [YAML: "Me la llevo."]
    }

    # Combinación: USAR PILAS en LINTERNA (Inverso)
    "USE_PILAS_PARA_LINTERNA_ON_LINTERNA_SIN_PILAS": lambda: crafting(
        "linterna_sin_pilas",      # Ingrediente 1 (ID).
        "pilas_para_linterna",    # Ingrediente 2 (ID).
        "linterna_con_pilas",     # ID del Nuevo Objeto resultante.
        "linterna_con_pilas.png",  # Imagen del nuevo objeto.
        "linterna_con_pilas_recogida" # Flag a activar.
    )
}

)

# Objeto B: Pilas (Lógica simétrica para permitir A sobre B y B sobre A)
s3.add_hotspot_data(
    name="pilas_para_linterna",
    image_file="pilas_linterna.png",
```



```
x=1050, y=370, scale=0.08,
label_id="BATTERIES",
flag_name="pilas_linterna_recogidas",
primary_verb="PICK UP",
actions={
    "PICK UP": "BATTERIES_PICK",

    # Combinación: USAR LINTERNA en PILAS
    "USE_LINTERNA SIN PILAS ON PILAS PARA LINTERNA": lambda: crafting(
        "linterna_sin_pilas",
        "pilas_para_linterna",
        "linterna_con_pilas",
        "linterna_con_pilas.png",
        "linterna_con_pilas_recogida"
    )
}

# Objeto Resultado: Linterna Completa (Se define fuera de pantalla)
# Es necesario definirlo para que el inventario sepa qué descripción mostrar. Lo dibujo en un sitio muy lejano para que no moleste.
s3.add_hotspot_data(
    name="linterna_con_pilas",
    image_file="linterna_con_pilas.png",
    x=-1000, y=-1000,           # Coordenadas negativas para que no se vea en la escena. Pero tiene que existir definido.
    label_id="FLASHLIGHT_FULL", # [YAML: "Una Linterna Encendida"]
    primary_verb="USE",
    actions={
        "LOOK AT": "FLASHLIGHT_FULL_LOOK", # [YAML: "Ahora sí, brilla con fuerza."]
        "USE": "FLASHLIGHT_FULL_USE",
    }
)
```



```
# 6. LÓGICA ESPECÍFICA POR PERSONAJE: LA PALA
#
# -----
s3.add_hotspot_data(
    name="pala",
    image_file="objects_pala.png",
    x=570, y=360, scale=0.1,
    label_id="SHOVEL",           # [YAML: "Pala Pesada"]
    flag_name="pala_recogida",
    primary_verb="PICK UP",
    actions={
        # Lógica condicional al COGER:
        "PICK UP": lambda: (
            # CASO 1: Si somos Bart (El fuerte)
            (
                inventory.add_item("pala", ITEM_NAMES["SHOVEL"], "objects_pala.png"),
                GAME_STATE.update({"pala_recogida": True}),
                # Eliminamos el objeto visualmente de la escena:
                [h.kill() for h in scene_manager.get_current_scene().hotspots.hotspots if h.name == "pala"],
                # Mensaje de éxito [YAML: "¡Humpf! Pesa un quintal..."]
                game_play_event(texto=OBJ_DESCS["SHOVEL_BART_PICK"], play_sound="medal")
            )
            if PLAYER_CONFIG["CHAR_ID"] == "Bart"

            # CASO 2: Si somos Gilo (El débil)
            # Mensaje de fallo [YAML: "Pesa demasiado. Necesito a Bart."]
            else game_play_event(texto=OBJ_DESCS["SHOVEL_GILO_FAIL"], speaker=player)
        ),
        # Descripción diferente según quién mire
        "LOOK AT": lambda: game_play_event(texto=OBJ_DESCS["SHOVEL_LOOK_BART"])
    }
)
```



```
        if PLAYER_CONFIG["CHAR_ID"] == "Bart"
        else game_play_event(texto=OBJ_DESCS["SHOVEL_LOOK_GILO"])
    }
)

# 7. AMBIENTACIÓN: OBJETO EN MOVIMIENTO (PÁJARO)
# -----
s3.add_ambient(
    image_file="crow8.gif",
    num_frames=8, anim_speed=120, # 8 frames de sprite sheet, paso cada frame a 120 milis.
    x=-50,                  # le doy una X negativa para que empiece fuera de la pantalla.
    scale=0.3,
    layer="back",           # Se dibuja detrás del personaje.
    solid=False,             # No choca.

    # Movimiento automático:
    move_to=(1600, 150),   # Destino final (fuera de pantalla a la derecha).
    move_speed=100,         # Velocidad de vuelo.
    loop_move=True,         # Al llegar al final, vuelve al inicio (bucle).

    label_id="BIRD",        # [YAML: "Pájarraco"]
    actions={
        "LOOK AT": "BIRD_LOOK" # [YAML: "Oh , mira, pajarillos !"]
    }
)

# 8. CAMBIO DE PERSONAJE (NPCs JUGABLES)
# -----
# NPC BART (Se usa para cambiar el control a él)
s3.add_hotspot_data(
    name="Bart",
```



```
image_file="bart_d.gif",
x=180, y=360, scale=0.6,
label_id="NPC_BART",           # [YAML: "Bart"]
flag_name="controlando_bart",# Si es True, este NPC se oculta (porque lo estás controlando).
primary_verb="TALK TO",
solid=True,
actions={
    "LOOK AT": "NPC_BART_LOOK",
    "TALK TO": "NPC_BART_TALK",
    # Al USAR: Cambia el control del jugador a "Bart".
    "USE": lambda: change_player_active("Bart"),
    "PUSH": "NPC_BART_PUSH"
}
)

# NPC GILO (Lo mismo para volver a Gilo)
s3.add_hotspot_data(
    name="Gilo",
    image_file="gilo_d.gif",
    x=180, y=360, scale=0.6,
    label_id="NPC_GILO",          # [YAML: "Gilo"]
    flag_name="controlando_gilo",
    primary_verb="TALK TO",
    solid=True,
    actions={
        "LOOK AT": "NPC_GILO_LOOK",
        "TALK TO": "NPC_GILO_TALK",
        "USE": lambda: change_player_active("Gilo"),
        "PUSH": "NPC_GILO_PUSH"
}
)
```



```
# 9. SALIDAS Y FINALIZACIÓN
# -----
# Evento al salir: Parar la música.
s3.on_exit = lambda: stop_scene_music()

# Definir zona de salida (Rectángulo invisible a la derecha del todo).
s3.add_exit(
    x=1550, y=0, w=50, h=GAME_AREA_HEIGHT, # Rectángulo de detección.
    target_scene="AVDA_PAZ",                 # ID de la escena destino.
    spawn_x=50, spawn_y=400                  # Dónde aparece el jugador en la nueva escena.
)

# ¡Importante! Añadir la escena al gestor global.
scene_manager.add_scene(s3)
```



16. ESCENA 5 – DEMO PARALLAX

```
# =====
# # --- ESCENA 5: EJEMPLO PARALLAX
# =====

# 1. INSTANCIACIÓN DE LA ESCENA
# Se crea el objeto Scene. Esta escena es especial porque utiliza el sistema de capas (Parallax). Como ejemplo uso webp.
s5 = Scene(
    "PARALLAX",                                # ID Interno de la escena (usado para cambios de escena y lógica).
    SCENE_NAMES["PARALLAX"],                      # Nombre para mostrar en UI/Debug.
    # (YAML es.yaml) -> "Parallax Demo"

    "parallax_middle.webp",                      # Imagen de fondo base (Fallback). Si el sistema parallax falla, se carga esta.
    "parallax_middle_bm.webp",                    # Máscara de navegabilidad (Bitmask). Define por dónde puede caminar el PJ (blanco=sí,
negro=no).

# --- CONFIGURACIÓN DE PARALLAX (Capas visuales) ---
parallax_paths=[
    "parallax_far.webp",                         # Capa 0: Fondo (Cielo/Montañas lejanas). Se pinta primero.
    "parallax_middle.webp",                       # Capa 1: Plano medio (Suelo/Edificios). Donde camina el PJ.
    "parallax_near.webp"                          # Capa 2: Primer plano (Arbustos/Niebla). Tapa al personaje.
],
# Factores de velocidad para el efecto de profundidad al mover la cámara.
parallax_factors=[
    0.0,      # Factor 0.0: La Capa 0 no se mueve con la cámara (fija o scroll automático).
    1.0,      # Factor 1.0: La Capa 1 se mueve sincronizada con la cámara (base).
    2.0      # Factor 2.0: La Capa 2 se mueve al doble de velocidad (efecto cercanía).
],
```



```
# Configuración de scroll automático independiente del jugador.  
# (Índice de capa, Velocidad X).  
auto_scroll_config=(0, -15.0),    # La Capa 0 (Cielo) se mueve sola a la izquierda a velocidad 15. Con 0 = automatico. Puedo  
poner otro  
  
# Escalado del personaje (Zoom).  
scale_range=(1.5, 2),            # Escala mínima (al fondo) 1.5x, Escala máxima (al frente) 2.0x.  
y_range=(350, 500)               # Rango Y en píxeles donde se aplica ese escalado (de y=350 a y=500).  
)  
  
# 2. LÓGICA "ON_ENTER" (Al entrar en la escena)  
# Se define una función lambda que ejecuta múltiples acciones al cargar la pantalla.  
s5.on_enter = lambda: (  
    # A) Reproducir música de ambiente.  
    # Archivo: 'fores_bird_pymapge.ogg', sin duración límite (0), volumen 70%.  
    play_scene_music("fores_bird_pymapge.ogg", duration_s=0, volume=0.7),  
  
    # B) Mostrar texto narrativo introductorio.  
    # Se busca la clave en el diccionario de descripciones cargado del YAML.  
    # (YAML es.yaml -> descriptions) -> "ESTA ES LA PANTALLA FINAL"  
    game_play_event(texto=OBJ_DESCS["SCENE5_INTRO_MSG"], text_time=8.0)  
)  
  
# 3. PREPARACIÓN DE CLAVES DINÁMICAS (Para Puzzles)  
# Esto es una buena práctica: Construir el nombre de la acción usando el nombre real del objeto.  
# Si en el YAML traduce "SHOVEL" a "Pala" o "Shovel", el código sigue funcionando.  
# El motor espera el formato: USE + NOMBRE_ITEM + _ON_ + NOMBRE_HOTSPOT  
  
# (YAML es.yaml -> items -> SHOVEL) -> "Pala Pesada"  
# Resultado esperado string: "USE_PALA_PESADA_ON_MARCA_FINAL" (El motor gestiona los espacios internamente)  
key_dig_shovel = f"USE_{ITEM_NAMES['SHOVEL'].upper()}ON_MARCA_FINAL"
```



```
# 4. DEFINICIÓN DE HOTSPOTS (Objetos interactivos)
# Definimos la "X" en el suelo que marca el final del juego.
s5.add_hotspot_data(
    name="marca_final",                      # ID único del hotspot en esta escena.
    image_file="x_the_end.png",               # Sprite gráfico de la X.
    x=620, y=400,                            # Coordenadas (x, y) donde se dibuja.
    scale=0.4,                                # Escalado del sprite al 40%.
    # Etiqueta visible al pasar el ratón.
    # (YAML es.yaml -> items -> MARK_X) -> "Marca en el suelo"
    label_id="MARK_X",
    primary_verb="LOOK AT",                   # Verbo por defecto si se hace doble clic o clic derecho.

    # Diccionario de Acciones (Verbo -> Reacción)
    actions={
        # Acción MIRAR:
        # (YAML es.yaml -> descriptions) -> "Una X gigante pintada en el suelo. Como en los mapas piratas."
        "LOOK AT": "MARK_LOOK",
        # Acción USAR (Genérica, sin objeto):
        # (YAML es.yaml -> descriptions) -> "Necesito algo para cavar."
        "USE":      "MARK_USE_FAIL",
        # Acción EMPUJAR:
        # (YAML es.yaml -> descriptions) -> "La tierra está muy dura."
        "PUSH":     "MARK_PUSH_FAIL",
        # --- RESOLUCIÓN DEL PUZZLE (Usar Pala en Marca) ---
        # Se define la clave específica para combinar objetos.
```



```
# Nota: Aquí se está usando hardcoded "USE_PALA_ON_MARCA_FINAL" en lugar de la variable `key_dig_shovel` creada arriba.
# Esto asume que el ID interno del ítem en inventario es "pala".
"USE_PALA_ON_MARCA_FINAL": lambda: (
    # Paso 1: Mostrar texto de éxito inicial.
    # (YAML es.yaml -> descriptions) -> "¡Aquí hay algo! Voy a cavar..."
    game_play_event(texto=OBJ_DESCS["MARK_DIG_START"], text_time=2.0),

    # Paso 2: Iniciar una Cutscene (Cinemática controlada)
    cutscene_manager.start_cutscene([
        # Evento 1: El personaje dice una frase.
        # (YAML es.yaml -> descriptions) -> "¡Crac! He dado con algo sólido."
        {"type": "SAY", "text": OBJ_DESCS["MARK_HIT_OBJ"], "time": 4.0},

        # Evento 2: Ejecutar función de final de juego.
        # Llama al gestor de finales para cambiar al estado ENDING.
        {"type": "FUNC", "func": ending_manager.start_ending}
    ])
)
}

# 5. LÓGICA "ON_EXIT" (Al salir de la escena)
# Importante para detener sonidos que no deben sonar en la siguiente pantalla (créditos o menú).
s5.on_exit = lambda: stop_scene_music()

# 6. REGISTRO FINAL
# Se añade la escena configurada al gestor global para que el motor pueda cargarla.
scene_manager.add_scene(s5)
```

17. UN POCO DE HISTORIA

“En el verano del 84, mientras Orwell se hacía realidad en las teles (y Van Halen lo petaba en la radio), yo descubrí la magia negra en casa ajena. Un chico mayor, medio amigo, medio guía espiritual, nos llevó a la casa de una familia bien. De esas que tenían piscina y paredes sin rozones. En el salón, apareció el hechizo: un ZX Spectrum conectado a la tele. Un teclado de goma con el poder de invocar marcianitos en pantalla. Ellos jugaban, yo miraba. Quizá me dejaron tocar una tecla, disparar un par de píxeles o mover el Pong, pero la mayoría del tiempo estuve hipnotizado, como si viera fuego por primera vez.” *Texto de mi libro ISBN disponible XD.

En 1985 tuve mi primer ordenador. Un AMSTRAD CPC664 de fósforo verde. Con los primeros discos que venían con el equipo ya había algunos juegos muy básicos escritos en Basic: teletenis, plaga, pulga y animal vegetal mineral. Lo bueno de estos juegos es que tenías acceso al código de forma inmediata. Nunca se me dio muy bien jugar; me gustaba más investigar cómo funcionaba aquella cosa. Ese primer año fue vivir mirando a verde.

Las primeras revistas con los Pokes y un libro para programar en Basic que me compró mi padre hicieron el invierno de un pueblo en el monte más llevadero. Al poco tiempo cayó en mis manos los primeros juegos sin gráficos. Eran las aventuras conversacionales. Además eran españolas. No me llamaron mucho la atención, pero sí que destripé su funcionamiento. Tenía 12 años; quería un poco de acción. Poco después llegó él, el gran juego español: La Abadía del Crimen. Terminar la Abadía, es un logro que tengo puesto en mi currículo. No era una aventura gráfica como las que se popularizaron posteriormente, pero sí que marcó un antes y un después en el género. Un par de años después en el instituto había un profesor de inglés, un poco avanzado a su época, que pedía la incipiente clase de informática y los viernes nos ponía la aventura “King ‘s Quest” ya en PCs con sus discos de 5¼. Eso ya era otra cosa. Pantallas con colores, y personajes que interactuaban. Era la Abadía del Crimen versión yanqui. Y en muy pocos años llegaron las grandes aventuras. La Trilogía del Dólar. Monkey Island, Fate of Atlantis y Day of the Tentacle. No se puede elegir entre una de ellas. Fueron los primeros juegos que realmente me engancharon desde el principio hasta el final. Ya en los primeros años de carrera, recuerdo el Colegio Mayor que cuando alguien descubría alguna cosa corría a otros pisos a dar el chivatazo. Sí, teníamos que haber estado estudiando, pero... Ese mismo año conocí también la salida de “Igor: Objetivo Uikokahonia”, la primera aventura gráfica desarrollada en España (1994). Aquello fue el gran desafío. Si Péndulo había desarrollado esa maravilla, yo quería hacer una igual. En la carrera ya estudiamos ASM, C++, etc. Así que me puse con ello, pero no tenía ni medios, ni herramientas ni el conocimiento necesario. Con algo de ayuda conseguí hacer algo



parecido a una aventura conversacional como el “Animal Vegetal Mineral” que fue el primer juego que había cargado en mi CPC casi 10 años antes y poco más. Había obtenido mejores resultados en Basic de chaval. Un par de años más tarde, instalé para C++ una librería que prometía maravillas: Allegro. Estuve unos meses cacharreando pero no tenía tiempo suficiente. Había que estudiar. Y ahí se acabó mi segundo intento de hacer una aventura gráfica: “Más duro que el rock”. Tenía nombre y todo, y por algún disco duro está el código que ya hacia cosas. Fue en cuarto de carrera en el año 2000 cuando me puse en serio. Ya con internet, se podía preguntar en foros, y había mucha información avanzada. Así que a ratos, y durante un año, me planté en 25mil líneas de código con mi primera aventura gráfica Point and Click terminada. Corría sobre la máquina virtual de java lo que la hacía multiplataforma, y también online. Creo que un logro para esa época. Tanto que 20 años después la prensa local se hizo eco y hasta [me hicieron una entrevista](#). En su época la intenté regalar con la prensa, pero no tenían ni idea de que hablaba. Recuerdo que a nivel local me dieron un premio en un concurso. Con lo que gané, me compré mi primera Fender Stratocaster. Desde esos años he seguido programando por hobby y algo a nivel profesional. Aunque ahora ya no tanto.

Aquel motor dejó de funcionar con las actualizaciones de los S.O. y tenía la espinita de hacer otro sistema que fuera más compatible. No soy mucho de Python, pero ahora mismo es el que me daba más opciones, y no nos vamos a engañar, los Bugs con la IA se corrigen en minutos. Así que durante los últimos meses, he desarrollado este motor que clona todas las características de aquellos Point and Click clásicos de principios de los 90. He intentado replicar los comportamientos y espero haber creado un motor funcional, para hacer las más propias y otros usuarios hagan las suyas. Gracias majos.



ENGINE & CODE:

- garba eduardogarbayo.com
- zainder.com Programmers
- riojawebs.com Graphics

SPECIAL THANKS:

- Python Community
- Pygame Developers
- LucasArts (For inspiration)
- Chir (Indy Java MAGE)

DONATIONS:

- If you want to support the development of PyCAPGE, you can donate at:
<https://www.zainder.com/donations/>