Manual de Programador

Proyecto Imagen Digital – Rubén Vázquez Angamarca

Funcionalidades del proyecto

Este proyecto ofrece 2 partes distintas que se pueden extraer y utilizarse en otros programas por separado:

- Generación de un mapa a partir de una imagen.
- Algoritmo de raycasting en 2D y 3D.

Instalación

- Instalar Python
- Instalar las librerías necesarias. Para ello, se abre una consola (*cmd* en windows) y se introducen los siguientes comandos en **negrita**.
    - Opencv: **pip3 install opencv-python**
    - Pygame: **pip3 install pygame**
    - Numpy: **pip3 install numpy**
- Abrir el programa y editar la ruta indicada en la línea 10 con la ruta de la imagen del mapa que se desea importar:
  image = cv2.imread("**./imagenes/lineas5.jpg**", cv2.IMREAD_GRAYSCALE)

Generación de mapa

Localización

Esta parte del proyecto se extiende desde la línea 6 a la línea 56 del código.

Entrada

Una imagen cuya ruta se especifica en la línea 10.

Salida

Una matriz del tamaño especificado por el usuario que representa un mapa siguiendo este código:

- casilla == 0: espacio vacío en el que el jugador puede caminar.
- casilla != 0: pared que el jugador no puede atravesar.

Funcionamiento

- Líneas 6 a 10: lectura de la imagen
- Líneas 11 a 15: transformación de la imagen para facilitar la detección de líneas
- Líneas 16 a 51: generación de la matriz a partir del mapa.
    - Línea 33: uso de **cv2.HoughLinesP()** para detectar las líneas de la imagen.
- Líneas 52 a 56: generación de una barrera de 1s alrededor del mapa para controlar posibles errores de generación.

Detección de líneas

Para la detección de líneas se ha utilizado el método **cv2.HoughLinesP(image, rho, theta, threshold, lines, minLineLength, maxLineGap)** con unos parámetros que se consideran adecuados para las imágenes con las que se ha probado el algoritmo. Aún así, los parámetros se pueden modificar para cambiar la detección de líneas. A continuación se indica la descripción de cada uno de ellos:

- **image**: Imagen de entrada.
- **rho**: Resolución en la distancia radial en píxeles. Los valores bajos permiten detectar líneas que están muy juntas o en ángulos pequeños, pero pueden aumentar el ruido en la detección.
- **theta**: Resolución en el ángulo en radianes. Los valores altos hacen que solo se detecten líneas a ángulos específicos, ignorando detalles finos en ángulos pequeños.
- **threshold**: Umbral de votos en el acumulador. Un valor más bajo aumenta el número de líneas detectadas, incluyendo las menos definidas.
- **lines**: Estructura de datos en la que se desea que se devuelvan las líneas.
- **minLineLength**: Longitud mínima del segmento de línea en píxeles. Valores altos evitan que se detecten líneas demasiado cortas y ruidosas, y solo se consideran los segmentos de línea más largos.
- **maxLineGap**: Distancia máxima entre segmentos de línea conectados para que se consideren una sola línea. Un valor alto permite combinar segmentos de línea con espacios más grandes entre ellos, útil para líneas continuas que se interrumpen ligeramente.

Raycasting en 2D y 3D

Localización

Esta parte del proyecto se extiende desde la línea 57 a la línea 227 del código.

Entrada

Una matriz que representa un mapa siguiendo este código:

- casilla == 0: espacio vacío en el que el jugador puede caminar.
- casilla != 0: pared que el jugador no puede atravesar.

Funcionamiento

- Líneas 57 a 98: declaración de constantes y variables.
- Líneas 99 a 129: inicializa el juego y sitúa al jugador en un hueco libre del mapa, devolviendo sus coordenadas. Si no encuentra un hueco libre, devuelve las coordenadas (-1, -1).
- Líneas 130 a 138: pinta el mapa en 2D.
- Líneas 139 a 165: lanzamiento de rayos. Se lanzan rayos desde el jugador hacia donde esté mirando y cuando el rayo choca con una pared, se obtiene la distancia del jugador a la pared. Esta información la utiliza de manera distinta según si está en modo 2D o 3D.
    - En modo 2D, colorea las paredes golpeadas por el rayo.
    - En modo 3D, pinta las paredes en la pantalla según su distancia al jugador.
- Líneas 166 a 227: gestión del bucle principal del juego y de los *inputs* del jugador.