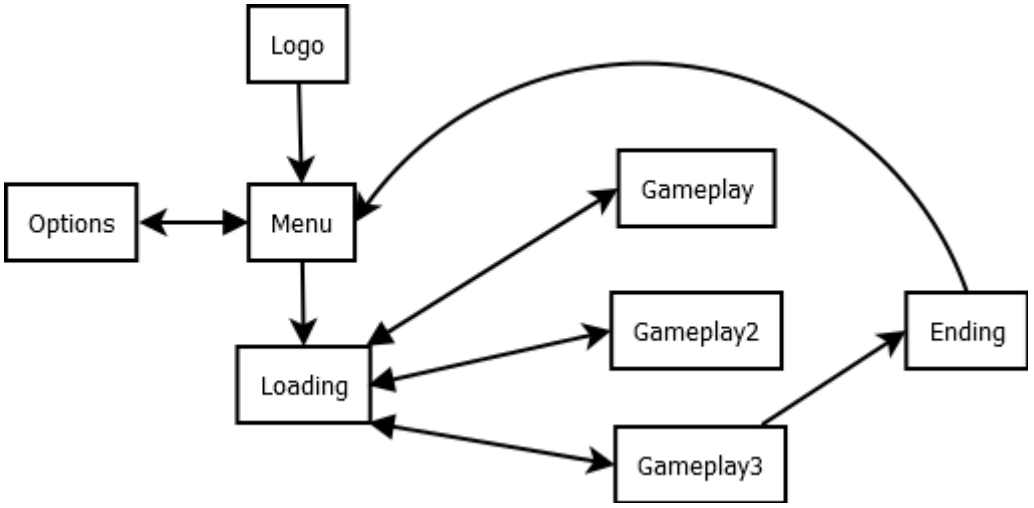
Flujo del juego



Transiciones

Logo => Menu

Por defecto la primera pantalla en cargar es el logo. Al cabo de tres segundos se producirá la transición al Menu de forma automática. Ya no es posible volver a la pantalla de Logo en esta ejecución.

Menu => Options

Esta transición se produce al pulsar la tecla ‘o’. Es posible volver a entrar en esta pantalla durante varias veces durante la ejecución.

Options => Menu

Esta transición se produce al pulsar la tecla ‘o’. Es posible volver a entrar en esta pantalla durante varias veces durante la ejecución.

Menu => Loading

Esta transición se produce al pulsar la tecla “Enter”. Es posible volver a repetirla pero para eso el jugador debe haber completado el juego.

Loading => Gameplay

Al inicio se preguntará al GameManager que nivel tiene que cargar. En el caso de que este sea el primero pasaremos a la pantalla de Gameplay.

Gameplay => Loading

Esta transición se produce cuando el jugador completa el primer nivel. En este caso encontrar la salida antes de que se agote el tiempo..

Loading => Gameplay2

Al inicio se preguntará al GameManager que nivel tiene que cargar. En el caso de que este sea el segundo pasaremos a la pantalla de Gameplay2.

Gameplay2 => Loading

Esta transición se produce cuando el jugador completa el primer nivel. En este caso encontrar la salida antes de que se agote el tiempo.

Loading => Gameplay3

Al inicio se preguntará al GameManager que nivel tiene que cargar. En el caso de que este sea el tercero pasaremos a la pantalla de Gameplay3.

Gameplay3 => Ending

Esta transición se produce cuando el jugador completa el primer nivel. En este caso encontrar la salida antes de que se agote el tiempo..

Gameplay3 => Gameplay3

En el caso de que el tiempo pase y el jugador no cumpla las condiciones de victoria el nivel se repetirá.

Ending => Menu

La transición se producirá cuando el usuario pulse la tecla “Enter”. No se puede volver a esta pantalla a no ser que el jugador complete los tres niveles otra vez.

Modificación del GenMeshCubicMap

Para esta tarea he creado un MeshManager para crear, obtener y eliminar el mesh bajo demanda.
Para la generación de los cubos he sustituído las comprobaciones que tenía para que sea compatible con los requisitos de la práctica. El problema de este método viene a la hora de cargar los modelos, puesto que ahora se pueden indicar modelos en las posiciones.
Para esto he creado un struct llamado object en el que guardo la posición donde se tiene que poner el modelo, el tipo de modelo que es y la textura que se tiene que aplicar.
Ahora bien de entrada no es posible ver cuantos elementos tenemos por lo que en la primera vuelta a los píxeles de la imagen cuento el número de modelos que hay que cargar.
Una vez se el número de modelos puedo reservar en memoria el tamaño que necesito para alojarlos con un malloc. Esto implica que tengo que crear un método que al llamarlo me libere toda esta memoria ocupada para no tener problemas con el heap.

Con esto tendría la generación del mesh y un puntero a una estructura de datos para los modelos.

Pantalla de carga

En la pantalla de carga llamo al levelManager. El levelManager se encarga de cargar el mapa del nivel y pasarselo al GenMeshCubicMap para obtener el mesh del mapa. En mi caso en cada llamada a esta función genero el mesh y después de finalizar el nivel llamo a la función de borrado.

Carga de modelos

Para cargar el modelo tengo varias partes implicadas. Por un lado necesito el propio modelo con su mesh. Para la gestión de modelos he creado el modelManager para cargar, devolver y eliminar modelos.
Seguidamente necesito una textura que aplicar al modelo. Para esto utilizaré el TextureManager que ya tengo creado.
Finalmente necesito especificar la posición donde se dibujará el modelo. Esto lo obtendré en el GenMeshCubicMap.
Estos tres requisitos los uno en un enum a la hora de generar el mesh. De modo que después de la pantalla de carga el meshManager tiene el mesh y el puntero al array de objetos con los modelos que tengo que poner en el nivel.
Solo tengo que pedir el puntero, recorrerlo y pedir al TextureManager y al ModelManager la textura y el modelo indicado.

Exploración de escenarios

He implementado la opción de volver a un mapa anterior. Por ejemplo ahora es posible ir al primer nivel desde el segundo.
Para hacer esto tengo un array de Vector3 con dos elementos para indicar la posición inicial y la posición final. Entonces a la hora de indicar la posición inicial de la cámara miro el valor del nivel anterior si el valor es más grande que el actual quiere decir que vengo de un nivel superior y tengo que empezar en la salida del nivel.

Cosas por hacer

Como se puede ver en el diagrama de flujo se puede ver que el screen_gameplay esta triplicado. Esto se debe a que estaba implementando diferentes mecánicas en cada nivel. Por ejemplo, el primer nivel solo sería encontrar la salida del laberinto. En el segundo para poder salir necesitas buscar tres objetos y llegar a la salida y en el último, a parte de buscar los objetos y la salida se debe hacer antes de que se acabe el tiempo.
Tal y como esta el juego ahora si que se podría hacer todo el gameplay en una sola clase porque es lo mismo para todas.