Memoria del videojuego 2D

Grupos 1 y 2

Bielsa Uche, Jaime (819033)
Daniel Soriano, Saúl (815743)
Lorente Guarnieri, Juan (816020)
Rimacuna Castillo, Eryka Liced (816778)
Mateo Trejo, Hugo (816678)
Lahoz Bernad, Fernando (800989)
Perbech Gállego, Ismael (815120)

9 de mayo de 2024

Ingeniería Informática

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

1. Introducción

El videojuego que se describe a continuación se trata de un clon de Lemmings, en concreto de la versión para Commodore Amiga estrenada en 1991, adaptado para el sistema operativo de Microsoft.

1.1. Detalles del juego

- Título: Lemmings
- Estudio: DMA Design (actualmente Rockstar)
- Género: Puzles / Estrategia
- Clasificación: ESRB Kids to Adults
- Plataforma original: Commodore Amiga
- Plataforma adaptada: Windows (x86-64)

1.2. Visión general del juego

El objetivo del juego es guiar a un grupo de Lemmings antropomorfizados a través de una serie de obstáculos hasta una salida designada. Se pide salvar un porcentaje mínimo de Lemmings para superar cada nivel. Para encaminarlos, se ha de gestionar la asignación de ocho habilidades únicas a Lemmings concretos que permitan alterar el entorno, afectar al comportamiento de otros Lemmings o despejar obstáculos con fin de crear un paso seguro para los demás hacia la salida.

1.3. Reglas

En cada nivel, se asigna al jugador un cierto número de Lemmings que puede utilizar para completar el nivel. Es muy importante administrarlos de manera que el jugador no se quede sin ellos, ya que perdería automáticamente. El número y tipo de habilidades disponibles pueden variar en cada nivel.

Hay un límite de tiempo para completar el nivel, el nivel finaliza cuando el temporizador llega a cero, o cuando no quedan Lemmings vivos en el nivel.

A lo largo de los niveles, los jugadores encontrarán diversos obstáculos y peligros que representan amenazas para los Lemmings, como agua, lava y trampas. Es importante utilizar sus habilidades para evitar estos peligros y garantizar su supervivencia.

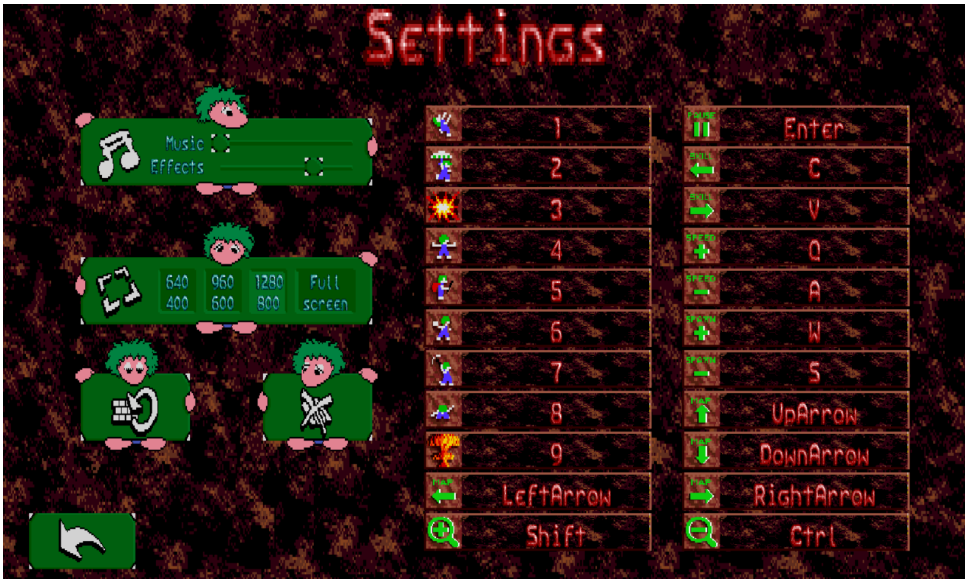
El éxito del nivel se determina según el número de Lemmings que llegan a la meta. Habrá un porcentaje predeterminado de éxito para el nivel. Si el número de Lemmings que llegan a la meta es inferior al porcentaje de éxito requerido, el jugador pierde el nivel. Si es igual o superior, el jugador habrá superado el nivel.

A medida que el jugador completa con éxito un nivel, automáticamente se accede al siguiente nivel. Esta progresión permite al jugador enfrentarse a desafíos más difíciles y complejos a medida que avanza en el juego.

2. Opciones de juego

El juego proporciona al usuario la opción de elegir jugar el nivel como un puzzle de estratégia o ver cómo son capaces de resolverlos por sí mismos.

- **Modo clásico:** el jugador debe ordenar a los Lemmings realizar movimientos para atravesar los obstáculos que se encuentran a lo largo del mapa (desde la puerta de salida hasta la puerta del cielo), de está forma podrán llegar a la puerta del cielo, dando por concluido el nivel.
- **Modo autoplay:** en este modo, los Lemmings se moverán automáticamente a través del nivel sin la intervención directa del jugador. El jugador puede observar cómo los Lemmings interactúan con los obstáculos para resolver el nivel por sí mismos.
- **Configuración (persistencia):**



El juego permite al usuario cambiar una serie de parámetros, con tal de adecuar el juego a sus preferencias individuales.

- **Ajuste del volumen:** el jugador tiene la capacidad de ajustar tanto el volumen de la música de fondo como de los efectos de sonido del juego.
- **Tamaño de pantalla:** el tamaño de la pantalla es variable, pudiendo elegir entre tres tamaños de ventana y pantalla completa.
- **Asignación de teclas:** es posible cambiar las teclas asignadas según la acción acorde a sus preferencias, personalizando la experiencia de juego del usuario.

Al guardar la configuración, como resulta común, se guarda para veces futuras en las que se abra el juego.

3. Niveles

Se han implementado, al igual que el juego original, un total de cuatro niveles de dificultad, para garantizar una experiencia de juego variada y desafiante.

- **Fun:** este nivel sirve como introducción al juego, presentando conceptos básicos y mecánicas fundamentales de manera gradual.
- **Tricky:** en este nivel, la complejidad de los escenarios aumenta significativamente, ofreciendo desafíos más difíciles y requerimientos estratégicos más avanzados.
- **Taxing:** este nivel está dirigido a jugadores con experiencia en el juego que buscan desafíos aún más difíciles y exigentes.
- **Mayhem:** considerado el nivel más difícil y desafiante del juego, Mayhem ofrece una experiencia extremadamente exigente y gratificante para los jugadores más expertos.

Cada uno de los treinta escenarios en cada uno de los niveles cuenta con una serie de características clave que definen la experiencia de juego, que son:

- **Mapa:** el diseño del mapa determina la disposición de los obstáculos y la ruta que los Lemmings deben seguir para llegar a la puerta de salida.
- **Tiempo:** se establece un límite de tiempo para cada escenario, desafiando a los jugadores a encontrar soluciones rápidas y eficientes.
- **Habilidades Disponibles:** se proporciona a los jugadores un conjunto limitado de habilidades que pueden utilizar para guiar a los Lemmings a través de los obstáculos, como cavar, construir, bloquear, entre otras.
- **Puertas de Entrada y Salida:** cada escenario cuenta con una puerta de entrada desde la cual emergen los Lemmings y una puerta de salida que representa el objetivo final del nivel.

Se ha implementado la colisión de los elementos no fijos (como los Lemmings) con el terreno en base a un sistema de capas alfa. El escenario es una imagen donde cada píxel tiene, aparte de sus valores RGB, un valor **α** (alfa) que corresponde a la transparencia del píxel. Se considerará que se puede colisionar con un píxel del terreno si su valor **α** supera cierto umbral. A partir de ahí, la detección de colisiones se realiza en base a un trazado de rayos.

Para algunos escenarios se necesitaba separar en distintas capas los fondos, ya que se necesitaba que se superpusieran una parte con el agua mientras que otra parte tendría que estar ocluida, también se requería de separar en otra capa las zonas con metales para llevar la lógica aparte y tener en cuenta que no se podía cavar este terreno.

Además algunas entidades requería de tener una hitbox que no se asemejaba con la capa alpha como los líquidos o los portales de salida, ya que el lemming no colisiona justo cuando toca con el pie el agua o cuando toca el borde del portal, para elle se implementaron triggers asociados a estas entidades permitiendo personalizar las hitboxes sin depender del canal alpha de la textura o para reducir la cantidad de objetos coleccionables, ya que en un mapa podría haber muchas entidades de agua en paralelo reduciéndolo a tener un solo trigger que ocupe todo el ancho del agua.

En el juego original la selección de niveles funcionaba con *passcodes* que debías ir guardando conforme superabas cada nivel. En este clon, se ha decidido dar la posibilidad de jugar a cualquier nivel desde el comienzo, permitiendo elegir la dificultad y el número del nivel desde un menú gráfico como el que se muestra a continuación:

4. Interfaz de partida

En la interfaz de juego se han mantenido los botones existentes en el juego original (habilidades de Lemmings, explosión grupal) y se han añadido funcionalidades que facilitan la interacción con la partida, como son: un botón de pausa, un multiplicador de velocidad y un editor de la velocidad de aparición de Lemmings. Así obtenemos los siguientes elementos en la interfaz de la partida:

1. Multiplicador de velocidad de juego.
2. Frecuencia de aparición de Lemmings.
3. Habilidades de los Lemmings.
4. Pausar partida.
5. Explosión grupal de Lemmings.
6. Minimapa de la partida.

Pausar el juego evita que el tiempo avance y permite examinar con calma la situación. Una mayor frecuencia de aparición provocará que los Lemmings aparezcan más rápido. Al comenzar un nivel se establece una velocidad inicial, que varía según el nivel.

Explotar a todos los Lemmings provoca que el nivel termine, pero si se logra salvar al porcentaje mínimo de Lemmings antes de explotarlos, el nivel se dará por superado. Si no ha aparecido ningún Lemming por el portal antes de forzar la explosión, el nivel terminará inmediatamente.

El minimapa muestra una versión simplificada del mapa de juego y de la posición de los Lemmings. Para desplazarse por el mapa es posible hacerlo moviendo el cursor hacia los lados de la pantalla de juego, arrastrando el foco del minimapa o pulsando en cualquier zona de este.

Además se puede hacer zoom sobre el mapa con las teclas asociadas.

Independientemente de la interfaz proporcionada, es posible también realizar acciones con el teclado. Se pueden cambiar las teclas en la configuración, pero por defecto son: 1-8 para las habilidades, 9 para explotar todos los Lemmings, Q/A para alterar la velocidad del juego o W/S de la aparición de Lemmings, ↑/↓ permite cambiar entre habilidades, ←/→ navegar por el mapa y Enter pausar el juego.

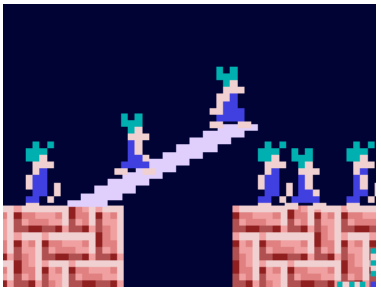
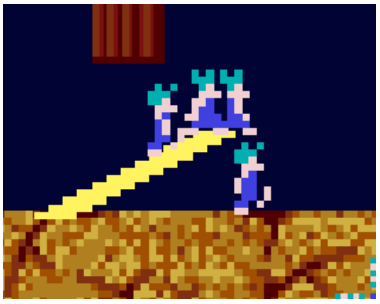
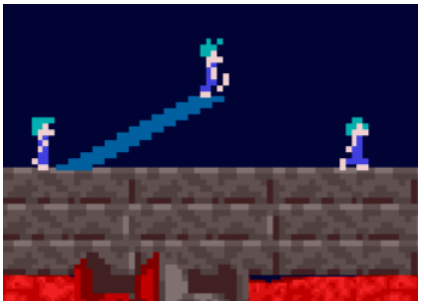
5. Personajes

5.1. Controlables: Lemmings

Los personajes que se controlan son los Lemmings, aunque no de forma directa. Desde que aparecen, se mueven lateralmente de forma automática. Si encuentran una pared, cambian de dirección. Se puedes asignar habilidades a los Lemmings para realizar un movimiento que altere el entorno o que les permita sortearlo:

- **Cavar hacia abajo:** cava hasta que encuentre un objeto al que no pueda cavar, por ejemplo el metal o el aire.
- **Cavar recto:** cava horizontalmente en la dirección que lleve antes de dar la orden. Para que la orden sea efectiva hay que seleccionar al Lemming con la acción cuando está justo delante del muro u objeto a cavar, de lo contrario se inicia la animación de cavar pero termina de forma inmediata y continua con la acción o movimiento previos. Hay materiales como el metal o los muros direccionales en los que no se puede cavar.
- **Cavar en diagonal:** cava diagonalmente con el objeto que se encuentre, como el suelo o un muro. Hay materiales como el metal o los muros direccionales en los que no se puede cavar.
- **Explosión:** el Lemming indica con una cuenta regresiva de 5 segundos sobre su cabeza cuando va a explotar, continuando con el movimiento que estuviera llevando a cabo. Cuando el tiempo se agote se parará en la posición en la que se encuentre y comenzará una animación donde el Lemming estallará destruyendo el área circundante. La explosión no afecta a los Lemmings aliados.
- **Construcción de escaleras:** construye escaleras pegando hasta 12 ladrillos seguidos uno encima del otro, si no tiene espacio para colocar el siguiente ladrillo a poner se acaba el movimiento antes. Los Lemmings no pueden chocar con la parte inferior de la escalera.

El color que adquiere la escalera a construir depende de la superficie donde se construye, como puede apreciarse en las imágenes de abajo.

Escalera Gris	Escalera Marrón	Escalera Azul
		

- **Escalada:** escala las paredes contra las que se choca. Si cuando escala se choca contra el techo cae al suelo, da la vuelta y sigue andando. Esta habilidad es permanente, es decir, escalará todos los muros que se encuentre.
- **Barrera:** el Lemming se para e impide el paso a otros Lemmings. Sobre este movimiento puede usarse el movimiento de explotar, permitiendo quitar Lemmings que bloqueen. El Lemming barrera sólo bloqueará a aquellos que choquen con sus manos, es decir, si un Lemming cae encima suyo se verá bloqueado y continuará avanzando en la misma dirección.
- **Paracaídas:** despliega un paracaídas durante la caída, permitiendo evadir la muerte por caída hasta el suelo. Este movimiento es una habilidad permanente, por lo que le permite al Lemming desplegar el paracaídas siempre que lo necesite.

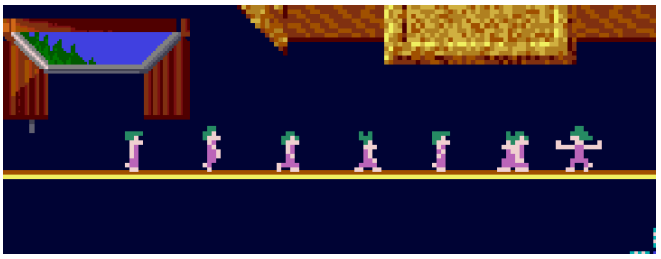
Los Lemmings pueden morir por las siguientes causas:

- Uso de la habilidad de explotar o de explosión grupal
- Caída al suelo desde demasiada altura.
- Ahogamiento con materiales líquidos.
- Mutilación, corte, quemadura, electrocución o aplastamiento por trampas en el mapa.
- Caída al vacio.

5.2. No controlables: Lemmings IA

Disponen de las mismas habilidades que los Lemmings, con las mismas restricciones en cada nivel disponible que los originales.

El jugador no puede interactuar con los Lemmings en el modo IA, a excepción de activar la explosión grupal, donde todos los Lemmings en el mapa explotarán.



6. Sonido

Tendremos por un lado música ambiente y por otro lado efectos de sonido que enriquecerán la experiencia de usuario.

6.1. Música

- **Música del prólogo:** suena durante la cinemática introductoria para presentar a los Lemmings a modo de breve introducción al juego. Comienza con el frío sonido del viento dando paso a una melodía desenfadada que invita al usuario a jugar y relajarse jugando.
- **Música de niveles:** en cada nivel sonará una pieza de música que tendrá asociada, de entre un total de 21. Algunas son totalmente originales, otras hacen referencia a composiciones de música clásica y otras incluso a piezas que hacen referencia a otros títulos de la misma compañía. Acompañan el diseño del nivel y generan en el jugador, tanto sensaciones de tranquilidad y desenfado, como generar tensión o una ligera sensación de extrañeza ayudando a la inmersión y la experiencia de juego.

6.2. Efectos de sonido

- **Let's go:** lo dicen los Lemmings en el inicio de cada nivel.
- **Oh no:** suena cuando un Lemming va a explotar.
- **Explode:** suena cuando un Lemming explota.
- **Splash:** suena cuando un Lemming cae sobre un fluido.

- **Glug:** suena cuando un Lemming se ahoga en un fluido, emitiendo un sonido gutural.
- **Splat:** suena cuando un Lemming se estrella contra el suelo. Suena a un Lemming al que le duele la caída y seguidamente hace un sonido gutural.
- **Fire:** suena cuando un Lemming cae en una trampa de fuego, es un audio de un objeto que se quema muy rápido.
- **Electric:** suena cuando un Lemming cae en una trampa eléctrica, es un audio que suena como un mini trueno.
- **Chain:** suena cuando un Lemming cae en una trampa de cadena, es una cuerda de cáñamo tensa que se encoge y luego produce un golpe seco.
- **Tenton:** suena cuando un Lemming cae en una trampa de pistón o de 10 toneladas, o de roca, es un sonido mecánico de una prensa.
- **Thunk:** suena cuando un Lemming cae en un cepo, es un sonido de un choque metálico, no muy exagerado.
- **Thud:** suena cuando un Lemming cae en una trampa de pinchos, suena a golpe seco.
- **Die:** suena cuando un Lemming cae al vacío, es un sonido de un Lemming gritando por su vida.
- **Door:** suena al inicio del nivel cuando la puerta que deja caer a los Lemming se abre, como una puerta de madera con bisagras mal engrasadas.
- **Yipee:** suena cuando un Lemming consigue escapar del nivel a través de la puerta de salida, suena como si un Lemming diese un bote “boing”.
- **Change Op:** suena cuando el jugador selecciona una habilidad en el HUD, suena a una cuerda de guitarra que reverbera.
- **Mouse Press:** suena cuando el jugador hace clic sobre un Lemming y le aplica una habilidad, sonando como una cuchara al golpear una cacerola.

7. Tecnologías

El videojuego ha sido desarrollado en el lenguaje de programación C++, utilizando el estándar 20, compilado para Windows con la versión de gcc ofrecida por el entorno MSYS2. Se ha hecho uso de la herramienta CMake para la configuración y compilación del proyecto, que nos ha permitido separar los fuentes en diferentes módulos, aprovechando también su compatibilidad con el IDE Visual Studio Code para agilizar el desarrollo.

El proyecto utiliza SDL2 como única biblioteca, que permite abstraer la interacción con los drivers de dispositivos multimedia para múltiples plataformas. Hemos utilizado esta biblioteca para gestionar:

- Ventanas y visualización del juego
- Música y sonidos
- Interacción del jugador mediante teclado y ratón

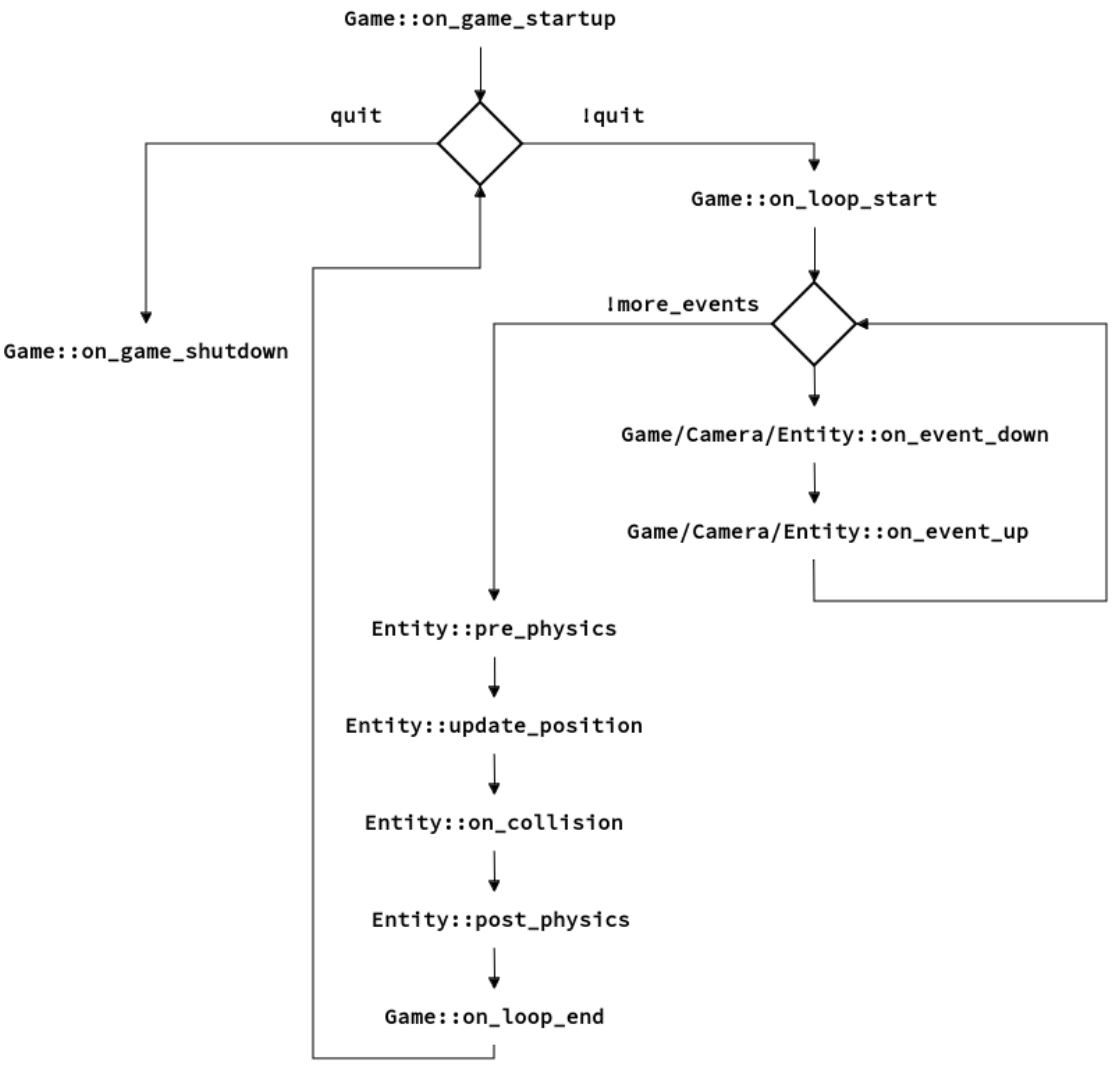
La mayor parte de los sprites del juego son los originales de la versión para el Atari Amiga, aunque ha sido necesario modificar ligeramente la interfaz. En esos casos se ha empleado el programa de edición de imágenes del proyecto gnu: GIMP.

8. Diseño interno

8.1. Motor general

El motor es el encargado de gestionar los recursos del sistema. Permite que el videojuego desarrolle su lógica mediante un conjunto de entidades. Para nuestro motor, una entidad es cualquier objeto que necesite recibir eventos, mostrarse por pantalla o interactuar con otras entidades. Cada entidad tiene como información activa una caja alineada con los ejes, que representa su posición y dimensiones, y una textura que la rellena. Adicionalmente, puede contar con otro tipo de variables como lo son su nombre, su velocidad, la gravedad por la que se ve afectada o las formas de colisionar con las demás entidades.

El motor consta de un bucle con el que gestiona la transmisión de eventos a las entidades, siguiendo el siguiente esquema:



En cada iteración del bucle, se comprueba los eventos de entrada y salida que ha generado el usuario, se calcula las físicas, se procesan las entidades creadas y destruidas y se visualiza por pantalla. Las entidades reciben los eventos que se producen en cada fase como invocaciones a métodos virtuales. Esto tiene como ventaja que permite un diseño simple y encapsulado de las entidades del videojuego, pero tiene el inconveniente de transmitir todos los eventos a todas las entidades, lo vayan a necesitar o no. Para reducir el número de llamadas virtuales innecesarias se facilita un sistema de suscripción a los eventos de entrada y salida, de tal forma que solo se propagará este tipo de eventos a aquellas entidades suscritas. Solo se ha hecho así para IO dado que es el único caso que influye sustancialmente en el rendimiento.

El motor también se encarga de ofrecer un reloj al videojuego con el cálculo del tiempo transcurrido entre cada frame: el *delta time*. Para calcularlo se utiliza un reloj monotónico que se media a lo largo de las últimas iteraciones y se resetea cuando crece 4 veces por encima de la media, con el fin de amortiguar irregularidades y evitar que crezca impredeciblemente en caso de bloqueo. En nuestro caso, este sistema de amortiguamiento se planteó como solución a un problema de la biblioteca SDL en Windows a la hora de arrastrar o redimensionar una ventana [1], el cual provocaba que el delta time creciera drásticamente y las magnitudes del sistema dependientes del tiempo (ej: la velocidad) presentasen comportamientos impredecibles.

8.2. Motor de colisiones y físicas

El motor distingue entre dos tipos de entidades: HUD, que solo interactúan con los eventos producidos por el usuario; estructuras, entidades que pueden colisionar pero no pueden moverse; y entidades móviles. Las entidades móviles colisionan con las estructuras y viceversa, pero entre un mismo grupo no colisionan (para optimizar el rendimiento).

Las entidades pueden ser configuradas para que las colisiones en las que participa tengan en cuenta la caja que la encierra o su textura. La colisión en función de la textura se realiza evaluando el canal alfa de la imagen que la conforma, de modo que solo las zonas no transparentes colisionan. Esto es especialmente relevante para el videojuego de los Lemmings.

Cada entidad móvil debe definir 4 cajas internas, generalmente una en cada borde de su caja de colisión y ajustadas a la forma interna de la entidad. Lo primero que comprobará el motor de físicas es si las cajas de colisión de dos entidades intersecan. En caso afirmativo, si la entidad requiere colisionar en base a su textura se procede a evaluar los píxeles de las texturas de ambas entidades, y si no lo requiere se evalúa con cuál de las 4 cajas internas interseca el objeto para conocer la dirección de la colisión.

Además de con otras entidades, también se puede colisionar con el ratón para recibir el evento de que se está siendo apuntada por el cursor.

Este sistema liga el aspecto visual (renderizado) con la lógica interna del motor y del juego, pero es necesario para permitir la destrucción granular del mapa tal y como hacía el juego original.

El motor también permite a las entidades lanzar rayos en cualquier dirección (vertical, horizontal o diagonal), que se colisionan con el resto de entidades apropiadamente con sus capas alfa utilizando ray marching de granularidad variable para averiguar la distancia de una entidad con el suelo, aunque en muchos casos es preferible usar estrategias más simples y ad hoc dado el coste del ray marching.

Los rayos horizontales son más rápidos dado que SDL almacena las Surface (texturas en CPU) por filas y por tanto se preserva la localidad de memoria.

8.3. Motor de render

Cada cámara está definida por dos cajas alineadas al eje: una que delimita los objetos que la cámara ve en el mundo, y otra que representa en qué parte de la pantalla se van a mostrar. El motor permite definir múltiples cámaras que renderizan imágenes independientes, que se pueden incrustar dentro de la cámara principal. Además pueden definirse propiedades que modifican el aspecto de todas las entidades de una misma clase, exclusivamente en esa cámara. Por ejemplo, que el minimapa de los lemmings se actualice en tiempo real se consigue renderizando la misma región del mundo desde otra cámara, a la que se le modifica la operación de blending (mediante la interfaz que proporciona SDL) para redimensionar y volver a colorear la textura del mapa.

Para renderizar qué es lo que ve una cámara se mantienen ordenadas las entidades en función de su eje z (profundidad) y se van pintando como sprites, siguiendo el algoritmo del pintor. Las texturas se gestionan internamente en una caché para evitar cargar varias imágenes desde fichero en un mismo frame y provocar tirones. El juego tiene control sobre la caché, permitiendo precargar varias texturas antes de comenzar a jugar un nivel, y vaciar la caché cuando ya no sean necesarias las texturas cargadas, por ejemplo, cuando se vuelve al menú.

Las texturas de las entidades pueden ser modificadas en tiempo de ejecución por el propio juego. Cuando una textura es modificada se activa una copia única de la misma, y la textura original se deja intacta para seguir siendo referenciada por el resto de entidades.

8.4. Inteligencia artificial

El sistema de IA es capaz de jugar a los lemmings. Dada la enorme complejidad de ciertos niveles el sistema solo es capaz de resolver algunos. Su comportamiento se puede ver en jugando en el modo “Auto play”.

El enfoque más realista para crear una Inteligencia Artificial capaz de completar todos los niveles es casi seguro que se encuentra en el aprendizaje por refuerzo, dado que muchos niveles más difíciles requieren mecánicas y decisiones notablemente difíciles de idear o deducir incluso para un humano.

Varios artículos científicos exploran la complejidad computacional de los lemmings, llegando a la conclusión de que, incluso dado un nivel genérico con un solo lemming y una sola habilidad, llegar a una solución es una prueba de NP-completitud. Para más de un lemming el problema se vuelve PSPACE-completo [2] [3] [4].

En su lugar se ha afrontado el problema con métodos clásicos, y se ha dividido el sistema en tres partes:

- La primera es el generador de casillas, que transforma un mapa cualquiera de los lemmings en un conjunto de casillas clasificadas según los píxeles de la textura del mapa y las posiciones de estructuras adicionales que forman parte de él.
- La segunda parte es el algoritmo de decisión, que recorre esas casillas en busca de un camino para completar el nivel, aplicando para ello las distintas habilidades.
- La tercera parte es el ejecutor, que teniendo la política generada por el paso anterior debe aplicarla durante un nivel.

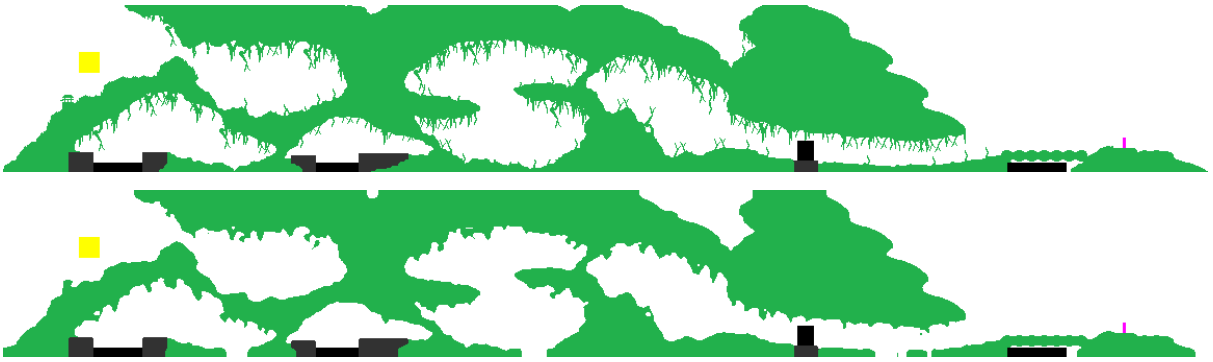
En la versión entregada del juego el modo de IA solo utiliza el algoritmo de decisión y el ejecutor, por lo que solo es capaz de resolver tres mapas para los que se han diseñado las casillas a mano. La integración de las cuadrículas construidas por el generador requiere un tiempo de desarrollo muy elevado dada la gran complejidad (computacional) de procesar una cuadrícula genérica (requiere una implementación cuidadosa y muy optimizada).

Por ello el generador se ha dejado a parte, y se ha invertido el tiempo en mejorar el resto de partes del juego.

8.4.1. Generador

El generador comienza con la imagen inicial del mapa, clasificando los píxeles en terreno y aire, según sean sólidos o transparentes. A la nueva imagen le añade los píxeles del resto de entidades que forman parte del mapa pero no son clasificables como terreno (metal y muros direccionales), y delimita las áreas de la imagen correspondientes a la zona de aparición, la meta y zonas de muerte (trampas y líquidos).

Después de obtener una matriz con el tipo de casilla se procesa la imagen aplicando un kernel de convolución gaussiana, sesgada (y normalizada) en el eje X que da mayor importancia a los cambios en el terreno horizontales que a los verticales. El procesamiento tiene el objetivo de eliminar píxeles sueltos del mapa que no aportan información y dificultan la agrupación de los píxeles en casillas. Es necesario darle importancia al eje X para evitar eliminar zonas de terreno por las que el lemming puede andar pero son lo suficientemente finas como para que el kernel gaussiano sin modificar las elimine.



Comparación de la textura del mapa con y sin filtro en el nivel Fun - 21.
El filtro elimina todos los píxeles que no interfieren en el movimiento del lemming y suaviza las superficies.

Tras esto se recorre cada una de las filas de la matriz en busca de agrupaciones de casillas del mismo tipo, para luego unir en vertical aquellas de igual longitud, hasta un límite parametrizable de altura en píxeles. Un valor razonable para el alto de las casillas es de 8 píxeles: es mayor a la altura que un lemming puede subir y es lo suficientemente pequeña para aproximar excavaciones horizontales del lemming.

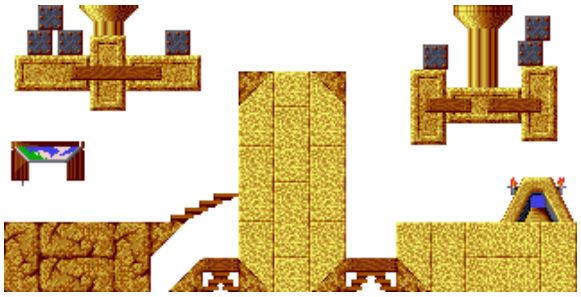
Agrupaciones de casillas del mismo tipo en el nivel Fun - 3.
Las plataformas horizontales son muy finas, pero no deben ser eliminadas por el filtro.

Una vez unidas en altura, los bloques resultantes se subdividen en cajas de una anchura tal que permita aproximar la posición del lemming desde su centro y que se acerque a la de los huecos en el suelo que se producen al cavar hacia abajo. Un buen valor para el ancho de la casilla es 8 píxeles.

Todo este proceso debe hacerse para facilitar el trabajo posterior de decisión, dado que para el algoritmo de búsqueda es imperante tratar las casillas como un todo e impide tener granularidad para elegir en qué punto de la casilla aplicar la habilidad. Con las casillas planteadas de aproximadamente 8x8, las posibles posiciones del lemming se dividen entre 64, en condiciones ideales.

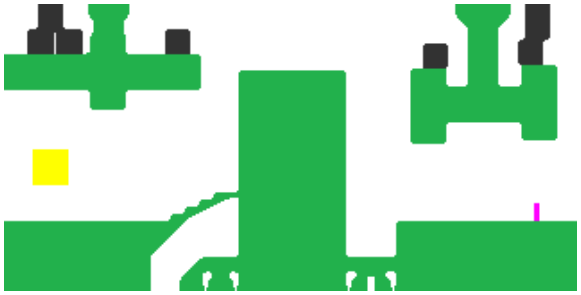
La estructura final es un conjunto de casillas de tamaño variable en el que cada casilla conoce a sus vecinas, formando un grafo doblemente conexo. Las casillas distinguen entre vecinas superiores, inferiores, a su izquierda y a su derecha.

Se aporta el siguiente esquema para visualizar el proceso completo:



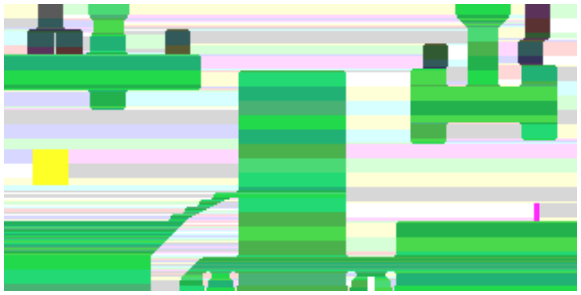
1) Textura del mapa

2) Clasificación de píxeles

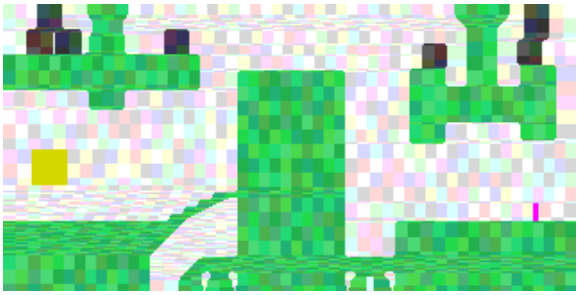


3) Suavizado

4) Agrupación por filas



5) Apilamiento de grupos similares



6) División en casillas

8.4.2. Algoritmo de decisión

El algoritmo de decisión es un algoritmo de búsqueda en profundidad. Iniciando en la casilla inicial (el recuadro amarillo que aparece en las imágenes anteriores) intenta encontrar un camino hasta la final.

Las acciones se distinguen entre obligatorias, como caer o caminar, u opcionales, como las habilidades de los lemmings. El camino que sigue el algoritmo representa el flujo principal de lemmings. Esto implica que sólo pueden resolverse niveles que requieran un único flujo de lemmings.

Estrategias como bloquear a los lemmings en un punto con un lemming muro, enviar a un único scout que escale un muro y abra la pared excavando desde el otro lado, para volver a unir todos los lemmings y atravesar juntos la pared requerirían permitir la creación de flujos secundarios, y ello conllevaría una explosión combinatoria inabarcable en casi todos los casos.

Cada vez que el algoritmo aplica una acción modifica el mapa original de casillas: al excavar hacia abajo se convierte en aire la casilla inferior y el algoritmo avanza una casilla hacia abajo. Poner un lemming muro consiste en invertir la dirección del flujo principal de lemmings, y construir un puente permite avanzar caminando a una casilla de aire y rellenarla con una casilla de puente.

Actualmente el algoritmo puede caminar, caer, cavar recto, cavar hacia abajo, construir puentes y convertirse en muro (invertir la dirección del flujo).

8.4.3. Ejecutor

El ejecutor recibe una política: un conjunto de acciones que tiene que ejecutar a lo largo de las casillas y que llevará al flujo de lemmings desde el inicio al final. Su trabajo es llevarla a cabo en el juego, tarea muy complicada dada la enorme cantidad de casos que hay en el juego.

Cada iteración de físicas el ejecutor comprueba la posición de los lemmings y selecciona aquellos dentro de la casilla en la que se esté ejecutando la política (el lemming que más avanzado vaya generalmente). A ellos deberá aplicarles la acción apropiada definida por la política para esa casilla.

El ejecutor distingue entre tres tipos de acciones:

- Del motor: son realizadas automáticamente por el motor. Ej: caminar.
- Únicas: son realizadas por un solo lemming (el más avanzado siempre). Ej: convertirse en muro.
- Totales: son realizadas por todos los lemmings que pasen por la casilla. Ej: caer con paraguas. Actualmente el sistema no soporta la habilidad del paraguas (aunque sería fácil de añadir) ni ninguna otra acción total, pero el ejecutor está preparado para ese tipo de acciones.

Una vez ha decidido a qué lemmings debe aplicar la acción queda aplicarla, para lo cual sigue reglas específicas para cada una que pueden durar varias iteraciones del ejecutor. Por ejemplo: el lemming muro se activa sólo cuando el lemming llega a un márgen en el centro de la casilla (o se está alejando de él, si se ha pasado), para ajustar mejor la posición de las casillas. Además, si el lemming acaba de caer debe esperarse a que se haya apartado de debajo de la caída del flujo, para evitar que el resto de lemmings le caigan encima.

En otras habilidades como la construcción o la excavación debe acordarse del lemming que está construyendo, e ir refrescando la habilidad cuando se pausen por algún motivo externo (como quedarse sin ladrillos o llegar a un hueco) hasta que salgan de la casilla asociada.

Cuando se haya completado la acción, el ejecutor avanza a la siguiente casilla y vuelve a iniciar el proceso.

9. Diseño del videojuego

En general el juego se divide en los siguientes puntos:

- **Memoria compartida**: Una clase que almacena toda la información y estado del juego y la partida para que las distintas entidades tengan acceso a toda la información.
- **Gestor de pantallas**: Se encarga de la lógica relacionada con cambiar entre los distintos menús, la partida y el video de introducción, en el que, una entidad deberá notificarle a través de la memoria compartida, por ejemplo de que se debe cambiar de menú, y para ello el gestor de pantallas está a la espera de cambios en sus variables compartidas y así cambiarla cuando sea necesario creando una transición a negro en la que al estar en negro es cuando se borran las entidades anteriores y se crean las nuevas del menú o nivel a generar.
- **Gestor de menús**: Se encarga de definir qué entidades se deben de crear cuando se cambie a un menú específico.
- **Gestor de niveles**: Se encarga de definir todas las entidades que se deben de crear para el nivel concreto que se tiene que cargar.
- **HUD**: Es un grupo de entidades generadas al iniciar cualquier partida, en las que cada una tiene una acción asignada como cambiar de habilidad o desplazarse por el mapa (minimapa).
- **Configuración**: Es un grupo de entidades que se generan en un menú en concreto que permiten cambiar ajustes principales del juego como ajustes de sonido, tamaño de ventana o configuración de teclas.
- **Persistencia**: Se encarga de que cada vez que se completa un nivel o se cambian las configuraciones que se actualice en un fichero la información correspondiente para que la próxima vez que se abra el juego, la información se mantenga.
- **Gestor de animaciones:** Para administrar las animaciones de las distintas animaciones se hace uso del delta time generado por el motor para que se cargue el siguiente sprite de la animación, el gestor no permite saltarse algunos frames de la animación ya que para la mayoría de entidades se necesita que haga una determinada acción en un frame concreto. Por ello como cada animación tiene un tiempo de duración, implica que en caso de aumentar demasiado la velocidad del juego la animación ya no iría más rápido debido al tiempo mínimo de coste de una iteración del bucle principal del motor. Por esto se ha decidido permitir aumentar la velocidad de la partida hasta 2.5 veces la velocidad normal.

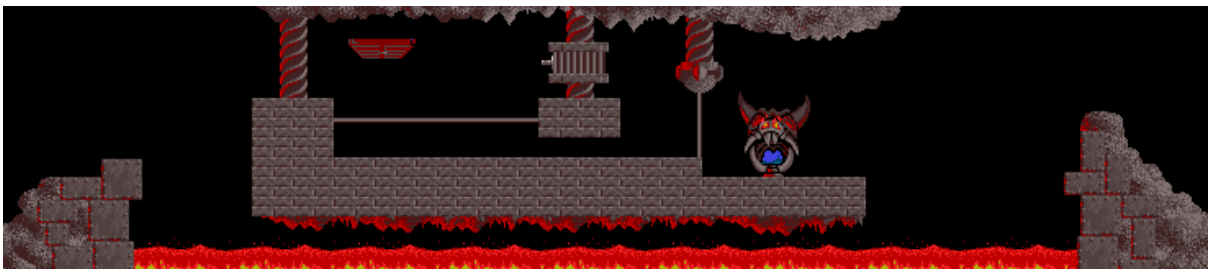
Cuando la partida comienza tras haber creado todas las entidades necesarias, estas empiezan a actuar entre sí, a través de la base de datos interna. La puerta inicial genera lemmings al ritmo apropiado, y los mismos van moviéndose por el mapa y respondiendo a I/O. Las entidades del HUD reflejan estos datos: muestran el número de cada habilidad, la habilidad activa, velocidad actual…

El propio cursor es una entidad que va siguiendo al ratón y va cambiando dependiendo de con qué está colisionando.

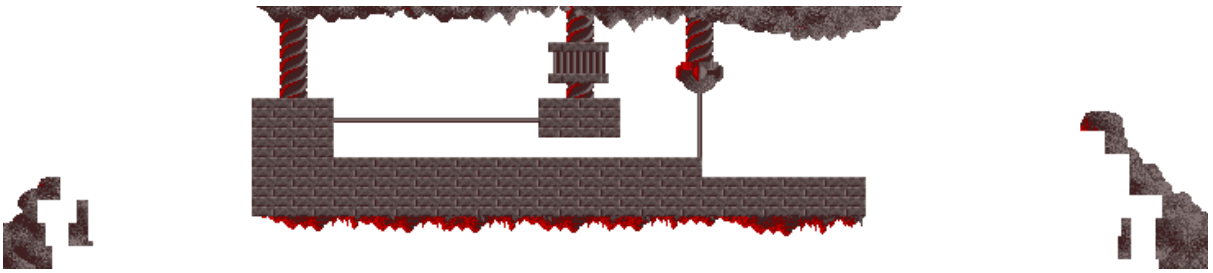
Para facilitar la creación de todos los niveles, se ha implementado un modo debug para las entidades que hay que colocar en el mapa como, puertas, portales,trampas o líquidos, permitiendo que mientras se ejecuta el programa se puedan mover las entidades por el mapa con unos botones y así copiar la ubicación y guardarla.

Toda la información obtenida sobre habilidades de niveles, tiempos, nº de lemmings, nº de lemmings a salvar, tipos de trampas, nombres de niveles… se ha obtenido de la wikipedia de los lemmings [5].

De esta misma web se ha obtenido los assets de los lemmings, entidades, mapas, sonidos y música, aunque hubo que separar en capas los mapas en los que había metales para poder diferenciarlos, y además borrar todas las entidades extras y el fondo ya que eran capturas de los niveles en sí.



Ejemplo de mapa en bruto de la wiki

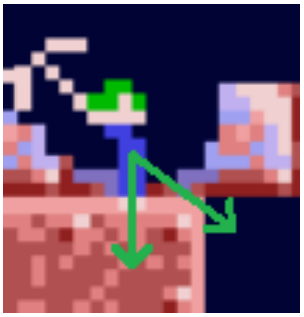


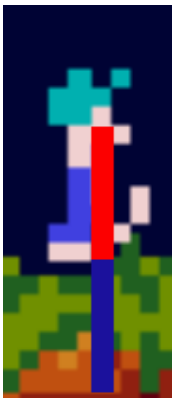
Ejemplo de imagen procesada eliminando fondo y entidades

Mientras tanto hubo que crear nuestros propios assets tanto algunos porque no se encontraban en ninguna parte o nuevos que requerimos para botones de menús, la principal herramienta utilizada para el procesamiento de las imágenes fue GIMP, también la utilizamos para separar y reposicionar todos los assets de los lemmings ya que no estaban bien posicionados y era mejor tenerlo en imágenes separadas debido al coste en memoria y tiempo que llevaría calcular que parte de la imagen representar.

Para simular el movimiento original los lemmings se definen 4 cajas alfa laterales que colisionan con el resto de entidades de la partida y así consiguiendo realizar acciones dependiendo de con qué entidad chocó, por ejemplo cuando colisiona con una pared, tiene que darse la vuelta o cambiar al estado de escalada si posee dicha habilidad, o si colisiona con una trampa, debería destruirse a sí mismo y accionar la animación de la trampa en caso de que la tuviere. Mientras tanto para ajustar la altura o comprobar distancias se utilizan 2 métodos.

La primera es lanzando rayos desde un punto y ver a qué distancia intersectan, este método se utiliza cuando queremos detectar a qué distancia está el techo cuando escalas para saber si te has chocado con él o cuando necesitamos comprobar si hay suelo debajo para caerse o dejar de cavar hacia abajo, también se utiliza cuando el minero (cavar diagonalmente) tiene un metal debajo o si en diagonal hay más mapa que cavar, evitar que empiece a cavar y así no clipear dentro del metal.





Y el segundo método es comprobar una hitbox con el canal alpha para que nos devuelva la altura máxima o si todos los pixeles no son alpha, esto lo utilizamos para ajustar la altura del lemming comprobando primero una caja para saber si tiene que subir el lemming (rojo), en caso de que nos devuelva el píxel más bajo, comprobamos otra caja (azul) para saber si tiene que bajar altura, y en caso de que devuelva el más bajo, el lemming tiene que cambiar de estado a caída. Y así de esta forma replicamos a la perfección el método del juego original, ya que sino no lo hacíamos con esta metodología habían bastantes niveles que no se podían completar.

Gracias al sistema de físicas con velocidades y gravedad se ha podido añadir las partículas de lemming explotando generando un efecto muy similar al original, dándole a cada partícula una velocidad aleatoria en el eje X y una velocidad aleatoria mayor que 0 hacia arriba. Para simular de mejor manera el juego original se quería hacer que los lemmings no explotaran a la vez, ya que en el original le costaba mas de un frame procesar la explosion, mientras que con el nuevo motor esto se hacía en un solo frame, para ello se ha decidido no poner el mismo contador para cada lemming, sino que dependa del orden de aparición generando un efecto de explosiones en serie.

Original

Nuevo

10. Reparto de Tareas

Motor de Videojuego	*Hugo*	*Fer*	*Ismael*	*Jaime*	*Saúl*	*Juan*	*Eryka*
Motor de físicas	🔵						
Sistema de colisiones (aabb, alfa, tipadas…)	🔵						
Motor de render	🔵	🔵					
Gestor de eventos	🔵	🔵					
Lógica general de entidades	🔵						
Gestión de vídeo y sonido		🔵					
Clases de ayuda (rigid body, rays, trigger, particle, fixed text…)	🔵	🔵					
Juego de los Lemmings	*Hugo*	*Fer*	*Ismael*	*Jaime*	*Saúl*	*Juan*	*Eryka*
Assets nuevos del menú y configuración				🔵	🔵		
Persistencia				🔵			
Assets del HUD, lemmings, mapas, trampas…						🔵	
Gestor del juego (generar lemmings, nº habilidades…)						🔵	
Cinemática inicial						🔵	
Lógica de interfaces						🔵	
Lógica de lemmings						🔵	
Habilidad de escalar y cavar horizontal							🔵
Habilidad de cavar diagonal					🔵		
Resto de habilidades						🔵	
Construcción de todos los mapas y niveles						🔵	
Minimapa		🔵				🔵	
Gestor de animaciones						🔵	
Sonidos			🔵				
Corrección de bugs				🔵	🔵	🔵	🔵

IA	***Hugo***	***Fer***	***Ismael***	***Jaime***	***Saúl***	***Juan***	***Eryka***
Algoritmo y ejecutor de la IA	🔵						
Conversión de mapas en tiles		🔵	🔵				
Interfaz del modo IA							🔵
Juego 3D	***Hugo***	***Fer***	***Ismael***	***Jaime***	***Saúl***	***Juan***	***Eryka***
Juego 3D		🔵	🔵				
Memoria	***Hugo***	***Fer***	***Ismael***	***Jaime***	***Saúl***	***Juan***	***Eryka***
Diseño interno	🔵	🔵					
Diseño del videojuego						🔵	
Descripción del juego				🔵	🔵		🔵

11.Referencias

[1] Windows: main thread is blocked when user resizes or moves a window
https://github.com/libsdl-org/SDL/issues/1059

[2] The hardness of the Lemmings game, or Oh no, more NP-completeness proofs
https://www.researchgate.net/publication/228571848_The_hardness_of_the_Lemmings_game_or_Oh_no_more_NP-completeness_proofs

[3] Lemmings is PSPACE-complete
https://arxiv.org/abs/1202.6581

[4] The Lemmings Puzzle: Computational Complexity of an Approach and Identification of Difficult Instances
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=23491c33af0409854a66389cfe28e92dc93385f3

[5] The Lemmings Wiki
https://lemmings.fandom.com/wiki/Lemmings