sphere wars

Autores
Juan Luis Burillo, 542083
Richard Elvira, 666800
Sandra Malpica, 670607
Adrián Milla, 557022

Juego

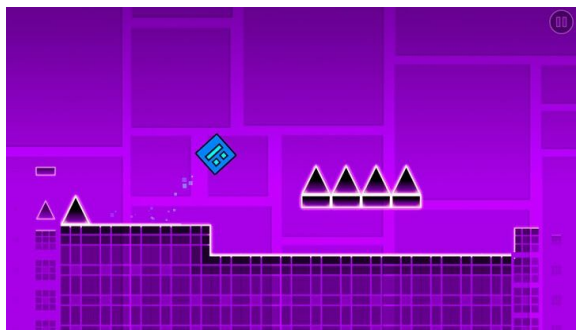
Se ha desarrollado un sencillo juego que se basa en un género muy explotado en la actualidad, sobre todo en el mundo del smartphone: el género de los “infinite runner”, los cuales se tratan de una pantalla infinita que no termina nunca y cuyo objetivo es conseguir la mayor puntuación antes de morir. En pantalla se encuentran diferentes obstáculos para evitar el avance del jugador y en algunas ocasiones hasta enemigos que tratan de cerrar el paso. A continuación se muestran las principales características del juego desarrollado: **Sphere wars**.

Inspiración

Las funciones y estética del juego se basan en otros juegos populares del mismo género o de las plataformas, se han tomado las decisiones pensando en muchos más pero los siguientes son los que más han influenciado a nuestro videojuego.

Geometry Dash

Plataformas que tiene unos niveles muy complejos en los que se deben efectuar saltos para evitar morir, la única acción que se puede realizar es saltar, ya que el jugador se mueve sin parar. Además del juego nos ha gustado la estética, que demuestra que con formas simples se pueden obtener buenos resultados.

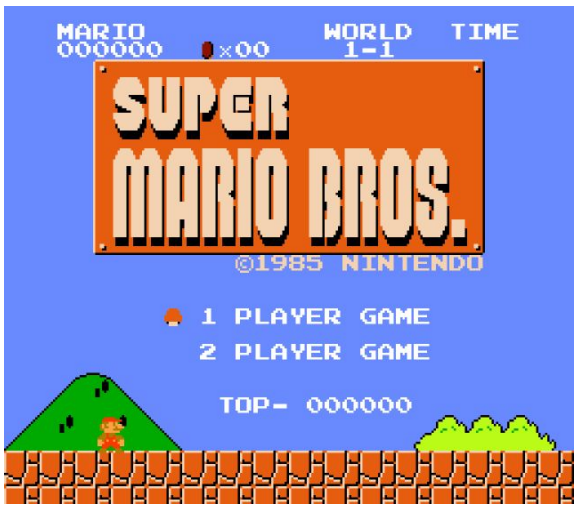


Jetpack Joyride

Runner infinito en el que se trata de llegar lo más lejos posible recogiendo monedas, en este juego se debe esquivar a los enemigos para continuar y el personaje se encuentra siempre en el mismo punto de la pantalla. El juego acaba al chocar con un obstáculo.

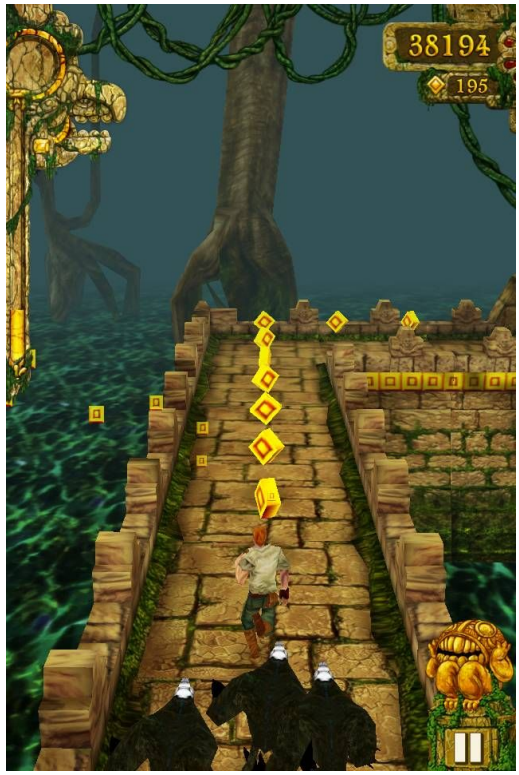
Super Mario Bross

Plataformas de las antiguas consolas, el cual es muy conocido y nos ha servido de inspiración por la estética de las plataformas y para fijarnos en los diferentes enemigos para crear una IA simple para los bots. En este caso, la de la tortuga roja que se ve en el pantallazo.



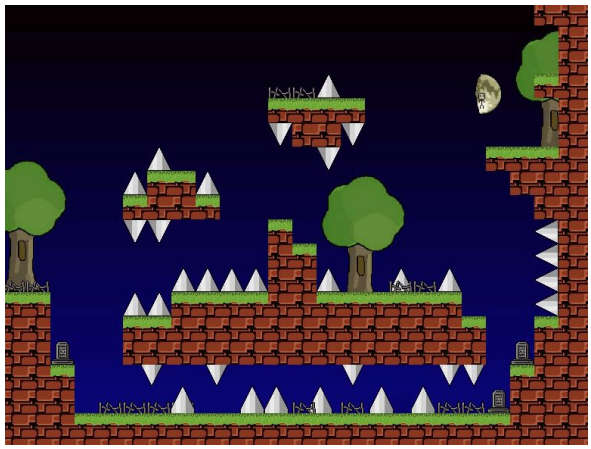
Temple Run

Runner infinito en 3D, trata de llegar lo más lejos posible recogiendo monedas. La estética está en 3D y se puede mover a izquierda y derecha para esquivar obstáculos. A diferencia de otros la partida no acaba al chocarse con un obstáculo pequeño, sino que se ralentiza al jugador dándole una nueva oportunidad y si se vuelve a chocar con otro obstáculo pequeño es cuando muere. Si pasa cierto tiempo se quita la ralentización.



I wanna be the guy

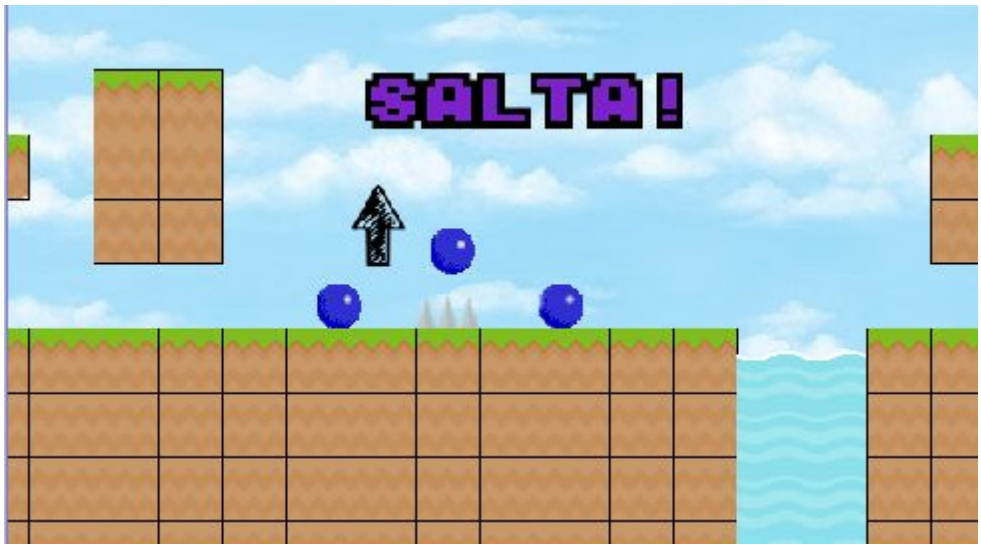
Juego de plataformas 2D independiente. Conocido por sus elementos de plataforma inusualmente difíciles, niveles no ortodoxos y elementos multimedia inspirados en otros videojuegos clásicos. De él aprendemos cómo un reto difícil puede suscitar adicción entre los jugadores.



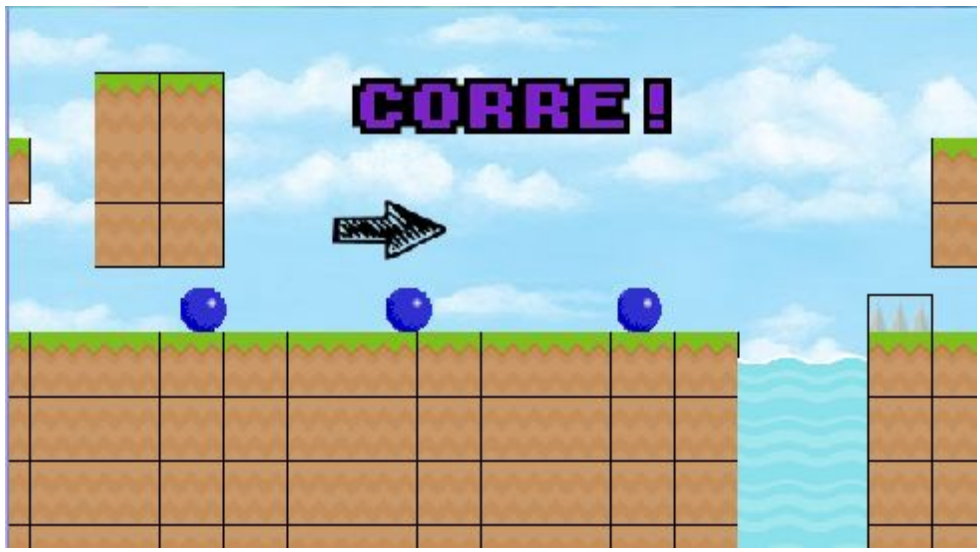
Mecánicas

Se ha decidido que nuestro juego contará con elementos de varios tipos de juegos, el personaje es una bola que tiene que llegar a alcanzar la mejor puntuación según el modo de juego. Se ha decidido que la forma de morir es golpear ciertos obstáculos como pinchos o enemigos, mientras que si se choca con plataformas simplemente no se avanza, pero si el borde de la pantalla alcanza a la bola entonces se muere. A continuación se muestran las distintas mecánicas que incorpora el juego.

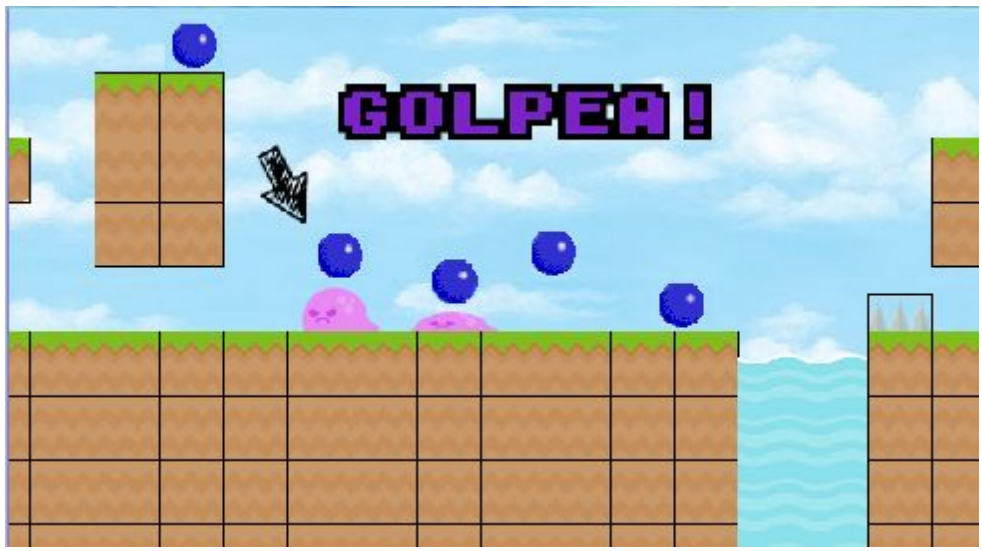
Salta, permite evitar obstáculos pulsando un botón, solo se puede realizar un salto cada vez. Es decir, no se puede saltar dos veces consecutivamente. Para volver a saltar la bola se debe volver a posar en el suelo antes. Si en medio de un salto choca con un obstáculo se pierde el impulso de subida.



Corre, realiza desplazamientos más rápidos para alcanzar lugares lejanos en los saltos y llegar a sitios que normalmente no se puede llegar, pero cuidado… ahora la pantalla se mueve más rápido y es más fácil estrellarse.

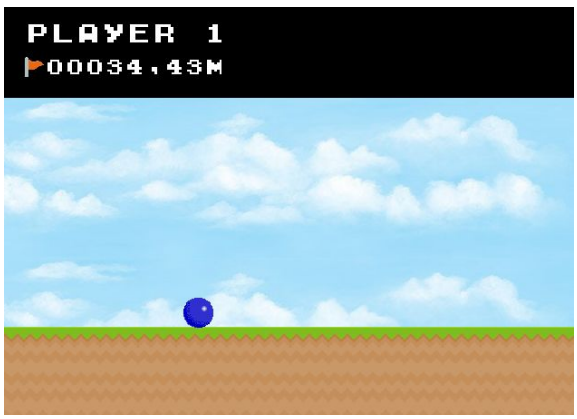


Golpea, salta sobre los enemigos para derrotarlos y dar un pequeño salto al golpearlo, pero no se puede ir de frente si se quiere seguir vivo.

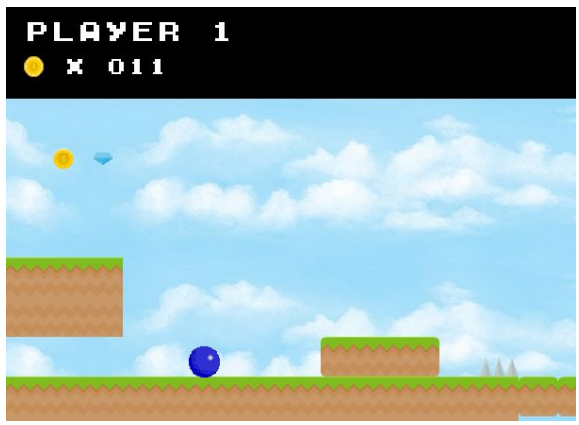


Modos de juego

Se han pensado dos modos distintos de juego dependiendo de cómo aumentar la puntuación. En el modo maratón lo más importante es llegar lejos sin morir. La distancia es la puntuación. En el modo cazatesoros, sin embargo, la puntuación aumenta al obtener tesoros (monedas y gemas), por lo que puede que llegues muy lejos sin conseguir una buena puntuación, o que te hagas con un botín en muy poco tiempo. Ambos modos de juego están disponibles para uno o dos jugadores: ¡la diversión se multiplica!



Maratón



Cazatesoros

Generación de mapas

Sphere wars cuenta con una generación de mapas pseudoaleatoria, no procedural. En cada juego nuevo se genera un mapa de forma aleatoria, componiendo unas unidades indivisibles previamente definidas en distinto orden cada vez. Cada una de estas unidades tiene un tamaño predefinido de 10 bloques de altura por 100 de ancho. Estas unidades han sido creadas manualmente por los miembros del equipo de desarrollo, para asegurar que hay caminos viables a través del mapa, aunque estos sean difíciles. En los mapas se pueden encontrar varios tipos de elementos:

Ambiente

Otra característica aleatoria de la generación de mapas es el ambiente del entorno. Hay varios sets de sprites que pueden aparecer: terreno normal, de castillo, fantasmagórico, desierto y nevado. Durante la generación del mapa se elige uno de estos estilos para todos los fragmentos. Este es un aspecto meramente estético, pero que le da cierta variabilidad al juego.

Elementos del mapa

Bloques de terreno:
Los bloques de terreno son la unidad básica del mapa, sobre los que la esfera puede moverse o saltar. También producen colisiones laterales, que implicarán que la esfera se quede atrapada y muera si permanece quieta en algún rincón. Cada bloque de terreno ocupa un bloque del mapa.

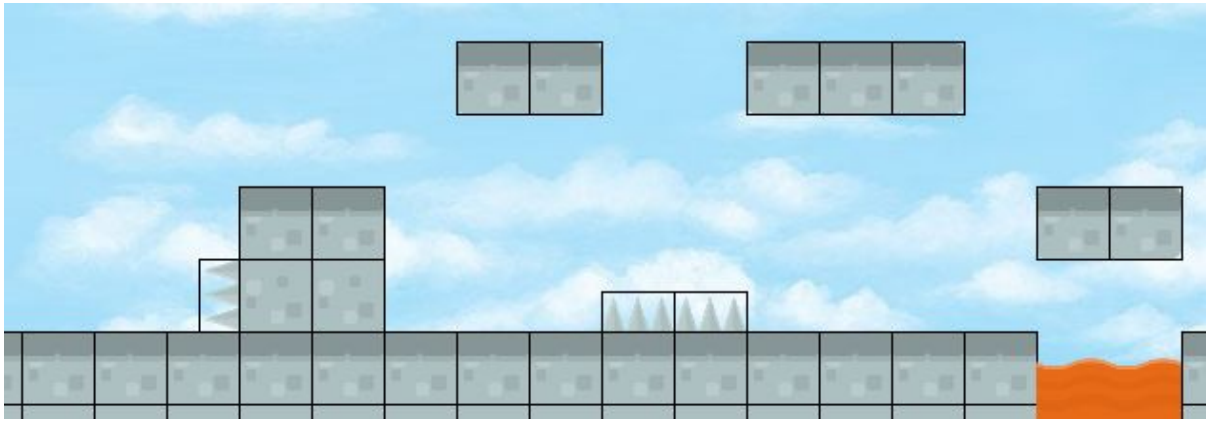
Bloques de líquido:
Un tipo de bloque que mata a la esfera si esta lo toca. Dependiendo del ambiente del terreno, el líquido puede ser lava o agua, pero sus efectos son los mismos. La superficie del líquido ocupa medio bloque, mientras que las capas inferiores ocupan los bloques enteros.

Pinchos:
Tocando los bloques de terreno se pueden encontrar tres afilados pinchos que matarán a la esfera en cuanto esta los roce. Su orientación(hacia dónde miran los pinchos) puede variar dependiendo de sobre qué cara del bloque se encuentre su parte plana. Al igual que la superficie del líquido, ocupan medio bloque.

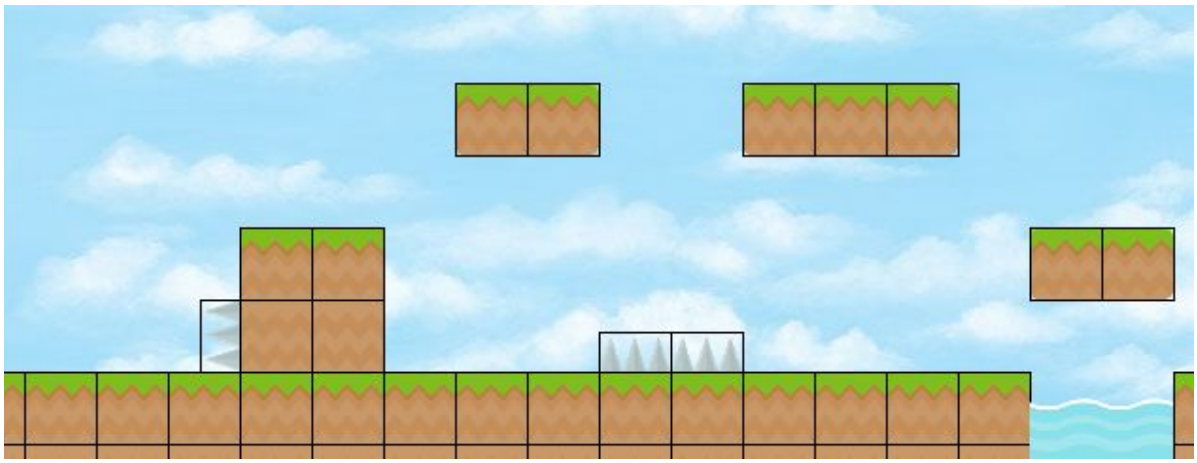
Tesoros:
Los tesoros se dividen entre monedas y diamantes, estos últimos de mayor valor, que flotan en distintos lugares del mapa. Sólo aparecen en el modo cazatesoros, y aumentan la puntuación de la esfera cuando los recoge.

Bots:
Los bots son enemigos sencillos que se reparten por los mapas y cuyo contacto es mortal. Sin embargo, se les puede matar saltando sobre ellos. Se habla de ellos en el apartado de IA.

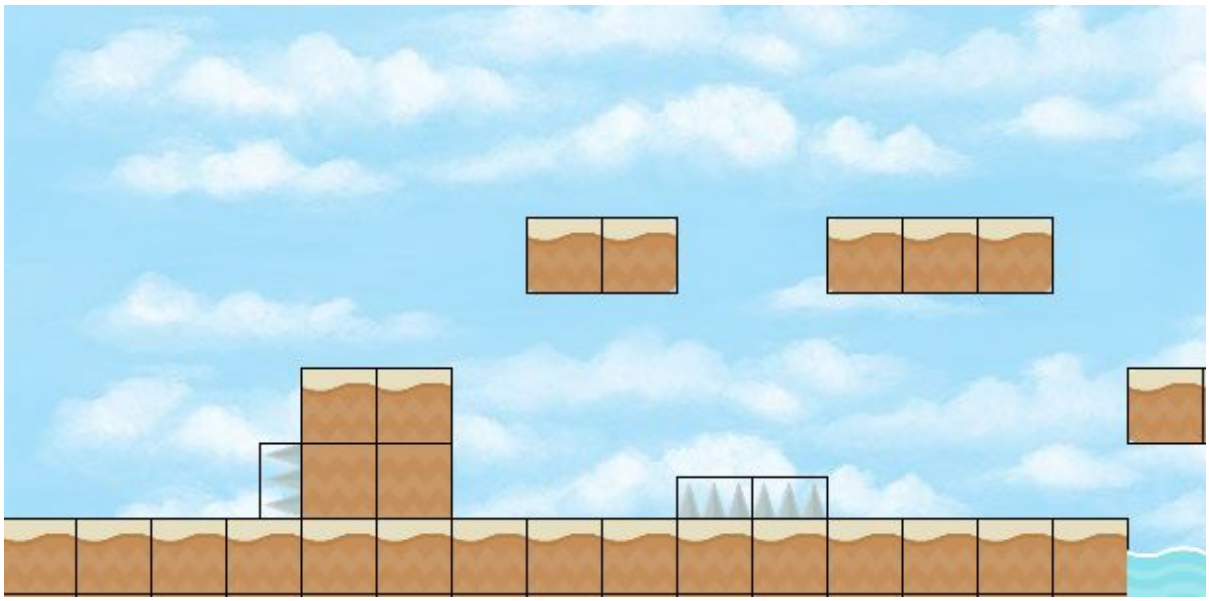
A continuación se muestran los cinco ambientes que hay en el juego, y una pequeña porción de un mapa:



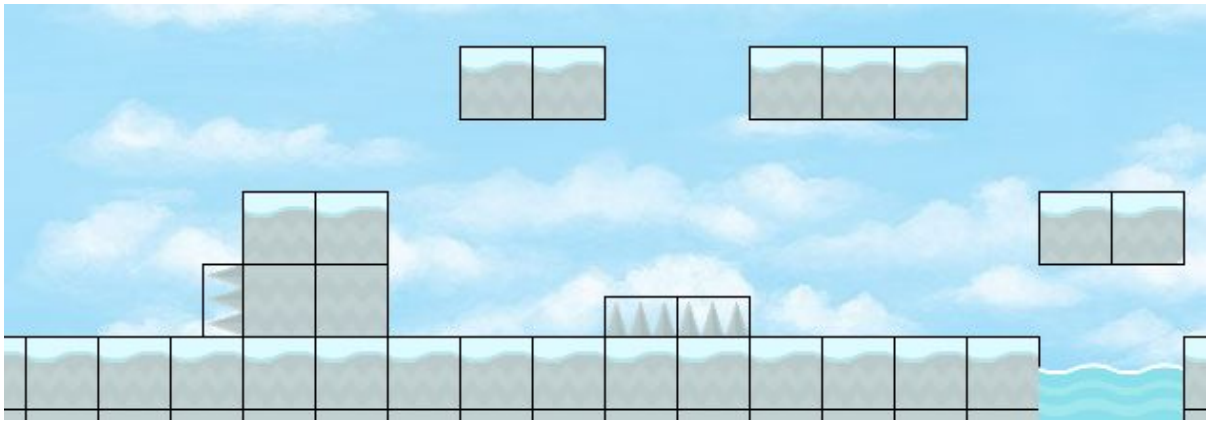
Terreno de tipo castillo



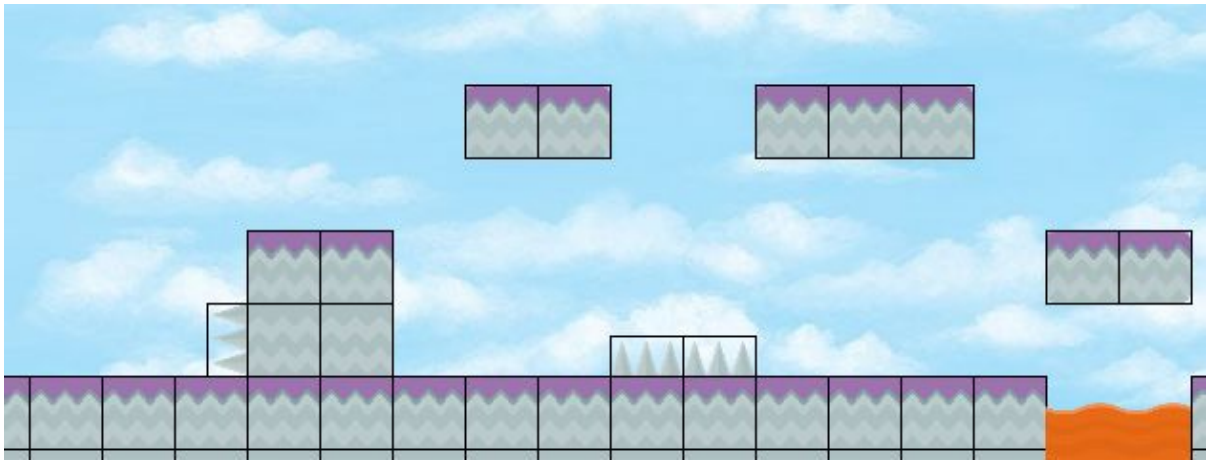
Terreno normal



Terreno desértico



Terreno nevado



Terreno fantasmagórico

Gráficos 2D

Estilo elegido

El estilo del juego está inspirado en otros plataformas de los que se ha hablado anteriormente. Es colorido, y posee varios elementos sencillos pero altamente combinables que proporcionan diversas estructuras de plataformas para el juego. Al ser el juego un infinite runner,carga mapas indefinidamente mientras la esfera no muera. Para evitar cálculos innecesarios sólo se calculan las colisiones y los movimientos de los objetos que se ven en la pantalla. De esta forma todos los bloques y obstáculos que se quedan atrás dejan de ser calculados. Además, según la estructura de mapas explicada en la sección anterior, sólo se cargan en memoria dos fragmentos de mapa (el actual y el siguiente) por lo que aunque el mapa sea infinito siempre ocupa un espacio reducido en memoria.
El mapa se desliza por la pantalla con un scroll horizontal, que puede modificar su velocidad si la pelota entra en Sprint. Por su parte, la pelota se encuentra estable en una posición X de la pantalla a no ser que se vea retrasada por algún obstáculo no letal. Si no muere antes, cuando se libere del obstáculo recuperará su posición en el eje X original progresivamente, aumentando de forma ligera su velocidad durante unos instantes. También se calcula la nueva posición en el eje Y de la pelota cada vez que esta salta. El funcionamiento de la lógica del juego se verá con más detalle en la sección de físicas.
En cuanto al fondo del modo 2D, se ha elegido utilizar la técnica del parallax para dar una mayor sensación de movimiento. Un fondo nublado se desliza a una velocidad menor que el mapa que pasa por delante, ayudando a dar una sensación de profundidad al juego.
Por último, cabe destacar que todos los recursos utilizados en la parte 2D del juego son o bien sprites abiertos que hemos encontrado o bien recursos creados o modificados por nosotros mismos. Se hablará con más detalle de la parte artística del videojuego en su sección correspondiente.

Parallax

Otro lugar en el que se puede ver el parallax es en los menús, implementados completamente en 2D y siguiendo la lógica del núcleo del juego. El fondo estándar de los menús se puede observar también el parallax, compuesto por un fondo estrellado que se desliza lentamente y una cordillera montañosa, más cercana, que se mueve más rápido dando una sensación de proximidad. Esta técnica está implementada de tal forma que se puedan añadir todas las capas que se quiera, siempre que el ancho de la imagen del fondo sea mayor que el de la pantalla.

Gráficos 3D

Librería utilizada

Para la realización del juego en 3D se ha utilizado la librería javax o java3D. Una librería basada en OpenGL para juegos en 3 dimensiones. Facilita el trabajo de la cámara, el tratamiento de imágenes y primitivas y la utilización de transformaciones elementales del mundo, las cuales han sido utilizadas para trasladar, rotar y deformar(para crear sencillas animaciones) los elementos estructurales del mundo en 3D. En concreto, se han utilizado las primitivas que ofrece esta librería: box(para los bloques de terreno y líquido), esfera (para el jugador), cilindro(para los bots de las babosas y el boss), y cono (para los pinchos del mapa), así como el OrbitBehavior para facilitar el comportamiento de la cámara.

Comportamiento de la cámara

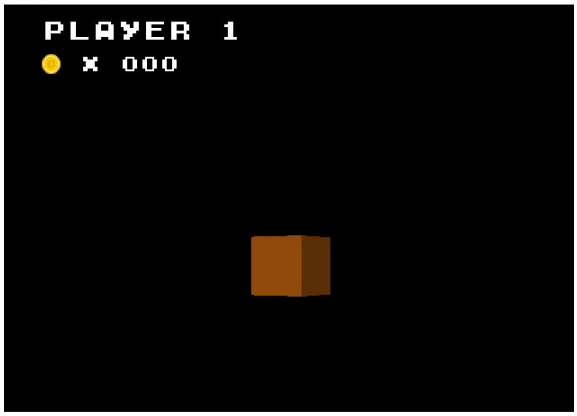
En el modo 3D la cámara es totalmente libre. Se puede rotar, trasladar y hacer zoom in o out alrededor de todo el mundo únicamente con el ratón. Como característica adicional, al pulsar una tecla la cámara vuelve a su posición original deshaciendo todas las transformaciones aplicadas sobre ella.

Adaptación desde 2D

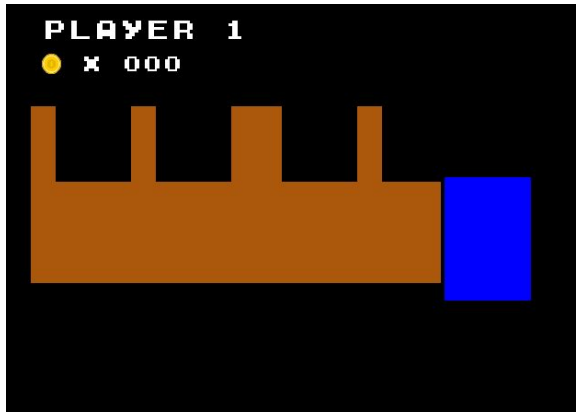
Se han utilizado formas simples para componer los diferentes objetos. Varias de ellas se combinan para crear estructuras más complejas. Los cubos, por ejemplo, se juntan formando líquidos o plataformas. Los cilindros representan a los bots, y conjuntos de conos forman cada bloque de tres spikes. Todo esto se agrupa en un bloque mayor, un bloque de mapa, cuya agrupación forma el mapa completo que se mueve de golpe. Para que la parte 3D se parezca lo más posible a la del 2D se han utilizado texturas sobre todos los objetos, partiendo de los sprites de terreno que se usaban anteriormente además de otras texturas sacadas de fotos (metálica para los spikes y de baba rosa para las babosas). Por último, la esfera tiene componentes de luz ambiental, especular y difusa asemejando los brillos del sprite usado en el modo 2D. El mapa también está iluminado con dos focos de luz direccional para que se aprecien distintas luces en los volúmenes del mapa.

Evolución del 3D

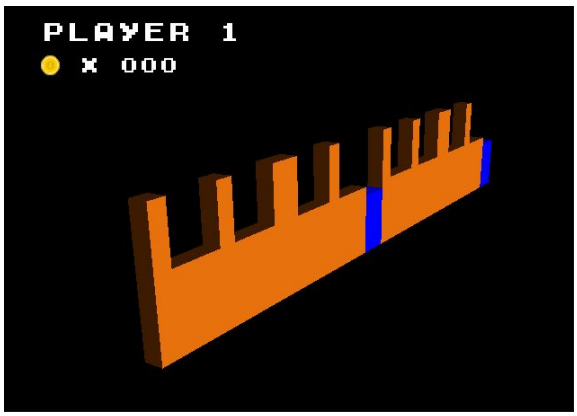
El 3D ha pasado por varias etapas, desde un inicio se han ido marcando pequeños objetivos alcanzables para después poder apoyarse en esos para alcanzar otros más ambiciosos.



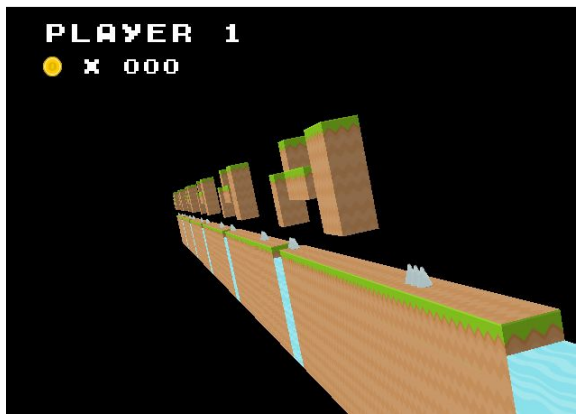
1 - Primer contacto



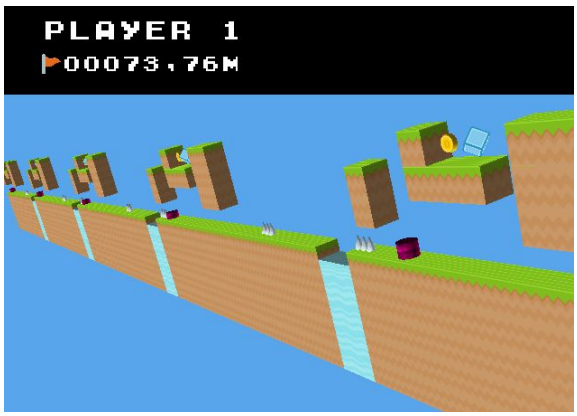
2 - Composición de un mapa



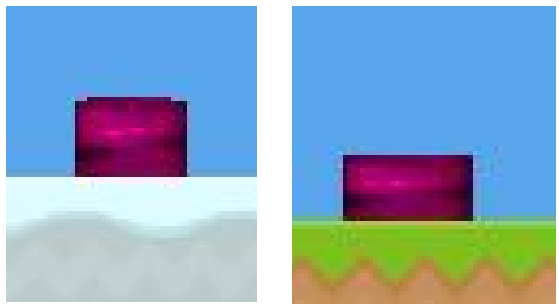
3 - Iluminación de la escena



4 - Aplicación de texturas



5 - Detalles e iluminación



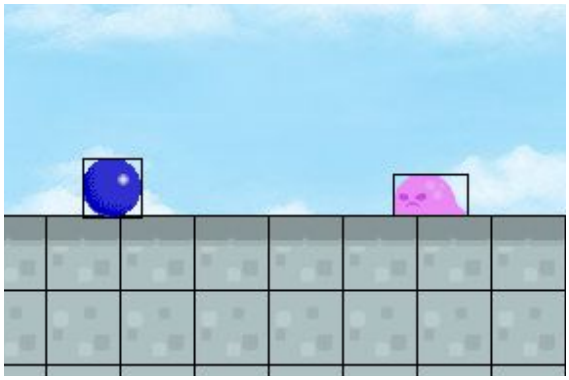
6 - Animación

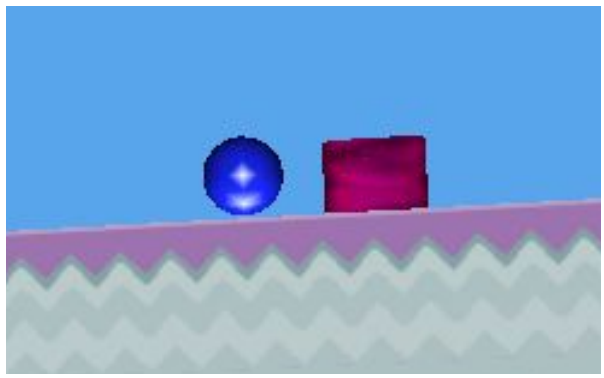
Inteligencia Artificial

A continuación se describe el comportamiento de los dos enemigos que se pueden encontrar en el juego.

Bot - babosa rosa

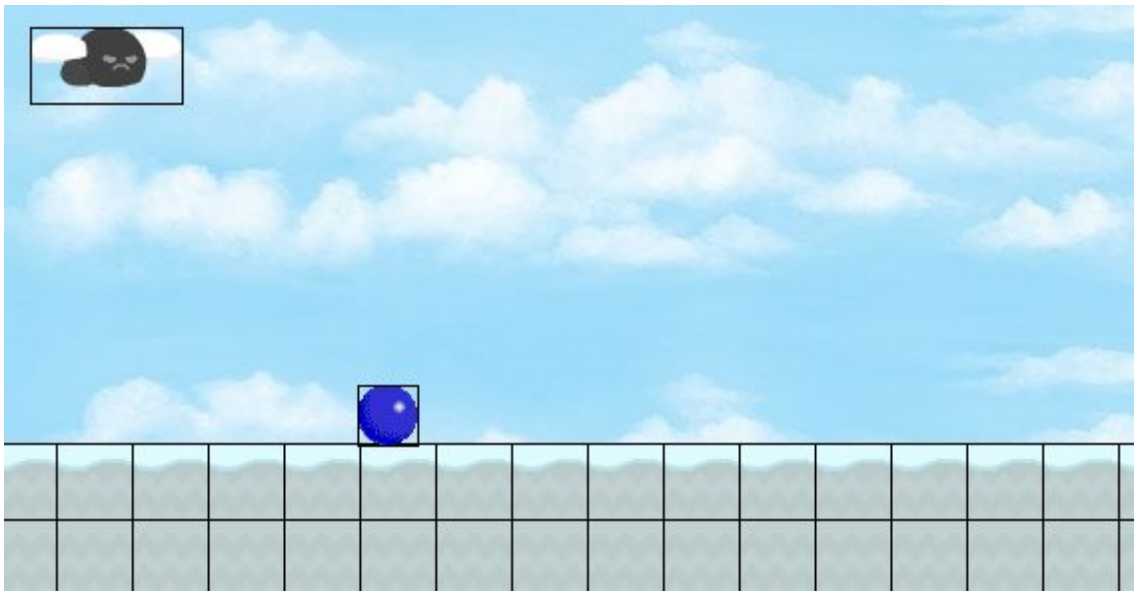
El bot del juego está inspirado en la tortuga roja del juego Super Mario Bros (red Koopa Troopa), aunque es un poco más sencilla. Se mueve en una dirección indefinidamente. Si encuentra un obstáculo o un precipicio, se da media vuelta. Tiene una única vida y mata a la pelota en cuanto la toca. Aparecen en algunos mapas. Se muestran dos capturas de la babosa, en el modo 2D y 3D



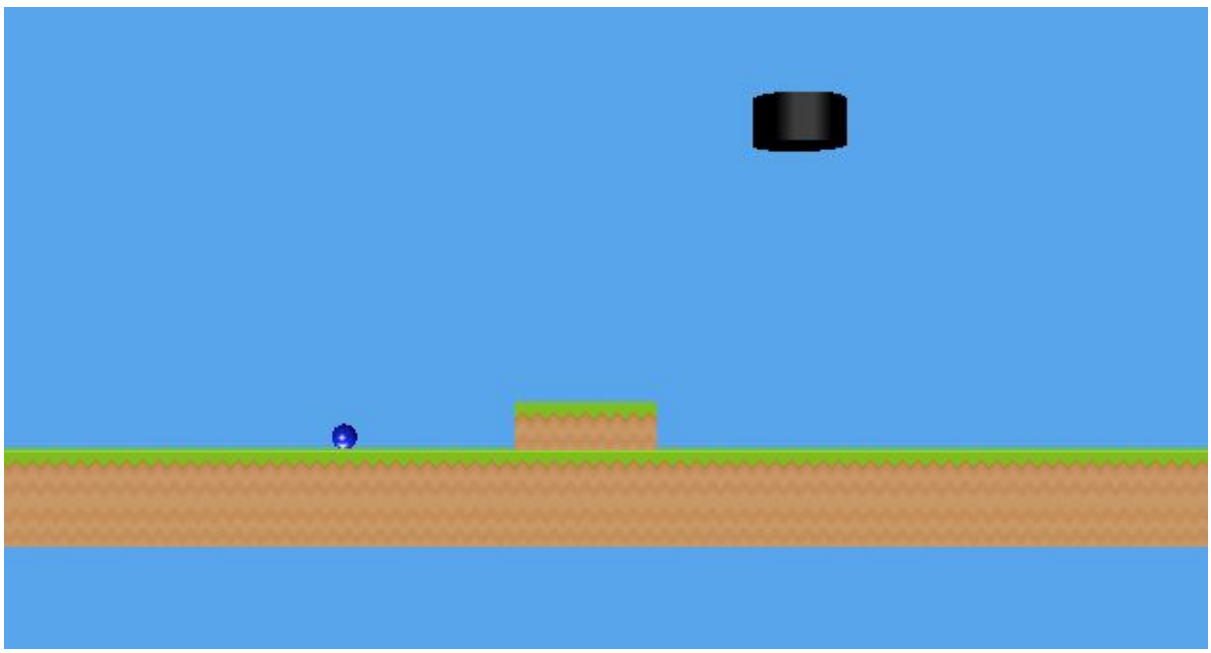


Boss - mosca negra

La inteligencia del jefe es ligeramente más avanzada que la del bot. Para empezar, el boss tiene no una sino 3 vidas. Su comportamiento depende de la vida que le queda en cada momento. Además, aparece al pasar un tiempo, independientemente del tipo de mapa en el que te encuentres. El comportamiento del boss es, a primera vista, errático. Realiza barridos de lado a lado de la pantalla combinados con descensos en picado a diferentes alturas. La probabilidad de que el descenso llegue más abajo es mayor que la de que se mantenga volando a la altura máxima. Además, cuanta más vida pierde, es más probable que no vuelva a la altura inicial, sino que se quede en alguna cercana al jugador o incluso por debajo de él. A medida que la vida del boss desciende, aumenta la probabilidad de que los ataques se dirijan hacia la pelota, siendo la velocidad del descenso proporcional a la distancia que le separa del jugador al inicio del movimiento. Entre los movimientos completos se produce un pequeño descanso. Por último, cuando le queda sólo una unidad de vida el boss cambia de color y se dedica a perseguir o esquivar al jugador más ferozmente. Cuando muere se inicia un contador de resurrección, por lo que siempre volverá a perseguir al jugador, si este no muere antes por algún otro obstáculo. Al ser un enemigo volador no se ve afectado por la gravedad.



Captura de una pelea entre el boss y la esfera en 2D



Captura de una pelea entre el boss y la esfera en 3D

Físicas

Lógica del juego en 2D

El elemento central del motor físico del juego es la esfera (el personaje con el que el jugador se maneja por el mundo de Sphere Wars). La esfera se ve afectada por la gravedad, por lo que cae después de realizar un salto o si no hay suelo debajo de ella. También choca con el resto de objetos del mundo. En general, todos los objetos menos el boss se encuentran en el mapa, que al ser básicamente una matriz cuadriculada permite calcular las colisiones con la esfera de una forma rápida y eficaz. Como se ha dicho antes, sólo se calculan las posiciones y colisiones con los objetos que se ven en pantalla, para minimizar cálculos. El cálculo de colisiones y movimientos se encuentra desdoblado del hilo de dibujo del juego y puede correr a una mayor velocidad para optimizar el cálculo de colisiones. En cada uno de estos “frames”, se obtiene la casilla en la que se encontraría la esfera en ese momento dentro del mapa y se buscan colisiones con las casillas adyacentes en sus 4-vecinos(lateral izquierda, superior, inferior y central ya que sabemos que no tiene nada detrás porque el movimiento hacia adelante es obligatorio). En caso de que haya colisión, se comprueba si ésta es letal o si sólo impedirá avanzar a la pelota y se toman las medidas oportunas al respecto.

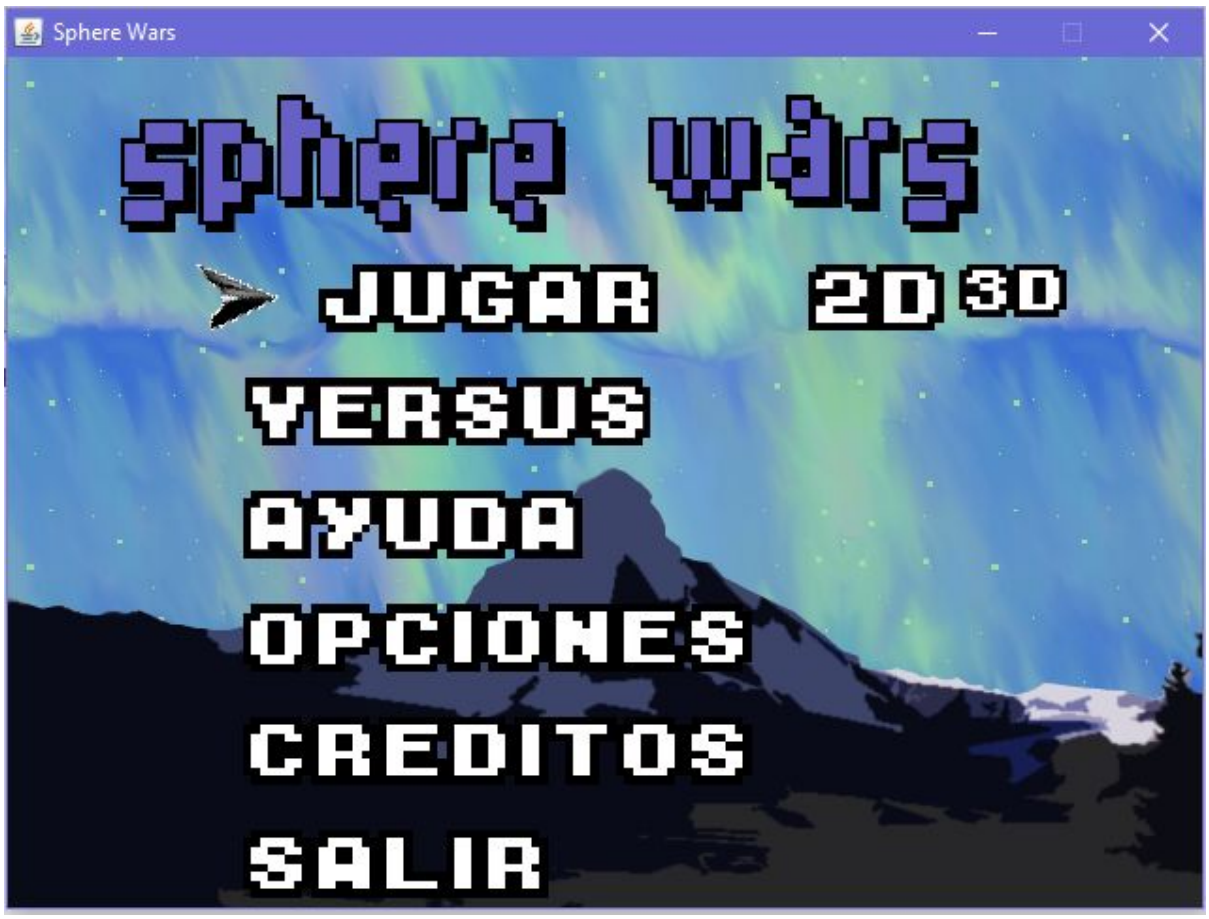
Lógica del juego en 3D

Ya que nuestro mundo en 3D se desarrolla a lo largo de un único bloque de profundidad, las físicas que hemos utilizado para la parte de 2D funcionan exactamente igual en el modo 3D, siendo completamente reutilizables.

Menús y rankings

Menús

Los menús son una constante que aparece a lo largo de todo el juego. Hay un menú principal que es a la vez pantalla de título, un menú de opciones, de pausa y de final de juego. Además el juego cuenta con créditos y con un pequeño menú de ayuda, además de un selector de modo de juego intermedio. Con todos estos menús se ofrecen las opciones de cambiar la resolución de la pantalla, activar o desactivar el sonido, cambiar o personalizar los controles de juego así como elegir si el juego se ejecuta en 2D o en 3D. Como se ha indicado anteriormente, los menús out-game cuentan con un parallax de fondo; mientras que los menús in-game - como el menú de pausa y el de fin de juego - son ligeramente transparentes para permitir que se vea el juego subyacente. Con estas opciones se pretende que el jugador se sienta más cómodo y pueda navegar con mayor facilidad por el juego.



Rankings

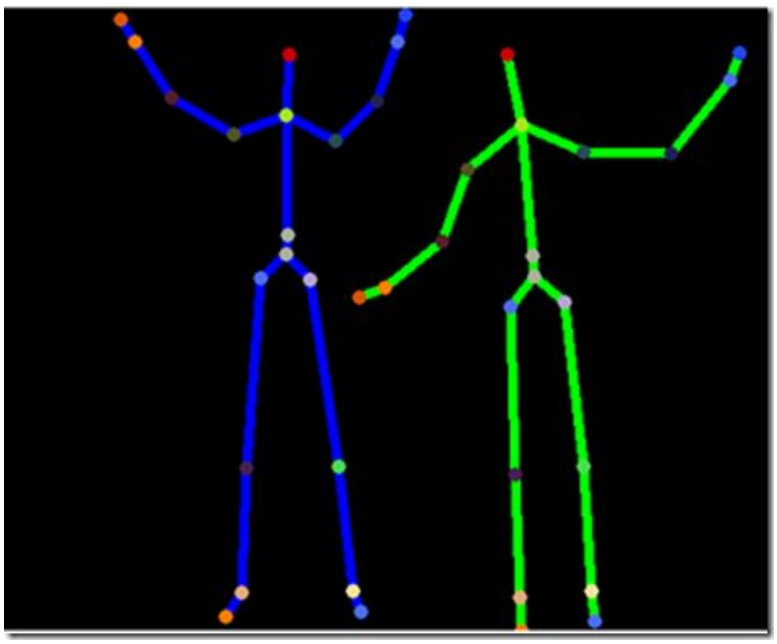
Como se ha explicado anteriormente, existen dos modos de juego con distintos tipos de puntuaciones. Para aumentar la sensación de competitividad del juego, éste cuenta con rankings persistentes para cada uno de los modos. Al terminar cada partida, si el jugador ha entrado en el top 10 de puntuación se le pide que introduzca tres letras y se guarda esa puntuación. En los créditos se pueden ver, además de a los creadores del videojuego, el top 3 de ranking de cada una de las modalidades, con la puntuación de cada puesto.

Arte del juego

Todo el contenido multimedia que aparece en el juego es copyleft, es decir, existe para su libre uso y modificación. Principalmente se están utilizando dos páginas: http://opengameart.org para los sprites del 2D (y texturas del 3D), los fondos y los efectos de sonido; y http://incompetech.com para la música de fondo del juego. Para cargar la música y reproducir los sonidos en el videojuego se han utilizado varias librerías de java. El sonido del juego está programado para hacer entrada y salida con fading de manera suave, y también cambia dependiendo de si el jugador se encuentra en los menús, en el juego o enfrentándose al jefe final. Para implementarlo se ha usado la librería de paulscode.com que reproduce archivos de audio de formato ogg .

Kinect

Al desarrollar el videojuego en un equipo de 4 personas, se ha añadido como característica adicional el uso del kinect como controlador de la esfera. El uso de este periférico se gestiona con la librería ufdw (Kinect for Java). Este dispositivo funciona reconociendo uno o varios esqueletos en su rango de visión. En cada frame de esqueleto recibido comprueba la posición de las manos del jugador. Si se alejan en la dirección que queremos (hacia arriba para saltar, por ejemplo) se manda una señal. Si la mano pasa suficiente tiempo quieta se resetea la posición de la mano, creando un nuevo punto inicial desde el que se iniciarán los nuevos movimientos. Cabe destacar que, al igual que en el menú de opciones se puede elegir qué teclas utilizar para saltar o correr, se puede elegir si la mano dominante (la del salto) es la izquierda o la derecha para el Kinect.



Ejemplo (externo a nuestro proyecto) de cómo el Kinect traquea dos esqueletos completos.

Problemas y retos

A lo largo de la realización del proyecto se han encontrado varios problemas. El principal de ellos ha sido calcular las colisiones de forma correcta entre la esfera y el resto de elementos del mundo, tanto en 2D como en 3D.
Otro reto al que el equipo se ha enfrentado es el equilibro de la jugabilidad del videojuego. Al estar trabajando sobre una idea propia en vez de sobre un videojuego ya existente ha habido muchas cosas que se han tenido que calibrar o rediseñar de una forma u otra.
Por otra parte, la utilización del Kinect también ha supuesto algún que otro problema. El más grave de ellos es que sólo funciona sobre determinados sistemas operativos (lo que es relativamente fácil de solucionar con una máquina virtual) y que además necesita un puerto USB 3.0. Esto nos ha limitado a poder utilizar el Kinect con uno sólo de los ordenadores de nuestro equipo de desarrollo, con sus consecuentes limitaciones.
En cuanto a la realización de mapas, también es difícil crear mapas que sean lo suficientemente difíciles como para resultar interesantes pero no tanto como para que sea imposible poder superarlos. Se han testeado los distintos mapas creados, tanto internamente entre los miembros del equipo como con beta testers “caseros” para intentar asegurar que todos los mapas se pueden conquistar.

Ideas que no se han llegado a implementar

Por la complejidad y tiempo limitado de este proyecto, hay varias ideas que se han planteado pero que finalmente no se han llegado a implementar. Entre ellas:

Implementar el modo 3D para 2 jugadores: Actualmente, el modo 3D sólo está disponible para un jugador (en cualquiera de los dos modos de juego)

Añadir nuevos modos de juego: Entre otros el modo contrarreloj en el que lo que cuenta es el tiempo, y el modo slasher en el que la puntuación aumenta matando a los enemigos que se cruzan en tu camino.

Kinect para dos jugadores: El segundo jugador sólo tiene la opción de jugar con el teclado (el primer jugador puede elegir si juega con teclado o con Kinect). Una de las mejoras que no se han podido llegar a realizar era permitir el modo versus (2 jugadores) con Kinect.

Mejoras del boss: Otra de las ideas que no hemos tenido tiempo de realizar ha sido hacer que el boss disparase proyectiles además de lanzarse en picado sobre nosotros. Además, se preveía que destrozase los bloques superiores con los que entrara en contacto, en vez de atravesarlos.

Mayor diversidad de enemigos: se podrían introducir nuevas IAs sencillas además de la de la babosa.

Atacar o devolver proyectiles: para herir a los enemigos o abrir nuevas rutas en el mapa.

Desplazamiento lateral en el modo 3D: Una de las ideas más enriquecedoras para el modo 3D era añadir dos carriles laterales como en el juego temple run, para enriquecer los movimientos de la esfera y su forma de evitar obstáculos, de tal forma que fuera más divertido moverse por el mapa y añadirle alguna funcionalidad real al modo 3D que el modo 2D no tuviera.

Generación procedural de mapas: Crear un algoritmo que permitiese crear los mapas de manera automática.

Cambio de la dificultad: Aunque hay mapas más sencillos que otros, no hay una opción en el menú que permita elegir la dificultad del juego.

Mejorar las animaciones: añadir más animaciones al 3D y a la esfera.

Cronograma

	Febrero	Marzo	Abril	Mayo
BrainStrom	4 horas.			
Definición del juego	11 horas.			
Busqueda de arte	15 horas.			
Búsqueda de librerías		30 horas.	25 horas.	
Estructura del proyecto	12 horas.			
Desarrollo				
1ªit - Kinect		15 horas.	5 horas	10 horas.
2ªit - Menús		20 horas.	6 horas.	37 horas
3ªit - Dibujo 2D	10 horas.	25 horas.	15 horas.	8 horas.
4ªit - Físicas		20 horas.	15 horas.	42 horas.
5ªit - IA			4 horas.	8 horas.
6ªit - Audio			15 horas.	10 horas.
7ªit - Dibujo 3D			32 horas.	20 horas.
8ªit - Opciones de resolución				20 horas.
9ªit - Mapas		10 horas.	2 horas.	15 horas.
10ªit - Testing				12 horas.

Detalles

- BrainStrom.
    - Febrero, tormenta de ideas para definir el tipo de videojuego a desarrollar.
- Definición del juego.
    - Febrero, elección del tipo de juego y definición de las características a implementar.

- Búsqueda de arte.
    - Febrero, búsqueda de imágenes para representar fondos, objetos 2D(Sprites), audio y efectos de sonido.
- Búsqueda de librerías.
    - Marzo, búsqueda de diferentes librerías de audio y comprobación de ellas para encontrar una que funcione como se necesita.
    - Abril, búsqueda de librerías para usar gráficos 3D, comparación entre ellas para elegir la más adecuada.
- Estructura del proyecto.
    - Febrero, definición de clases y esqueleto del proyecto, preparación de un entorno de GitHub para el proyecto.
- 1ªit - Kinect.
    - Marzo, investigación del kinect y prueba de proyectos de ejemplo para su comprensión.
    - Abril, extraer esqueleto de un jugador.
    - Mayo, implementación de controles del kinect en el juego.
- 2ªit - Menús.
    - Marzo, creación de los menús principales y de opciones (sprites, funcionamiento del cursor, etc.)
    - Abril, creación del menú de pausa y final de juego (con los rankings)
    - Mayo, se añade la funcionalidad a las opciones recién programadas y se añaden el menú de ayuda y los créditos.
- 3ªit - Dibujo 2D.
    - Febrero, primeros pasos para dibujar elementos en 2D mediante sprites.
    - Marzo, dibujo de mapa en pantalla mediante sprites y movimiento del mapa.
    - Abril, inclusión de bot con su animación para cambiar de sprites al moverse.
    - Mayo, inclusión del jefe con su animación para simular el vuelo.
- 4ªit - Físicas.
    - Marzo, colisiones entre objetos del mapa y el jugador.
    - Abril, colisión entre el bot y el jugador para establecer las muertes.
    - Mayo, colisión entre el boss y el jugador, adaptar las colisiones a la vista 3D.
- 5ªit - IA.
    - Abril, implementación de la lógica del bot.
    - Mayo, implementación de la lógica del boss basada en probabilidades.
- 6ªit - Audio.
    - Abril, mirar las librerías y creación del gestor.
    - Mayo, programación de la transición suave y la entrada del audio en general, además de los efectos de sonido.
- 7ªit - Dibujo 3D.
    - Abril, primeros pasas para dibujar un mundo en 3D, entendimiento de la librería, dibujado de primeros mapas.
    - Mayo, dibujado del resto de elementos, se establece la posición de la cámara y se da la posibilidad de hacer una navegación libre.
- 8ªit - Opciones de resolución.
    - Mayo, se añaden opciones para cambiar la resolución de la pantalla.
- 9ªit - Mapas.

    - Marzo, creación del lector y contenedor de los mapas.
    - Abril, implementación de toda la lógica de movimiento del mapa y creación de mapas de prueba.
    - Mayo, creación de mapas de juego.
- 10ªit - Testing.
    - Mayo, testeo del juego para encontrar y arreglar bugs, así como probar que todos los mapas pueden ser resueltos.

Reparto de tareas

Se ha realizado un reparto de tareas en el que cada uno se ha encargado de una parte y en tareas más complejas se han realizado entre varios,

- **Juan Luis Burillo:** BrainStrom, definición del juego, búsqueda de librerías, físicas, dibujo 3D, mapas, testing.
- **Richard Elvira:** BrainStrom, definición del juego, búsqueda del arte, búsqueda de librerías, estructura del proyecto, menús, dibujo 2D, IA, dibujo 3D, mapas, testing.
- **Sandra Malpica:** BrainStrom, definición del juego, búsqueda de librerías, menús, IA, dibujo 3D, opciones de resolución, mapas, testing.
- **Victor Adrian Milla:** BrainStrom, definición del juego, búsqueda del arte, búsqueda de librerías, kinect, audio, mapas, testing.

Conclusiones

Se ha comprobado la enorme complejidad que tiene desarrollar una juego a pesar de ser uno sencillo, el mayor problema es implementar unas físicas de colisiones y que se calculen de forma eficiente ya que se debe llegar a un compromiso de eficiencia para que el juego se ejecute a suficiente velocidad para que no vaya a saltos.

La forma de implementar las formas 3D y animarlas ha servido para afianzar los conocimientos adquiridos en gráfica sobre las diferentes transformaciones a los diferentes objetos.

A pesar de haber logrado desarrollar el juego y haber cumplido los objetivos que se habían propuesto, al final siempre se observan posibles mejoras que implementar por lo que queda una sensación de que siempre queda algo que mejorar.

El trabajo ha servido para poner sobre un proyecto todos los conocimientos adquiridos e ir viendo cómo poco a poco se va desarrollando y creciendo, otorgando mucha satisfacción cuando se ve funcionando, no obstante hay ocasiones en que la gran cantidad de horas invertidas y los problemas hallados pueden ser frustrantes para los miembros del trabajo.