

Documentación Asteroids

30/05/2018

Tomás Rodríguez Martín

Índice

• Manual de variables	2
◦ Asteroids.h	2
◦ Disparo.h	2
◦ EnemyShip.h	3
◦ ListaAsteroid.h	3
◦ ListaDisparos.h	4
◦ ListaPolys.h	4
◦ Math_aux.h	4
◦ Ships.h	5
◦ Ui.h	5
◦ Stats.h	6
◦ Asteroids.cc	6
• Manual de funciones	8
◦ Asteroids.h	8
◦ Disparo.h	9
◦ EnemyShip.h	10
◦ Game.h	11
◦ ListaDisparos.h	16
◦ ListaPolys.h	18
◦ ListaAsteroids.h	20
◦ Math_aux.h	22
◦ Stats.h	24
◦ Ships.h	25
◦ Ui.h	28
◦ Asteroids.cc	31
• Manual de usuario	32
◦ Base de Datos	42
◦ Controles	43
◦ Instalación	43
• Análisis post mortem	44

Manual de variables

Asteroids.h

struct asteroid
<ul style="list-style-type: none">• esat::Vec2 *puntos_locales;• esat::Vec2 *puntos_globales;• esat::Vec2 pos;• esat::Vec2 v;• int vertices;• int size;• int tipo;
<p>Estructura que uso para almacenar los datos del asteroide. Contiene los puntos locales, los puntos globales, la posición, la velocidad, los vértices que tiene el polígono que define el asteroide, el tamaño del mismo y el tipo de asteroide.</p>

Disparo.h

struct disparo
<ul style="list-style-type: none">• esat::Vec2 pos;• esat::Vec2 v;• double time;
<p>Esta estructura guarda los datos de los disparo. Contiene la posición, la velocidad y una variable de tiempo para destruirlos una vez se cumple dicho tiempo.</p>

Enemyship.h

struct EnemyShip
<ul style="list-style-type: none">• esat::Vec2 *puntos_locales;• esat::Vec2 *puntos_globales;• esat::Vec2 pos;• esat::Vec2 v_dir;• float v;• bool live;• int size;
<p>Estructura que uso para almacenar los datos de la nave enemiga. Contiene, al igual que la nave, los puntos locales y globales de la misma y la posición. Además tenemos, en este caso, un vector director, la velocidad, un booleano para saber cuando está viva y un tamaño.</p>

ListaAsteroid.h

struct NodoAsteroid
<ul style="list-style-type: none">• asteroid val;• NodoAsteroid *nextNodo;• NodoAsteroid *prevNodo;
<p>Estructura que uso para almacenar los nodos de la lista que gestiona los asteroides. Contiene el struct del asteroide, un puntero al siguiente nodo y un puntero al anterior nodo.</p>

ListaDisparos.h

struct NodoDisparo
<ul style="list-style-type: none"> • disparo val; • NodoDisparo *nextNodo; • NodoDisparo *prevNodo;
<p>Estructura que uso para almacenar los nodos de la lista que gestiona los disparos.</p> <p>Contiene el struct del disparo, el puntero al siguiente nodo y el puntero al anterior nodo.</p>

ListaPolys.h

struct NodoPoly
<ul style="list-style-type: none"> • poly val; • NodoPoly *nextNodo; • NodoPoly *prevNodo;
<p>Estructura que uso para almacenar los nodos de la lista que gestiona los polígonos en los que se dividen los asteroides.</p> <p>Como viene siendo habitual tenemos el struct del polígono, el puntero al siguiente nodo y el puntero al anterior nodo.</p>

Math_aux.h

struct Er
<ul style="list-style-type: none"> • float a,b,c;
<p>Estructura que uso para almacenar los datos de la ecuación general de la recta.</p> <p>Contiene tres floats que definen la ecuación de esta forma:</p> <ul style="list-style-type: none"> • $ax + by + c = 0$

struct poly
<ul style="list-style-type: none">• esat::Vec2 *points;• int vertices;
Estructura que uso para almacenar los datos del polígono. Contiene los puntos que definen el polígono y el número de vértices del mismo.

Ships.h

struct ship
<ul style="list-style-type: none">• esat::Vec2 puntos_locales[6];• esat::Vec2 puntos_globales[6];• float ShipRads;• esat::Vec2 pos;• esat::Vec2 v_dir;• esat::Vec2 v_force;• float v;• float SpeedUp;• float friction;• float invtime;• bool collisionable;
Estructura que contiene los datos de la nave. Contiene los puntos locales, los puntos globales, el giro de la nave en forma de radianes, la posición, el vector que define dónde está mirando la nave, el vector de fuerza que define la dirección a la que se mueve la nave, la velocidad, la fricción que hace que la nave se vaya parando progresivamente, el tiempo de invencibilidad y un booleano para saber si esta en esta en modo invencible.

Ui.h

int nrows
Guarda el número de filas que devuelve la consulta sql.

Stats.h

struct Stats
<ul style="list-style-type: none">• int puntuacion;• int lives;
Estructura que guarda las estadísticas de la partida. Almacena la puntuación de la partida actual, y el número de vidas que le queda al jugador.

struct bd
<ul style="list-style-type: none">• char user[50]="";• char password[50]="";• char name[50]="";• char subname[50]="";• char country[50]="";
Estructura que almacena los datos del jugador. Contiene un string con el nombre de usuario, la contraseña, el nombre, el apellido y el país.

asteroids.cc(main)

Stats stats
Declaración de la estructura de las estadísticas de la partida.

ship Nave
Declaración de la estructura de la nave.

EnemyShip NaveEnemiga
Declaración de la estructura de la nave enemiga.

```
NodoAsteroid *asteroides;
```

Declaración de la estructura de la lista de los asteroides.

```
NodoDisparo *disparos;
```

Declaración de la estructura de la lista de los disparos.

```
NodoDisparo *disparos_enemigos;
```

Declaración de la estructura de la lista de los disparos enemigos.

```
sqlite3 *db;
```

Declaración de la base de datos sqlite.

```
bd user;
```

Declaración de la estructura de los datos del usuario.

Manual de funciones

Asteroids.h

Funcion	Draw_asteroid
Tipo de dato que devuelve	void
Parámetros de entrada	asteroid *a
Explicación	Función que pinta el asteroide en pantalla, se transforman cada uno de los puntos mediante matrices de transformación, escalando según su tamaño y trasladandolo a la posición. Finalmente se pinta en pantalla.

Funcion	Init_asteroid
Tipo de dato que devuelve	void
Parámetros de entrada	asteroid *a, int size
Explicación	Función que inicializa el asteroide. Inicializa el tipo de asteroide, los puntos locales según el tipo de asteroide, el tamaño del asteroide, que recibe por parámetro y el vector de velocidad.

Funcion	UpdatePos
Tipo de dato que devuelve	void
Parámetros de entrada	asteroid *a
Explicación	Actualiza la posición del asteroide sumandole el vector velocidad a la posición. Además, controla que se cambie la posición al rebasar los límites.

Disparo.h

Funcion	Init_disparo
Tipo de dato que devuelve	void
Parámetros de entrada	disparo *a, ship *nave
Explicación	Función que inicializa los disparos de la nave. Inicializa la posición del disparo con la posición de la nave, la velocidad del disparo y el tiempo en el que se ha disparado.

Funcion	Init_disparo
Tipo de dato que devuelve	void
Parámetros de entrada	disparo *a, EnemyShip *nave, esat::Vec2 v
Explicación	Función que inicializa el disparo de la nave enemiga. La posición del disparo será nuevamente la de la nave, la enemiga en este caso, el vector velocidad se saca de restar la posición de la nave(la nave del jugador) con la de la nave enemiga(la posición de la nave se pasa por parámetro en este caso). Luego se normaliza el vector y se escala para darle la velocidad. Finalmente se asigna el tiempo en el que es inicializado el disparo.

Funcion	Draw_shot
Tipo de dato que devuelve	void
Parámetros de entrada	disparo *a
Explicación	Función que pinta el disparo. Se inicializa otro punto para dibujar la línea y por último se dibuja la línea en pantalla.

Funcion	UpdatePos
Tipo de dato que devuelve	void
Parámetros de entrada	disparo *a
Explicación	Actualiza la posición del disparo, sumandole la velocidad en cada frame y actualiza la posicoin en caso de que rebase los limites de la pantalla.

Enemyship.h

Funcion	InitShip
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *nave
Explicación	Inicializa la posición de la nave en un lado aleatorio da la pantalla, dependiendo del lado cogerá un vector director diferente. Además se inicializa la velocidad, el tamaño de la nave y un booleano para saber si esta viva o no.

Funcion	DrawShip
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *nave
Explicación	Dibuja la nave en pantalla. Transforma los puntos locales de la nave, escalando a su tamaño y trasladandola a la posición. Primero se pinta el contorno de la nave y luego las líneas de su interior.

Funcion	UpdatePos
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *nave
Explicación	Actualiza la posición de la nave, además de controlar que se actualice cuando rebasan los límites.

Funcion	UpdateState
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *nave
Explicación	Actualiza el estado de la nave. Cuando la nave está muerta, la actualiza a viva con un aleatorio.

Funcion	UpdateDir
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *nave
Explicación	Actualiza aleatoriamente el vector director de la nave, cambiando la dirección en el eje y.

Game.h

Funcion	Disparo
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo **ListaDisparos, ship *nave
Explicación	Inicializa los disparos de la nave. Si el jugador pulsa el espacio, crea un nuevo disparo y lo inserta en la lista de disparos.

Funcion	DivideAsteoroid
Tipo de dato que devuelve	void
Parámetros de entrada	NodoPoly **p, asteroid *a
Explicación	Función que divide los asteroides en diferentes polígonos. Dependiendo del tipo de asteroide se divide en diferentes polígonos, se inicializan los polígonos, se le dan los respectivos puntos globales del asteroide y se insertan en la lista de polígonos.

Funcion	TestColPolys
Tipo de dato que devuelve	bool
Parámetros de entrada	NodoPoly **p, esat::Vec2 v
Explicación	Testea la colisión de un punto con la lista de polígonos.

Funcion	DivideAsteroidCol
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **a,int size,esat::Vec2 pos
Explicación	Función que genera dos asteroides nuevos. Inicializa los asteroides y los inserta en la lista. La posición que recibe por parámetro es la del asteroide destruido.

Funcion	ColShotAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **a, NodoDisparo **d, Stats *stats
Explicación	<p>Comprueba la colisión de los disparos de la nave con los asteroides.</p> <p>Recorre la lista de disparos, y comprueba la colisión con los asteroides.</p> <p>Una vez se detecta colisión se suman los puntos correspondientes, crea dos asteroides nuevos y finalmente elimina el asteroide y el disparo que han colisionado.</p>

Funcion	ColShotAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **a, NodoDisparo **d
Explicación	<p>Comprueba la colisión de los disparos de la nave enemiga con los asteroides.</p> <p>Esta función es exactamente igual a la anterior, salvo que en este caso no se suma ninguna puntuación.</p>

Funcion	DeadTimeShots
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo **d
Explicación	<p>Función que elimina los disparos una vez superan el tiempo de actuación.</p> <p>Recorre la lista de disparos comprobando que hayan superado el tiempo de vida, si es así los elimina de la lista.</p>

Funcion	ColShipAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave, NodoAsteroid **a, Stats *stats
Explicación	Comprueba la colisión de los asteroides con la nave. Coge los tres puntos de fuera de la nave y comprueba la colisión de cada uno de ellos con los asteroides. Si colisionan se resta una vida en las estadísticas.

Funcion	ColEnemyAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *nave, NodoAsteroid **a
Explicación	Comprueba la colisión de los asteroides con la nave enemiga. Al igual que la función anterior recorre la lista de asteroides y comprueba la colisión con los puntos de la nave enemiga.

Funcion	Disparo
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo **ListaDisparos, EnemyShip *enemiga, ship *nave
Explicación	Función que genera los disparos de la nave enemiga. Cuando la nave está viva dispara siguiendo un aleatorio, la dirección del disparo se saca de restar la posición de la nave con la de la nave enemiga.

Funcion	ColEnemyAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *nave, NodoAsteroid **a
Explicación	<p>Función que detecta las colisiones entre la nave enemiga y los asteroides.</p> <p>Como de costumbre, se recorre la lista de asteroides y se comprueba la colisión con cada uno de los puntos de la nave enemiga.</p> <p>Si colisiona se elimina el asteroide y se pone en falso el estado de vida de la nave.</p>

Funcion	ColShotEnemy
Tipo de dato que devuelve	void
Parámetros de entrada	EnemyShip *enemiga, NodoDisparo **d, Stats *stats
Explicación	<p>Detecta la colisión de los disparos del jugador con la nave enemiga.</p> <p>Se recorre la lista de disparos del jugador y se comprueba si colisiona con los polígonos que conforman la nave enemiga.</p> <p>Si colisiona la nave enemiga muere y el jugador recibe puntos por ello.</p>

Funcion	ColShotShip
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave, NodoDisparo **d, Stats *stats
Explicación	<p>Detecta la colisión de los disparos de la nave enemiga con la nave del jugador.</p> <p>Esta función es igual a la anterior, salvo que cuando hay colisión el jugador pierde una vida.</p>

Funcion	CheckDeath
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState, Stats *stats
Explicación	Detecta si no le quedan vidas al personaje, y cambia el estado del juego si así fuera.

Funcion	CheckWin
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState, NodoAsteroid *asteroides
Explicación	Detecta si no quedan asteroides en la lista y cambia el estado del juego si así fuera..

Funcion	void InitGame
Tipo de dato que devuelve	void
Parámetros de entrada	ship *Nave, EnemyShip *NaveEnemiga, Stats *stats, NodoDisparo **disparos, NodoDisparo **disparos_enemigos, NodoAsteroid **asteroides
Explicación	Inicializa todos los aspectos del juego. Inicializa los asteroides, las estadísticas, la nave enemiga y la nave del jugador.

ListaDisparos.h

Funcion	CrearListaDisparo
Tipo de dato que devuelve	NodoDisparo*
Parámetros de entrada	void
Explicación	Funcion para inicializar la lista. Simplemente devuelve NULL.

Funcion	EsvaciaLista
Tipo de dato que devuelve	bool
Parámetros de entrada	NodoDisparo *Lista
Explicación	Comprueba si la lista está vacía. Simplemente comprueba si la lista es iguala a NULL, si es así devuelve true, si no false.

Funcion	EliminarNodo
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo *Nodo, NodoDisparo **Lista
Explicación	Elimina un nodo de la lista. Se le pasa por parámetro la lista y el nodo que se desea eliminar. Se enlazan los nodos adyacentes y se elimina el nodo que se pasa por parámetro.

Funcion	BorrarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo **Lista
Explicación	Recorre la lista de principio a fin borrando los nodos a su paso.

Funcion	DeadTimeShots
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo **d
Explicación	Función que elimina los disparos una vez superan el tiempo de actuación. Recorre la lista de disparos comprobando que hayan superado el tiempo de vida, si es así los elimina de la lista.

Funcion	InsertarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo **Lista, disparo val
Explicación	Inserta un nodo al principio de la lista. Recibe el valor del nodo por parámetro, crea el nodo y lo inserta por el inicio.

Funcion	MostrarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo *Lista
Explicación	Recorre la lista dibujando todos los disparos.

Funcion	MoveDisparos
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo **Lista
Explicación	Recorre la lista actualizando la posición de los disparos.w

ListaPolys.h

Funcion	CrearListaPoly
Tipo de dato que devuelve	NodoPoly*
Parámetros de entrada	void
Explicación	Funcion para inicializar la lista. Simplemente devuelve NULL.

Funcion	EsvaciaLista
Tipo de dato que devuelve	bool
Parámetros de entrada	NodoPoly *Lista
Explicación	Comprueba si la lista está vacía. Simplemente comprueba si la lista es iguala a NULL, si es así devuelve true, si no false.

Funcion	EliminarNodo
Tipo de dato que devuelve	void
Parámetros de entrada	NodoPoly *Nodo, NodoPoly **Lista
Explicación	Elimina un nodo de la lista. Se le pasa por parámetro la lista y el nodo que se desea eliminar. Se enlazan los nodos adyacentes y se elimina el nodo que se pasa por parámetro.

Funcion	BorrarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoPoly **Lista
Explicación	Recorre la lista de principio a fin borrando los nodos a su paso.

Funcion	InsertarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoPoly **Lista, poly val
Explicación	Inserta un nodo al principio de la lista. Recibe el valor del nodo por parámetro, crea el nodo y lo inserta por el inicio.

ListaAsteroids.h

Funcion	CrearListaAsteroide
Tipo de dato que devuelve	NodoAsteroid*
Parámetros de entrada	void
Explicación	Funcion para inicializar la lista. Simplemente devuelve NULL.

Funcion	EsvaciaLista
Tipo de dato que devuelve	bool
Parámetros de entrada	NodoAsteroid *Lista
Explicación	Comprueba si la lista está vacía. Simplemente comprueba si la lista es iguala a NULL, si es así devuelve true, si no false.

Funcion	EliminarNodo
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid *Nodo, NodoAsteroid **Lista
Explicación	Elimina un nodo de la lista. Se le pasa por parámetro la lista y el nodo que se desea eliminar. Se enlazan los nodos adyacentes y se elimina el nodo que se pasa por parámetro.

Funcion	BorrarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **Lista
Explicación	Recorre la lista de principio a fin borrando los nodos a su paso.

Funcion	InsertarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoPoly **Asteroid, asteroid val
Explicación	Inserta un nodo al principio de la lista. Recibe el valor del nodo por parámetro, crea el nodo y lo inserta por el inicio.

Funcion	MostrarLista
Tipo de dato que devuelve	void
Parámetros de entrada	NodoDisparo *Lista
Explicación	Recorre la lista dibujando todos los asteroides.

Funcion	MoveAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **Lista
Explicación	Recorre la lista actualizando la posición de los asteroides.

Funcion	InitAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **Lista
Explicación	Inicializa los asteroides del juego. Crea tres asteroides y los inserta en la lista.

Funcion	MoveAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **Lista
Explicación	Recorre la lista actualizando la posición de los asteroides.

Math_aux.h

Funcion	Vec2xScalar
Tipo de dato que devuelve	esat::Vec2
Parámetros de entrada	const esat::Vec2 &v, float scalar
Explicación	Multiplica las componentes del vector por un escalar. Devuelve el vector resultante .

Funcion	Vec2minusVec2
Tipo de dato que devuelve	esat::Vec2
Parámetros de entrada	const esat::Vec2 &v, const esat::Vec2 &w
Explicación	Resta las componentes del primer vector con las del segundo. Devuelve el vector resultante.

Funcion	Vec2plusVec2
Tipo de dato que devuelve	esat::Vec2
Parámetros de entrada	const esat::Vec2 &v, const esat::Vec2 &w
Explicación	Suma las componentes de los dos vectores. Devuelve el vector resultante.

Funcion	Vec2xScalar
Tipo de dato que devuelve	esat::Vec2
Parámetros de entrada	const esat::Vec2 &v, float scalar
Explicación	Recorre la lista actualizando la posición de los asteroides.

Funcion	Module
Tipo de dato que devuelve	float
Parámetros de entrada	const esat::Vec2 &v
Explicación	Devuelve el módulo del vector.

Funcion	MoveAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **Lista
Explicación	Recorre la lista actualizando la posición de los asteroides.

Funcion	Vec2Normalized
Tipo de dato que devuelve	esat::Vec2
Parámetros de entrada	const esat::Vec2 &v
Explicación	Devuelve el vector normalizado. Se saca el modulo y se dividen las componentes del vector por el mismo. Devuelve el vector resultante.

Funcion	Vec2toEr
Tipo de dato que devuelve	Er
Parámetros de entrada	const esat::Vec2 &v, const esat::Vec2 &w
Explicación	Dados dos puntos saca la ecuación de la recta. Devuelve la ecuación de la recta resultante.

Funcion	TestDotEr
Tipo de dato que devuelve	bool
Parámetros de entrada	esat::Vec2 v, Er r
Explicación	Comprueba la posición de un punto respecto de la recta. Si el resultado es positivo devuelve true, si no devuelve false.

Funcion	ColVec2Poly
Tipo de dato que devuelve	bool
Parámetros de entrada	esat::Vec2 d,poly p
Explicación	<p>comprueba la colisión de un punto respecto de un polígono.</p> <p>Va transformando dos puntos en una recta y comprueba la posición del punto pasado por parámetro respecto a esta.</p> <p>Si alguna da negativa significa que no colisiona, si no habrá colisión y retornara true.</p>

Funcion	MoveAsteroids
Tipo de dato que devuelve	void
Parámetros de entrada	NodoAsteroid **Lista
Explicación	Recorre la lista actualizando la posición de los asteroides.

Stats.h

Funcion	InitStats
Tipo de dato que devuelve	void
Parámetros de entrada	Stats *player
Explicación	Inicializa las estadísticas de la partida.

Funcion	InitBd
Tipo de dato que devuelve	void
Parámetros de entrada	bd *mybd
Explicación	Inicializa los parámetros de la base de datos.

Ships.h

Funcion	InitShip
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Inicializa los parámetros de la nave.

Funcion	DrawFire
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Dibuja el fuego de la nave cuando acelera.

Funcion	DrawShip
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Dibuja la nave en pantalla. Si el jugador está presionando el acelerador dibuja el fuego también.

Funcion	RotateShip
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Rota la nave si pulsas los botones de dirección laterales.

Funcion	CheckVforce
Tipo de dato que devuelve	void
Parámetros de entrada	ship * nave
Explicación	Comprueba que el vector que impulsa la nave no se pase de velocidad. Si el módulo del mismo supera el límite se capta a ese límite.

Funcion	DecreaseVforce
Tipo de dato que devuelve	void
Parámetros de entrada	esat::Vec2 &v,float f
Explicación	Disminuye el vector de fuerza. Devuelve el resultado de la resta.

Funcion	SpeedUp
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Controla la aceleración de la nave. Cuando pulsas el botón de aceleración en una dirección, se genera un vector en la dirección donde hemos pulsado. Este vector se suma al vector fuerza y se guarda en el mismo vector fuerza, actualizando la misma. Si no acelera va aumentando la fricción y se frena la nave progresivamente.

Funcion	UpdateVdir
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Actualiza el vector director de la nave. Coge el punto de arriba de la nave y le resta la posición de la misma sacando así la dirección a la que está mirando.

Funcion	UpdatePos
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Actualiza la posición de la nave. Le suma a la posición el vector de fuerza. Además actualiza la posición cuando nos salimos de la pantalla.

Funcion	UpdateColStatus
Tipo de dato que devuelve	void
Parámetros de entrada	ship *nave
Explicación	Actualiza la capacidad de la nave de ser colisionable. Si el tiempo de colisión es menor del tiempo actual ya es colisionable.

Ui.h

Funcion	callback
Tipo de dato que devuelve	static int
Parámetros de entrada	void *unused, int count, char **data, char **columns
Explicación	Función auxiliar de sqlite que pasa por pantalla las filas de la tabla resultante de la consulta. Además incrementa la variable global de número de filas cada vez que es llamada, para saber si la consulta ha dado algún resultado.

Funcion	PrintUi
Tipo de dato que devuelve	void
Parámetros de entrada	Stats *stats, ship *nave
Explicación	Imprime por pantalla la interfaz de usuario en la partida. Concretamente, las vidas que le quedan al jugador y la puntuación que tiene en ese momento.

Funcion	Col
Tipo de dato que devuelve	bool
Parámetros de entrada	esat::Vec2 *v, esat::Vec2 w
Explicación	Función de colisión sencilla entre un punto y un caja. Se usa básicamente para detectar donde pulsa el jugador sobre el menú.

Funcion	PrintMainMenu
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState, bd *bduser
Explicación	Imprime el menú principal en pantalla. Dependiendo de donde se pulse cambiará el estado del juego de un sitio a otro. Resetea los parámetros de la base de datos.

Funcion	bool CheckUser
Tipo de dato que devuelve	bool
Parámetros de entrada	bd *b, sqlite3 *db
Explicación	Comprueba que el usuario introducido en el login pertenezca a la base de datos. Se crea la consulta con un string concatenando los campos y se ejecuta. Si el número de filas que se devuelve es mayor que 0 significa que existe un usuario con esos parámetros, luego retornará true. Si no existe retornará false.

Funcion	PrintLoginMenu
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState, bd *b, int *campo, sqlite3 *bd
Explicación	Imprime el menú de login. Detecta en qué campo estas pulsando y rellena el array con las letras que recibe por teclado. Si los parámetros que se han introducido estan en la base de datos entra en partida, si no se resetean los campos.

Funcion	InsertToDateBase
Tipo de dato que devuelve	bool
Parámetros de entrada	bd *b, sqlite3 *db
Explicación	Inserta en la base de datos los parámetros del usuario. Como en el login se concatena los campos para conformar la consulta y se ejecuta posteriormente.

Funcion	PrintRegisterMenu
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState, bd *b, int *campo, sqlite3 *db
Explicación	Imprime el menú de registrar usuario. Al igual que en el login detecta en qué campo has pulsado y captura el input del teclado en el mismo.

Funcion	PrintGameOverMenu
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState
Explicación	Imprime por pantalla el menú de fin de partida.

Funcion	UpdateDateBase
Tipo de dato que devuelve	void
Parámetros de entrada	bd *b, sqlite3 *db, Stats *stats
Explicación	Actualiza la base de datos con la puntuación del jugador. La consulta está hecha de tal forma que si la puntuación es más alta de la que ya tenía se actualiza en la base de datos.

Funcion	PrintWinMenu
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState, Stats *stats, bd *bduser, sqlite3 *bd
Explicación	Imprime el menú de victoria.

Asteroids.cc(main)

Funcion	GameLoop
Tipo de dato que devuelve	void
Parámetros de entrada	void
Explicación	Bucle de la partida. Se ejecuta continuamente mientras estés jugando.

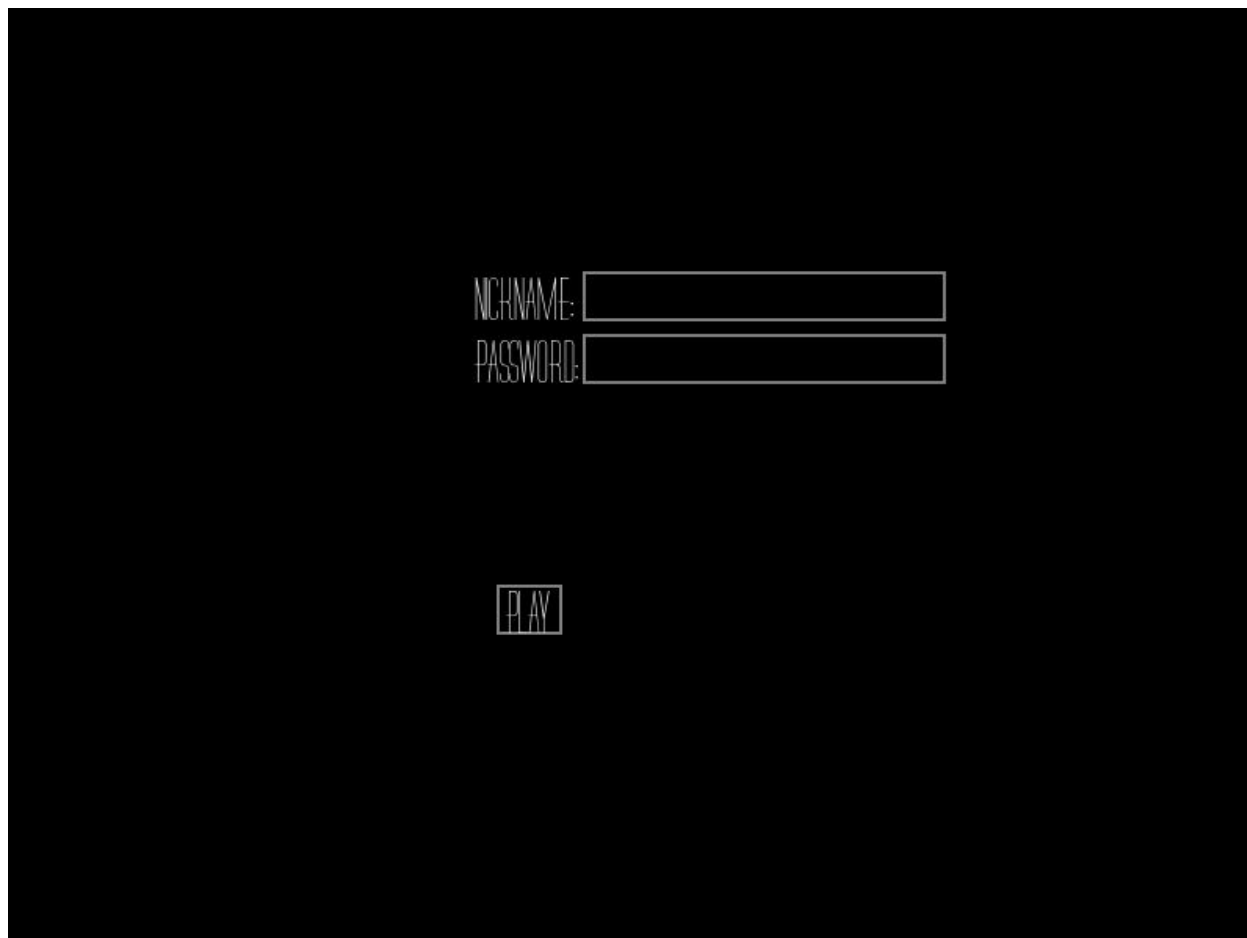
Funcion	MainLoop
Tipo de dato que devuelve	void
Parámetros de entrada	int *GameState, bd *bduser, int *campo
Explicación	Bucle principal del juego. Dependiendo del estado del juego irá pasando por las distintas partes del juego.

Manual de usuarios

Una vez iniciamos el juego nos encontramos con el menú de inicio, en el que tenemos dos botones. Botón de login para entrar en la cuenta de un usuario registrado y botón de registro para registrar un usuario.



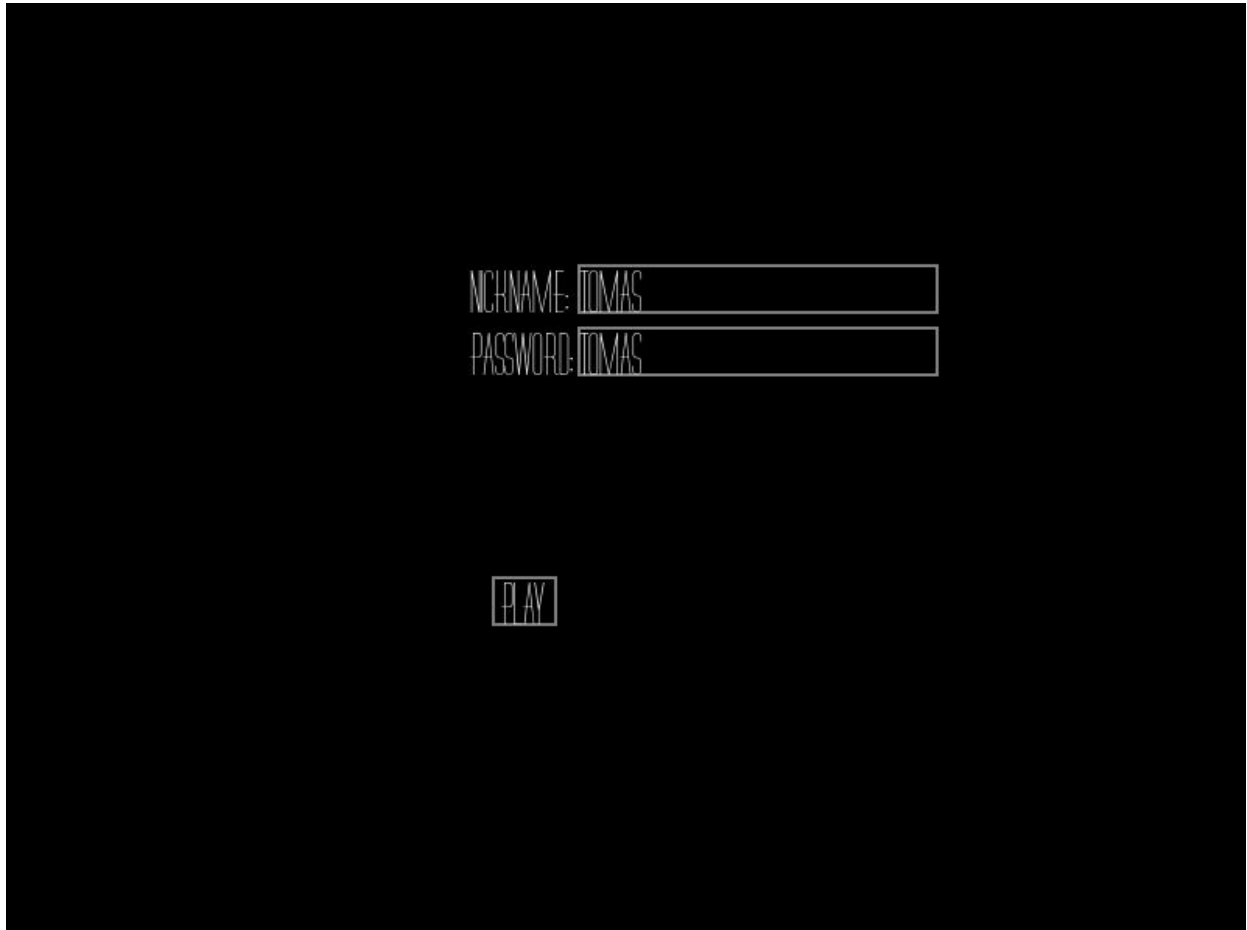
Si pulsamos en el botón de login nos saldrá la opción de loguear un usuario. Nos saldrá este menú donde podemos ingresar nombre y contraseña pulsando en cada uno de sus campos.

A login form is displayed on a solid black background. It consists of two white-outlined rectangular input fields stacked vertically. To the left of the top field is the label 'NICKNAME:' and to the left of the bottom field is the label 'PASSWORD:'. Below these fields is a single button with the word 'PLAY' inside a square frame.

NICKNAME:

PASSWORD:

Una vez que ponemos los datos del usuario podemos darle al play.



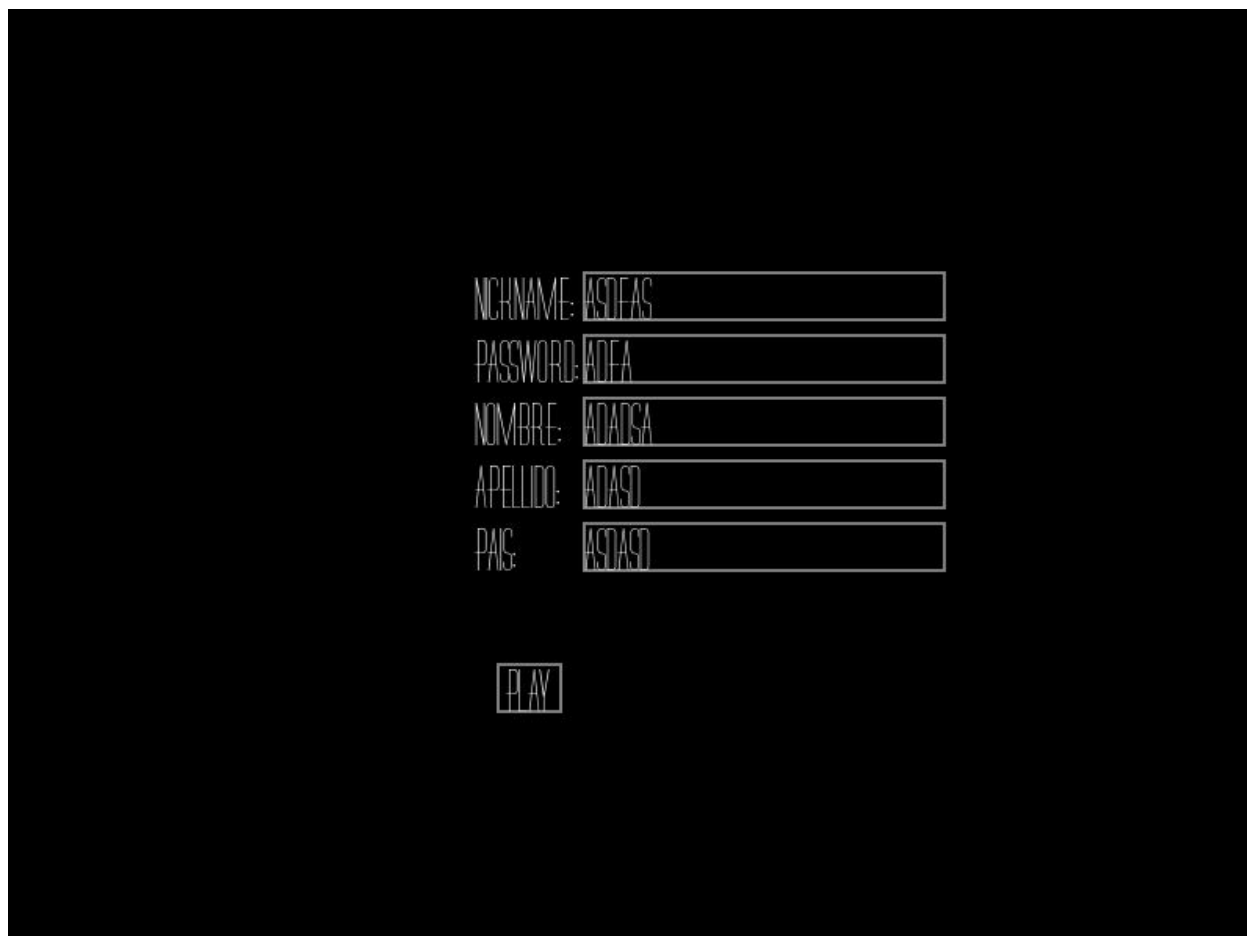
Si en vez de haber dado en la opción de login, le hubiéramos dado a la opción de registro, nos saldría este menú. Al igual que el login podemos rellenar los campos pulsando sobre ellos y escribiendo acto seguido.



Registration form fields:

- NICKNAME:
- PASSWORD:
- NOMBRE:
- APELLIDO:
- PAIS:

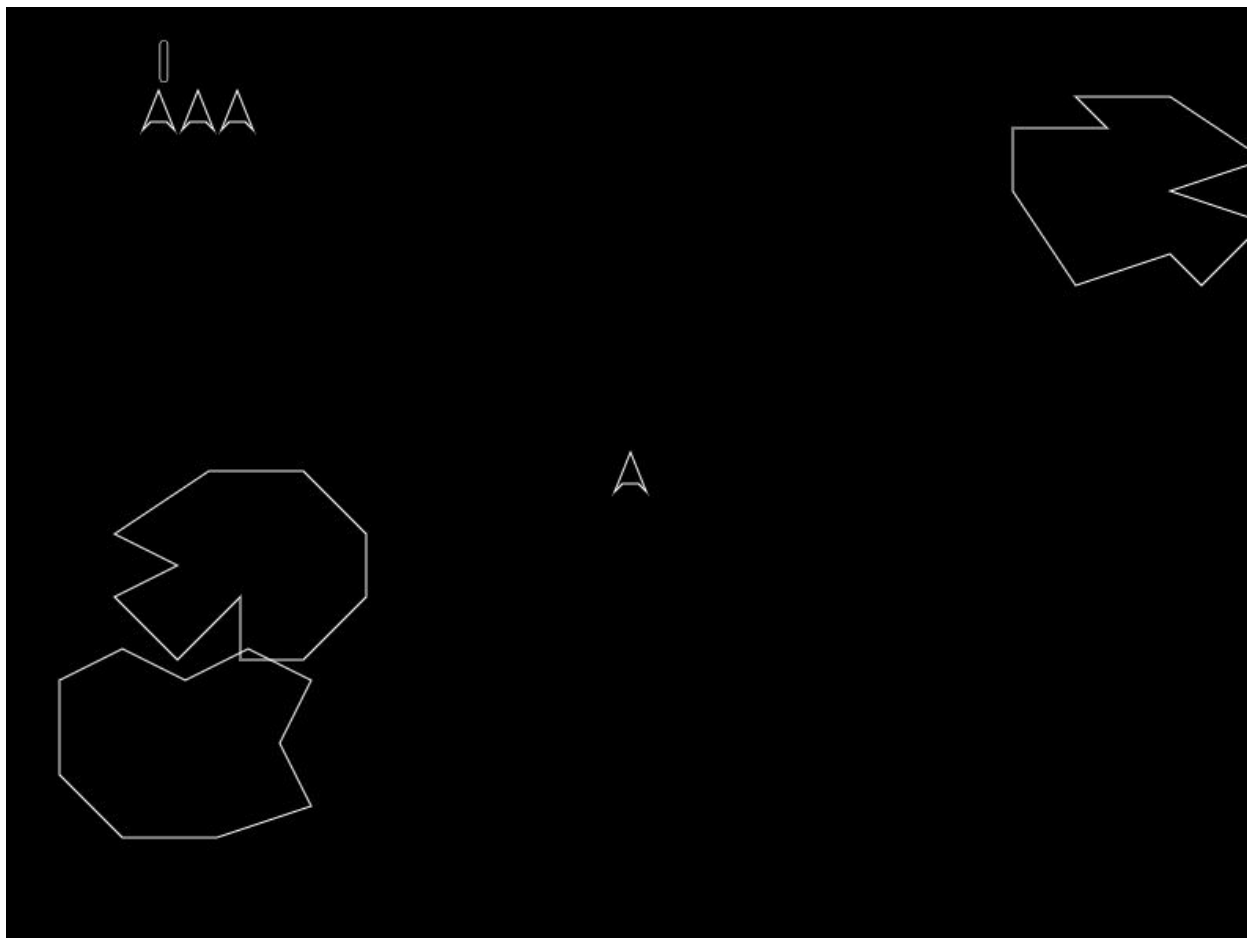
Una vez que tengamos los datos introducidos podemos darle al play.



```
NICKNAME: 
PASSWORD: 
NOMBRE: 
APELLIDO: 
PAIS: 

[PLAY]
```

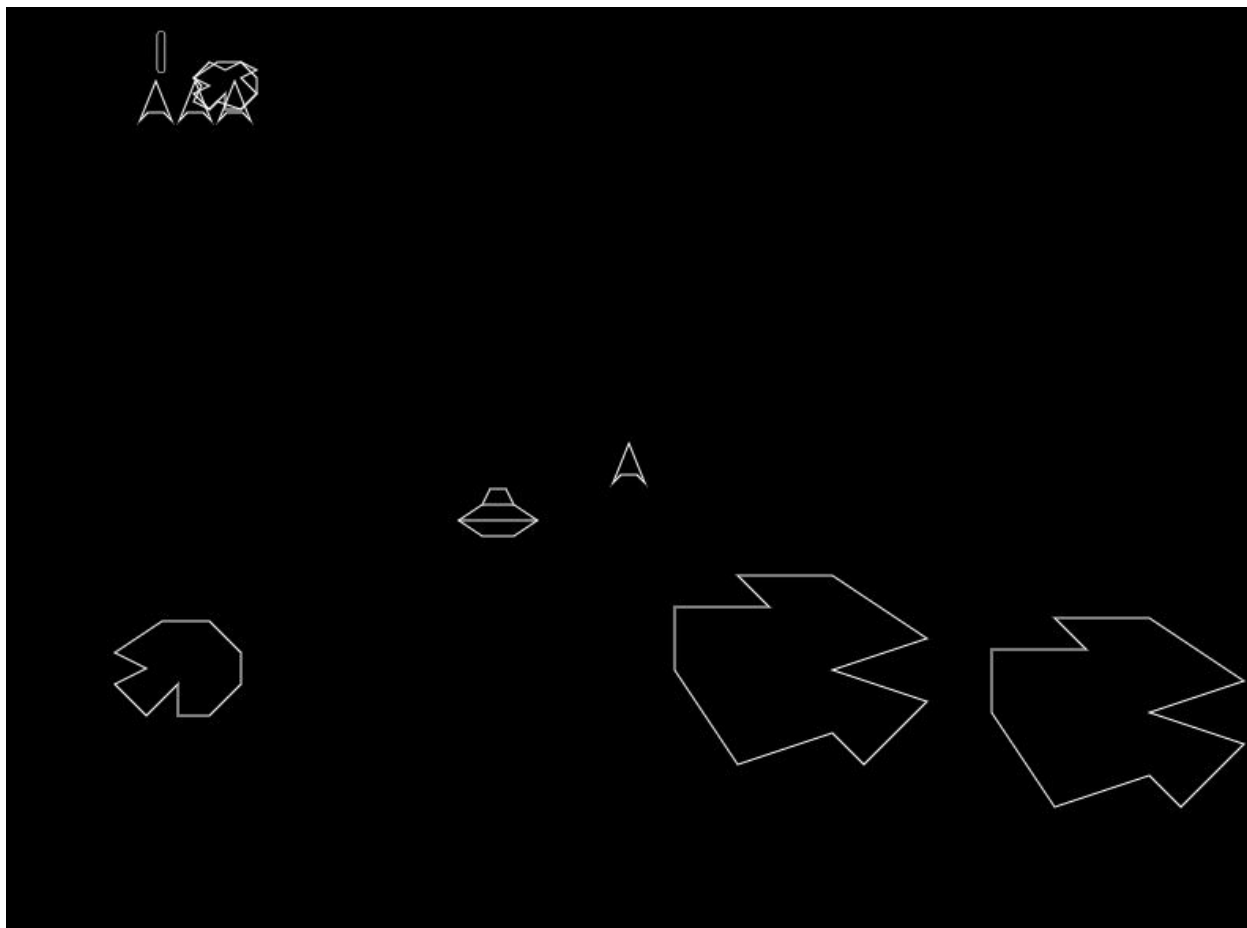
Una vez dentro del juego nos encontraríamos con el siguiente panorama. Como podemos ver tenemos tres asteroides rondando por la pantalla, la nave en el centro y en la parte superior derecha las vidas restantes y la puntuación.



Una vez transcurrida parte de la partida, podemos ver que si colisionamos con los asteroides perdemos una vida y si los destruimos con los disparos vamos ganando puntuación. Además al destruir asteroides estos se dividen en dos asteroides más pequeños.



Eventualmente saldrá en escena la nave enemiga, esta aparecerá por un lado aleatorio de la pantalla y podrá disparar al jugador. Al matar esta nave ganaremos más puntuación. Además esta nave da más puntuación que los asteroides.



Si perdemos todas las vidas significa que hemos perdido. Siendo así nos saldrá este menú, en el que podemos ir al menú principal para cambiar de usuario o jugar de nuevo.



Por el contrario, si destruimos todos los asteroides habremos ganado la partida. Nos saldrá el siguiente menú donde tenemos la puntuación final que hemos obtenido y la opción de ir al menú principal o volver jugar.










Base de Datos

La base de datos que gestiona los usuarios es la siguiente. Como podemos ver tenemos siete columnas, el id del usuario que es una variable numérica que se incrementa mediante vamos agregando usuarios, el nombre de usuario, la contraseña, el nombre, el apellido, el país y su puntuación máxima.

Además decir que el nombre de usuario es clave unica, lo que quiere decir que no pueden haber dos usuarios con el mismo nombre.

Por otro lado la puntuación máxima tiene como valor por defecto cero, para evitar conflictos al momento de actualizar la puntuación máxima.

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate
1	User_Id	INTEGER						NULL
2	Nickname	CHAR (50)						NULL
3	Name	CHAR (50)						NULL
4	Password	CHAR (50)						NULL
5	Country	CHAR (50)						NULL
6	Subname	CHAR (50)						NULL
7	HighScore	INTEGER						0

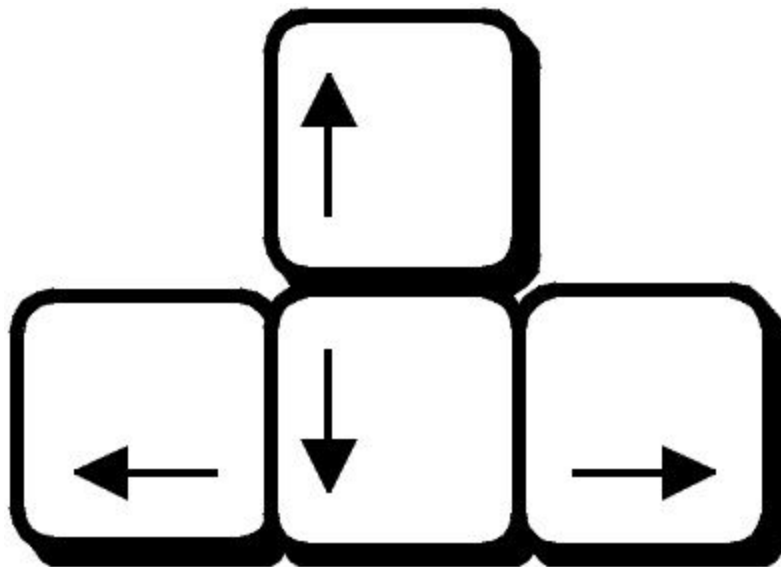
Este es el estado actual de la base de datos, como podemos ver los usuarios que no han ganado ninguna partida tienen la puntuación máxima a 0.

	User_Id	Nickname	Name	Password	Country	Subname	HighScore
1	1	ASDFASDF	ASDFASDF	ADSFASDF	ASDFASDF	ASDFASD	0
2	2	TOMAS	TOMAS	TOMAS	TOMAS	TOMAS	1960
3	3	ASDFASD	ASDFASDF	SDFASDF	ASDFASDF	ASDFASDF	0
4	4	ASDF	ASDFAS	ASDFA	FDSD	ADDAS	0
5	5	ASDAS	ASASAA	ASSAAS	AASASASA	SASAS	1760
6	6	ANAH	HANA	ANA	ESPA;A	GONZALEZ	1710
7	7	PEPE	ADSFA	PEPE	ADASD	ADSFADS	1540
8	8	ASDFAS	ADADSA	ADFA	ASDASD	ADASD	1860

Controles

Los controles son muy simples:

- Movimiento: Teclas direccionales.
 - Rotar: Teclas de izquierda y derecha.
 - Acelerar: Tecla arriba.



- Disparo: Tecla 1.

Instalación

- Extraer el archivo Asteroids.rar en una carpeta.
- Ejecutar asteroids.exe.

Análisis post mortem

Me ha faltado tiempo para completar todo lo que tenía pensado, se que es completamente culpa mía, pero quería que quedara constancia de ello.

Principalmente me ha faltado el sonido y el top 10 de jugadores, que por falta de tiempo y cogerlo todo a última hora los he tenido que dejar en el tintero.

En pocas palabras me ha faltado planificarme mejor y no infravalorar el trabajo que llevaba el proyecto, además, teniendo en cuenta que estuve de viaje un par de semanas por el tema de la boda, tenía que haberle metido más trabajo al principio a las diferentes asignaturas para no llegar tan ahogado al final del curso.