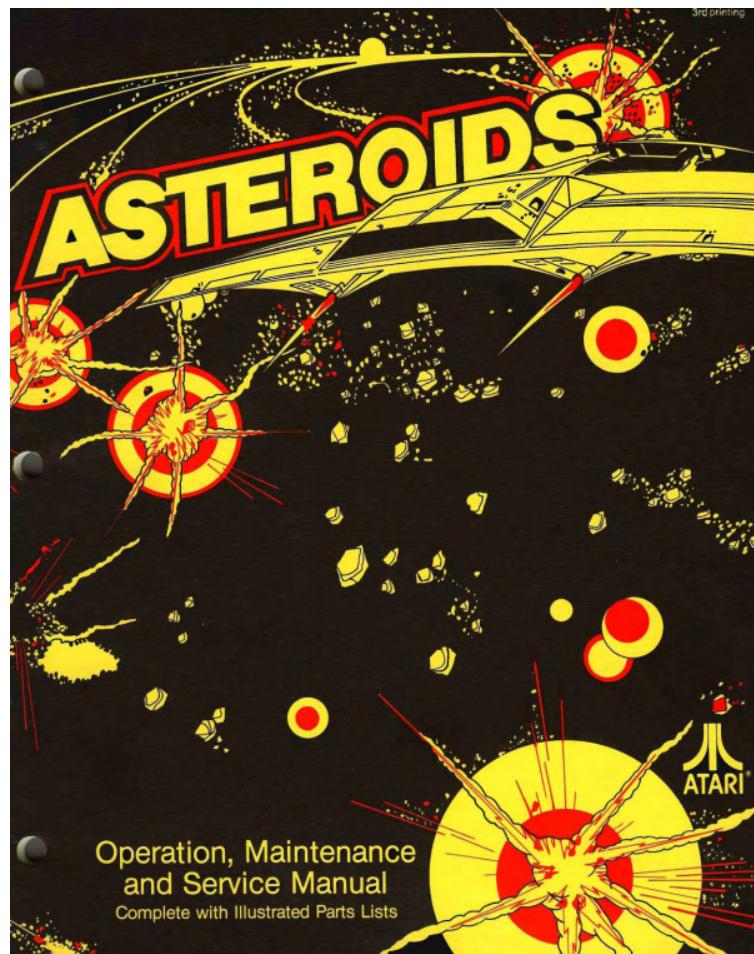


ASTEROIDS
Documentación



Nombre Asignatura: Historia del Videojuego
Módulo Asociado: 1PVG
Profesor: Gustavo Aranda
Curso: 2022/2023
Autor/es: Carlos Mazcuñán Blanes



Índice/Index

1.- Estructuras.....	7
1.1 bool_error_codes.....	7
1.2 TBaseFigure.....	7
1.3 TColor.....	7
1.4 TEntity.....	8
1.5 TFigure.....	8
1.7 THighscore.....	9
1.8 TPlayer.....	9
1.10 TUser.....	10
1.11 TWindowSettings.....	10
2.-Constantes.....	11
2.1 Define.....	11
2.2 Esat::Vec2.....	11
2.3 Enteros.....	11
3.-Variables.....	12
3.1 Bool.....	12
3.2 bool_error_codes.....	12
3.3 Char.....	12
3.4 Double.....	13
3.5 esat::SpriteHandle.....	13
3.6 esat::Vec2.....	13
3.7 File.....	13
3.8 Float.....	14
3.9 Enteros.....	14
3.10 TBaseFigures.....	14
3.11 TEntity.....	15
3.12 TFigureInter.....	15
3.12 THighscore.....	15
3.13 TPlayer.....	15
3.14 TUser.....	15
3.15 WindowSettings.....	15
4.- Procedimientos.....	16
AdminWindow.....	16
AnimationDeadAsteroids.....	16
AsignMemoryUser.....	16
AsteroidScore.....	17
AutoAdmin.....	17
AutoUser.....	17



ChangePlayer.....	17
CheckFile.....	18
CheckHighscoreFile.....	18
CleanAsteroids.....	18
CleanGame.....	19
CleanUpStars.....	19
CleanUser.....	19
CollideOvniAsteroid.....	20
CollideOvniShip.....	20
CollideShipAsteroid.....	20
CollideShootAsteroid.....	21
CollideShootOvni.....	21
CollideShoot_of_Ovni_Ship.....	21
ControlsWindow.....	22
CreditsWindow.....	22
DeleteUser.....	22
DrawAsteroid1.....	22
DrawAsteroid2.....	23
DrawAsteroid3.....	23
DrawAsteroid4.....	23
DrawFondo.....	23
DrawFigure.....	24
DrawLives.....	24
DrawUi.....	24
Edge.....	25
EnterAdminMode.....	25
ExitAdminMode.....	25
Fondo.....	25
Game.....	26
GameOver.....	26
GameOverWindow.....	26
GestionVentanas.....	26
GetPlayerScore.....	27
HyperSpace.....	27
InitBaseAsteroids.....	27
InitBaseParticulas.....	27
InitBaseShoot.....	28
InitCircle.....	28
InitEntities.....	28
InitFigures.....	29



InitFondo.....	29
InitGame.....	29
InitGameAsteroids.....	30
InitGameParticulas.....	30
InitGameShoot.....	30
InitHighscore.....	31
InitMainMenu.....	31
InitPlayer.....	31
InitPointAsteroids.....	32
Inmortality.....	32
LoadScore.....	32
LoadUser.....	33
LoginWindow.....	33
ManageSpawnTimer.....	33
MoveEntities.....	34
MoveFondo.....	34
MoveOvni.....	34
MoveShip.....	34
PauseWindow.....	35
PowerUpLive.....	35
ReadScore.....	35
ReadUserData.....	35
ReleaseFigures.....	36
RotateFigure.....	36
ResetUser.....	36
SaveData.....	36
ScreenText.....	37
ShowHighScore.....	37
SignUpWindow.....	37
SpawnAsteroids.....	37
SpawnBullet.....	38
SpawnOvni.....	38
SpawnOvniBullet.....	38
SpawnParticulas.....	38
Style.....	39
TransformFunction.....	39
UpdateHighScoreFile.....	39
UserWindow.....	39
UpgradeLevel.....	40
WelcomeWindow.....	40



WindowManager.....	40
5.- Funciones.....	41
esat::Vec2 AddVec2.....	41
int AsteroidInScreen.....	41
TEntities *AvailableInPool.....	41
bool DimensionVector.....	42
bool EntityColision.....	42
esat::Vec2 FloatMultVec2.....	42
float vec2 Magnitude.....	43
esat::Vec2 Normalize.....	43
int NumberUsers.....	43
int RankingScore.....	43
esat::Vec2 SolveEqMat2.....	44
esat::Vec2 SubVec2.....	44
esat::Mat3 UpdateBase2.....	44
Int UpdateHighScore.....	44
bool UserExist.....	45
int ValidateLoginData.....	45
int ValidateSignUpData.....	45
6.- Información del juego original.....	46
6.1 Ficha técnica.....	46
6.2 Información hardware.....	46
6.4 Historia del desarrollo.....	46
6.5 Descripción general del juego.....	47
6.6 Curiosidades.....	47
6.7 Tabla de puntuaciones.....	47
7.- Controles.....	48
8.- Gestión de usuarios.....	49
8.1 Pantalla de bienvenida.....	49
8.2 Pantalla inicio de sesión.....	50
8.3 Pantalla de usuario.....	51
8.4 Pantalla registro de usuario.....	52
8.5 Pantallas de créditos y de controles.....	53
8.6 Pantalla de administrador.....	54
9.- Gameplay.....	55
8.1 Antes de jugar.....	55
Selección de modos de juego.....	55
Mejores puntuaciones.....	55
8.3 En el juego.....	56
Modo clásico (Play 1P):.....	56



Modo 2 jugadores (Play 2P).....	58
Modo Hardcore 1 jugador (Hardcore 1P).....	58
Modo Hardcore 2 jugadores (Hardcore 2P).....	59
Ovni.....	59
Ventana de pausa.....	60
Game Over.....	61
Créditos o Monedas.....	61
Guardado de partida.....	61
10.- Guía de instalación.....	62
10.1 Instrucciones.....	62
10.2 Instrucciones en caso de error.....	62
11.- Análisis postmortem.....	63
Bibliografía.....	63



1.- Estructuras

Todas las estructuras están declaradas en common.cc

1.1 bool_error_codes

La estructura `bool_error_codes` está compuesta de variables **booleanas** y gestiona los errores a la hora de verificar los campos de un usuario

<code>wrongpassword</code>	contraseña incorrecta o nula
<code>wrongname</code>	nombre incorrecto o nulo
<code>wrongsurname</code>	apellido incorrecto o nulo
<code>wrongnickname</code>	apodo incorrecto o nulo
<code>wrongcountry</code>	país incorrecto o nulo
<code>wrongprovince</code>	provincia incorrecta o nula
<code>wrongbirthday</code>	cumpleaños incorrecto o nulo
<code>wronggmail</code>	e-mail incorrecto o nulo
<code>wronggg_user</code>	usuario al LogIn incorrecto o nulo
<code>wronggg_password</code>	contraseña al LogIn incorrecta o nula

1.2 TBaseFigure

La estructura `TBaseFigure` está compuesta de un **entero** y una última variable tipo `esat::Vec3*`. Se utiliza para inicializar los puntos de una figura.

<code>esat::Vec3 *points</code>	puntos de la figura
<code>int vertex</code>	vértices de la figura

1.3 TColor

Contiene 4 variables **unsigned char** para controlar los colores (RGBA) de las figuras por pantalla.

<code>r</code>	red
<code>g</code>	green
<code>b</code>	blue
<code>a</code>	alpha



1.4 TEntities

TEntities se encarga de almacenar todos los datos necesarios de las entidades.

bool active	gestión de “vida” de las entidades
bool inmortal	gestión de “inmortalidad” de las entidades
esat::Vec2 dir	dirección
esat::Vec2 vel	velocidad
float rotate_vel	velocidad de rotación
float inmortal_timer	contador de inmortalidad
float respawn_timer	contador de reaparición
TFigure figure	Información para dibujar la figura

1.5 TFigure

Estructura que almacena los datos necesarios para dibujar la figura por pantalla.

esat::Vec2 *tr_points	lista de puntos transformados de la figura
esat::Vec3 *points	lista de puntos de la figura
int vertex	número de vértices de la figura
TColor color	color de la figura
TTransform transform	transformación de la figura

1.6 TFigureInter

Estructura que almacena los datos necesarios para realizar el efecto del fondo de la gestión de usuarios.

esat::Vec2 pos	Posición por pantalla
esat::SpriteHandle star	Sprite
float gravity	Velocidad de caída
float size	Tamaño de cada estrella



1.7 THighscore

Estructura que almacena los datos necesarios para generar la tabla de mejores puntuaciones.

char *user	cadena de caracteres con el nombre del usuario
int highscore	puntuación que entra dentro de la tabla

1.8 TPlayer

Estructura que almacena los datos necesarios para el jugador.

char *score_text	cadena de caracteres que guarda la puntuación para poder dibujarla por pantalla.
int score	puntuación de cada jugador
int n_score_to_up_live	puntuación necesario para aumentar 1 vida
int lives	vidas de cada jugador

1.9 TTransform

Estructura que almacena los datos necesarios para transformar las figuras

esat::Mat3 transform_matrix	matriz de transformación de cada figura
esat::Vec2 scale	escalado de cada figura
esat::Vec2 origin	origen de cada figura
esat::Vec2 pos	posición en pantalla
float origin_rotate	origen de rotación de cada figura
float rotate	rotación en radianes de cada figura



1.10 TUser

TUser almacena los datos necesarios para la creación de un usuario.

char *name	nombre del usuario
char *nickname	apodo del usuario
char *surname	apellido del usuario
char *password	contraseña del usuario
char *repeatpassword	validación de contraseña del usuario
char *country	país del usuario
char *province	provincia del usuario
char *mail	e-mail del usuario
int credits	créditos de juego del usuario
int *birthday	fecha de nacimiento del usuario
int id	señalizador del usuario
int score	puntuación del usuario
int highscore	mejor puntuación del usuario

1.11 TWindowSettings

Estructura que almacena variables booleanas para la gestión de ventanas.

bool welcome	ventana de bienvenida
bool login	ventana de inicio de sesión
bool signup	ventana de registro de usuario
bool ctrls	ventana de controles
bool crdts	ventana de créditos
bool game	estado de juego
bool usr	ventana con información de usuario
bool admin	ventana de administrador
bool admin_signup	ventana de creación de usuario en modo administrador
bool edit_user_for_admin	ventana de editar usuario en modo administrador
bool game_over	ventana de juego finalizado



2.-Constantes

Todas las constantes están declaradas en common.cc

2.1 Define

delta	conversión de frames a segundos
--------------	---------------------------------

2.2 Esat::Vec2

Esat::Vec2 está compuesto por {float x, float y};

kBigAsteroid	{40.0f, 40.0f}; Escala del asteroide grande
kMidAsteroid	{25.0f, 25.0f}; Escala del asteroide mediano
kLittleAsteroid	{10.0f, 10.0f}; Escala del asteroide pequeño
kLittleOvni	{10.0f, 10.0f}; Escala del Ovni pequeño
kBigOvni	{25.0f, 25.0f}; Escala del Ovni grande

2.3 Enteros

kUserSize	190; bits que ocupa un TUser en el fichero
kHighScoreSize	68; bits que ocupa un THighscore en el fichero
kNAsteroidBase	4; Número de diferentes asteroides que hay en el juego
kAsteroidPool	50; Reserva de memoria para los asteroides
kShootPool	30; Reserva de memoria para las balas
kParticulasPool	200; Reserva de memoria para las partículas
knOvni	1; Número de ovnis por pantalla
n_players	2; Número de jugadores
kNEntities	1 + knOvni + kAsteroidPool + kShootPool + kParticulasPool; Número de entidades
knVertexCircle	60; constante para calcular un círculo
n_figures	200; número de estrellas para el efecto del fondo



3.-Variables

Todas las variables están declaradas en common.cc

3.1 Bool

viewpass	controla que se vea o no la contraseña al iniciar sesión o registrarse
check_file_admin_window	comprueba el fichero para el modo administrador
multiplayer	gestiona los modos 1 jugador o 2 jugadores
pause	gestiona la ventana de pausa del juego (input 'P')
exit_game	gestiona el cierre del programa (botón "Exit")
bool_hyperspace	gestiona el tiempo que el player es inmortal cuando se ha teletransportado
hardcore	gestiona los modos de juego "hardcore"

3.2 bool_error_codes

e_codes	códigos de error de inicio y registro de usuario.
----------------	---

3.3 Char

*g_user	cadena de caracteres que se utiliza para comprobar el usuario en el inicio de sesión
*g_password	cadena de caracteres que se utiliza para comprobar la contraseña en el inicio de sesión
*P1	cadena de caracteres que dibuja por pantalla el jugador que está jugando en ese momento (Player 1)
*P2	cadena de caracteres que dibuja por pantalla el jugador que está jugando en ese momento (Player 2)
*copyright	cadena de caracteres que muestra por pantalla: "1979 ATARI"



3.4 Double

g_last_frame	control de fps, esta variable se iguala a la función esat::Time();
---------------------	--

3.5 esat::SpriteHandle

keys	hoja de sprites que contiene las teclas de input
arrows	sprite para las flechas izquierda y derecha del teclado
spacebar	sprite para la barra espaciadora del teclado
g_key	sprite para la tecla 'G'
up_key	sprite para la flecha hacia arriba del teclado
esat_logo	sprite del logo de ESAT
ship_lives	sprite con forma de la nave para indicar las vidas de la misma
white	hoja en blanco
logo	sprite logo Asteroids

3.6 esat::Vec2

Esat::Vec2 está compuesto por {float x, float y};

*circle	puntos para calcular un círculo
----------------	---------------------------------

3.7 File

*regist	fichero que almacena los datos de todos los usuarios
*highscore_file	fichero que almacena los datos de las 10 mejores puntuaciones



3.8 Float

counter_bullet_ovni	{0.0f} contador para la aparición de la bala del ovni
counter_movement	{0.0f} contador para gestionar el movimiento del ovni
counter_ovni	{0.0f} contador para gestionar la aparición del ovni
d	distancia entre cada vértice de la circunferencia

3.9 Eteros

asteroids_screen	asteroides que hay en ese momento por pantalla
cont_init_asteroids	contador de asteroides que deben de aparecer en cada nivel
current_player	para gestionar cuál de los dos jugadores está jugando en ese momento
level	nivel del juego
user_id_for_admin	identificador de cada usuario para que se puedan gestionar correctamente desde el modo administrador
swap	variable que gestiona la situación del programa, utilizándose en un "switch"

3.10 TBaseFigures

*base_asteroids	puntero (kAsteroidBase) de los datos necesarios para dibujar los 4 tipos de asteroides
base_ovni	datos necesarios para dibujar el ovni
*base_particulas	puntero (2) de los datos necesarios para dibujar los dos tipos de partículas que
base_shoot	datos necesarios para dibujar las balas



3.11 TEntities

*game_asteroids	puntero (kAsteroidPool) de todos los asteroides que aparecen por pantalla
**global_entities	puntero (kNEntities) de punteros de todas las posibles entidades que aparecen por pantalla
ovni	entidad del ovni
*particulas	puntero (kParticulasPool) de todas las posibles partículas que aparecen por pantalla
ship	entidad de la nave que maneja el jugador
*shoots	puntero (kShootPool) de todos los posibles disparos que pueden aparecer por pantalla

3.12 TFigureInter

*figures	puntero (200) de todas las “estrellas” que aparecen en el efecto del fondo de la gestión de usuarios
-----------------	--

3.12 THighscore

aux_hs	variable auxiliar para leer las mejores puntuaciones del fichero “highscore_file”
---------------	---

3.13 TPlayer

*player	puntero (n_players) que gestiona ambos jugadores posibles del juego
----------------	---

3.14 TUser

user	TUser que inicia sesión
*aux_user	Tuser que se utiliza de auxiliar para muchas funciones

3.15 WindowSettings

config	gestiona todas las ventanas del menú principal
---------------	--



4.- Procedimientos

AdminWindow	—
Declaración	main2.cc
Llamada	Durante la gestión de usuarios cuando entras en el modo administrador
Descripción	Ventana administrador

AnimationDeadAsteroids	TTransform tr int n_asteroids_to_spawn float dead_scale
Declaración	Ship2.cc
Llamada	Cada vez que hay un asteroide colisiona con la nave, el ovni o algún disparo
Descripción	Hace aparecer varios asteroides de una escala menor.

AsignMemoryUser	TUser *u
Declaración	main2.cc
Llamada	Al inicializar los datos para la gestión de usuarios
Descripción	Asignar memoria para los usuarios



AsteroidScore	TPlayer *player int asteroid_type
Declaración	ship2.cc
Llamada	Cada vez que el jugador dispara a un asteroide
Descripción	Los puntos que recibe el jugador por derribar cada asteroide

AutoAdmin	—
Declaración	main2.cc
Llamada	En la ventana de inicio de sesión
Descripción	Con la tecla “Alt” se rellenan los campos automáticamente para iniciar sesión en modo administrador.

AutoUser	—
Declaración	main2.cc
Llamada	En la ventana de inicio de sesión
Descripción	Con la tecla “Shift” se rellenan los campos automáticamente para iniciar sesión con un usuario

ChangePlayer	—
Declaración	ship2.cc
Llamada	Cada vez que un jugador pierde una vida con una colisión entre la nave y las entidades con las que colisiona.
Descripción	Cambia de un jugador a otro en los modos “multiplayer”



CheckFile	—
Declaración	main2.cc
Llamada	Al inicializar todos los datos necesarios para la gestión de usuarios
Descripción	Comprueba que existe el fichero de registros, y si no es así crea uno y lo abre en modo “wb+”

CheckHighscoreFile	—
Declaración	highscore.cc
Llamada	cada vez que actualizamos el fichero “highscores”
Descripción	Comprueba que existe el fichero de mejores puntuaciones y si no es así crea uno y lo abre en modo “wb+”

CleanAsteroids	—
Declaración	Draw_asteroids.cc
Llamada	Al liberar la memoria de los datos necesarios de la interfaz
Descripción	Libera los puntos de los asteroides decorativos dibujados en la pantalla de bienvenida



CleanGame	—
Declaración	main2.cc
Llamada	Al cerrar la parte del juego
Descripción	Libera memoria necesaria para el juego y le asigna el valor '0' a curren_player

CleanUpStars	—
Declaración	FondoInterfaz.cc
Llamada	En el momento que se libera toda la memoria de la gestión de usuarios
Descripción	Libera memoria de los elementos necesarios para generar el efecto del fondo

CleanUser	TUser *u
Declaración	main2.cc
Llamada	En el momento que se libera toda la memoria de la gestión de usuarios
Descripción	Libera toda la memoria que se había asignado en el procedimiento "AsignMemoryUser"



CollideOvniAsteroid	TEntities *ov TEntities *as int n_asteroids
Declaración	ship2.cc
Llamada	En el bucle de juego
Descripción	Si tanto el ovni como el asteroide están activos y detecta una colisión, hace las funciones competentes para dicha colisión

CollideOvniShip	TEntities *ov TEntities *sh
Declaración	ship2.cc
Llamada	En el bucle de juego
Descripción	Si tanto el ovni como la nave están activos y detecta una colisión, hace las funciones competentes para dicha colisión

CollideShipAsteroid	TEntities *pool_1, int n_entities1 TEntities *pool_2, int n_entities2 TPlayer *player
Declaración	ship2.cc
Llamada	En el bucle de juego
Descripción	Si tanto el asteroide como la nave están activos y detecta una colisión, hace las funciones competentes para dicha colisión



CollideShootAsteroid	TEntities *pool_1, int n_entities1 TEntities *pool_2, int n_entities2
Declaración	ship2.cc
Llamada	En el bucle de juego
Descripción	Si tanto el asteroide como el disparo están activos y detecta una colisión, hace las funciones competentes para dicha colisión

CollideShootOvni	TEntities *bullet int n_bullet TEntities *ov
Declaración	ship2.cc
Llamada	En el bucle de juego
Descripción	Si tanto el ovni como el disparo de la nave están activos y detecta una colisión, hace las funciones competentes para dicha colisión

CollideShoot_of_Ovni_Ship	TEntities *bullet TEntities *ship
Declaración	ship2.cc
Llamada	En el bucle de juego
Descripción	Si tanto el disparo del ovni como la nave están activos y detecta una colisión, hace las funciones competentes para dicha colisión



ControlsWindow	—
Declaración	main2.cc
Llamada	En la ventana de bienvenida al pulsar el botón “controls”
Descripción	Una pequeña ventana la cual indica los controles del juego

CreditsWindow	—
Declaración	main2.cc
Llamada	En la ventana de bienvenida al pulsar el botón “credits”
Descripción	Una pequeña ventana la cual indica los créditos del programa

DeleteUser	int aux_id
Declaración	main2..c
Llamada	Desde el modo administrador de la gestión de usuarios al darle al botón “delete”
Descripción	Elimina el usuario seleccionado

DrawAsteroid1	esat::Mat3 m int n_points
Declaración	Draw_asteroids.cc
Llamada	En la ventana de bienvenida
Descripción	Dibuja un tipo de asteroide, decorativos al lado de cada botón del menú de bienvenida



DrawAsteroid2	esat::Mat3 m int n_points
Declaración	Draw_asteroids.cc
Llamada	En la ventana de bienvenida
Descripción	Dibuja un tipo de asteroide, decorativos al lado de cada botón del menú de bienvenida

DrawAsteroid3	esat::Mat3 m int n_points
Declaración	Draw_asteroids.cc
Llamada	En la ventana de bienvenida
Descripción	Dibuja un tipo de asteroide, decorativos al lado de cada botón del menú de bienvenida

DrawAsteroid4	esat::Mat3 m int n_points
Declaración	Draw_asteroids.cc
Llamada	En la ventana de bienvenida
Descripción	Dibuja un tipo de asteroide, decorativos al lado de cada botón del menú de bienvenida

DrawFondo	—
Declaración	FondoInterfaz.cc
Llamada	Dentro del procedimiento “Fondo”
Descripción	Dibuja las “estrellas” del efecto del fondo



DrawFigure	TFigure figure
Declaración	ship2.cc
Llamada	En el bucle de juego, dentro del procedimiento llamado “Game”
Descripción	Dibuja las figuras que se le pase como parámetro

DrawLives	TPlayer *p float pos_x
Declaración	ship2.cc
Llamada	Dentro del procedimiento “DrawUi”
Descripción	Dibuja por pantalla las vidas de cada usuario con el ícono de la nave del juego

DrawUi	—
Declaración	ship2.cc
Llamada	En el bucle de juego, dentro del procedimiento llamado “Game”
Descripción	Dibuja los puntos y las vidas de los jugadores que estén jugando



Edge	esat::Vec2 *pos
Declaración	common.cc
Llamada	En el bucle de juego, dentro del procedimiento llamado “Game”
Descripción	Controlar los bordes de la pantalla para cuando una entidad sobrepase uno, aparezca en el borde contrario

EnterAdminMode	—
Declaración	main2.cc
Llamada	Cuando entramos en el modo administrador
Descripción	Hacemos los preparativos competentes para entrar en el modo administrador.

ExitAdminMode	—
Declaración	main2.cc
Llamada	Al salir del modo administrador
Descripción	Hacemos los preparativos competentes para salir del modo administrador

Fondo	—
Declaración	FondoInterfaz.cc
Llamada	Durante la gestión de usuarios
Descripción	Dibuja y le da físicas a las “estrellas” del efecto del fondo en la gestión de usuarios



Game	—
Declaración	main2.cc
Llamada	Al momento de iniciar la parte del juego, en un switch en el main
Descripción	El bucle del mismo juego

GameOver	—
Declaración	main2.cc
Llamada	En el procedimiento “Game”
Descripción	Si los jugadores se quedan sin vidas, te lleva a la ventana de Game Over

GameOverWindow	—
Declaración	main2.cc
Llamada	Cuando el jugador se queda sin vidas
Descripción	Ventana de “Fin de Juego”

GestionVentanas	—
Declaración	main2.cc
Llamada	En un switch del main
Descripción	Es la función que gestiona todas las ventanas de la gestión de usuarios



GetPlayerScore	—
Declaración	highscore.cc
Llamada	Al terminar una partida
Descripción	Guarda la puntuación que ha realizado el jugador al acabar la partida

HyperSpace	<code>esat::Vec2 *position</code>
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Gestiona la mecánica de juego de teletransportarse

InitBaseAsteroids	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Inicializa todos los puntos y vértices que son necesarios para dibujar los cuatro tipos de asteroides

InitBaseParticulas	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Inicializa todos los puntos y vértices que son necesarios para dibujar las partículas de la animación de destrucción de cualquier entidad



InitBaseShoot	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Inicializa todos los puntos y vértices que son necesarios para dibujar los disparos

InitCircle	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Se calcula la circunferencia para generar la animación de destrucción de las entidades

InitEntities	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Da memoria a “**global_entities” y se asigna todas las entidades



InitFigures	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Se inicializan los datos necesarios de las entidades de la nave y el ovni, y se llama a los procedimientos “InitGameAsteroids”, “InitGameShoots” e “InitGameParticulas”

InitFondo	—
Declaración	FondoInterfaz.cc
Llamada	Al inicializar los datos necesarios para la parte de gestión de usuarios
Descripción	Inicializa los datos necesarios para crear el efecto del fondo

InitGame	—
Declaración	main2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Engloba los procedimientos necesarios para inicializar el juego y se le asignan valor a algunas variables necesarias.



InitGameAsteroids	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Inicializa los datos necesarios para la transformación de los asteroides y para sus físicas

InitGameParticulas	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Inicializa los datos necesarios para la transformación de las partículas de la animación de destrucción y para sus físicas

InitGameShoot	—
Declaración	ship2.cc
Llamada	Al inicializar todos los datos necesarios de la parte del juego
Descripción	Inicializa los datos necesarios para la transformación de las balas y para sus físicas



InitHighscore	THighscore *hs
Declaración	highscore.cc
Llamada	En cualquier momento que se le necesite dar memoria a una variable THighscore
Descripción	Da memoria a la cadena de dentro de la estructuras THighscore e inicializa la variable entera a 0

InitMainMenu	—
Declaración	main2.cc
Llamada	En el momento que sea necesario inicializar los datos para la parte de gestión de usuario y el menú principal
Descripción	Engloba todas las funciones de inicialización de datos de la parte de gestión de usuarios e individualmente también inicializa otros datos necesarios

InitPlayer	—
Declaración	ship2.cc
Llamada	Al inicializar los datos necesarios para la parte del juego
Descripción	Inicializa los datos necesarios para cada uno de los jugadores



InitPointAsteroids	—
Declaración	Draw_asteroids.cc
Llamada	Al inicializar los datos necesarios para el menú principal y la parte de gestión de usuarios
Descripción	Inicializa los puntos necesarios de cada uno de los tipos de asteroides que se dibujan en el menú principal

Inmortality	TEntities *entity float segs
Declaración	ship2.cc
Llamada	Cada vez que la nave pierde una vida o se teletransporta
Descripción	Hace inmune a la nave durante unos pocos segundos

LoadScore	THighscore hs
Declaración	highscore.cc
Llamada	En el momento que sea necesario escribir los datos competentes de las variables THighscore
Descripción	Escribe los datos de las variables THighscore en el fichero



LoadUser	int aux_id
Declaración	main2.cc
Llamada	Cuando se edita un usuario en el modo administrador
Descripción	Carga los datos del usuario que se haya seleccionado en una variable TUser

LoginWindow	—
Declaración	main2.cc
Llamada	Cuando se pulsa en el botón “Login” de la ventana de bienvenida
Descripción	Ventana de inicio de sesión para poder jugar

ManageSpawnTimer	TEntities *entity float segs int mode
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Gestiona el tiempo en pantalla de algunas entidades, como los disparos o las partículas. También el tiempo que tardará la nave en reaparecer al perder una vida



MoveEntities	<code>&(((*(*global_entities + i))))</code>
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Gestiona el movimiento de todas las entidades del juego

MoveFondo	—
Declaración	FondoInterfaz.cc
Llamada	En el procedimiento “Fondo”
Descripción	Le da físicas a las “estrellas” del efecto del fondo

MoveOvni	TEntities *aux
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Gestiona el movimiento de los ovnis

MoveShip	TEntities *entitie
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Gestiona el movimiento de la nave con sus respectivas físicas



PauseWindow	—
Declaración	main2.cc
Llamada	En el procedimiento “Game”
Descripción	Acciona una ventana de pausa del juego

PowerUpLive	—
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Le incrementa una vida al jugador cuando supera el umbral de 10.000 puntos

ReadScore	THighscore *hs
Declaración	highscore.cc
Llamada	En el momento que sea necesario leer los datos competentes del fichero de mejores puntuaciones
Descripción	Lee los datos de las variables THighscore en el fichero

ReadUserData	TUser *u
Declaración	main2.cc
Llamada	En los momentos que es necesario leer los datos de un usuario del fichero de usuarios
Descripción	Lee los datos de las variables TUser del fichero de usuarios



ReleaseFigures	—
Declaración	ship2.cc
Llamada	En el momento que salimos de la parte del juego
Descripción	Libera la memoria de los datos necesarios de la parte del juego

RotateFigure	TEntities *entitie
Declaración	ship2.cc
Llamada	Dentro del procedimiento “Game”
Descripción	Rota la nave

ResetUser	TUser *user1
Declaración	main2.cc
Llamada	Cuando le damos al botón “clear” o “return” en la ventana de creación de un usuario
Descripción	Limpia las cadenas de texto

SaveData	TUser *u
Declaración	main2.cc
Llamada	Al momento de crear o eliminar un usuario
Descripción	Guarda en el fichero de usuarios los datos de un usuario



ScreenText	TPlayer *player, float pos_x
Declaración	ship2.cc
Llamada	Dentro del procedimiento “DrawUI”
Descripción	Dibuja por pantalla las puntuaciones de cada jugador, el jugador que está jugando en ese momento y la fecha y palabra que salen abajo en el centro mientras juegas

ShowHighScore	—
Declaración	highscore
Llamada	Al iniciar sesión y darle al botón “Play”;
Descripción	Muestra por pantalla las 10 mejores puntuaciones que estén guardadas en el fichero

SignUpWindow	—
Declaración	main2.cc
Llamada	Al darle al botón, de la ventana de bienvenida, “SignUp” o al de “Edit” o “Create” del modo administrador
Descripción	Ventana de dada de alta de usuario

SpawnAsteroids	TTransform tr
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Hace aparecer un asteroide



SpawnBullet	TTransform tr
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Hace aparecer una bala de la nave

SpawnOvni	—
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Hace aparecer al Ovni

SpawnOvniBullet	TTransform tr
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Hace aparecer una bala del Ovni

SpawnParticulas	TTransform tr
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Hace aparecer las partículas de la animación de destrucción



Style	—
Declaración	main2.cc
Llamada	Al iniciar el programa
Descripción	Setea los colores de los botones y ventanas de ImGui

TransformFunction	TFigure *figure
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Transforma mediante la matriz básica de transformación todas las figuras que se dibujan por pantalla

UpdateHighScoreFile	THighscore *hs
Declaración	highscore.cc
Llamada	Al guardar los datos de las variables de tipo THighscore
Descripción	Actualiza el fichero de mejores puntuaciones

UserWindow	—
Declaración	main2.cc
Llamada	Al iniciar sesión con cualquier usuario
Descripción	Ventana con la información del usuario que haya iniciado sesión y con los botones necesarios para seleccionar cada modo de juego



UpgradeLevel	TPlayer *p
Declaración	ship2.cc
Llamada	En el procedimiento “Game”
Descripción	Aumenta el nivel cuando no hay asteroides por pantalla, al hacer esto se incrementará el número de asteroides aparecidos

WelcomeWindow	—
Declaración	main2.cc
Llamada	Al entrar en el bucle de la ventana gráfica
Descripción	Ventana de bienvenida del juego con los botones necesarios de navegación

WindowManager	int n
Declaración	main2.cc
Llamada	Cada vez que se cambia de ventana en la parte de gestión de usuarios y menú principal
Descripción	Contiene un switch que gestiona una estructura de booleanas las cuales gestionan qué ventana está abierta en cada momento, para que no se superpongan la una a la otra



5.- Funciones

esat::Vec2 AddVec2	esat::Vec2 A esat::Vec2 B
Declaración	common.cc
Llamada	Cada vez que se necesite sumar dos variables tipo esat::Vec2
Descripción	Suma dos variables tipo esat::Vec2

int AsteroidInScreen	—
Declaración	ship2.cc
Llamada	En el procedimiento “UpgradeLevel”
Descripción	Comprueba que no haya asteroides en pantalla para subir de nivel

TEntities *AvailableInPool	TEntities *pool const int pool_size
Declaración	ship2.cc
Llamada	En todo momento que necesitemos comprobar si hay memoria suficiente
Descripción	Comprueba si hay memoria suficiente de la que hemos reservado al principio de la partida, en caso de que la haya devuelve un TEntities



bool DimensionVector	esat::Vec2 p1_1, esat::Vec2 p1_2, esat::Vec2 p2_1, esat::Vec2 p2_2, float offset_col = 0.0f
Declaración	ship2.cc
Llamada	En la función “EntityColision”
Descripción	Calcula la distancia entre dos rectas, si se cumple la condición devuelve “verdadero”

bool EntityColision	TEntities entity1, TEntities entity2, float offset_col = 0.0f
Declaración	ship2.cc
Llamada	Cada vez que se quiere comprobar una colisión entre diferentes entidades
Descripción	Comprueba si hay colisión, su la hay devuelve “verdadero”

esat::Vec2 FloatMultVec2	esat::Vec2 A, float B
Declaración	common.cc
Llamada	Cada vez que se necesite hacer dicha operación
Descripción	Multiplica un vector por un escalar



float vec2 Magnitude	esat::Vec2 v
Declaración	common.cc
Llamada	Cada vez que se necesite hacer dicha operación
Descripción	Calcula el módulo de un vector

esat::Vec2 Normalize	esat::Vec2 v
Declaración	common.cc
Llamada	Cada vez que se necesite hacer dicha operación
Descripción	Normaliza un Vector

int NumberUsers	—
Declaración	main2.cc
Llamada	Cada vez que se necesite saber el número de usuarios
Descripción	Calcula el número de usuarios dados de alta en el fichero

int RankingScore	—
Declaración	highscore.cc
Llamada	Cada vez que se necesite saber el número THighscore
Descripción	Calcula el número de THighscore que hay en el fichero



esat::Vec2 SolveEqMat2	esat::Vec3 first_eq, esat::Vec3 second_ec
Declaración	common2.cc
Llamada	En la función “DimensionVector”
Descripción	Resuelve una ecuación de segundo grado

esat::Vec2 SubVec2	esat::Vec2 A, esat::Vec2 B
Declaración	common.cc
Llamada	Cada vez que se quiera realizar esta operación
Descripción	Resta dos vectores

esat::Mat3 UpdateBase2	float x float y
Declaración	Draw_asteroids.cc
Llamada	Se le pasa por parámetro a las funciones DrawAsteroids[1-4]
Descripción	Para transformar los asteroides que se dibujan en el menú principal

Int UpdateHighScore	THighscore *hs
Declaración	highscore.cc
Llamada	Al recoger los datos de puntuación del jugador
Descripción	Actualiza el fichero de mejores puntuaciones



bool UserExist	char *us char *pass
Declaración	main2.cc
Llamada	Al presionar el botón de iniciar sesión
Descripción	Si no coincide el usuario con alguno que esté guardado en el fichero devolverá “false” y no se podrá iniciar sesión

int ValidateLoginData	char *user char *pass
Declaración	main2.cc
Llamada	Al presionar el botón de iniciar sesión
Descripción	Comprueba que se hayan llenado los campos antes de iniciar sesión

int ValidateSignUpData	TUser user
Declaración	main2.cc
Llamada	Al presionar el botón de “Submit” en la ventana de “SignUp”
Descripción	Comprueba que se hayan llenado todos los campos necesarios para dar de alta un usuario



6.- Información del juego original

6.1 Ficha técnica

Nombre del juego	Asteroids
Fabricante	Atari Inc.
Año desarrollo	1979
Categoría	Shooter

6.2 Información hardware

- > Utiliza un monitor vectorial monocromo.
- > Procesador genérico MOS 6502 @ 1.5Mhz, que utiliza un generador digital de vectores (DVG) con el que comparte 2Kb de RAM.
- > Código almacenado en ROM de 6Kb.
- > 2Kb adicionales de ROM para datos de los vectores de los gráficos.
- > 1Kb adicional de RAM para los datos durante la ejecución.

Desarrolladores principales: **Lyle Rains y Ed Logg**

Compositores de la música: **Howard Delman (circuitería para los sonidos del juego)**

6.4 Historia del desarrollo

Siguiendo la estela de la temática del espacio en los videojuegos, Atari decide desarrollar un juego mezcla de los clásicos Computer Space y Space Invaders. Decide cambiar la dinámica tradicional del juego, permitiendo al jugador mover su nave por toda la pantalla. Se seleccionó un monitor vectorial porque permitía más resolución que los monitores de barrido de la época y permitía más precisión a la hora de apuntar a los objetivos.

El juego fue un éxito y vendió más de 70.000 unidades, generando un beneficio para Atari de unos 150 millones de dólares por la venta de las máquinas y a los operadores unos 500 millones de dólares por la recaudación de los jugadores.



6.5 Descripción general del juego

Nos encontramos a los mandos de nuestra nave espacial en medio de un sin fin de asteroides que deberemos esquivar o destruir para evitar estrellarnos contra ellas. Al destruir un asteroide, éste se partirá en trozos más pequeños, incrementando el número de rocas a evitar y haciendo que destruir las rocas más pequeñas sea más difícil. Para hacer más interesante aún nuestra aventura, aparecerán también de vez en cuando naves enemigas que dispararán contra nosotros.

Los controles de la nave están compuestos únicamente por botones, para controlar la rotación a derecha/izquierda, el empuje, disparo y la posibilidad de saltar a una zona aleatoria de la pantalla (hiperespacio).

6.6 Curiosidades

Cada 10.000 puntos el jugador recibe una vida extra, que aparece dibujada como un triangulito en la parte superior de la pantalla. Si el jugador es lo suficientemente hábil como para ir acumulando vidas, el juego se acaba ralentizando debido al tiempo de redibujado de todos los elementos de la pantalla.

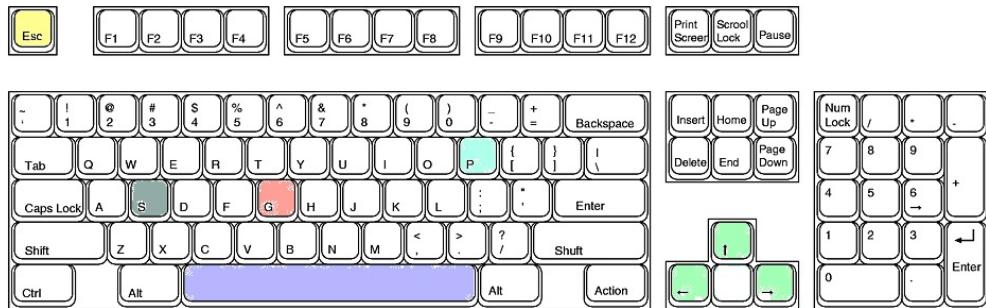
Es uno de los primeros juegos en incorporar elementos de física, como la inercia en la acción.

6.7 Tabla de puntuaciones

Asteroide Grande	20pts.
Asteroide Mediano	50pts.
Asteroide Pequeño	100pts.
Ovni Grande	200pts.
Ovni Pequeño	1000pts.



7.- Controles



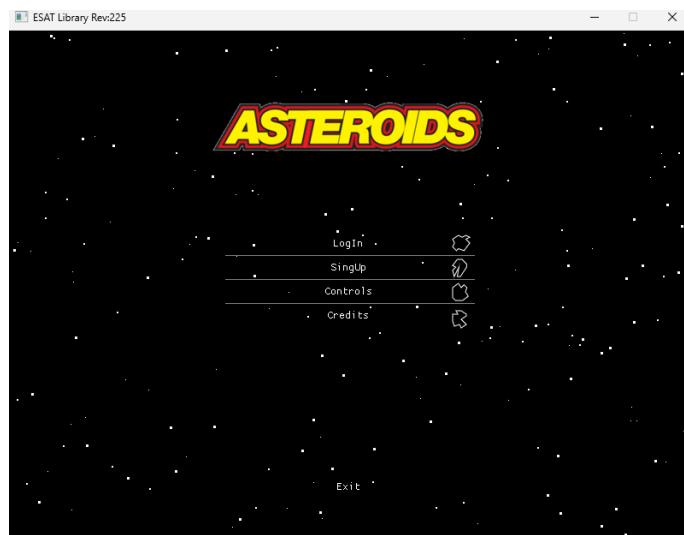
Los controles que necesitamos para controlar el juego son:

- **Flechas**: Flecha hacia arriba sirve para que la nave acelere, y las flechas laterales para que rote.
- **Espacio**: La nave disparará todas las veces que la tecla de espacio se pulse.
- **G**: Mecánica del “hiperespacio”, la nave se teletransporta a una posición aleatoria de la pantalla cuando esta sea pulsada.
- **P**: Se pausa el juego y saldrá una [ventana emergente](#).
- **Escape**: Para cerrar el programa.
- **S**: Durante el juego, para [guardar](#) datos de partida.



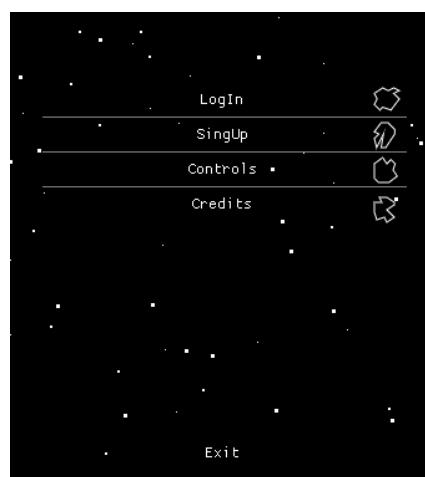
8.- Gestión de usuarios

8.1 Pantalla de bienvenida



Pantalla bienvenida del Asteroids.

Al iniciar el programa lo primero que nos aparecerá será una pantalla con el logo del juego original, junto a, más abajo, un menú con 4 botones para navegar por las diferentes opciones de la parte de gestión de usuarios e interfaz. Más abajo encontramos un botón de “Exit” para cerrar el programa.



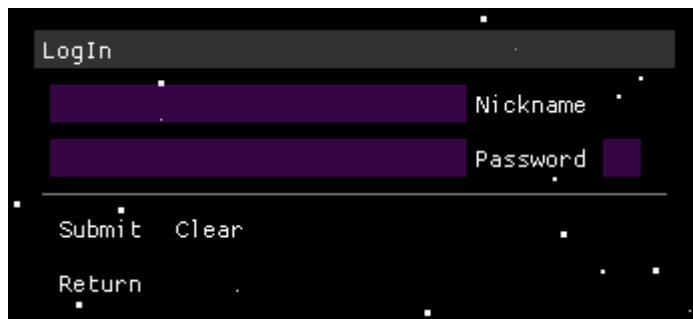
Menú botones pantalla bienvenida Asteroids

Aquí podemos ver los 5 botones, el primero botón: “[LogIn](#)” nos llevará a una pantalla la cual podremos iniciar sesión o entrar al modo administrador. El segundo botón: “[SingUp](#)” abre una ventana la cual podemos darnos de alta como usuario. El botón “[Controls](#)” abre una ventana con los controles principales para poder jugar y por último: “[Credits](#)” abre una ventana con los créditos del juego.



8.2 Pantalla inicio de sesión

La pantalla de inicio de sesión muestra los campos necesarios para iniciar sesión, se accede a través de la ventana de bienvenida pulsando el botón “LogIn”. Con el botón “Clear” limpiaremos los campos que se hayan rellenado, “Return” lo utilizaremos siempre que queramos volver a la [pantalla anterior](#), y una vez tengamos los campos necesarios rellenos, podremos entrar al juego pulsando el botón “Submit”.



Ventana de inicio de sesión Asteroids.

Si nos dejamos algún campo por llenar y pulsamos el botón “Submit” nos aparecerán los siguientes mensajes por pantalla:



Mensaje de error 1 en la ventana de Inicio de sesión.



Mensaje de error 2 en la ventana de Inicio de sesión.

Si pulsamos en el recuadro de al lado de la palabra “Password” podremos ver nuestra contraseña, en vez de verla en formato de asteriscos.



Mensaje de error al introducir un usuario que no está dado de alta.

Al introducir un usuario que no se encuentra en el fichero nos saldrá un mensaje por pantalla.

8.3 Pantalla de usuario



Pantalla de usuario.

En esta pantalla aparecerá la información competente del usuario que haya iniciado sesión, los diferentes [modos de juego](#) y en el caso que haya una partida guardada, aparecerá un botón el cual cargará la última partida guardada, más abajo hay un botón para volver a la pantalla anterior. A esta pantalla se llega desde la pantalla de [inicio de sesión](#) al introducir un usuario correcto.

Al seleccionar un modo de juego, nos llevará a la [pantalla de mejores puntuaciones](#) y podremos empezar a jugar.

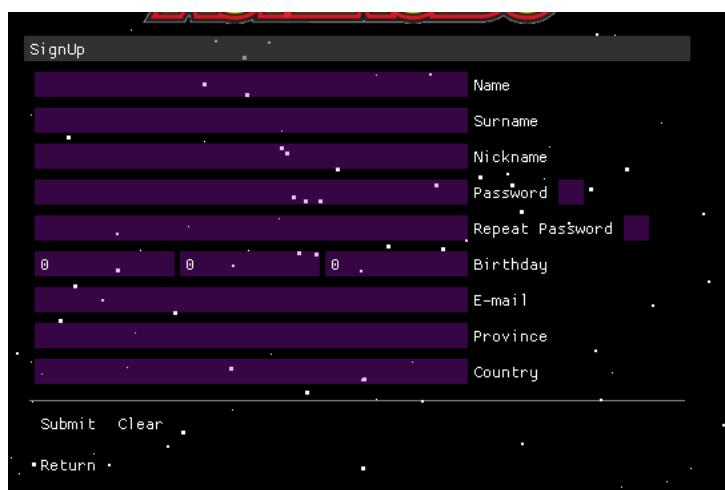


Si el usuario no tiene créditos no podrá jugar, y le saldrá el siguiente mensaje:



Pantalla con mensaje de “créditos insuficientes”

8.4 Pantalla registro de usuario



Pantalla de registro de usuario.

La pantalla de registro de usuario presenta de forma ordenada los campos necesarios para dar de alta a un usuario. Se accede a esta ventana desde el menú principal pulsando el botón “SignUp”

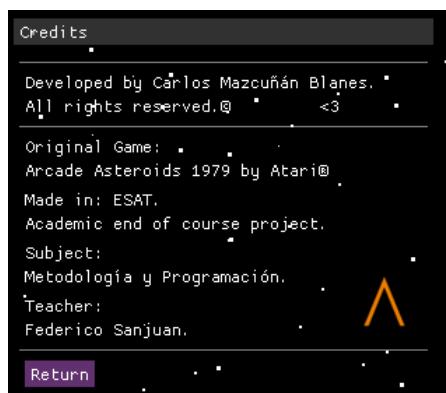
Los botones inferiores tienen la misma funcionalidad en esta pantalla como en la pantalla de “[LogIn](#)”



Al darle a “Submit” y los campos no estén correctamente llenados, saldrá un mensaje de aviso en rojo el cual nos indicará que hay un error. En el caso que esté todo correcto, el usuario se guardará correctamente, se le añadirán 5 créditos y nos devolverá automáticamente a la [ventana de bienvenida](#), donde ya podremos seguir navegando.

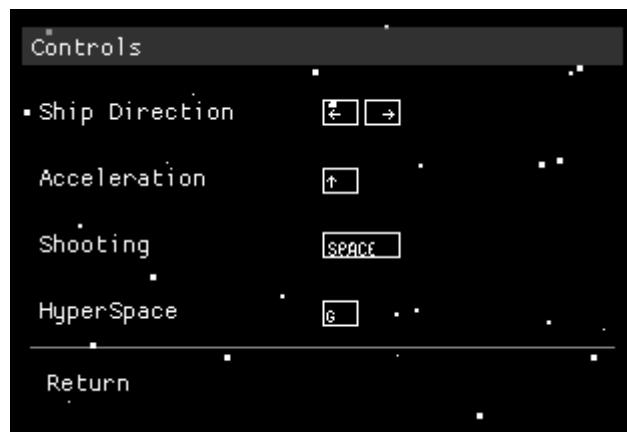
8.5 Pantallas de créditos y de controles

En la pantalla de créditos encontraremos información competente del proyecto escolar. Accedemos a ella desde el botón “Credits” de la [pantalla de bienvenida](#).



Pantalla de créditos.

En la pantalla de controles encontramos unas pequeñas imágenes que nos indican los controles que necesitamos para jugar. Accedemos a ella con el botón “Controls” que encontramos en la [pantalla de bienvenida](#).



Pantalla de controles.

En ambas pantallas encontramos un botón de “Return” para volver a la [pantalla anterior](#).



8.6 Pantalla de administrador



Ventana de administración.

La ventana de administrador muestra los usuarios que hay registrados en el fichero, mostrando su apodo, nombre, apellido y los créditos que le quedan.

En esta pantalla encontramos 3 botones, aparte del botón de “Return” que es para volver a la [pantalla anterior](#). Las funcionalidades de los 3 botones son:

- Add User: Dar de alta a un usuario desde el modo administrador.
- Edit User: Editar el usuario seleccionado desde el modo administrador.
- Delete User: Eliminar el usuario que se haya seleccionado.

Al editar o crear cualquier usuario desde el modo administrador se le podrá añadir o quitar todos los créditos que se deseé.



Ventana de edición de usuario en modo administrador.

Para entrar al modo administrador, deberemos de poner las credenciales correctas: nickname -> admin; password -> a123456.



9.- Gameplay

9.1 Antes de jugar

Selección de modos de juego



Diferentes modos de juego

En la [ventana de usuario](#), al iniciar sesión nos encontramos 4 botones con los diferentes modos de juego:

- > [Play 1P](#): Modo de juego clásico de 1 jugador.
- > [Play 2P](#): Modo de juego igual que el clásico pero con dos jugadores de manera alterna, cada vez que un jugador pierde una vida, será el turno del otro.
- > [Hardcore 1P](#): Modo de juego de un jugador con una dificultad añadida, la cual hace que empieces con una vida menos y la aparición de asteroides al inicio sea mucho mayor.
- > [Hardcore 2P](#): Modo de juego igual que “Hardcore 1P” pero con dos jugadores de manera simultánea.

Mejores puntuaciones

Al iniciar sesión satisfactoriamente y seleccionar un modo de juego, nos llevará a una pestaña la cual nos mostrará las 10 mejores puntuaciones que estén guardadas en el fichero local, ordenadas de mayor a menor.

1/ CARLOSMAZCU - 29910 PTS.
2/ MARKEL - 15690 PTS.
3/ CARLOSMAZCU - 14900 PTS.
4/ MARKEL - 14420 PTS.
5/ CHULETA - 10300 PTS.
6/ CARLOSMAZCU - 7740 PTS.
7/ CARLOSMAZCU - 7280 PTS.
8/ CARLOSMAZCU - 7280 PTS.
9/ CARLOSMAZCU - 7280 PTS.
10/ CARLOSMAZCU - 7280 PTS.



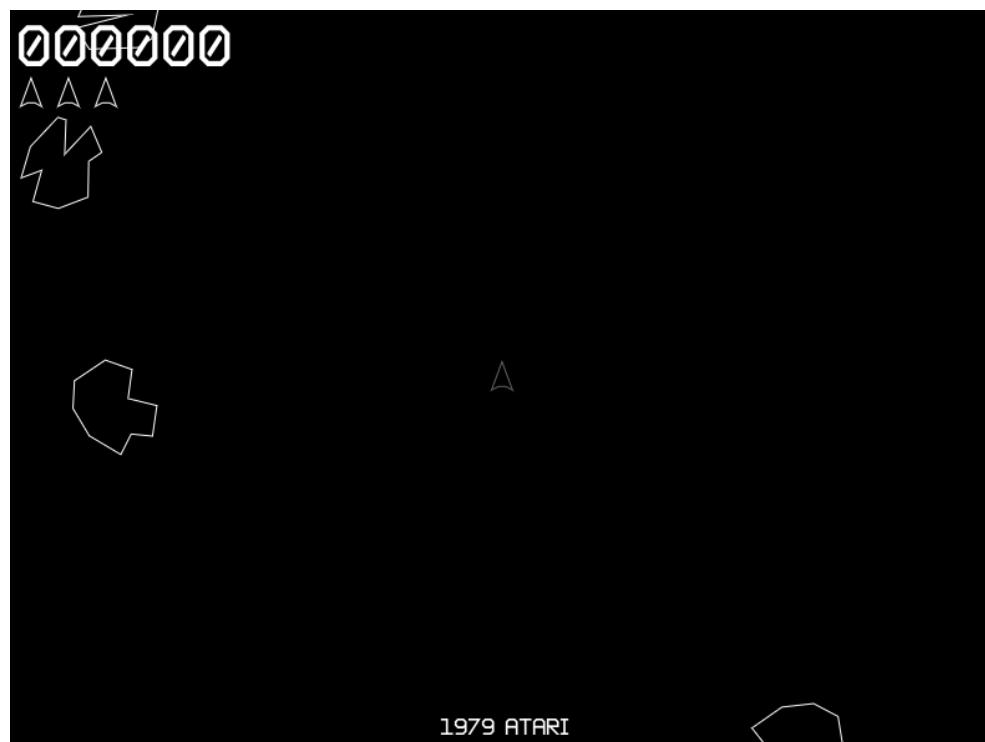
Pantalla mejores puntuaciones

9.2 En el juego

Al darle al botón de “Play” que se encuentra en la ventana de [mejores puntuaciones](#), iniciaremos el juego. Al comenzar, tendremos nuestra nave posicionada en el centro, con una inmunidad de unos segundos, la cual se representa con la nave dibujada en color gris. También aparecerán 4 asteroides de tamaño grande moviéndose en una dirección aleatoria entre la pantalla gráfica.

En todos los modos de juego se mantienen las mecánicas básicas del modo clásico.

Modo clásico (Play 1P):



Pantalla dentro del juego.

Arriba a la izquierda tendremos unos números los cuales reflejan la puntuación del jugador, junto a unos iconos iguales a los de la nave, que indican las vidas del usuario. Abajo del todo se muestra el fabricante y el año en el que se desarrolló el videojuego original.

Cada vez que se destruye un asteroide, este se dividirá en dos asteroides de tamaño mediano, y este de tamaño mediano, cuando se le dispare, se dividirá en dos asteroides de tamaño pequeño. Estos pequeños asteroides, si se les dispara, desaparecerán.

Cada vez que el jugador supere el umbral de los 10.000pts. se le sumará 1 vida.



Captura de pantalla durante el juego

Cada vez que una entidad llegue a un borde de la pantalla, ésta aparecerá en el borde contrario.

En el momento que no queden asteroides por pantalla, automáticamente, aparecerán el número de asteroides del nivel anterior añadiendo dos más.

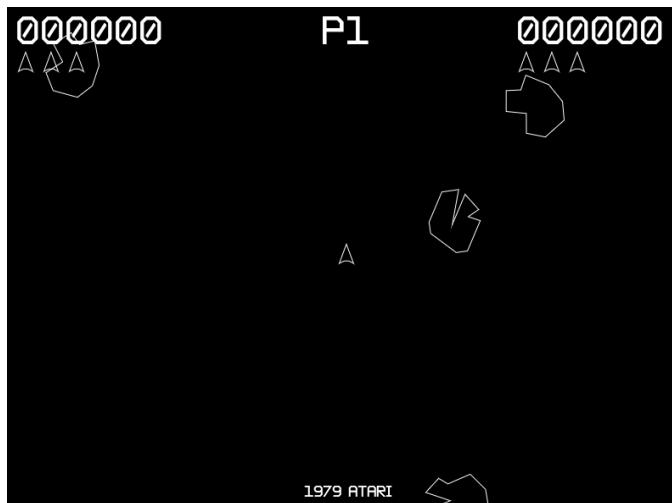
Los asteroides no colisionan entre sí, pero si uno de ellos colisiona con nuestra nave, perderemos una vida y reaparecemos en el centro de la pantalla a los pocos segundos, con una inmunidad de unos segundos más, en la cual podremos disparar.

Al pulsar la tecla 'G' nos teletransportamos a cualquier punto de la pantalla gráfica, al aparecer, tendremos la inmunidad durante unos segundos, pero no podremos disparar, ya que siempre te puedes teletransportar y habría un fallo de diseño del juego.

El juego se acabará cuando el jugador se quede sin vidas ([Game Over](#)).



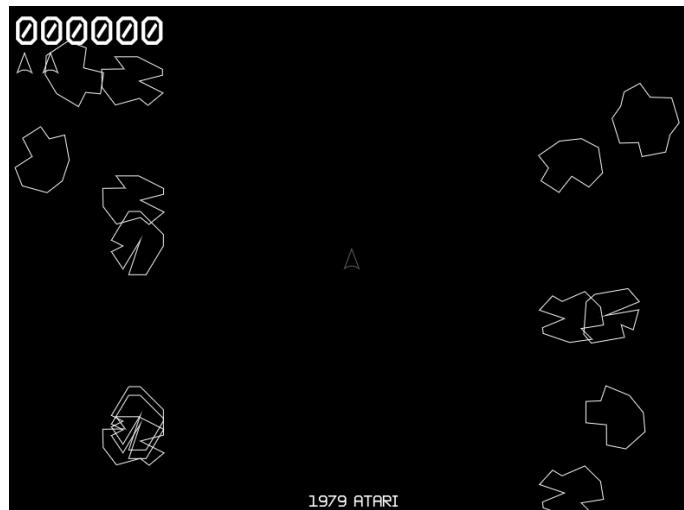
Modo 2 jugadores (Play 2P)



Captura de pantalla en el modo de juego 2 jugadores

En este modo de juego, se nos mostrará las puntuaciones y vidas de cada jugador en cada esquina de la pantalla, y arriba en el centro se nos indicará con 'P1' o 'P2' el turno de cada jugador. El juego se acabará cuando ambos jugadores se queden sin vidas.

Modo Hardcore 1 jugador (Hardcore 1P)

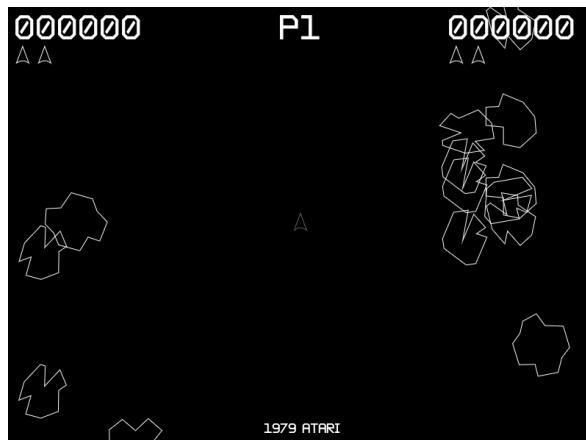


Captura del modo de juego Hardcore 1 jugador

En este modo de juego, al iniciar aparecemos con 2 vidas y un mayor número de asteroides.



Modo Hardcore 2 jugadores (Hardcore 2P)

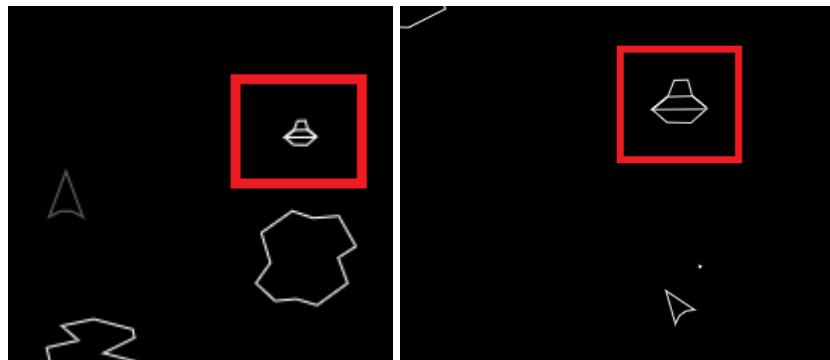


Captura del modo de juego Hardcore 2 jugadores

En este modo de juego se fusionan el modo [2 jugadores](#) clásicos y el modo [hardcore 1P](#), entonces se jugará en una mayor dificultad con dos jugadores de manera alterna.

Ovni

El ovni puede aparecer con dos tamaños diferentes:



Capturas de los diferentes tamaños de los ovnis

Los ovnis aparecen cada cierto tiempo con uno de los dos tamaños aleatoriamente, se mueven por la pantalla gráfica de izquierda a derecha o de derecha a izquierda, y cada cierto tiempo suben o bajan cierta distancia.

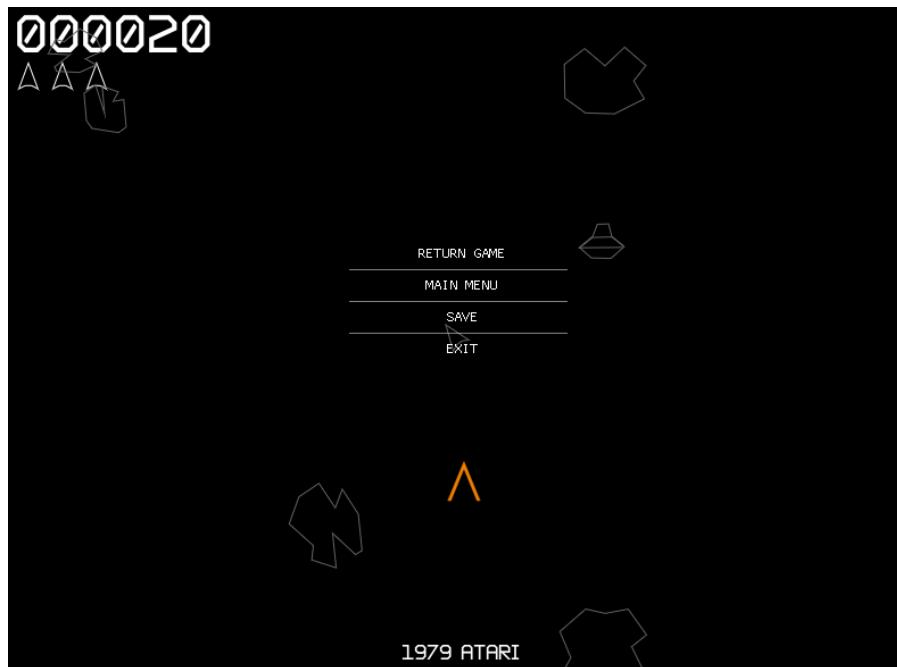
Pueden colisionar tanto con la nave del jugador y los asteroides, si este colisiona con algún asteroide, los puntos **no** se sumarán a la puntuación del jugador.

Este también dispara cada cierto tiempo predefinido en la dirección de la nave del jugador, si el disparo le da, le quitará una vida al jugador.

Para derrotar al ovni, solo basta con dispararle.



Ventana de pausa



Captura de pantalla de la ventana de pausa

Durante el juego, si pulsamos la tecla 'P', se activará una ventana de pausa, la cual hará que se bloqueeen todas las acciones del juego, pero se seguirá viendo detrás todas las entidades que estaban en ese momento en pantalla, las vidas y puntuación de los jugadores.

En esta ventana tendremos 4 botones, el primero que reanudará la partida que estamos jugando en ese momento ([Return Game](#)). El siguiente, que nos enviará directamente a la pantalla de bienvenida del menú principal ([Main Menu](#)). El tercer botón que nos guardará los datos necesarios de nivel para cuando se vuelva a iniciar sesión se pueda cargar la última partida guardada ([Save](#)). Y por último un botón que cerrará el programa por completo ([Exit](#)).



Game Over



Captura de pantalla de la ventana de "Game Over"

Cuando el jugador o los jugadores se queden sin vidas, el juego terminará y nos saltará una ventana de fin del juego, con un único botón el cual tiene la utilidad de volver a la ventana de bienvenida del menú principal.

Créditos o Monedas

Al [crear un usuario](#) se le asignará automáticamente 5 créditos, los cuales salen reflejados en la ventana de información del jugador una vez se inicia sesión.

Cada partida jugada costará un crédito, en cualquier modo, y la única manera que un usuario consiga más créditos es que el administrador los añada o entrar dentro del top 10 [mejores puntuaciones](#), lo cual se le asignará 5 créditos al usuario.

Guardado de partida

Mientras estamos en partida, si presionamos la tecla 'S' se guardan ciertos datos en un fichero, necesarios para cargar la partida en el mismo punto del juego, como el nivel, las vidas, la puntuación, etc.

Desde la [ventana de pausa](#), presionando el botón "Save" también guardaremos los datos.

Desde la [ventana de usuario](#), al iniciar sesión, si existe una partida guardada, nos saldrá un botón "Load Game", que nos cargará automáticamente la partida guardada.



10.- Guía de instalación

10.1 Instrucciones

📁 .vscode	10/05/2023 16:14	Carpeta de archivos
📁 documentacion	30/05/2023 20:17	Carpeta de archivos
📁 Imgui	10/05/2023 16:14	Carpeta de archivos
📁 Lib_Graph	10/05/2023 16:14	Carpeta de archivos
📁 Resources	30/05/2023 20:04	Carpeta de archivos
📄 Asteroids	30/05/2023 20:07	Archivo de origen C++
executable Asteroids	30/05/2023 20:17	Aplicación
exe compilacion	06/02/2023 12:56	Archivo por lotes de Windows
imgui	30/05/2023 20:17	Opciones de configuración

Listado de archivos al descomprimir Asteroids.zip

1. Descomprimir el archivo Asteroids.zip
2. Ejecutar Asteroids.exe

📁 Asteroids	30/05/2023 20:23	Carpeta comprimi...	12.454 KB
-------------	------------------	---------------------	-----------

Captura de pantalla del archivo Asteroids.zip

10.2 Instrucciones en caso de error

1. Eliminar Asteroids.exe
2. Abrir la consola de comandos de Visual Studio
3. Acceder al directorio en el que se encuentra el juego
4. Ejecutar el siguiente comando:
`compilación "nombre de la raíz" Asteroids.cc`
5. Si la compilación ha sido correcta, se habrán generado varios archivos, uno de ellos Asteroids.exe, el nuevo ejecutable compilado en el equipo de destino.
6. Ejecutar Asteroids.exe

```
C:\Asteroids>compilacion C: Asteroids.cc|
```

Captura de ejemplo del comando de compilación

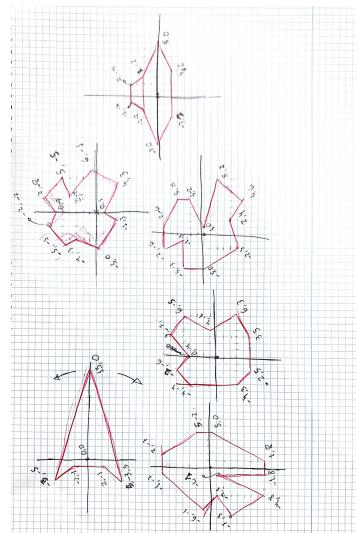


11.- Análisis postmortem

La realización de este proyecto me ha supuesto un reto y un desafío, a la par que una experiencia de aprendizaje y mejora en mis conocimientos de programación y geometría.

Dividí el proyecto en dos partes definidas, la gestión de usuarios y el gameplay. En primera instancia me puse a pensar y diseñar las ventanas con las cuales interactúa el usuario para poder iniciar sesión, darse de alta y demás utilizando una librería llamada ImGui, y fue la primera vez que investigaba y aprendía sobre librerías externas, lo que es otra parte positiva en el desarrollo del proyecto. Considero que la parte más complicada de esta parte fue saber gestionar correctamente los usuarios desde el modo administrador.

Una vez finalizada la primera parte satisfactoriamente, me adentré en el gameplay. Lo primero de todo fue diseñar la nave y los enemigos respetando al máximo el diseño del juego original. Con boli y libreta con hojas de recuadros dibujé todas las entidades y averigüé los puntos necesarios para realizar el dibujo vectorial. Una vez todo dibujado, era la hora de realizar las físicas de cada objeto y sus diferentes comportamientos, añadiendo colisiones y mecánicas necesarias.



Bocetos de diseño del juego

Después de juntar las dos partes principales, empecé con el guardado de partida y la tabla de mejores puntuaciones. Algo que se me complicó, hasta que diseñé el algoritmo correcto de ordenación.

Por último solo quedaba retocar detalles y solucionar ciertos bugs que fueron apareciendo durante el desarrollo del proyecto



Bibliografía

Manual técnico Asteroids original / imagen de portada:

[Asteroids - Arcade - Manual - gamesdatabase.org](#)

Información general del juego::

[Arcadeología - \(arcadeologia.es\)](#)

Imagen del teclado:

[Keyboard clipart coloring, Keyboard coloring Transparent FREE for download on WebStockReview 2023](#)

Video Longplay Asteroids Original:

[\(188\) Arcade Longplay \[900\] Asteroids - YouTube](#)

Software utilizado para la edición de imágenes:

[Paint - Apps on Google Play](#)