EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Gestió. Pla 2001

Títol: INTEGRACIÓ DE LLIBRERIES DE SO POSICIONAL AMB EL MOTOR DE RENDER OGRE3D

Document: Memòria

Alumne: Hug Beneït Gonzàlez

Director/Tutor: Mateu Sbert Casasayas
Departament: Informàtica i Matemàtica Aplicada
Àrea: LSI

Convocatòria (mes/any): 06/2008

Índex

1-Introducció:

Un videojoc és un joc que involucra a algú amb un espai d'efectes visuals i auditius, en un dispositiu electrònic amb una pantalla i una sèrie de perifèrics que permeten la interacció. Aconseguir el màxim de realisme es un dels objectius principals dels dissenyadors de videojocs i per això cal integrar en un mateix entorn diferents components tals com: un motor de render, un motor de física, un d' intel·ligència artificial, un motor de so, etc. El motor de so és un dels elements clau del videojoc, ja que el motor de render pot tenir molts bons gràfics, però sense un sistema de so amb posicionament 3D no es pot apreciar ni gaudir del videojoc en la seva totalitat.

La indústria del videojoc avança constantment sense parar, i cada cop acapara més i més camps. Un d'ells és la música i els sons. Tot i que moltes vegades no se'ls presti massa atenció ja que queden ofuscats pels gràfics, és un apartat molt important dins del videojoc.

Heu provat mai de jugar al vostre videojoc preferit amb els altaveus apagats?

Ràpidament notareu una gran diferència, trobareu que no és el mateix videojoc, i es que la música (Banda Sonora com ja tenen molts videojocs), els sons i els efectes són igual d'importants que la qualitat dels gràfics, ja que d'aquesta forma aconseguirem una major immersió del jugador en el videojoc.

Antigament els videojocs no tenien so, després van aparèixer les xiuletades que emetien les plaques base dels pc. Actualment tenim targetes de so amb una qualitat inimaginable de processament de so.

En definitiva, tractar la música com alguna cosa que ha d'ésser allà per tradició o costum, seria un error ja que l'hem de tractar com un escaló més a superar dia a dia en la creació d'un videojoc.

Nosaltres en aquest projecte tractarem amb la música dels videojocs i intentarem aprendre com ho fan els motors de videojocs per introduir la música i els sons dins dels videojocs.

A continuació farem un breu repàs a la història de la música i sons ens els videojocs i seguidament presentarem l'emmarc i l'abast del projecte.

1.1 Història

1.1.1 Sistemes de 8 bits i música xip

En els anys 1970, quan els videojocs van començar a florir com a forma d'entreteniment, la música s'emmagatzemava en mitjans físics com els cassets i els discs de vinil analògicament. Aquests components eren cars i propensos a avaries sota un ús constant, fent-los poc adequats per al seu ús en les. Un mètode més econòmic de tenir música en un videojoc era usar mitjans digitals, usant un microprocessador específic per a generar ones analògiques convertint codis digitals a impulsos elèctrics enviats a un altaveu. D'aquesta mateixa manera es generaven els efectes sonors.

Fig.1 Comodore 64

L'aproximació al desenvolupament de la música per a videojocs en aquest període solia fer-se mitjançant un simple generador de tons o síntesi FM per a simular instruments per a les melodies i usant un «canal de soroll» per a simular sons de percussió.

Fig.2 Els primers ordinadors mancaven de capacitats sonores reals (primer model de IBM, 1981).

Respecte als ordinadors IBM PC compatibles, els més antics estaven limitats al PC speaker de 1 bit de l'estàndard original, que resultava clarament insuficient per a generar sons. Els desenvolupadors de jocs que van usar la seqüenciació MIDI va ser usada per a aconseguir la síntesi FM, usant wavetables amb mostres predeterminades, la qualitat de les quals variava enormement d'un fabricant de targetes a un altre.

1.1.2 Síntesi i mostreig digital en els anys 1980 i 1990

Fig.3 Commodore Amiga 500

El primer ordinador personal que va fer ús del processament digital de senyals en la forma del

mostreig va ser el Commodore Amiga el 1985. El xip de so del computador presentava inicialment quatre convertidors digital-analògic de 8 bits. En lloc de generar simplement una forma d'ona que sonés com una xiuletada més o menys simple, com fa la síntesi FM, això va permetre reproduir petites mostres pre-gravades de so residents en memòria a través del xip de so.

A mesura que el cost de la memòria magnètica en forma de disquets queia, l'evolució de la música de videojocs en l'Amiga (i el seu desenvolupament en general) es va moure en certa forma cap al mostreig.

Fig.4 Sound Blaster 16

En els IBM PC compatibles, el so mostrejat va començar la seva introducció amb el llançament de la Sound Blaster de Creative en 1989, que va suposar una solució genèrica assequible per als usuaris de PC que volien contar amb característiques sonores avançades. La targeta incloïa un port per a maneta de jocs, suport MIDI compatible AdLib, un port estàndard per a targetes secundàries com la seva pròpia Wave Blaster o productes d'altres companyies, i permetia l'enregistrament i reproducció de so digital a 8 bits i 22,05 kHz (posteriorment 44,1 kHz) per a un únic canal estèreo. No obstant això, això no va suposar l'adopció massiva del so mostreig en els jocs de PC a causa de la seva incapacitat per a reproduir més d'una mostra cada vegada. La música seqüencial continuaria sent la més comuna en els jocs de PC fins a mitjans dels anys 1990, quan el CDROM es va popularitzar.

1.1.3 Música pregravada i streaming en l'era moderna

L'acostament predominant dels videojocs en format CD va anar, a pesar de la millora que havia experimentat el maquinari sonor, aprofitar la reproducció directa de so pregravat. Contar amb música completament pregravada tenia molts avantatges quant a qualitat sonora respecte a la seqüenciació: podien utilitzant-se tants instruments com es volgués en la producció en estudi, gravant una sola pista que seria reproduïda durant el joc. La qualitat era l'únic factor limitador quant a l'esforç dedicat a l'enregistrament de la pròpia pista. Els costos d'emmagatzematge que prèviament havien preocupat molt van anar en bona mesura de la mà dels mitjans òptics, que es convertiria en el mitjà de distribució preferit per a videojocs. La qualitat del CD àudio van permetre músiques i veus realment indistingibles de qualsevol altra font o gènere de música.

1.1.4 Aplicacions actuals i desenvolupaments futurs

La Xbox 360 té suport per a Dolby Digital, gravant i reproduint mostres de 16 bits a 48 kHz, suportant descompressió per maquinari i fins a 256 canals simultanis. A pesar d'aquesta potència i flexibilitat, cap d'aquestes característiques ha suposat un canvi important en la forma en la qual la música de videojocs per a l'última generació de consoles domèstiques es produeix. Els PC compatibles segueixen confiant en dispositius de tercers per a la reproducció del so en els videojocs, sent encara de llarg Sound Blaster el principal fabricador de targetes d'expansió de so i continuant el desenvolupament dels seus productes a bon ritme.

La tecnologia futura, a part de la seva major potència, tampoc presenta cap canvi fonamentar en la creació de música per a videojocs. La PlayStation 3 suporta diversos tipus de tecnologies de so

desenvolupant, incloent Dolby TrueHD i DTS-HD. La Nintendo Wii compartirà moltes característiques de so amb la consola de l'anterior generació, la Nintendo GameCube, incloent el suport per a Dolby Pro Logic II. Aquestes característiques, no obstant això, no són noves sinó més aviat millores de les quals ja estan en ús.

1.2- Emmarc del projecte:

El grup de recerca i investigació GiLab està desenvolupant un conjunt de llibreries d'il·luminació per a ajudar als programadors a millorar el rendiment i la visualització dels seus productes (videojocs). L'objectiu d'aquest projecte és integrar dins del motor gràfic de treball (Ogre3D) un motor de so i convertir el motor de render actual en un motor de videojocs en fase beta (ja que també és necessari el motor de física, IA,...).

Aquest projecte està emmarcat dins d'un projecte més gran anomenat Game Tools.

Game Tools és un Projecte Europeu desenvolupat per diferents universitats Europees per generar un framework per a desenvolupadors, que conté, llibreries, funcions i eines per a ajudar a la generació d'aplicacions 3D en temps real.

En aquest projecte s'utilitza Ogre3D (www.ogre3d.org) un motor de render de codi lliure.

1.3 El Projecte GameTools:

El projecte **GameTools** és un projecte de la Unió Europea que conjuntament amb altres universitats d'Àustria, França, Hongria i Espanya, i amb col·laboració amb socis de la indústria Europea s'han ajuntat per crear un conjunt de llibreries de gràfiques per a rendering en temps real 3D d'última generació.

Aquestes llibreries inclouen Geometria (LoD), Visibilitat, i Il·luminació Global per a PC i amb extensions previstes per a les consoles PS2,XBox, PS3 i Xbox369.

Aquestes llibreries estan desenvolupades sobre Ogre i Shark3D.

Per a més informació es pot anar a la web (www.gametools.org).

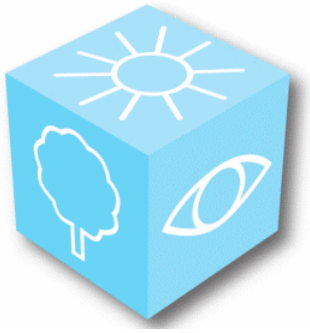


Fig.5 Logo del projecte GameTools

1.4 Abast del projecte:

L'objectiu principal d'aquest projecte és implementar un sistema de so completament integrat amb la plataforma de Render Ogre3D que permeti de forma senzilla i intuïtiva des d'un model de classes més abstracte la fàcil programació dels sons dins del sistema de render.

Es vol crear una plataforma escalable des de la qual sigui fàcil per als diferents desenvolupadors de targes gràfiques implementar els seus drivers de manera abstracta.

Per aconseguir això es crearà un model el qual estarà separat el driver de la tarja de so i les diferents classes del sistema.

Els fabricants només han d'implementar un seguit de funcions d'una classe abstracta per fer que el seu driver sigui utilitzable.

Finalment es vol crear una plataforma des de la qual sigui fàcil crear algoritmes per a millorar la realitat dels sons en diferents tipus d'ambients.

El sistema proporcionarà diferents efectes de so i els algoritmes es poden basar en aquests diferents efectes simplement canviant els paràmetres.

Per a realitzar aquesta tasca necessitarem que els drivers dels fabricants implementin un seguit d'efectes de so amb una sèrie de paràmetres que han de ser els mateixos per a tots els diferents drivers.

Per assolir aquest objectiu ens caldrà fer :

- Estudi i comparació de les diferents llibreries de so que hi ha en el mercat i elecció de la/les més adient/s. Utilitzarà les llibreries amb **so Surround** i suport per a estèreo, headphone, 4 canals, so 5.1 i 7.1.
- Disseny de la capa d'abstracció per aïllar al programador del motor de render del de so que s'utilitza en cada cas. Aquest disseny permetrà afegir-hi nous drivers de diferents fabricants simplement implementant una classe.
- Implementació del mòdul d'integració de so en el motor de forma que el programador i/o a l'usuari pugui **triar de manera gràfica el motor de render de so** en que vol executar l'aplicació.
- Dissenyar i implementar els mòduls que permetin **formats d'àudio comprimit**. Tals com OGG-Vorbis.
- Dissenyar i implementar els mòduls que permetran un ampli conjunt d'efectes **DSP** que es podran aplicar als sons en temps d'execució, tals com reverberació, eco, ...
- S'hauran de realitzar un seguit de proves i testos per comprovar el correcte funcionament de les llibreries.

La implementació es realitzarà amb C++ sobre l'IDE de desenvolupament Visual Studio 2005.

2- Estudi previ:

2.1 Estudi del motor de render.

2.1.1 Ogre3D

OGRE (*Object-Oriented Graphics Rendering Engine*) és un enginy de render 3D orientat a l'escena escrit en C++ dissenyat per fer fàcil i intuïtiu pels desenvolupadors que desenvolupen aplicacions gràfiques 3D accelerats per hardware. La llibreria de classes abstrau al programador dels detalls de les capes inferiors com Direct3D i OpenGL i proveeix una interfície basada en objectes del món en classes d'alt nivell.

OGRE té una gran comunitat i va entrar com a projecte de Sourceforge.net el Març del 2005. Ha estat utilitzat en jocs comercials com Ankh i Pacific Storm.

2.1.1.1 Informació General:

Tal com el seu nom diu, OGRE és només un sistema de render. I com a tal, el seu propòsit és proveir solucions generals per al render de gràfics. També conté altres utilitats com classes vectors, matrius, gestió de memòria, etc., que són considerats complementaris. No és una solució "all-in-one" en termes de desenvolupament de joc, o simulació, ja que no proveeix suport per a àudio o física, per exemple.

Això es podria considera com una desavantatge d'OGRE, però també pot ser vist com un avantatge de l'enginy ja que permet triar al desenvolupador quina llibreria física, àudio, input,... vol fer servir i així permet als desenvolupadors d'OGRE focalitzar-se en el tema de gràfics en comptes d'altres sistemes. OGRE suporta explícitament les llibreries OIS (input), SDL (gràfics 2D) i CEGUI (llibreries per fer menús), i inclou un "Cg toolkit".

Actualment OGRE està publicat sota una llicència dual (una és LGPL, i l'altre s'anomena *OGRE Unrestricted License* (OUL)), per poder triar la que més escaigui, ja que molts dels editors refusen utilitzar free/open-source software en aquest mercat en particular.

2.1.1.2 Característiques:

OGRE té un disseny orientat a objectes amb una arquitectura que permet la fàcil addició de plugins , el qual el fa altament modular.

OGRE es basa en scene graphs, amb un gran varietat d'scene managers, com per exemple octree, BSP i *Paging Landscape*, juntament amb una beta-stage portal-based (encara en desenvolupament).

OGRE és completament multi-platforma, amb suport per a OpenGL i Direct3D. Pot renderitzar el mateix contingut en diferents plataformes sense que el programador hagi de tenir en compte la plataforma sobre la que esta programant, reduint així la complexitat. Actualment els binaris pre-compilats estan disponibles per a Linux, Mac OS X, i la major part de versions del Windows.

OGRE també suporta Vèrtex i Fragment shaders programes escrits en GLSL, HLSL, Cg i assembler.

El scene manager de paisatge té suport per a LOD, el qual pot ser creat automàticament o manualment.

La part d'animació té suport complet via hardware per a l'animació via esquelets.

OGRE també té un compositing manager amb un llenguatge d'scripting i un complet postprocessing per a efectes com HDR, blooming, saturació, brightness, i soroll. Un sistema de partícules amb un extensible i personalitzable emissor de partícules.

Les llibreries també permeten el debug de memòria i la càrrega de recursos d'arxius exteriors.

Hi ha exportadors disponibles per a la gran majoria de modeladors 3D els quals inclou:3D Studio Max, Maya, Blender, LightWave, Milkshape, Sketchup i més.

Un llarg llistat de característiques sobre OGRE es pot trobar aquí.

2.1.1.3 Un exemple de CubeMaping en Ogre.

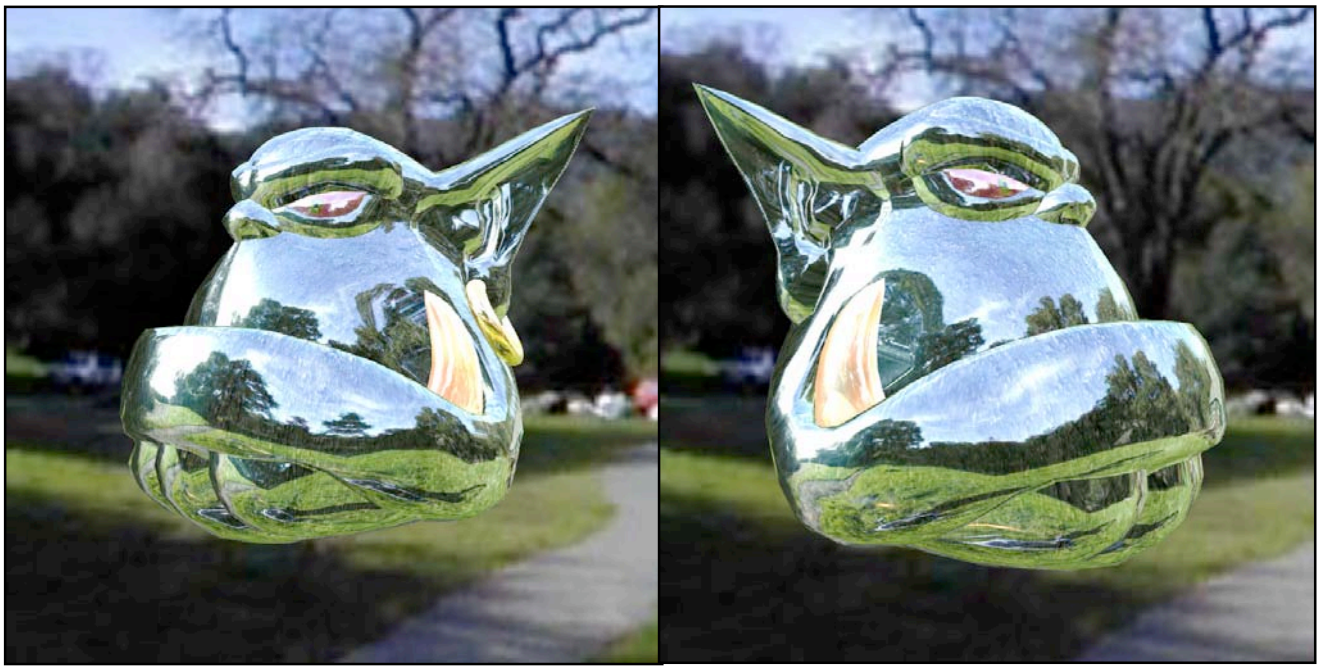


Fig.6 Exemple de cubemaping en el motor de render Ogre3D

2.2 Triar la llibreria de so:

Un aspecte molt important de la nostra aplicació resideix en la tria de les diferents llibreries de render de so disponibles en el mercat actual. Per a dur a terme aquesta cerca ens hem servit del buscador www.google.com per trobar-les.

Un cop recopilada la informació, l'hem estructurada separant dins de cada llibreria que hem trobat en avantatges i inconvenients que presenten cadascuna d'elles, finalment hem fet la tria en funció de les que creiem més interessants.

2.2.1 OpenAL

OpenAL es un SDK de multi-plataforma, lliure per a algunes d'elles. Segons la mateixa pàgina, és un SDK bastant poderós per a la fabricació de videojocs 3D i 2D. Té suport per a varis sistemes com Linux, Windows, Mac, entre d'altres. S'ha utilitzat en una gra quantitat de videojocs, tals com Battlefield 2142, Doom 3, Unreal Tournament 2003/2004, Lineage 2, ...

Link: http://www.openal.org/

Avantatges:

- Codi Lliure i obert.
- Gratuïta.
- Multi plataforma Win32, linux, MacOS
- Múltiples efectes reberb, echo,...

Inconvenients:

- Només reprodueix WAV (tot i que s'hi pot integrar amb facilitat ogg).
- Hardware bastant exclusiu quan es volen fer coses més avançades com efectes de so, filtres,...

2.2.2 irrKlang Pro

Avantatges:

- Multi plataforma: Windows 95, 98, NT, 2000, XP, Vista, Linux, MacOSX.
- Efectes de so: Chorus, Compressor, Distortion, Echo, Flanger Gargle, 3DL2Reverb, ParamEq and WavesReverb
- Reprodueix múltiples formats, .WAV, .MP3, .OGG, .MOD, .XM, .IT, .S3M...

Inconvenients:

- No donen el codi font per modificar.
- Té un cost econòmic, tot i que no és gratuït (en aplicacions comercials).

License Type	Summary	Cost
IrrKlang non commercial	You can use irrKlang freely for non-commercial products. Just download and go!	**0€**, it's free.
irrKlang Pro shareware	License for irrKlang to by used in a shareware or low cost product which is sold for a very low price (less than 14€) AND for an additional unlimited amount of products sold for less than 5€. Includes all irrKlang Pro features, on all supported platforms.	**65€*** buy now
irrKlang Pro mid-price	License for irrKlang to by used in one product which is sold for a price less than 25€. Includes all irrKlang Pro features, on all supported platforms, updates, access to preliminary patches, priorized support via mail.	**145€*** buy now
irrKlang Pro full	License for irrKlang to by used in one commercial product. Includes all irrKlang Pro features, on all supported platforms, updates, access to preliminary patches, priorized support via mail.	**390€*** buy now

2.2.3 Miles

Avantatges:

- Corre sobre una gran varietat de plataformes ps3,wii, xbox360, pc...
- Llegeix MP3
- Pot reproduir una gran varietat d'efectes de so.

Inconvenients:

- És molt car: $3,000.00
- No és codi lliure.

2.2.4 Fmod

Avantages:

- Multiplataforma:
    - Windows (32bit and 64bit)

    - Macintosh (PPC and x86)
    - Linux (32bit and 64bit)
    - Sony PS2, PS3 and PSP
    - Microsoft Xbox and Xbox 360
    - Nintendo Gamecube and Wii
- Suport tècnic de pagament i professional.
- Programable amb C++ i C#
- Canals virtuals.
- Efectes:
    - Oscillators, including sine, square, saw up, saw down, triangle and noise wave oscillators
    - Low-pass filters with resonance parameters
    - High-pass filter with resonance parameters
    - Echo
    - Flange
    - Distortion
    - Normalizer
    - Parametric EQ
    - Real-time Pitch Shifter (changes pitch not playback speed!)
    - Chorus
    - Reverb

Inconvenients:

- Té un preu massa elevat excepte per a us no comercials (llavors és gratuït).

Product	Description	Cost Per Product
FMOD Ex (including Designer)	First Platform	$ 6,000 USD
FMOD Ex (including Designer)	Subsequent Platforms	$ 3,000 USD
FMOD3	First Platform	$ 3,000 USD
FMOD3	Subsequent Platforms	$ 1,500 USD

2.2.5 Audiere

Aquest enginy és bastant simple, ofereix les característiques més simples per a la incorporació de so 2D en els jocs.

Avantatges:

-És molt fàcil d'utilitzar, ja que té una interfície molt amigable.

Inconvenients:

-No compleix les expectatives 3D per a l'aplicació, és a dir, no fa el càlcul de com s'ha d'escoltar per l'auricular dret i esquerra, simplement reprodueix sons en 2D.

2.2.6 DirectSound (DirectX)

DirectSound és un component de software de la biblioteca de DirectX, proveït per Microsoft, propietari del sistema operatiu Windows. Proporciona una interfície directa entre les aplicacions i els drivers de la tarja de so, permetent a les aplicacions produir so i música. A demés de proporcionar el servei essencial de passar dades d'àudio a la targeta de so, proporciona moltes capacitats necessàries tals com gravació i mescla de sons; efectes de so como reverberació, eco, flanger; posicionar sons en l'espai 3D (espaciament d'audio 3D), captura de sons d'un micròfon o d'una altra entrada i controlar la captura d'efectes durant la captura d'àudio.

Avantatges:

-Es poden reproduir tots els efectes sonors encara que no es tingui el hardware adequat, s'implementa per software (al contrari que OpenAL).

-Gratuït.

Inconvenients:

-No és multiplataforma.

-No és codi lliure.

-Nomes reprodueix WAV.

-Gairebé tot ho implementa per software per tant és molt costós per a la CPU.

2.3 La meva recomanació.

La meva recomanació seria utilitzar la llibreria OpenAL, ja que és 100% lliure, i té pràcticament les mateixes funcionalitats que qualsevol de les altres llibreries comercials.

En quant al problema de la reproducció de WAV té fàcil solució afegint-hi les llibreries ogg-vorbis, que permetrien la entrada d'àudio codificat de forma més senzilla.

També s'implementarà amb DirectSound ja que és gratuït i d'aquesta manera aconseguiríem tots els efectes que no tenim amb l'OpenAL.

Si un usuari té una targeta X-Fi de Creative podrà aprofitar millor les característiques de la seva targeta utilitzant OpenAL que DirectSound.

2.3.1 Overview d'OpenAL:

OpenAL és una API d'àudio multiplataforma desenvolupada per Creative Labs per al renderitzat eficient d'àudio posicional i multicanal en tres dimensions. Està ideada per al seu us en videojocs.

L'API està disponible per a les següents plataformes: Mac OS, Linux (tant per OSS com per ALSA), *BSD, Solaris, Irix, Microsoft Windows, Sony PlayStation 2, Microsoft Xbox i Nintendo GameCube.

Al contrari que la especificació d'OpenGL, la especificació d'OpenAL es divideix en dos seccions: el nucli, consisteix en crides a funcions, i l'API ALC, s'utilitza per a la gestió dels contextos de renderizat, us i bloqueig de recursos, etc. de manera multiplataforma.

Amb la intenció d'afegir funcionalitats extres en el futur, OpenAL utilitza un mecanisme basat en extensions. Cadascuna pot incloure les seves pròpies extensions en la distribució de OpenAL, el qual permet agregar funcionalitats de hardware propietari.

El funcionament global d'OpenAL es pot dividir en objectes font, oients i búffers d'àudio. Un objecte font conté un punter a un búffer, a més d'una sèrie d'atributs com la velocitat, posició, direcció o intensitat de l'emissor de so. Un oient conté informació sobre la velocitat, posició i orientació del sistema de referència, a demés del guany general aplicat a tot so. Només pot existir un oient. Els búffers contenen la informació del so en format PCM, bé en 8 o 16 bits, en format mono o estèreo. El motor de renderizat s'encarrega de tots els càlculs necessaris com la atenuació, doppler, etc.

Història

OpenAL va ser desenvolupat en un principi per Loki Software amb la finalitat d'ajudar a portar els jocs de Windows a Linux. Després de que Loki desaparegués, el projecte va ser mantingut un temps per free software/open source community — però ara albergat per Creative Technology i suportat per Apple i free software/open source .

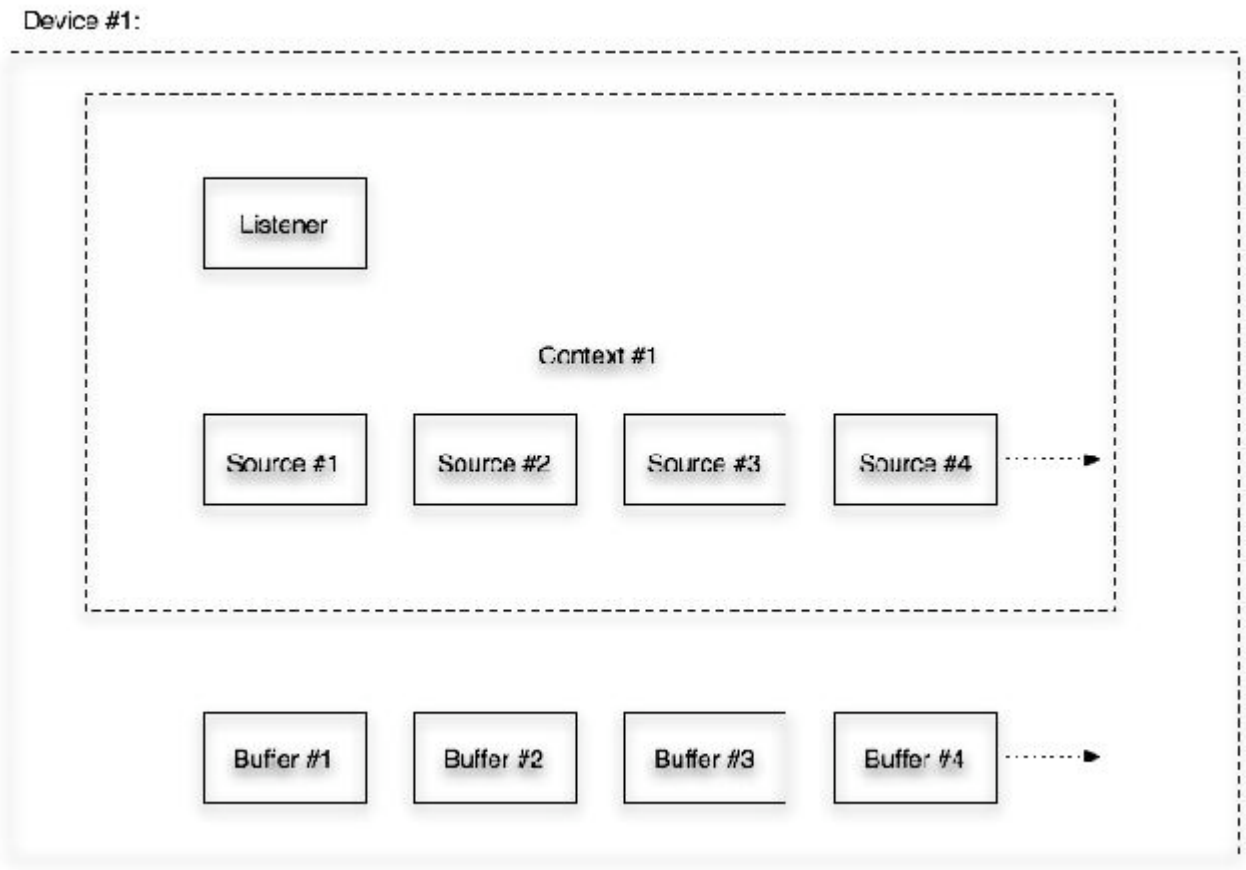


Fig.7 Estructura bàsica de l'OpenAL

OpenAL està estructurat en emissors (sources), àudio búffers i un sol oient (listener). Un

emissor té un punter a un búffer i conté la velocitat, la posició, la direcció, intensitat del so. L'oient conté la seva velocitat, posició, direcció i guany general aplicat a tots els sons. Els búffers contenen l'àudio pròpiament dit (bits) en format PCM, tant en 8 com en 16 bits, en mono o estèreo.

Per tal de proveir funcionalitats addicionals en el futur, OpenAL utilitza un mecanisme d'extensions que permet als diferents venedors la inclusió de les seves pròpies extensions.

Aplicacions que utilitzen l'API OpenAL:

Jocs:

id Software engine-based games such as Doom 3, Jedi Knight 2, Jedi Knight: Jedi Academy, Quake 4 and Prey.

Unreal Technology based games such as Unreal 2, Unreal Tournament 2003, Unreal Tournament 2004, Unreal Tournament 3, Postal², and America's Army.

Battlefield 2, Battlefield 2142, Freedom Fighters, Hitman, Psychonauts, and Colin McRae: DiRT.

Numerous free software/open-source games also use OpenAL. Several examples include War§ow and Warzone 2100.

Altres aplicacions:

Blender - 3D modelling and rendering tool.

Unity - 3D game engine and game creation IDE.

Una llista més exhaustiva pot ser trobada a la web d'OpenAL.

2.3.2 Environmental audio extensions (EAX)

Environmental audio extensions (o **EAX**) són un conjunt de processament digital de senyals en temps real per a audio, està present a Creative Labs, targes de so Sound Blaster i Creative NOMAD/Creative Zen .

La aspiració d'EAX és crear un ambient més acurat en els videojocs que simuli un so ambiental més proper al nostre món real. A partir de l'EAX 2.0, la tecnologia simplement es basava al voltant d'efectes a través de les targes Sound Blaster Live! que tenien el xip d'àudio EMU10K1. Aquest xip permetia afegir efectes a sons MIDI tals com reverb i chorus.

La majoria de les següents versions de EAX van coincidir amb l'increment del nombre de sons simultanis que era capaç de processar la tarja de so:

EAX 1.0 suportava 8 veus, EAX 2.0 suportava 32 (Live!), EAX Advanced HD (EAX 3.0) suportava 64 (Audigy), EAX 4.0 un altre cop 64 (Audigy 2), i finalment EAX 5.0 suporta 128 veus (i 4 efectes per cada un) (X-Fi).

Actualment les targes de so X-Fi permeten processar els següents efectes de so:

- EAXReverb
- Standard Reverb
- Chorus
- Distortion
- Echo
- Flanger
- Frequency Shifter
- Vocal Morpher
- Pich shifter
- Ring Modulator
- Auto-Wah
- Compressor
- Equalizer

Jocs que utilitzen EAX

2.3.3 Overview de DirectSound (DirectX)

DirectSound és un component de software de la biblioteca de DirectX de Microsoft, que resideix en una computadora amb el sistema operatiu Windows. Proporciona una interfície directa entre les aplicacions i els drivers de la tarja de so, permetent a les aplicacions produir sons i música. A més de proporcionar el servei essencial de passar dades d'àudio a la tarja de so, proporciona moltes capacitats necessàries tals com gravació i barreja de sons; addició d'efectes de so com reverberació, eco, flanger; posicionar els sons en l'espai 3D, captura de sons de un micròfon o de una altra entrada i controlar la captura d'efectes durant la captura d'àudio.

DirectSound també permet a vàries aplicacions d'una forma convenient compartir l'accés a la tarja de so al mateix temps. La seva capacitat per a reproduir el so 3D va afegir una nova dimensió als videojocs. També proporciona la capacitat als jocs de modificar una seqüència musical en resposta a aconteixements del joc en temps real, per exemple: el ritme de la música pot accelerar en quant la acció augmenta.

Després de molts anys de desenvolupament DirectSound és avui en dia una API molt madura i que proveeix moltes altres capacitats útils, com la capacitat de reproduir el so multicanal i els sons d'alta resolució. Mentres que directSound va ser dissenyat per a ser utilitzat en joc, un nombre de professionals d'àudio aprofiten ara moltes de les seves capacitats.

DirectSound3D (DS3D) és una addició al sistema de DirectX de Microsoft el qual va intentar estandarizar l'àudio 3D sota Microsoft Windows, introduït amb el DirectX 3 el 1996.

Iniciant des de DirectX8 en endavant, DirectSound i DirectSound3D (DS3D) són conjuntament Anomenats com **DirectX Audio**.

2.4 Planificació:

Un cop triades les llibreries fem un planing sobre la quantitat de temps que ens portarà dissenyar tota la aplicació.
Per fer-ho utilitzarem el programa Microsoft Project.

Nom de la tasca:	Duració:	Inici:	Final:
1. Estudiar llibreries Ogre	155d	mar 11/09/07	lun 14/04/08
2. Estudiar llibreries OpenAL	122d	vie 26/10/07	lun 14/04/08
3. Estudiar llibreries DirectSound	118d	jue 01/11/07	lun 14/04/08
4. Fer Diagrama de classes i fitxes de cas d'us	11,5d?	lun 19/11/07	mar 04/12/07
5. Codificació de les classes més importants	30d	mar 04/12/07	mar 15/01/08
6. Proves per comprovar el correcte funcionament de les classes	16d	mar 15/01/08	mié 06/02/08

7. Codificació de la demo de prova 2	25d?	lun 28/01/08	vie 29/02/08
8. Comprovació de la demo 2	4d?	mar 26/02/08	vie 29/02/08
9. Codificació dels efectes de so DSP	32d?	vie 29/02/08	lun 14/04/08
10. Codificació de la demo de prova1 amb efectes de so	15,5d?	mar 15/04/08	mar 06/05/08
11. Comprovació de la demo 1	5,5d?	mar 06/05/08	mar 13/05/08
12. Codificació de la demo 3 per a fer els testos de velocitat	13d?	lun 28/04/08	mié 14/05/08

Tal com podem observar tant les llibreries Ogre3D com les de OpenAL i DirectSound s'estaran estudiant pràcticament durant tot el projecte ja que la seva desconeixença fa que les haguem de consultar molt sovint.

Un cop après una part de les llibreries (la part bàsica d'afegir i reproduir sons), començarem a fer els diagrames de classes i les fitxes de cas d'ús.

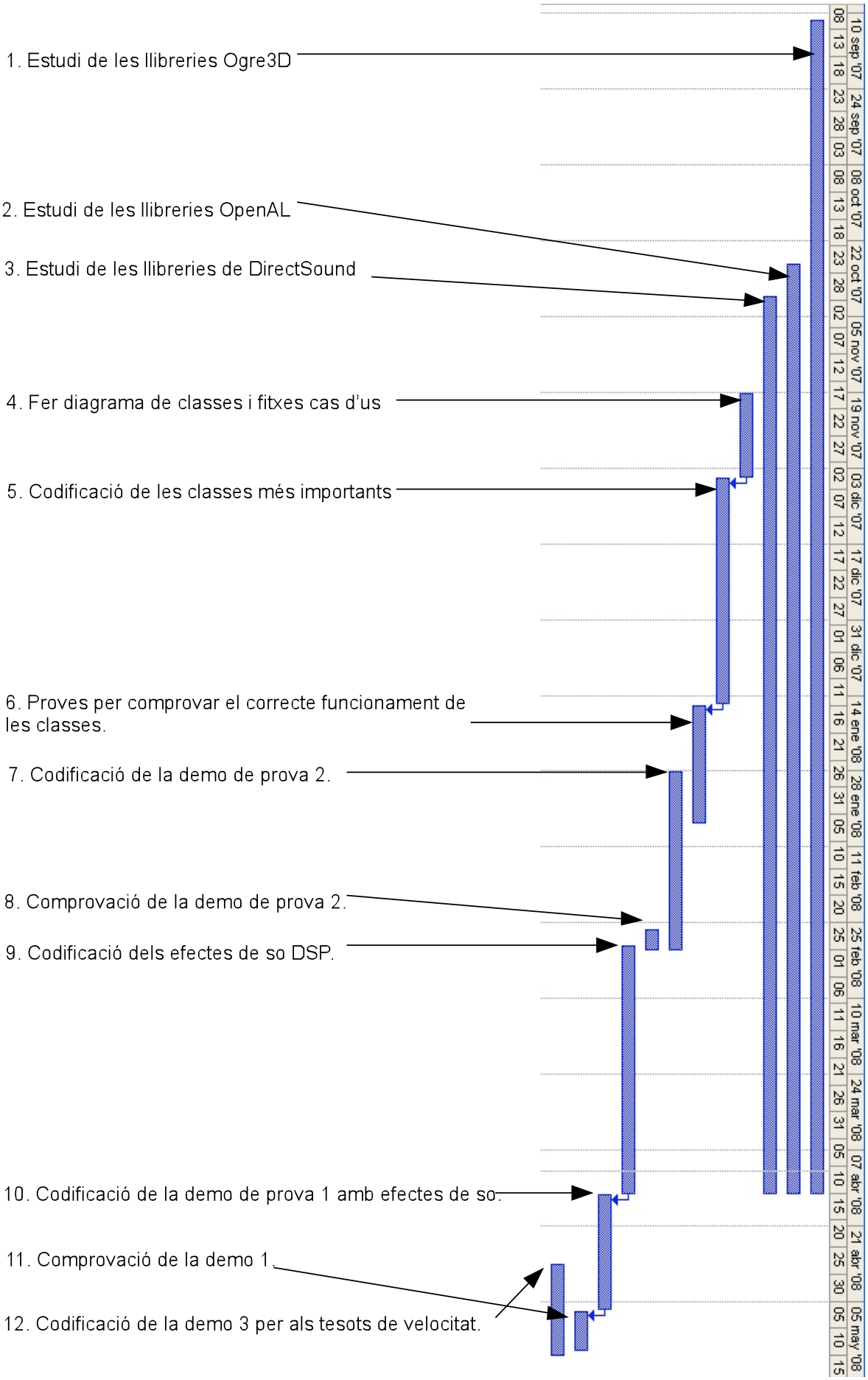
Òbviament gran part dels diagrames que es van fer al principi es van haver de polir molt degut a la desconeixença de les llibreries.

Seguidament varem començar a desenvolupar les classes més importants (deixant de banda la part d'efectes de so ja que varem preveure que les diferències entre elles farien enrevessar massa el codi si es tenien en compte llavors).

Per comprovar el correcte funcionament de les classes varem codificar jocs de proves a mesura que s'afegeixen noves funcionalitats al projecte.

Seguidament varem procedir a la codificació de les classes per als efectes de so (DSP) i es va realitzar una altra demo de proves.

Finalment varem codificar els jocs de proves i els testos de rendiment per testejar i comparar els dos drivers de so.


1. Estudi de les llibreries Ogre3D
2. Estudi de les llibreries OpenAL
3. Estudi de les llibreries de DirectSound
4. Fer diagrama de classes i fitxes cas d'us
5. Codificació de les classes més importants
6. Proves per comprovar el correcte funcionament de les classes.
7. Codificació de la demo de prova 2.
8. Comprovació de la demo de prova 2.
9. Codificació dels efectes de so DSP.
10. Codificació de la demo de prova 1 amb efectes de so.
11. Comprovació de la demo 1.
12. Codificació de la demo 3 per als tesots de velocitat.
10 sep '07
24 sep '07
08 oct '07
22 oct '07
05 nov '07
19 nov '07
03 dic '07
17 dic '07
31 dic '07
14 ene '08
28 ene '08
11 feb '08
25 feb '08
10 mar '08
24 mar '08
07 abr '08
21 abr '08
05 may '08

3-Anàlisi i Disseny

3.1 Requeriments de l'aplicació:

Tal com hem dit a la presentació el que nosaltres volem és fer un wrapper (embolcall) de sistema de so per a Ogre3D, és a dir una capa d'abstracció per a que el programador no hagi de pensar sobre quina llibreria de render de so s'està executant.

Els requeriments que demanem a les llibreries que volem implementar són:

- **So Surround**. Suport per a stereo, headphone, 4 canals, so 5.1 i 7.1.
- **Espacialització 3D**. Els sons s'escoltaran com si estiguessin en un entorn 3D.
- **DSP**. Un ampli conjunt d'efectes que es poden aplicar als sons en temps d'execució, tals com reverberació, eco, ...
- Capacitat per reproduir **formats d'àudio comprimit**. Suport per a streaming del format OGG-Vorbis.
- **Atenuació** dels sons en relació a la distància.
- **Modificació** de la freqüència dels sons.
- **Sons direccionals** amb atenuació fora dels cons.
- **Obstrucció/atenuació** amb el terreny.
- **Elecció del sistema de render de so** (OpenAL o DirectSound) per part del programador/usuari.
- Els sons que es troben fora del rang per a ser escoltats són apagats **automàticament.**

3.2 Metodologia de programació:

Com que estem davant d'un projecte nou i "innovador" i no tenim experiència prèvia en aquest tipus de projectes, hem decidit utilitzar la metodologia X-trem Programming.

X-trem Programming és un dels diversos processos àgils de desenvolupament.

La XP està pensada per respondre als canvis en els requeriments d'un projecte. En entorns on el client no té una idea molt clara del que es vol o on els requeriments poden canviar ràpidament és on es treu més profit de la XP. En aquests entorns una metodologia tradicional molt probablement fracassaria, mentre que la XP està pensada per adaptar-s'hi. Tant aquests projectes com qualsevol altre tenen una nivell de risc, les regles i pautes marcades per la XP també permeten reduir-ne els riscos.

És una metodologia pensada per a equips petits (en aquest cas un sol programador). Es recomana que els equips siguin d'entre 2 i 12 membres. Els equips petits permeten adaptar-se més fàcilment als canvis.

Característiques:

Desenvolupament iteratiu i incremental: petites millores, una rere l'altra.

Proves unitàries **contínues**, freqüentment repetides i automàtiques, incloent proves de regressió.

Programació per parelles: es recomana que les tasques de desenvolupament es duguin a terme per dues persones en un mateix lloc. Se suposa que la major qualitat del codi escrit d'aquesta forma (el codi és revisat i discutit mentre s'escriu) és més important que la possible pèrdua de productivitat immediata.

Freqüent **interacció de l'equip de programació amb el client** ó usuari. Es recomana que un representant del client treballi juntament amb l'equip de desenvolupament.

Correcció de tots els errors abans d'afegir una nova funcionalitat. Fer entregues freqüents.

Defactorització del codi, es a dir, reescriure certes parts del codi per tal d'augmentar la seva llegibilitat i mantenibilitat però sense modificar el seu comportament.

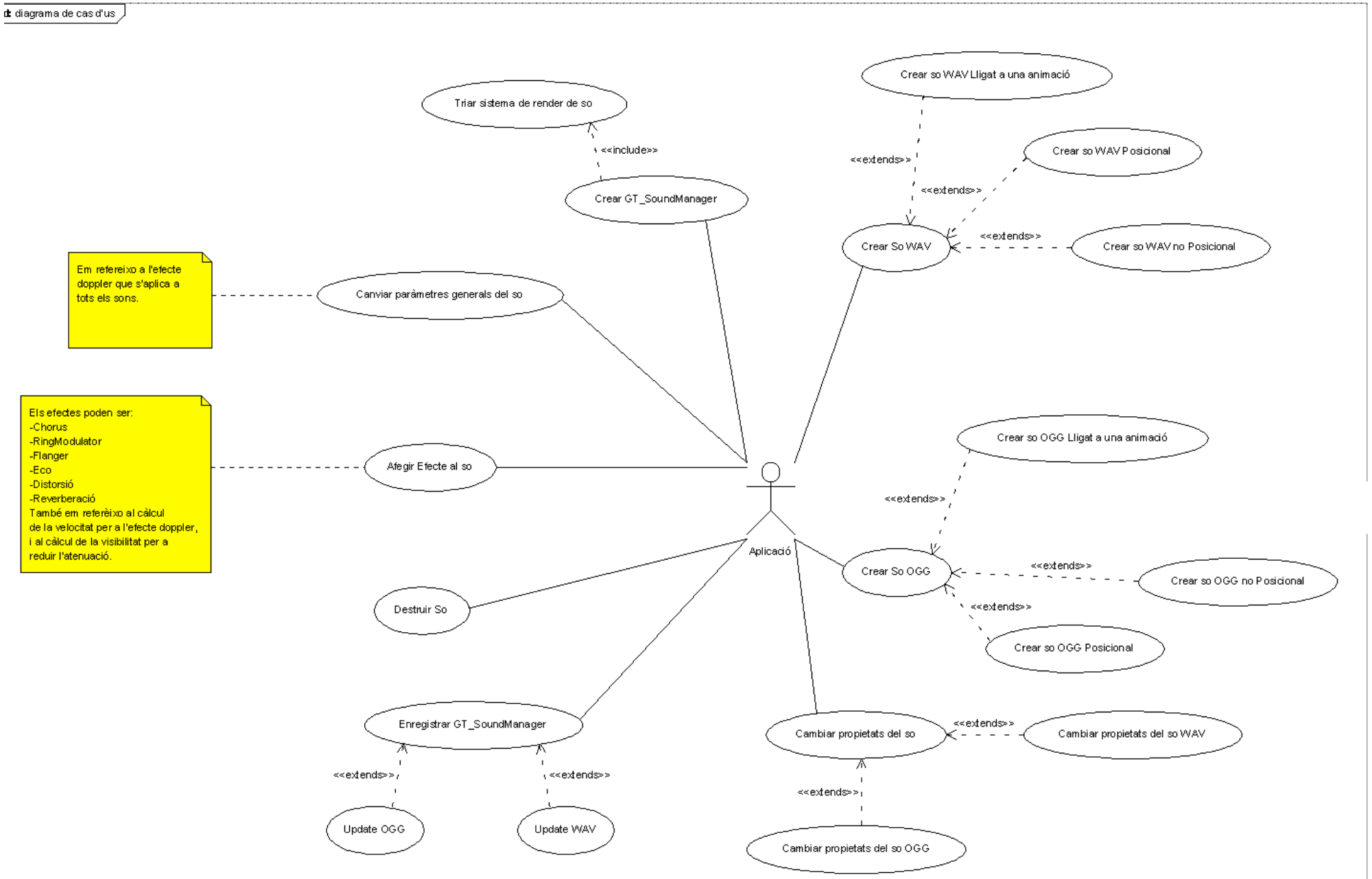
Propietat del codi compartida: en lloc de dividir la responsabilitat en el desenvolupament de cada mòdul en grups de treball diferents, aquest mètode promou que tot el personal pugui corregir i estendre qualsevol part del projecte. Les freqüents proves de regressió garanteixen que els possibles errors seran detectats.

Simplicitat en el codi: és la millor manera de que les coses funcionin. Quan tot funcioni es podrà afegir funcionalitat si és necessari.

3.3 Diagrames de cas d'ús

Per complir les anteriors característiques que ens han demanat que tingui el sistema de so, hem proposat el següent diagrama de casos d'ús.

En els diagrames de cas d'us que hem adjuntat a sota són les funcions més importants que es cridaran des de l'aplicació.


d: diagrama de cas d'us
Triar sistema de render de so
<<include>>
Crear GT_SoundManager
Crear so WAV Lligat a una animació
<<extends>>
Crear so WAV Posicional
<<extends>>
Crear So WAV
<<extends>>
Crear so WAV no Posicional
Em refereixo a l'efecte doppler que s'aplica a tots els sons.
Canviar paràmetres generals del so
Els efectes poden ser:
-Chorus
-RingModulator
-Flanger
-Eco
-Distorsió
-Reverberació
També em referèixo al càlcul de la velocitat per a l'efecte doppler, i al càlcul de la visibilitat per a reduir l'atenuació.
Afegir Efecte al so
Aplicació
Crear so OGG Lligat a una animació
<<extends>>
Crear So OGG
<<extends>>
Crear so OGG no Posicional
<<extends>>
Crear so OGG Posicional
Destruir So
Enregistrar GT_SoundManager
<<extends>>
Update OGG
<<extends>>
Update WAV
Cambiar propietats del so
<<extends>>
Cambiar propietats del so WAV
<<extends>>
Cambiar propietats del so OGG

3.4 Diagrama de classes:

Tal com hem dit a l'inici de l'anàlisi el nostre objectiu és aïllar al programador del driver que està utilitzant. Per aconseguir això varem implementat el següents diagrames de classe amb els patrons de disseny adients:

3.4.1 Diagrama de classe per parts:

3.4.1.1 La Classe GT_SoundManager i els seus entorns

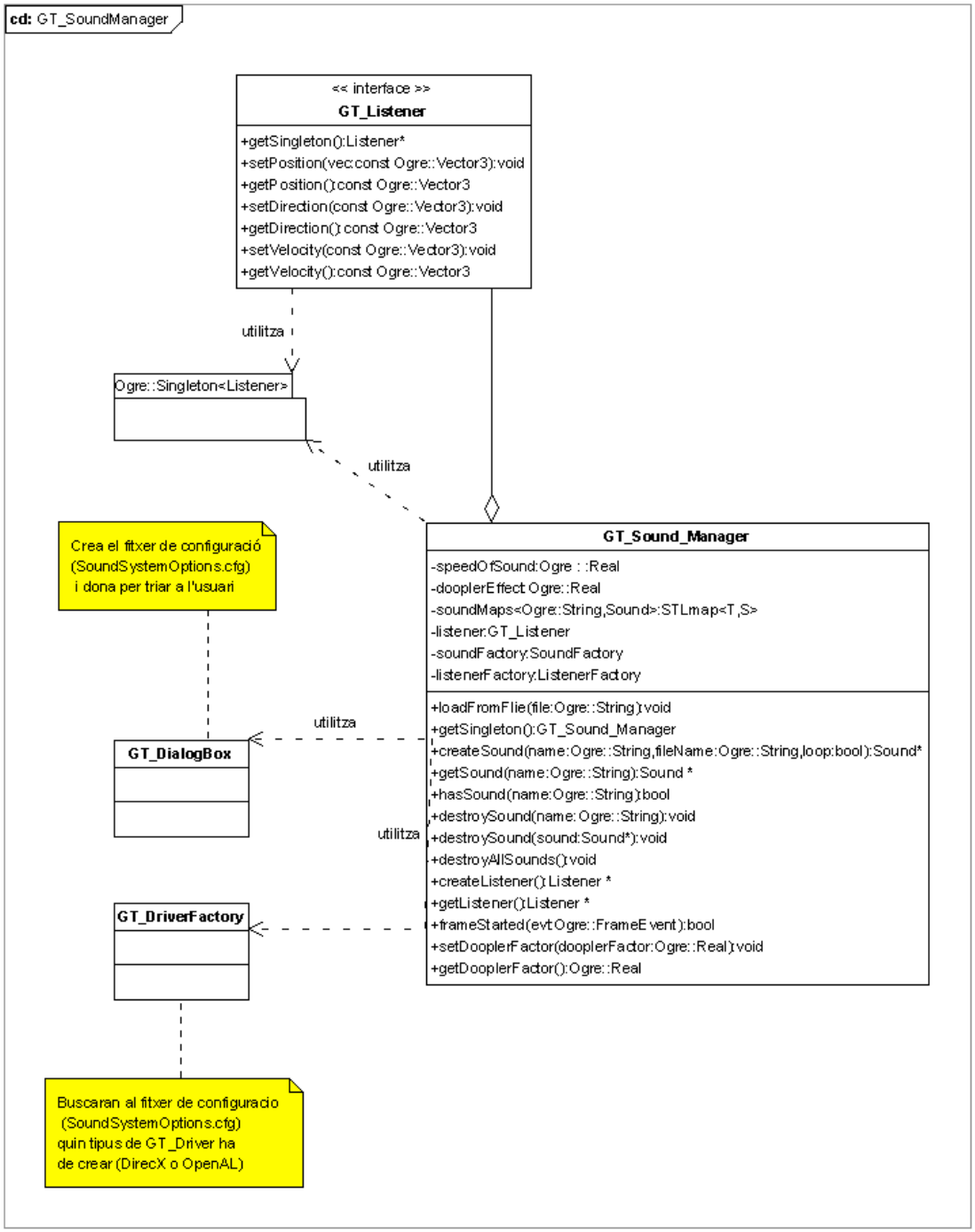


Fig.8 La classe GT_SoundManager i els seus entorns

Ens hem basat en l'estil com Ogre instància i crea els objectes.

Ogre ho fa mitjançant un altre objecte, és a dir fa servir el patró builder per a la creació d'un objecte més complex tal com pot és el so.

La classe GT_SoundManager:

La classe **GT_SoundManager** és el centre d'operacions en el qual es **creen**, es **busquen** i es **destrueixen** els sons.
La classe conté un hashmap amb els sons que hi ha dins del sistema.
No és necessari que tots els sons que estiguin dins del **GT_SoundManager** s'estiguin reproduint a l'hora, la classe **GT_SoundManager** triarà per nosaltres en quin moment s'ha d'escoltar o s'ha d'aturar un so en funció de la distància. Això ho fa mitjançant la classe **Ogre::FrameListener** de la qual deriva i la qual és cridada cada frame i fa aquesta comprovació.
També fa la **inicialització/alliberament** del sistema de so i la tria del driver mitjançant un fitxer de configuració o manualment.
És l'encarregat de calcular **l'efecte doppler** i emmagatzemar la **velocitat del so**.
La feina més important del **GT_SoundManager** és la de moure a cada frame cada so i moure també l'oient.

En el següent diagrama mostrem la inicialització de la classe **GT_SoundManager**.

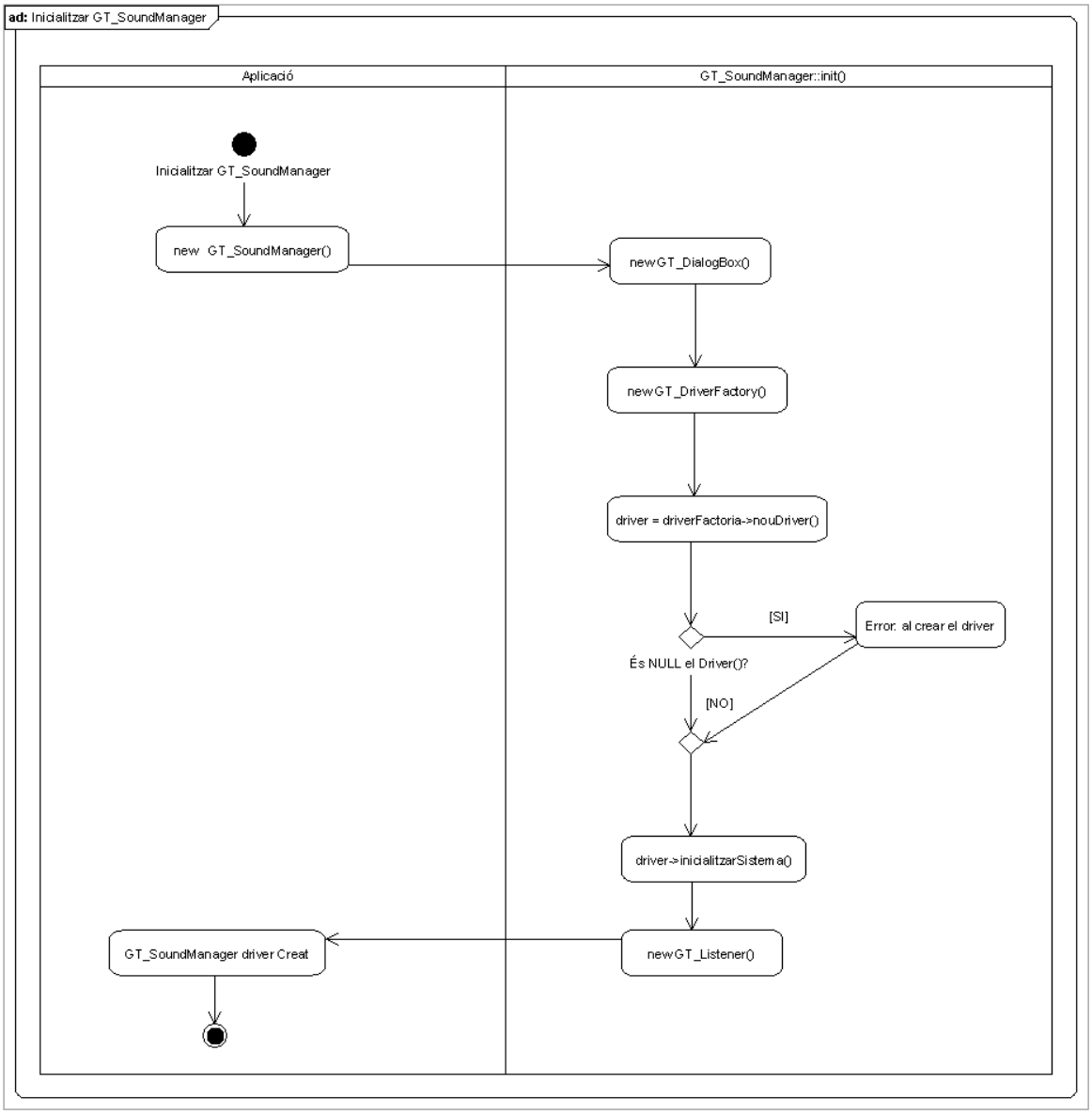


Fig.9 Inicialització de la classe GT_SoundManager

La classe GT_Listener:

La classe **GT_Listener** és l'oient de la escena.
En aquesta classe bàsicament s'utilitzen les funcions per **lligar l'oient a una càmera** o a un **Ogre::SceneNode**.
Hem de tenir en compte que només podrem crear un sol **GT_Listener**.

Aquí tenim el diagrama sobre com s'actualitza l'oient mitjançant la funció update() que crida cada frame la classe **GT_SoundManager**.

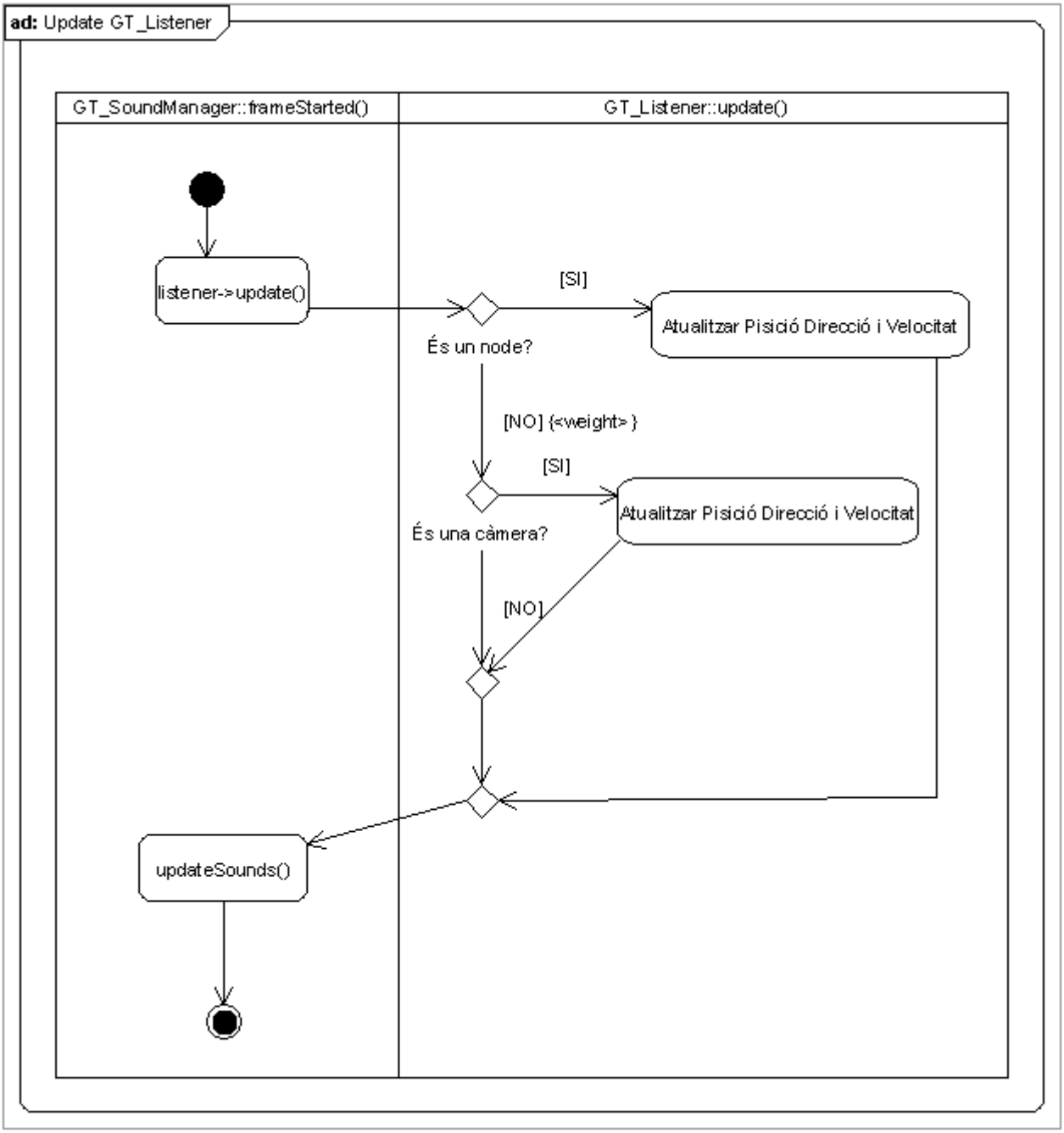


Fig.10 Update del GT_Framelistener

La classe GT_DialogBox:

-La classe **GT_DialogBox** és la encarregada de mostrar un diàleg a l'usuari per a que triï el sistema de render que vol utilitzar. Un cop triat el guarda en un fitxer de configuració i és el que utilitzarà a partir de llavors per a triar el sistema de render de so.

La classe GT_Driver_Factory:

-La classe **GT_Driver_Factory** llegeix el fitxer de configuració creat per la classe **GT_DialogBox** i instancia el driver convenient amb els paràmetres adequats.

3.4.1.2 La Classe GT_Sound i els seus entorns

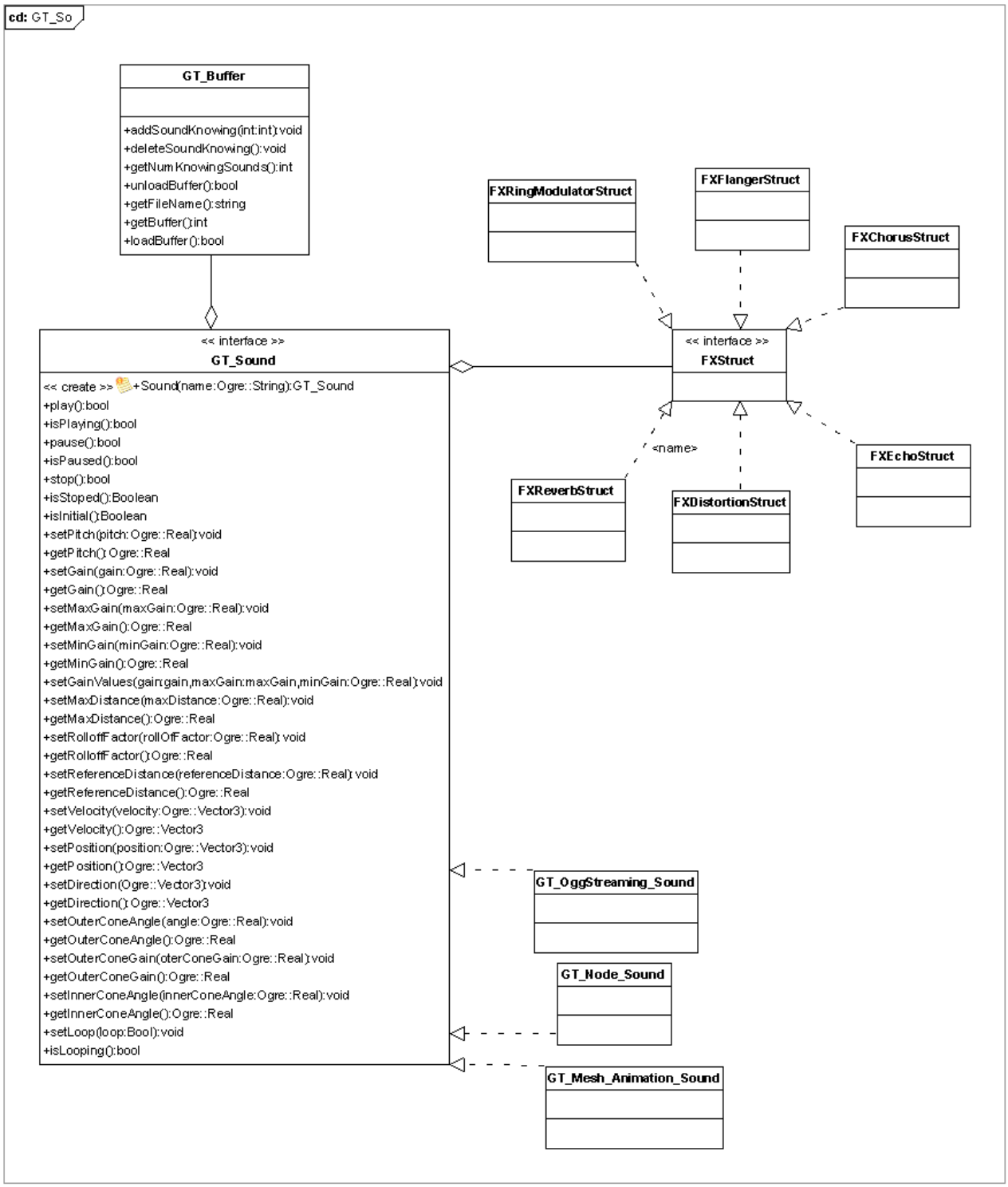


Fig.11 La classe GT_Sound i els seus entorns

La classe GT_Sound:

La classe **GT_Sound** és una classe abstracta i és el so pròpiament dit, es pot pausa, aturar, reproduir,...
Té varis paràmetres que es poden ajustar segons sigui necessari (so amb un con de direcció, amb un guany màxim i mínim,...)

Mai s'ha d'instanciar una classe **GT_Sound** fent un "new" ja que llavors no s'actualitzaria cada frame, per a la creació d'un so s'utilitzarà la classe **GT_SoundManager** que és qui controla l'actualització de les posicions del so.
De la classe **GT_Sound** deriven 3 classes més que són:

GT_Node_Sound, és la classe que es lliga un so a un node i que depenent de la distància serà audible o no.

GT_Mesh_Animation_Sound és la classe que s'utilitza per lligar sons a mesh i animacions de dins de la escena, si per exemple tenim un ninja que produeix un so quan està fent una animació aquest so només serà audible quan s'estigui reproduint la animació.

GT_OggStreaming_Sound és com la classe **GT_Node_Sound** però en comptes de reproduir arxius WAV són OGG.

El programador pot crear noves classes depenent de les seves necessitats, simplement derivant de la classe **GT_Sound** i reimplementant els mètodes que convingui.

També cal esmentar que la classe **GT_Sound** no inicialitza el so quan es crea la classe sinó quan s'ha de reproduir per els altaveus, és a dir, estem prou a prop com per que el so sigui audible, o la animació del mesh s'ha activat. És a dir la classe es converteix en una màquina d'estats la qual pot passar entre els següents estats:

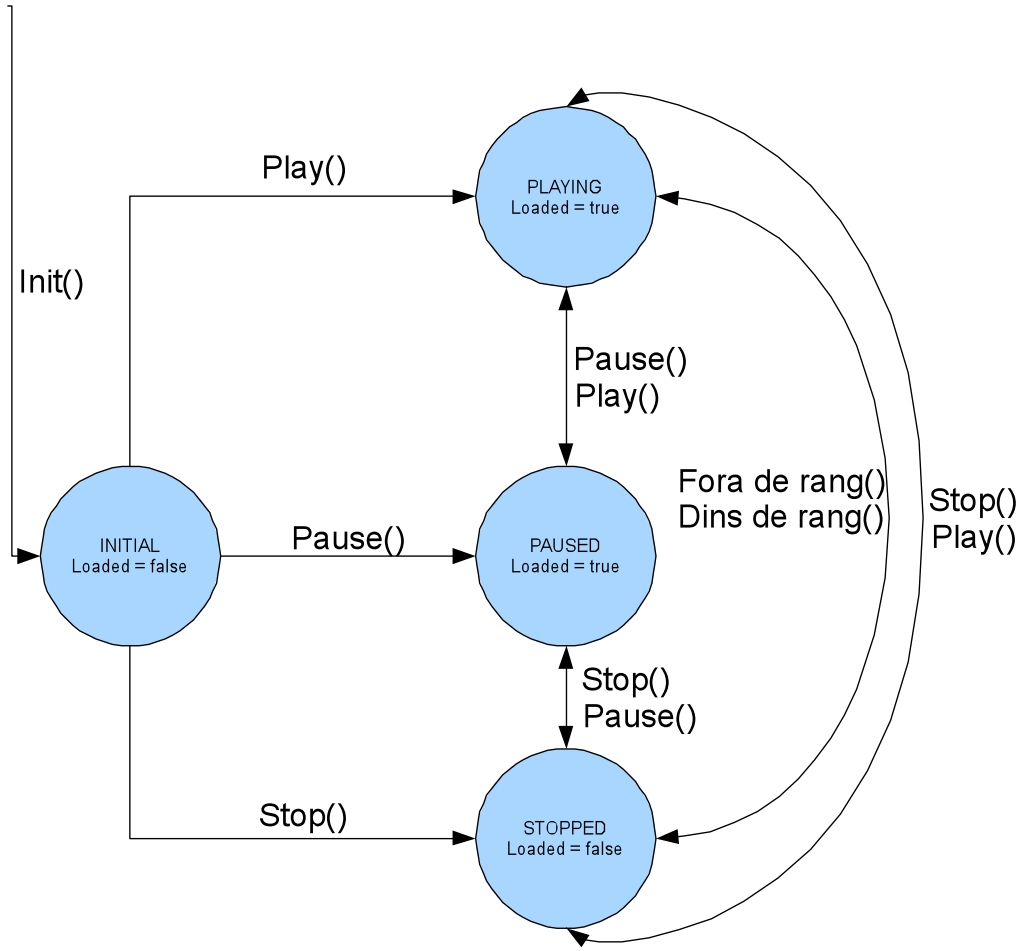


Fig.12 Màquina d'estats de la classe GT_Sound

Podem observar que el so pot passar de PLAYING a STOPPED si entra/surt del rang d'audició del so. D'aquesta manera estalviem memòria de sons que no fagi falta reproduir-los.

Aquest diagrama mostra l'actualització dels sons a cada frame.

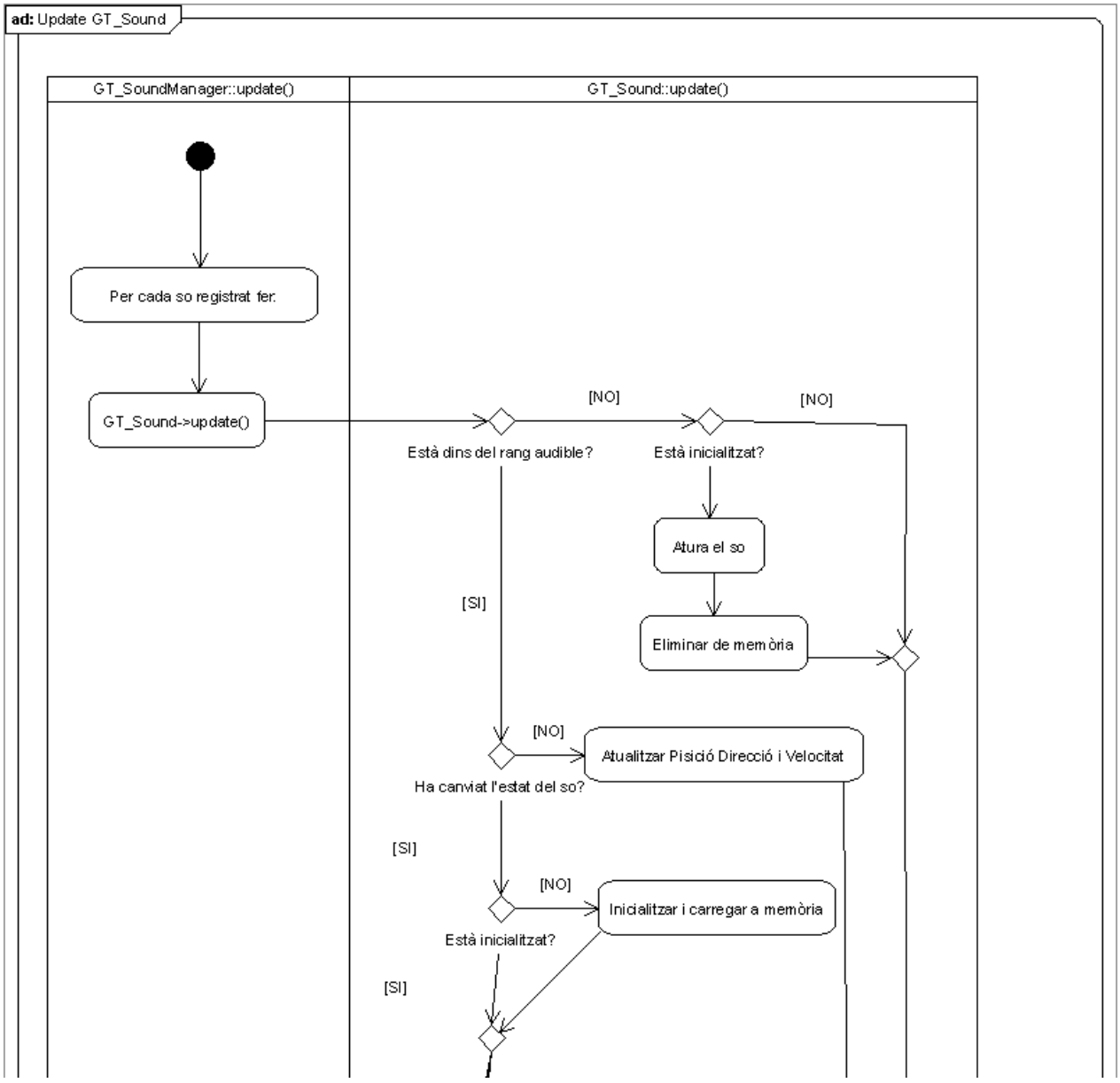


Fig.13 Diagrama d'actualització de les classes GT_Sound Part I

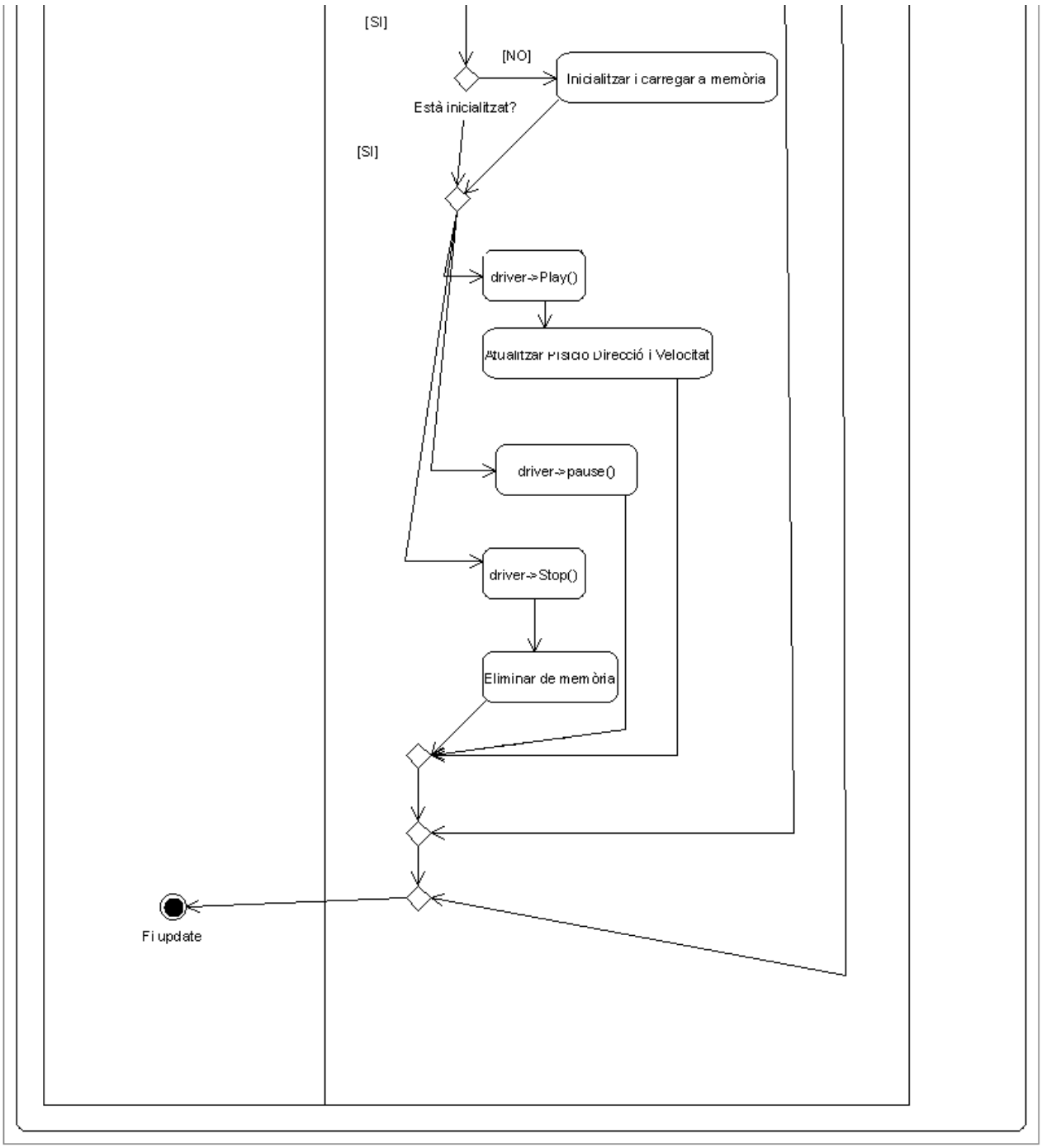


Fig.14 Diagrama d'actualització de les classes GT_Sound Part II

En el següent diagrama mostro com es porta a terme la creació d'un so genèric dins de la classe **GT_SoundManager**.

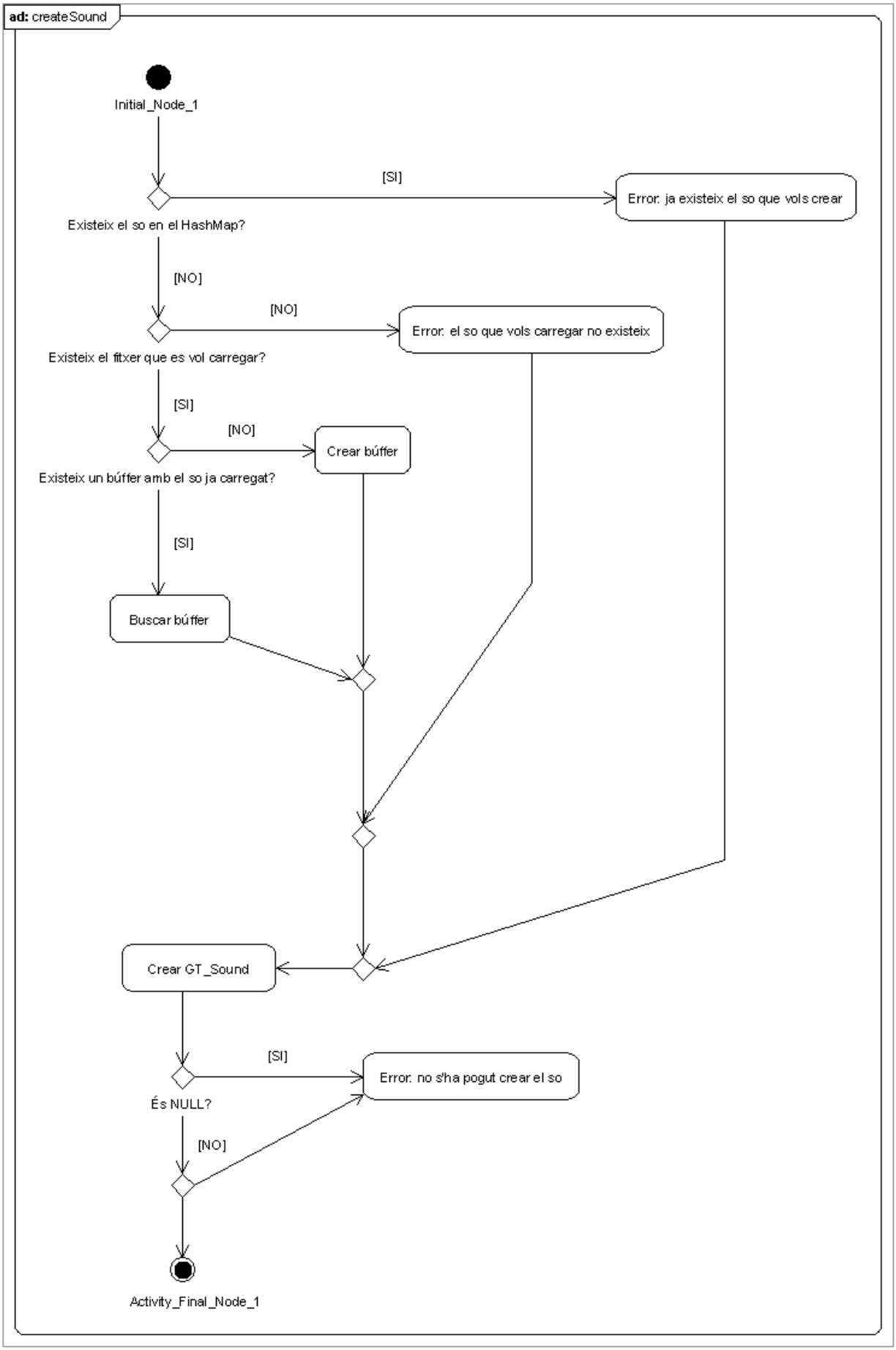


Fig.15 Creació de la classo GT_Sound des de la classe GT_SoundManager

La classe FX_Struct:

La classe **FX_Struct** és una classe abstracta on s'emmagatzemen les variables per a cada efecte de so diferent que es vol reproduir. Depenent de la instanciació podrem tenir un objecte del tipus FXEchoStruct en el qual li podrem passar els paràmetres "Amplitud de la sortida" i "Temps d'espera entre el primer 'tap' i el segon"... depenent de la estructura que instanciem.

Aquesta classe es passa com a paràmetre a la classe **GT_Sound** per quan es vol utilitzar un efecte de so.

La classe GT_Buffer:

-La classe **GT_Buffer** que és la encarregada de manejar els búffers de cada so, s'utilitza per a la compartició de búffers si el driver ho suporta. En el cas que no ho suporti (com és el cas del driver de DirectSound) cada so tindrà el seu espai en memòria.

Aquesta classe té un comptador del nombre de sons que l'apunten, quan el comptador arriba a 0 descarrega el búffer de memòria automàticament per evitar consumir espai.

Cada **GT_Sound** conté un punter a un sol búffer. La classe **GT_SoundManager** té tota la col·lecció de búffers que hi ha a memòria.

Cada cop que es crea un so, la classe **GT_SoundManager** comprova si ja havia creat anteriorment un búffer per aquell fitxer, en cas afirmatiu incrementa el valor del nombre de sons que apunten al búffer en +1.

En el cas que no existeixi cap búffer associat, en crea un de nou.

Cada cop que es destrueix un es resta -1 al búffer i si ha arribat a 0 s'esborra de la memòria.

3.4.1.3 Els drivers i l'abstracció

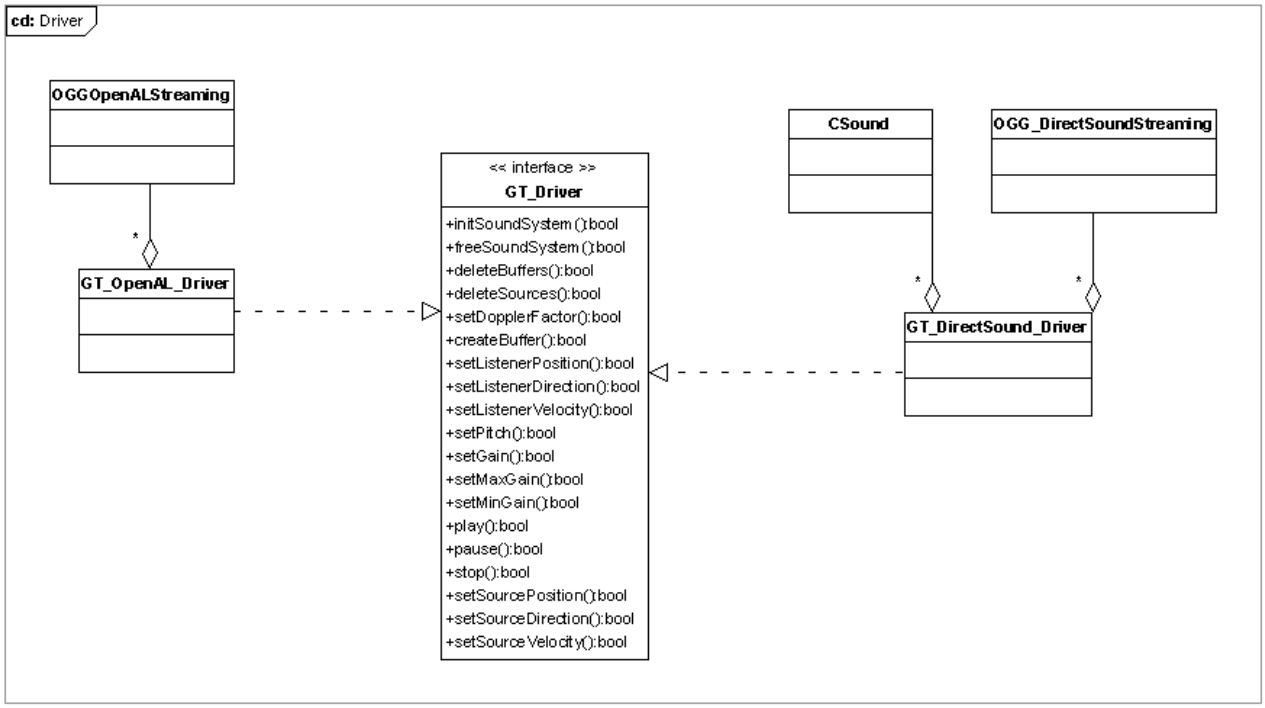


Fig.16 Els drivers de la tarja de so

En aquesta part mostrem les classes que formen part de la capa d'abstracció del driver.

La classe GT_Driver:

-La classe **GT_Driver** és una classe abstracta la qual implementen els drivers que volen que es suportin dins del sistema.

En el nostre cas tenim 2 implementacions:

-GT_OpenAL_Driver per la part dels drivers d'OpenAL.

-GT_DirectSound_Driver per la part dels drivers de DirectSound.

Cada classe utilitza unes altres classes per a implementar la classe abstracta i ajudar-se a organitzar millor.

Aquestes classes depenen de la organització que el programador vol donar per implementar el **GT_Driver**.

En el nostre cas hem creat una classe per manegar els sons OGG i en el cas de DirectSound ens hem ajudat d'una classe que ens proporcionava el propi sistema (Csound).

3.4.2 Diagrama de classe complet:

Aquest és el diagrama de classes complert, aqui es pot observar com cacen totes les classes anteriorment esmentades.

Aquest és el **diagrama de classes complert** que s'ha utilitzat per a lligar amb els diferents drivers que es volien implementar.

Aquest diagrama no és necessari per als que vulguin utilitzar el **SDK** sinó que serveix per a si es vol modificar el comportament intern de la aplicació o afegir noves funcionalitats, nous drivers, nous tipus de sons…

Podem observar que la classe **GT_SoundManager** té 2 hashmap.
El primer guarda el nom dels sons i les classes **GT_Sound** i l'altre guarda el nom del fitxer de música i el búffer que la conté.
També conté un punter a un driver que es va passant entre totes les classes que intenten accedir al sistema de so

Nota: en vermell són les classes que proporciona Ogre.

Nota 2: per a més informació sobre les funcions, mireu la documentació de les classes.

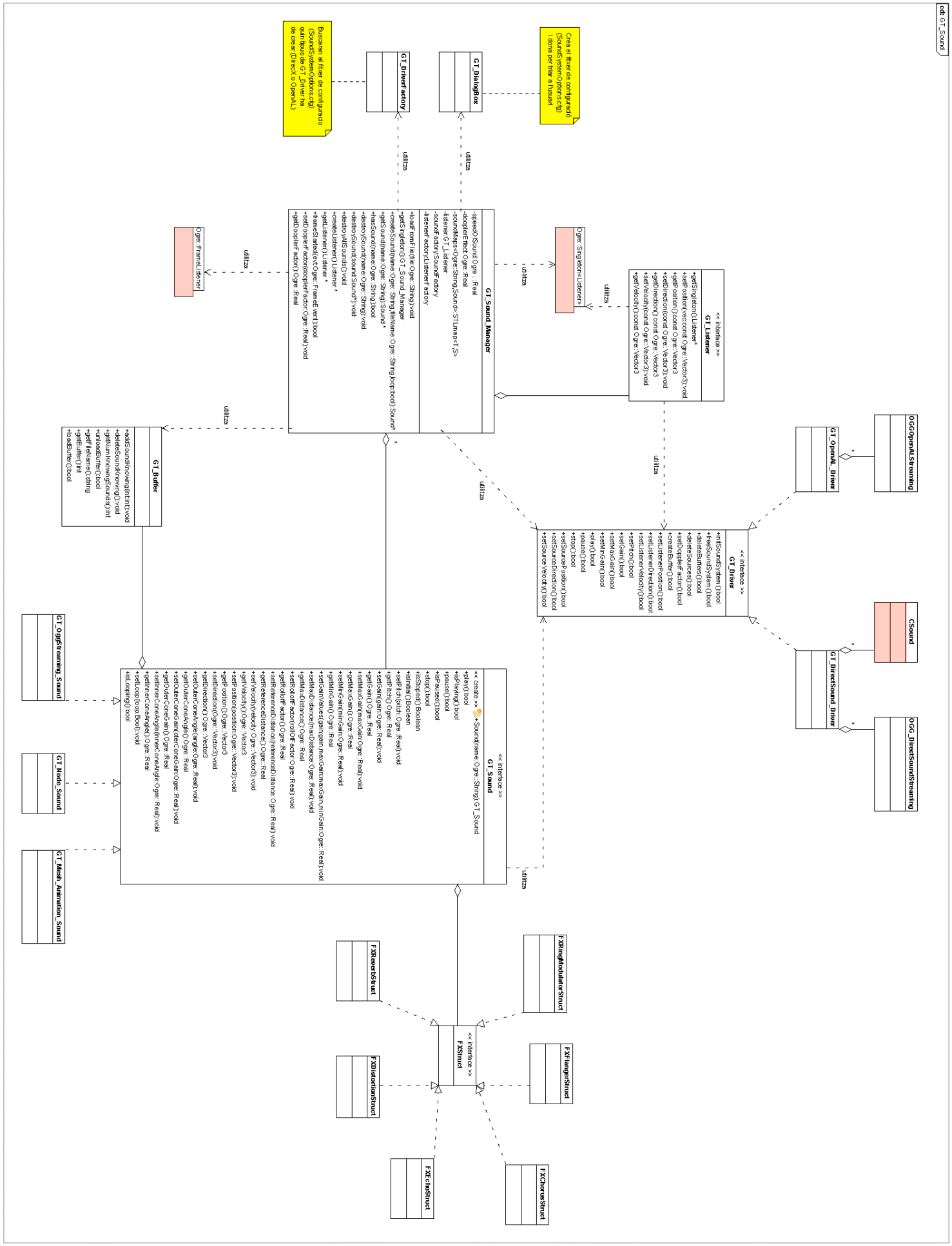


Fig.17 Diagrama de classes complet.

3.5 Patrons de disseny implementats:

En la construcció d'aquest software podem observar els següents patrons de disseny utilitzats per a resoldre els diferents problemes que hem tingut.

3.5.1 Singleton:

Necessitàvem que tant la classe **GT_SoundManager** com la classe **GT_Listener** fossin úniques ja que només es permet la creació d'un listener dins de cada context d'OpenAL i DirectSound, per aquest motiu hem utilitzat el patró **singleton** per al **GT_SoundManager** i el **GT_Listener** ja que només poden haver-hi una sola classe en tot el sistema.

Per implementar això ens hem ajudat de la classe **Ogre::Singleton** la qual registràvem les classes que volíem que fossin úniques per poder accedir-hi i que no es creessin noves classes d'elles mateixes.

3.5.2 Builder:

El procés de creació d'un objecte **GT_Sound** és un procés complicat, a part que la classe **GT_SoundManager** s'ha de guardar la referència. Per això hem utilitzat el patró **Builder** a la classe **GT_SoundManager**, que és l'encarregat de la creació dels sons.

3.5.3 Factory Method:

El patró **Factory Method** l'hem utilitzat per a crear els diferents tipus de driver OpenAL o DirectSound depèn del fitxer de configuració que creem. L'implementa la classe **GT_Driver_Factory**.

3.5.4 Facade:

El patró **Facade**, ens proveeix d'una interfície unificada i simple per a accedir a un grup de classes del sistema. Per fer que els drivers de DirectSound i OpenAL es comportin de la mateixa manera, apliquem una única interface per ambdós, a partir d'aquí cada fabricant que vulgui que es suporti el seu driver haurà d'implementar una única classe (**GT_Driver**).

3.5.5 Proxy:

Ens varem trobar amb el problema que si es carregaven molts sons a memòria el sistema es saturava ràpidament i el driver OpenAL tenia un límit de sons que podia carregar simultàniament. Per aquest motiu varem utilitzat el patró **Proxy** per a la creació dels sons en la tarja de so ja que d'aquesta manera només carreguem a memòria els sons que estrictament s'han de reproduir, d'aquesta manera s'aconsegueix reduir el cost de memòria que suposa tenir un so a memòria carregat, a part que el driver d'OpenAL té una limitació amb el nombre de sons que pot tenir emmagatzemats i que pot reproduir.
La classe que implementa aquest patró és la **GT_Sound** que només en el cas que es vulgui escoltar pels altaveus un so, no l'instanciarà.

4-Implementació:

En aquest apartat d'implementació comentarem alguns dels problemes més rellevants que s'han plantejat i quines han estat les solucions aportades.

4.1 Format del fitxer de propietats:

En aquest projecte necessitàvem guardar un fitxer amb la configuració que havia triat l'usuari/programador per instanciar el driver OpenAL o DirectSound.

Primer de tot varem crear un dialog box amb l'ajuda de Microsoft Visual Studio en la qual hi haguessin 4 drop list per poder seleccionar:

1. L'enginy de render de so (OpenAL o DirectSound).
2. El controlador de sortida de DirectSound.
3. El controlador d'entrada de DirectSound (tot i que no es fa servir en aquest projecte).
4. El controlador d'entrada/sortida d'OpenAL.

Els drop list de DirectSound estan bloquejats ja que DirectSound agafa la sortida que millor s'ajusta a les preferències que el sistema operatiu té configurat. En canvi OpenAL deixa triar depenent si el sistema té vàries targes de so.

Fig.18 Tria del sistema de render de so.

El format del fitxer de propietats s'utilitza per tenir guardat quines són les preferències de l'usuari sobre quin és el driver que vol utilitzar, i quins són els dispositius d'entrada/sortida que utilitzarà.

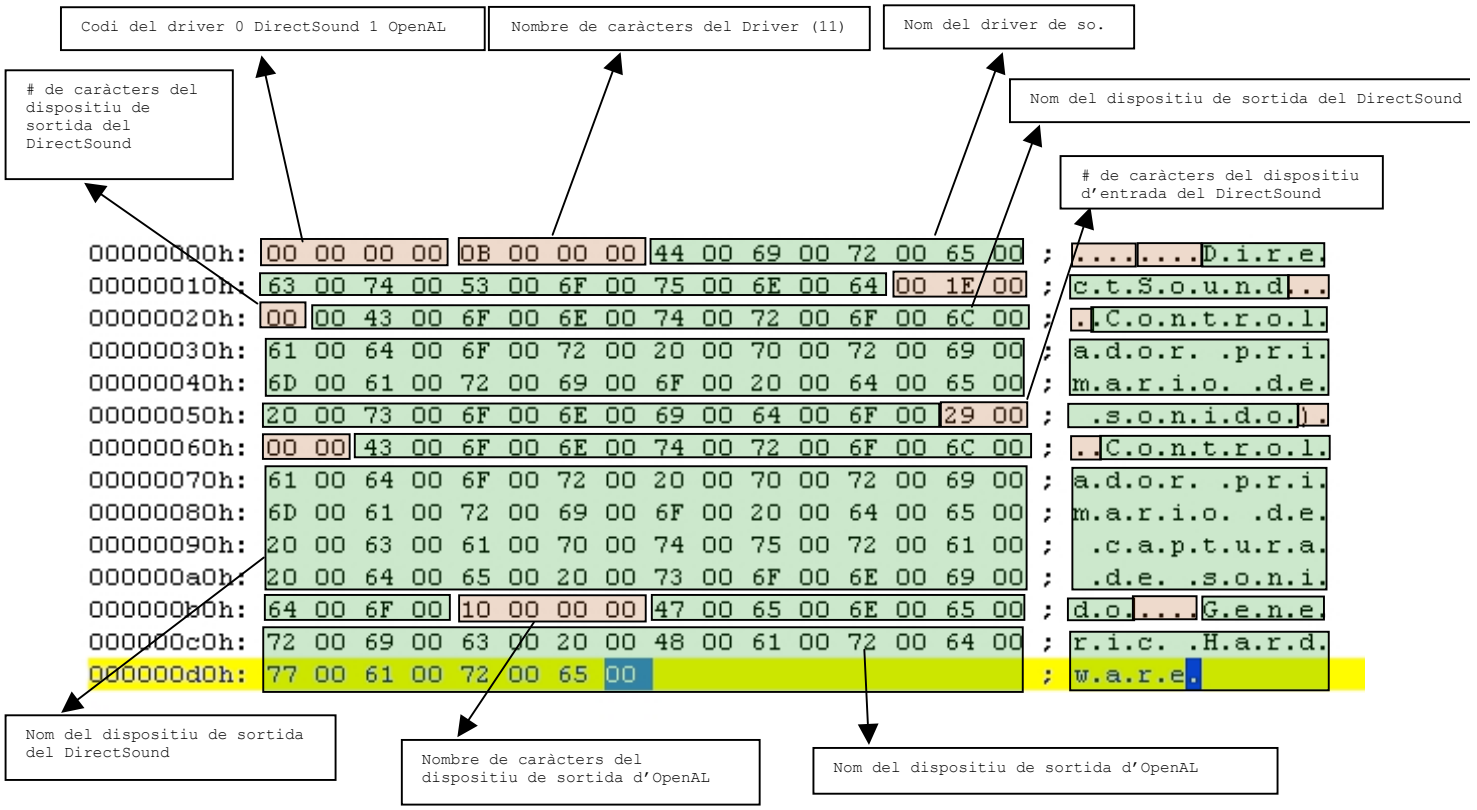
La classe GT_SoundManager quan s'inicialitza demanarà a l'usuari que triï quin sistema de render vol utilitzar sempre que no trobi el fitxer de configuració ("SoundSystemOptions.cfg") a la carpeta des de on s'executa l'aplicació.

Un cop l'usuari pitja OK es genera el següent fitxer de configuració.

S'ha triat utilitzar funcions a baix nivell (fopen, fread, ...) en comptes de crear un fitxer XML per evitar sobrecarregar més el sistema amb dll, ja que és molt més senzill d'aquesta manera.

És possible, però, que si el sistema creix més s'hagi de reimplementar utilitzant XML per millorar el conjunt de propietats que poden agafar els diferents sistemes de so.

En el fitxer que es crea és el següent. S'ha obert amb un editor hexadecimal per poder observar els valors dels enters (color vermell) i els dels caràcters (color verd).



Nota: el fitxer no es veu correctament en el notepad ja que el tipus de dades que hem guardat són del tipus wchar_t ja que les llibreries DirectSound obliguen a aquest tipus de dades en comptes de char*.

Wchar_t és un tipus de dades que s'utilitza per a guardar tipus de caràcters Unicode. Aqui es pot trobar més informació: http://www.conclase.net/c/librerias/estructura.php?tip=lwchar_t

4.2 Efectes de so:

Un problema que ens varem trobar a l'apartat d'efectes de so és que hi ha efectes de sons que no existeixen amb OpenAL i viceversa, això va provocar que haguéssim de triar els efectes de so que fossin comuns en ambdós drivers.

La taula amb la correspondència entre els efectes sonors d'OpenAL i DirectSound és la següent. També s'hi adjunta una descripció de què fa cada efecte:

OpenAL	DirectSound	Descripció:
AL_EFFECT_CHORUS	DSFXChorus	L'efecte Chorus, és el resultat de barrejar una senyal amb vibrato amb la senyal sense processar. El resultat és similar al de un parell d'instruments que toquen a l'uníson de tal manera que un d'ells es desafina lleugerament.
AL_EFFECT_DISTIORTION	DSFXDistortion	Afegeix una distorsió i un filtre passa-baixos semblants als de les guitarres elèctriques.
AL_EFFECT_ECHO	DSFXXEcho	Efecte eco que es produeix per la reflexió de les ones i que retorna atenuada.
AL_EFFECT_FLANGER	DSFXFlanger	Barreja de la ona original amb una còpia molt poc retardada i amb una freqüència diferent.
AL_EFFECT_RING_MODULATOR	DSFXGargele	Multiplica la senyal d'entrada per una senyal portadora, el resultat és un tremolo o un efecte harmònic.
AL_EFFECT_EAXREVERB	DSFXI3DL2Reverb	La reverberació és un fenomen derivat de la reflexió del so consistent en una lleugera prolongació del so un cop s'ha extingit l'original, degut a les ones reflexades. Aquestes ones reflexades patiran un retard no superior a 100ms que és el valor de persistència acústica, temps que correspon a una distància de 34 metres a la velocitat del so. Quan el retard és més gran parlem d'eco.

També varem trobar el problema que cada efecte sonor tenia uns paràmetres que variaven en funció del driver.

Per a solucionar varem fer unes taules que reunien els paràmetres que es feien servir per el mateix i en quines unitats es representaven.

Per exemple la següent taula mostra la correspondència entre els valors de l'efecte chorus en OpenAL i DirectSound, una descripció de cada paràmetre i sobre quin rang de valors treballen:

AL_EFFECT_CHORUS	DSFXChorus	Descripció:	Rang de Valors:
AL_CHORUS_WAVEFORM	LONG lWaveform	Forma de la senyal que controla el temps d'espera de la senyal.	OpenAL: 0-sin 1-triangle default: 1 DirecSound: DSFXCHORUS_WAVE_TRIANGLE o DSFXCHORUS_WAVE_SIN default sin
AL_CHORUS_PHASE	LONG lPhase	Fase diferencial entre les LFO's esquerra i dret.	OpenAL: -180 fins 180 default:90 DirecSound: DSFXCHORUS_PHASE_MIN a DSFXCHORUS_PHASE_MAX. Default DSFXCHORUS_PHASE_90
AL_CHORUS_DELAY	FLOAT fDelay	Temps d'espera fins que es torna a reproduir el so.	OpenAL: 0.0 fins 0.016 default: 0.016 DirecSound: DSFXCHORUS_DELAY_MIN a DSFXCHORUS_DELAY_MAX default 16ms
AL_CHORUS_DEPTH	FLOAT fDepht	Percentatge entre en el que el temps d'espera és modulat.	OpenAL: 0 fins 1 default: 0.1 DirecSound: DSFXCHORUS_DEPTH_MIN a DSFXCHORUS_DEPTH_MAX default 10
AL_CHORUS_FEEDBACK	FLOAT fFeedback	Controla la quantitat de senyal	OpenAL:

AL_EFFECT_CHORUS	DSFXChorus	Descripció:	Rang de Valors:
		que alimenta el chorus.	-1 fins 1 default: 0.25 DirecSound: DSFXCHORUS_FEEDBACK_MIN a DSFXCHORUS_FEEDBACK_MAX default 25

Un cop fetes totes les taules de correspondència havíem de triar quin dels dos drivers era el que feia la conversió entre els valors d'una i els valors de l'altre.
Per exemple dins de la funció d'afegir efecte echo prèviament s'ha de transformar els valors que es passen per paràmetre per ajustar-los als valors que accepta el driver DirectSound:

```
void GT_DirectSound_Driver::modifyChorusValues( Ogre::Real *values ){
          //Si és 0 posem el valor DSFXCHORUS_WAVE_TRIANGLE en cas contrari DSFXCHORUS_WAVE_SIN
          values[0] = values[0] ? DSFXCHORUS_WAVE_TRIANGLE : DSFXCHORUS_WAVE_SIN;

          //Modifiquem el valor d'OpenAL que va des de -180 fins a 180 per que vagi de 0 a 4
          values[1] = (values[1]+180)/90;

          //Modifiquem el valor d'OpenAL que va des de 0 fins a 0.016 per que vagi de 0 a 160
          values[2] = values[2]*1000;

          //Modifiquem el valor d'OpenAL que va des de 0 fins a 1 per que vagi de 0 a 100
          values[3] = values[3]*100;

          //Modifiquem el valor d'OpenAL que va des de -1 fins a 1 per que vagi de -100 a 100
          values[4] = values[4]*100;
}
```

A la part de l'annex s'hi ha adjuntat les taules amb les referències entre els paràmetres dels diversos efectes de so que s'han implementat.
En el cas que un desenvolupador volgués afegir un so que no estigués en ambdós drivers, hauria de tenir en compta

4.3 Obstrucció de sons:

Per finalitzar la implementació, hem afegit una funcionalitat la qual es pot cridar des de cada so en la que es controla si hi ha línia de visió directe entre la font emissora de so i l'oient; en el cas que no hi hagi línia de visió directa, el so s'atenuarà en funció del paràmetre que se li passa a la funció.

```
so->setObstructionOnOff( 0.5, mSceneMgr );
```

El primer paràmetre és l'atenuació que s'hi aplicarà i el segon paràmetre és l'Scene Manager el qual buscarà la obstrucció dels sons.

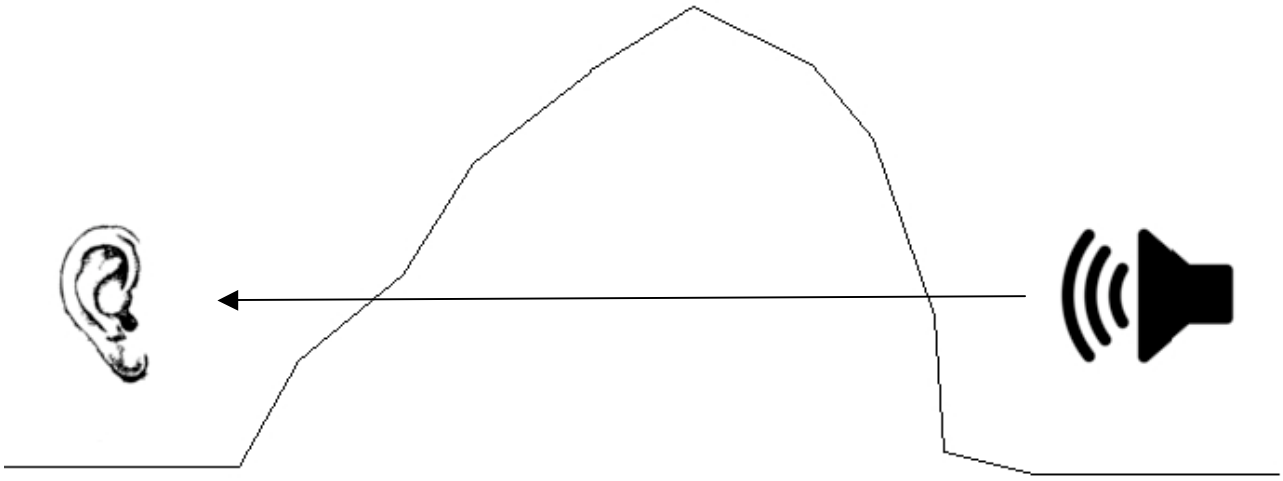


Fig.19 Representació sobre com s'ha dut a terme la obstrucció de

La implementació es fa llençant un raig des de l'emissor del so fins el receptor. Es calcularà simplement si hi ha algun objecte entre mig i en cas afirmatiu es canvia el paràmetre del guany actual per el definit pel programador.

Hi ha altres mètodes que es poden fer servir, per millorar l'algorisme anteriorment esmentat i pot ser una bona línia per començar/continuar investigant.

Per a més informació es pot llegir la documentació d'OpenAL SDK anomenada Effects Extension Guide el capítol introductori (Pàgines 8 a 18).

5- Fase de proves de l'aplicació:

5.1 Jocs de proves:

Un cop finalitzada la implementació s'han realitzat un conjunt de jocs de proves per comprovar diferents aspectes:

1. Detecció d'errors en la implementació.
2. Com s'escolten els sons amb les dues llibreries escollides.
3. Comprovació de totes les funcionalitats implementades.
4. Velocitats dels diferents drivers i espai en memòria que ocupen.

5.1.1 Joc de proves 1:

En aquest joc de proves el que volem intentar demostrar és la absència d'errors en una aplicació amb paràmetres correctes i paràmetres incorrectes per veure quin comportament tenen les classes, quines excepcions salten,...

Els sons que es presenten en aquesta demo són sense espacialització (2D) ja que les funcions sobre les que ens centrem són la creació, la destrucció, el guany, el canvi de freqüència,...

En aquest joc de proves farem un conjunt de proves a les funcions de :

- Creació del **GT_SoundManager**.
- Obtenció d'un punter al del singleton del **GT_SoundManager**.
- Posar els diferents valors de l'efecte doppler per comprovar quan falla,...
- Creació d'un so WAV inexistent.
- Creació d'un so1 WAV existent.
- Posar un con interior, exterior i un guany.
- Reproduir el so1.
- Canviar el guany màxim, el guany mínim, i tornar a deixar-lo com estava.
- Canviar el pitch (freqüència) del so.
- Creació d'un so OGG inexistent.
- Creació d'un so2 OGG existent.
- Posar un con interior, exterior i un guany.
- Reproduir el so2.

- Canviar el guany màxim, el guany mínim, i tornar a deixar-lo com estava.
- Canviar el pitch (freqüència) del so2.
- Pausar el so1, reproduir el so1, pausar el so2, aturar el so1, destruir el so1, reproduir el so2, destruir el so 2 i destruir la tots els sons (cap).

5.1.1.1 Conclusió:

El resultat del test ha estat satisfactori, les funcions que havien de fallar han fallat (per exemple si creem un so amb un arxiu que no existeix) i les funcions que havien de funcionar ho han fet correctament i els sons s'han escoltat tal com s'esperaven.

Els dos drivers s'escolten igual és a dir no hi ha diferència entre un driver o un altre en aquesta prova.

5.1.2 Joc de proves 2:

Aquest joc de proves es desenvolupa per verificar la espacialització 3D dels sons, sons amb cons de volum, canvi del driver de so, comprovar la utilització dels sons en animacions, sons amb efectes de reverberació, eco, flanger,... i atenuació per causa d'obstruccions del terreny entre l'emissor de so del Ninja2 i l'oient.

Aquest joc de proves ens trobem en un món amb un terreny en el qual hi ha 2 ninjes que quan es mouen emeten so.

Les tecles configurades són les següents:

Per moure la càmera s'ha de mantenir pitjat el boto dret del ratolí.

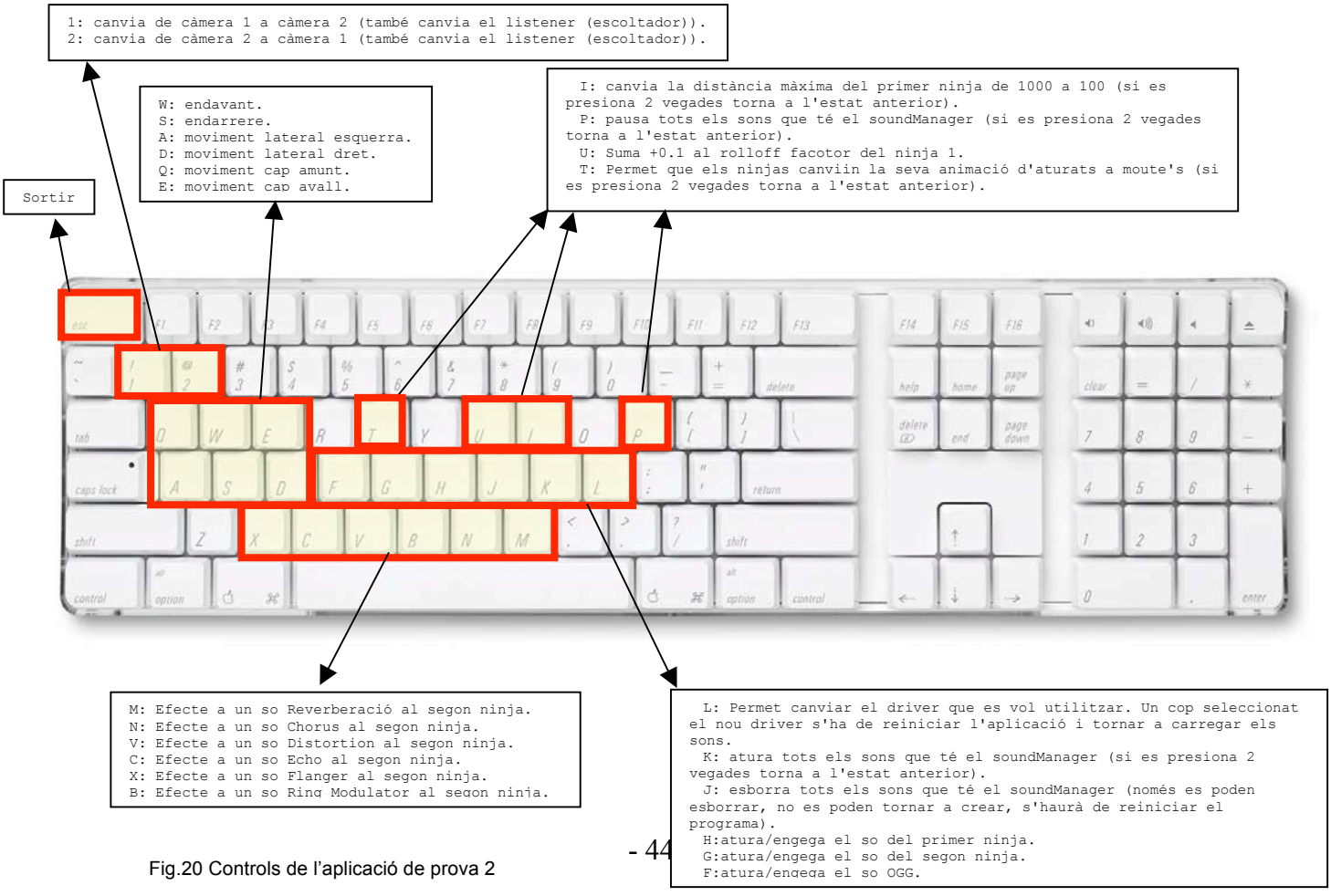


Fig.20 Controls de l'aplicació de prova 2

Fig.21 Captura de pantalla de l'aplicació de prova 2

5.1.6.2.1 Conclusió:

Els resultats de l'aplicació són satisfactoris, s'han trobat alguns errors i bugs, però s'han solucionat. Observem que en OpenAL el so és més intens i més vibrant que amb el driver de DirectSound.

També és molt important destacar que els efectes de so amb OpenAL tenen una qualitat molt millor que els de DirectSound. Això es deu a que el driver DirectSound processar els sons via software i la qualitat dels algoritmes són pitjors, per tant la qualitat dels sons deixa molt que desitjar.

DirectSound pot reproduir tots els efectes de so, mentres que OpenAL depèn del hardware sobre el que s'està executant pot (o no) reproduir-los.

5.1.3 Joc de proves 3:

Aquest joc de proves s'ha realitzat per comprovar la fàcil utilització de les llibreries en un "minijoc" en el que has de matar tants fantasmes com siguin possibles, la dificultat es troba en que pràcticament no es veu res i t'has de guiar pel que es sent als auriculars.

També està pensat per comprovar el correcte funcionament de les funcions de rolloff.

En aquesta demo, l'oient està situat en el node del ninja en comptes d'estar situat a la càmera com ho havíem fet en l'anterior demo.

Fig.22 Controls de l'aplicació de prova 3

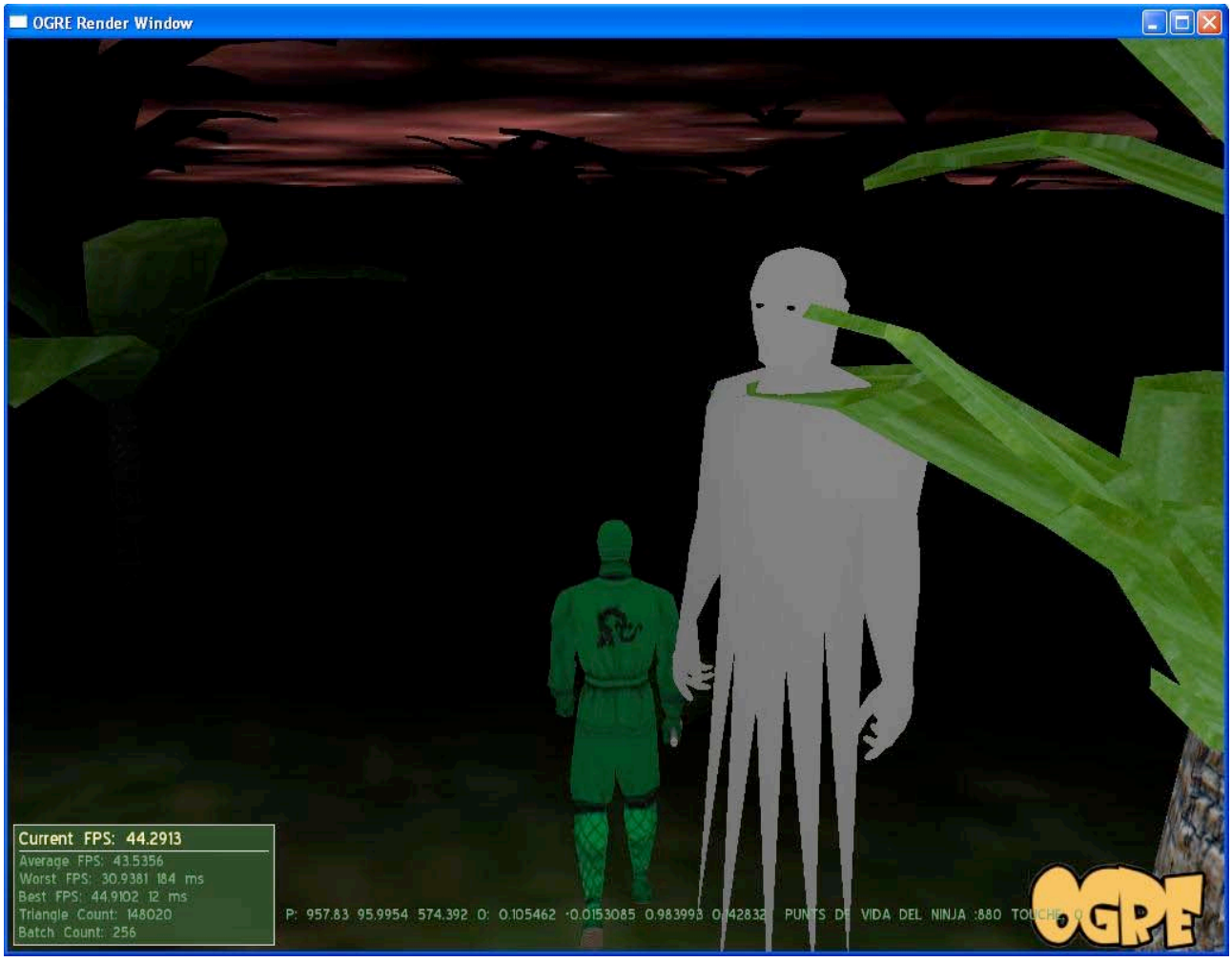


Fig.23 Captura de pantalla de l'aplicació de prova 3

5.1.3.1 Conclusió:

Hem observat novament la diferència entre la qualitat de so de DirectSound i la d'OpenAL.

Hem observat també que per programar aquesta demo s'ha necessitat més temps per trobar els sons i els models 3D que per programar l'apartat de sons, ja que la abstracció esmentada ho fa molt fàcil per al programador.

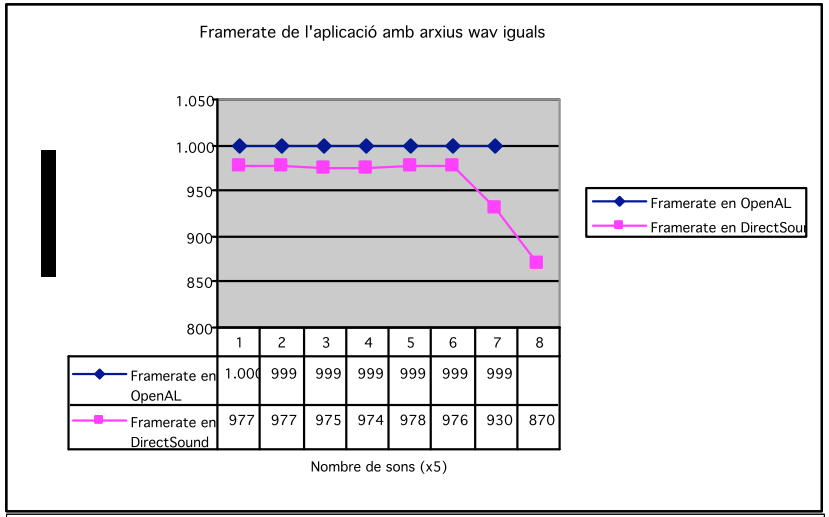
5.1.4 Joc de proves 4:

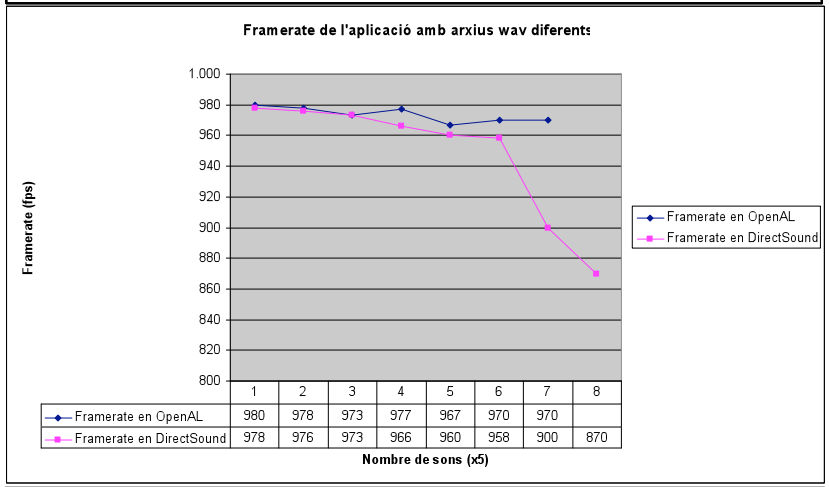
Degut a la complexitat d'analitzar els diferents sistemes de so s'utilitzarà aquesta aplicació per fer les següents proves:

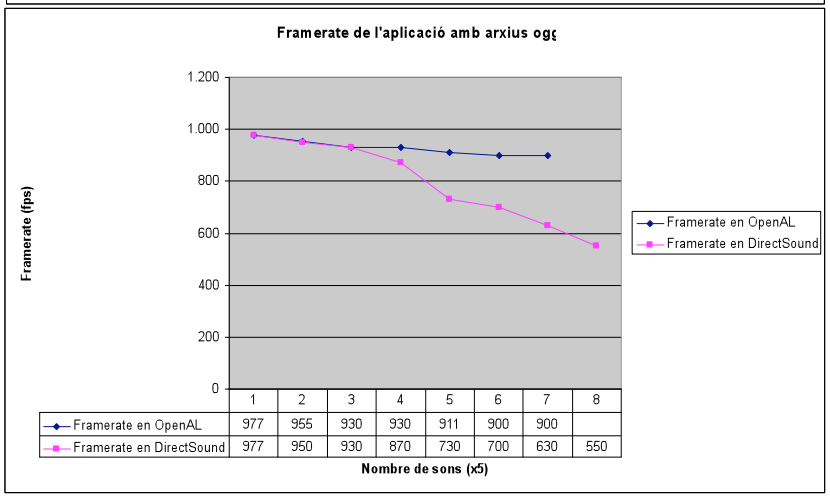
En aquesta aplicació podem afegir/treure sons WAV que comparteixen búffer (en el cas de OpenAL), sons **WAV** que no comparteixen búffer, i sons **OGG**.

Totes les proves es faran en una aplicació **Ogre3D** amb el mínim de triangle per renderitzar, amb una optimització de l'executable per al processador, per assegurar que pràcticament tot el que es processa sigui del sistema de so.

Els sons que es carreguen són "**roar.wav**" **230KB** i "**King.ogg**" **2,986KB**, la mida del búffer del so **OGG** és de **65500bytes**.







-Comprovarem el framerate en funció del nombre de sons que es reprodueixen

Nota: l'eix de coordenades X és el nombre de sons multiplicat per 5.

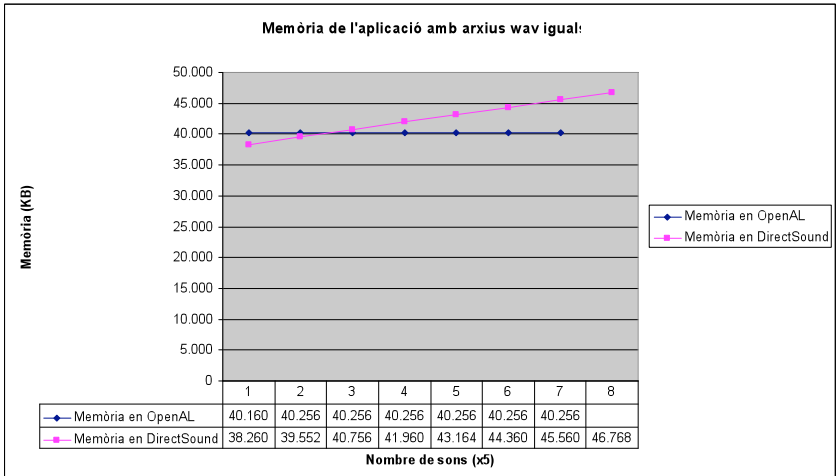
Aquests testos serviran per determinar quina és la càrrega de la CPU mitjançant el framerate de la aplicació, com menys framerate tingui l'aplicació significa que està destinant més recursos a processar el sistema de so que el sistema de render i per tant baixarà el framerate.

Tal com podem observar en la primera i la segona gràfica tant el framerate d'**OpenAL** com el de **DirectSound** són bastant semblants fins que arribem als 30 sons, a partir de llavors el framerate del **DirectSound** degenera i es torna lineal amb una pendent negativa.

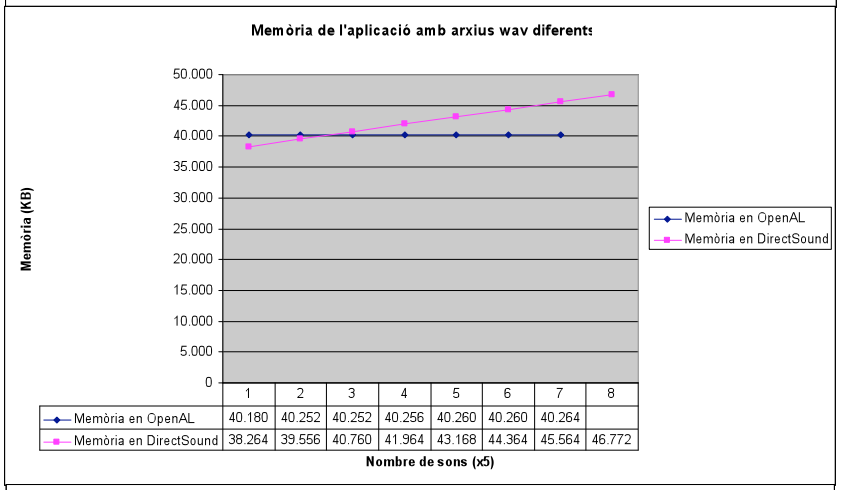
El framerate de **OpenAL** sempre és millor que el de **DirectSound**.

En el cas de l'aplicació OGG tots dos sistemes de render de so disminueixen el seu framerate però **OpenAL** ho fa molt menys que **DirectSound** amb molta presència de sons.

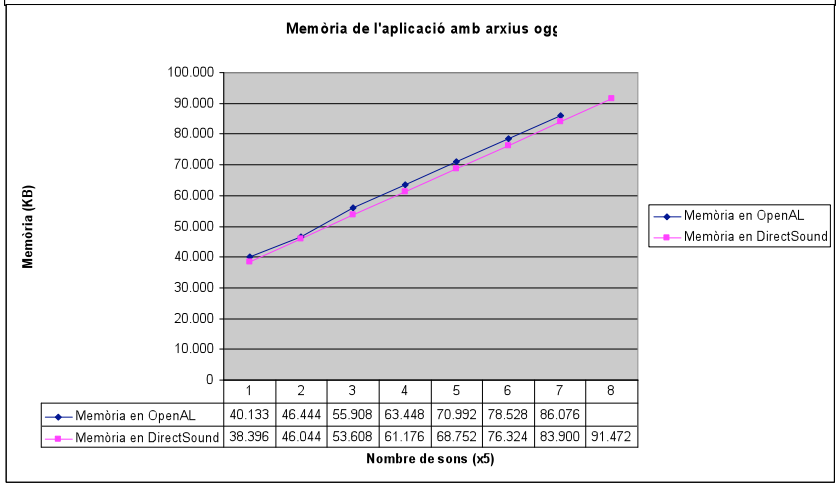
Cal destacar que **OpenAL** només pot reproduir fins a 35 sons mentres que **DirectSound** no té límit.



	1	2	3	4	5	6	7	8
Memòria en OpenAL	40.160	40.256	40.256	40.256	40.256	40.256	40.256	
Memòria en DirectSound	38.260	39.552	40.756	41.960	43.164	44.360	45.560	46.768



	1	2	3	4	5	6	7	8
Memòria en OpenAL	40.180	40.252	40.252	40.256	40.260	40.260	40.264	
Memòria en DirectSound	38.264	39.556	40.760	41.964	43.168	44.364	45.564	46.772



	1	2	3	4	5	6	7	8
Memòria en OpenAL	40.133	46.444	55.908	63.448	70.992	78.528	86.076	
Memòria en DirectSound	38.396	46.044	53.608	61.176	68.752	76.324	83.900	91.472

-Comprovarem la memòria utilitzada en funció del nombre de sons que es reprodueixen

Nota: l'eix de coordenades X és el nombre de sons multiplicat per 5.

Aquests testos tenen com a funció comprovar la memòria utilitzada per l'executable i posteriorment afegint sons.

Hem de matisar que el driver **OpenAL** comparteix els diferents búffers WAV, és a dir, que si dos sons han de reproduir el mateix fitxer, **OpenAL** només utilitzarà un sol búffer. Per altra banda el driver **DirectSound** cada so té el seu búffer.

Tal com podem observar tant en el primer gràfic com en el segon, **OpenAL** manté **constant** la memòria **RAM** utilitzada. Això succeeix perquè **OpenAL** emmagatzema els sons en la memòria de la tarja de so **X-RAM**, en el cas que la tarja de so no tingués memòria l'emmagatzemaria en memòria **RAM** del PC.

Pel contrari **DirectSound** no aprofita la característica de la memòria en la tarja de so i cada cop que s'insereix un so el carrega en memòria principal. És per això que la inserció de sons és **lineal en funció del nombre de sons**.

Amb els sons OGG tant **OpenAL** com **DirectSound** la seva inserció provoca un augment de la memòria principal ja que ningú utilitza la memòria de la tarja de so i cada so no comparteix el búffer de so a causa que es reprodueix en streaming el qual fa impossible aquest fet.

5.1.5 Altres proves realitzades:

Aquí mostrem algunes de algunes proves que hem realitzat al marge de les anteriors per comprovar les capacitats d'ambdós drivers.

-Quin és el màxim nombre de sons (iguals/diferents) que suporta el sistema de so:

El màxim nombre de sons que permet **OpenAL** varia d'una tarja a una altra però la majoria suporten entre 30 i 128 sons simultanis.

DirectSound no té límits de so, es poden carregar tants sons com memòria tingui el pc.

-Quina és la mida màxima de l'arxiu WAV que suporta el driver de so:

Pel que podem experimentar **OpenAL** posa un màxim de 30MB a un únic arxiu mentres que **DirectSound** no té màxim.

6- Conclusions

L'objectiu del projecte era implementar un envoltori per aïllar al programador del driver que estigués utilitzant. Això s'havia d'integrar amb el motor de render Ogre3D.

Hem utilitzat les llibreries OpenAL i DirectSound i hem agafat les funcionalitats comunes que tenien.

Hem creat una llibreria (.lib) que el programador pot utilitzar per a incloure de manera estàtica dins dels seus projectes.

Podem dir que s'han assolit tots els objectius que ens varem plantejar abans de la realització i durant l'estudi del projecte, les característiques del qual són les següents:

- **So Surround**. Suport per a stereo, headphone, 4 canals, so 5.1 i 7.1.
- **Espacialització 3D**. Els sons s'escoltaran com si estiguessin en un entorn 3D.
- **DSP**. Un ampli conjunt d'efectes que es poden aplicar als sons en temps d'execució, tals com reverberació, eco, ...
- Capacitat per reproduir **formats d'àudio comprimit**. Suport per a streaming del format OGG-Vorbis.
- **Efecte doppler.**
- **Atenuació** dels sons en relació a la distància.
- **Modificació** de la freqüència dels sons.
- **Sons direccionals** amb atenuació fora dels cons.
- **Obstrucció/atenuació** amb el terreny.
- **Elecció del sistema de render de so** (OpenAL o DirectSound) per part del programador/usuari.
- Els sons que es troben fora del rang per a ser escoltats són apagats **automàticament** i eliminats de memòria**.**

Hem afegit a més 4 demostracions/testos fets amb les llibreries de so per comprovar el correcte funcionament d'aquestes.

Els mòduls i classes són independents entre ells i permeten la fàcil addició de noves funcionalitats i noves llibreries. De fet aquest projecte està pensat per a futures recerques i investigacions de posicionament del so i algoritmes d'obstrucció.

En el cas que es volguessin eines de domini públic o multiplataforma, es podria recompilar el codi només utilitzant el driver d'OpenAL (ja DirectSound és propietari de Microsoft i com a tal només corre sobre Windows).

6.1 Millores futures:

Aquestes són algunes de les millores futures que es podrien afegir en projectes futurs per a

millorar-lo.

- Captura de so amb el micròfon.
- Fer una classe que faci streaming d'arxius WAV.
- Poder reproduir arxius MP3.
- Millor selecció del sistema de so que permeti seleccionar la freqüència dels arxius de so que es reproduiran.
- Crear un editor de sons amb el qual es puguin afegir sons en el videojoc de forma més interactiva creant fitxers XML.
- Un millor manegament dels efectes de so, ja que ara per ara s'han de posar i treure manualment. Es podria millorar fent que els efectes de so estiguessin definits dins d'un cub 3D i quan un so entres dins del cub s'apliqués de forma automàtica l'efecte. Tal com està definit ara és el programador el que li ha de dir quan es vol que s'activi i es desactivi un efecte de so.
- Millorar el sistema perquè de forma automàtica llançant rajos al voltant d'un so ell pugui decidir quin efecte o amb quina intensitat s'ha d'aplicar un efecte.
- Per exemple si un so entra en una església llençarà uns quants rajos per fer una mitja sobre quin espai es troba i llavors es podria trobar els valors per posar dins d'un efecte de reverberació.
- Separar els diferents drivers en llibreries que es carreguin dinàmicament (.dll) depenent del driver que es seleccioni

7-Bibliografía

7.1 C++:

http://www.conclase.net/c/

http://www.zator.com/Cpp/index.htm

http://msdn2.microsoft.com/es-es/default.aspxhttp://www.zator.com/Cpp/E1_4_4b.htm

http://arco.inf-cr.uclm.es/~david.villa/doc/repo/librerias/librerias.html

7.2 Llibreries de render

7.2.1 Ogre3D:

[GRE06] Gregory Junker , "*Pro Ogre3D programming*", Apress, 2006

http://www.ogre3d.org

http://www.ogre3d.org/wiki/

http://www.ogre3d.org/phpBB2/

7.3 Llibreries de so:

7.3.1 OpenAL:

http://www.openal.org/

http://www.devmaster.net/articles.php?catID=6

http://worldspace.berlios.de/openal/index.html

http://www.edenwaith.com/products/pige/tutorials/openal.php

http://developer.creative.com/articles/article.asp?cat=1&sbcat=31&top=38&aid=147

http://www.ogre3d.org/phpBB2/viewtopic.php?p=100754

7.3.2 General:

http://www.insomniacgames.com/tech/techpage.php

http://www.codepixel.com/content/view/4690/29/

http://niozero.blogspot.com/2007/10/physic-sound-library.html

http://www.maximopc.org/articulos/sb_x-fi_xtreme_music_p1.html

http://www.ojgames.com/article/articleview/956512/

8-Annex

8.1 Quan utilitzar DirectSound o OpenAL:

Molts cops ens haurem de decantar cap a una driver o un altre depenent sobre quin hardware volem que corri la nostra aplicació. En aquest apartat dono les avantatges i les desavantatges d'utilitzar un o l'altre.

-**DirectSound** s'utilitzarà en els casos que vulguem aplicar Efectes ja que no tothom té targes de so **Creative** o que suportin **EAX**.
-**OpenAL** s'utilitzarà en la resta de casos ja que és més estable, implementa tot el que pot per hardware i és més independent de la freqüència i bitrate dels sons, també és millor perquè comparteix els sons en memòria cosa que el **DirectSound** no fa.

-Quant utilitzar ogg o wav:

-**Ogg** s'utilitzarà quan s'hagin de reproduir sons molt grans que fan impossible posar-los en memòria. S'ha de tenir en compte que cada so tindrà el seu espai de memòria (no la comparteixen), també s'ha de tenir en compte que els sons **OGG** no se li poden aplicar efectes de so.
-**Wav** s'utilitzarà quan s'hagin de reproduir sons petits i que n'hi hagi molts ja que comparteixen memòria (en **OpenAL**).

Tots els arxius de so **WAV** i **OGG** han d'estar gravats en el format 44100Hz 16bits i mono per que puguin ser reproduïts en qualsevol dels dos sistemes de so.

Avantatges amb:

-**OpenAL**:
-Pot processar els sons via software o via hardware.
-És més estable que el driver de DirectSound, en les proves de framerate no era tant variable com el driver de DirectSound.
-Els fitxers Wav no cal que estiguin gravats en el format 44.100KH i 16bits ell ja entén la resta de formats.

-**DirectSound**:
-Pot reproduir tants sons a l'hora com faci falta.
-Pot processar fitxers Wav molt grans.

-Inconvenients amb :

-**Comuns de OpenAL i DirectSound**:
-Només permet 1 efecte de so per cada so.
-No es poden crear efectes sobre sons OGG.

-**OpenAL**:
-és dependent del hardware (EAX) per a reproduir efectes de so com chorus, reverb... tot i que actualment la gran majoria de targes de so com a mínim implementen l'efecte de reverberació.
-té un límit de sons que pot reproduir a l'hora segons via tarja de so o via hardware (uns 30 en targes "normals" i 128 en targes sound blaster x-fi series).
-té un arxiu de mida màxima que pot carregar a memòria (30MB).
-té un límit d'efectes de so que depenen de cada tarja de so.

-**DirectSound**:

-No comparteix búffers de memòria per reproduir sons que són el mateix.
-Els efectes no són mai accelerats via hardware i tenen una qualitat pèssima.

8.2 Utilització de les llibreries:

1. Per utilitzar les llibreries, és necessari tenir instal·lat el Visual Studio 2005 SP1, Microsoft DirectX SDK (la versió que faig servir és August 2007) i Ogre SDK 1.4.6 for Visual Studio 2005.

2. Cal incloure els directoris d'inclusió addicional i els directoris de biblioteca addicionals.

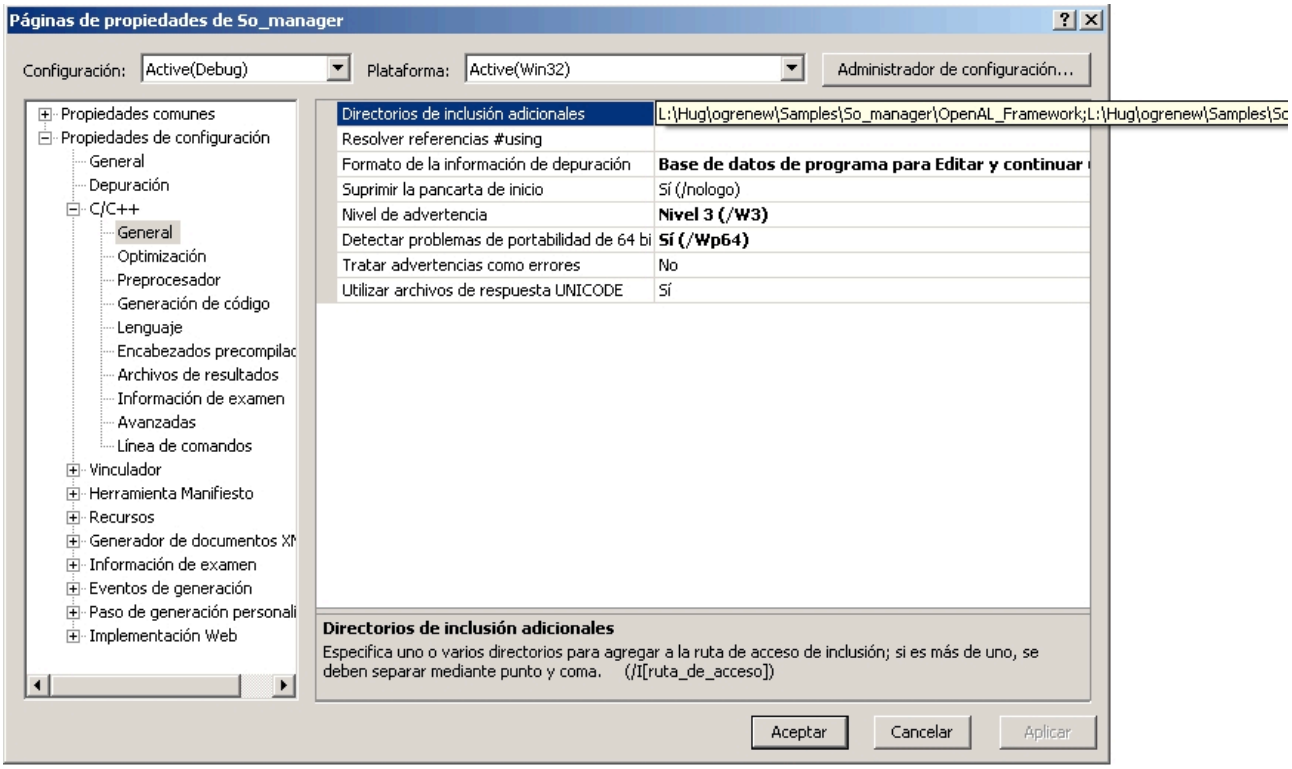


Fig.24 Propietats C++/generals de l'aplicació

3. Cal incloure llavors les següents llibreries (en aquest estricte ordre):

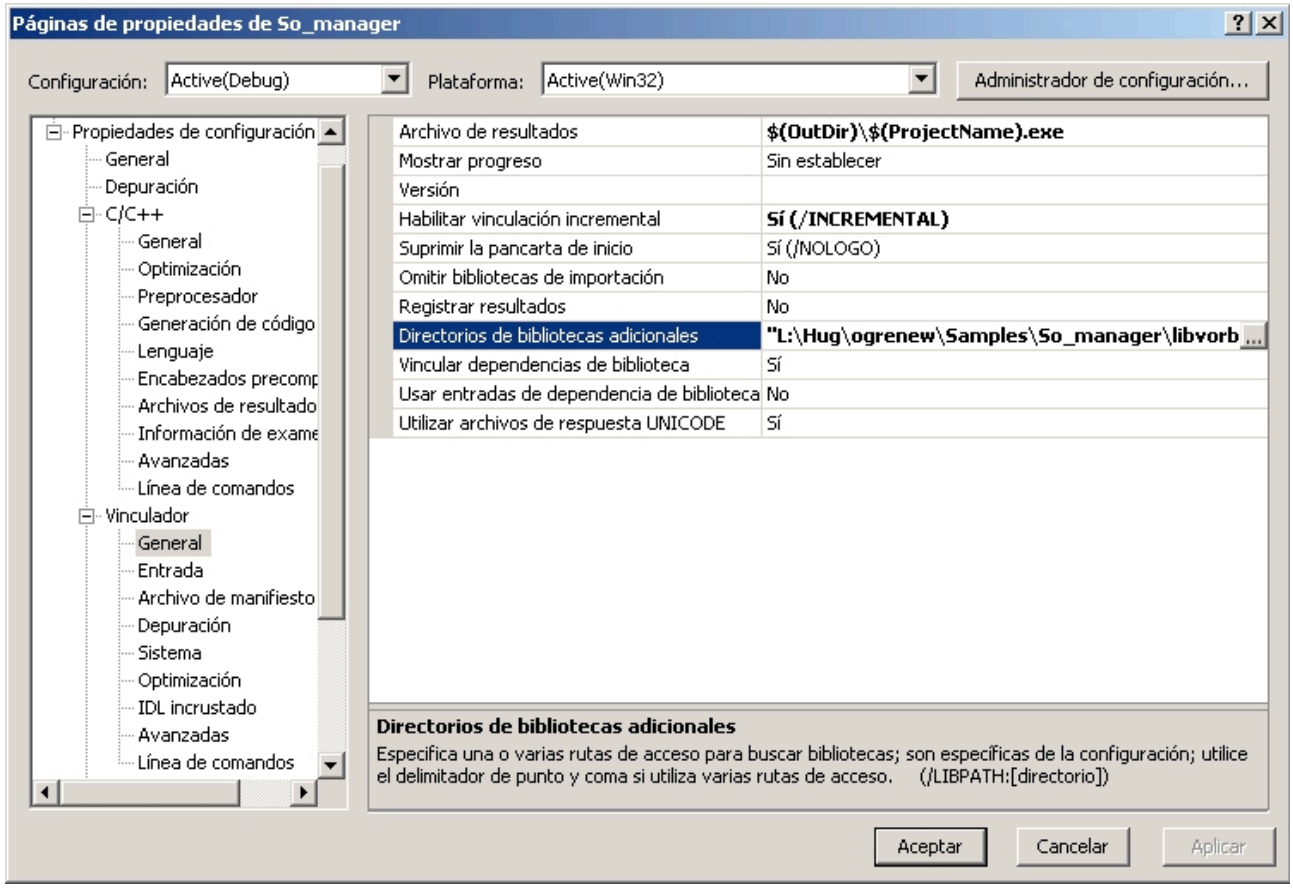


Fig.25 Propietats del Vinculador/General de l'aplicació

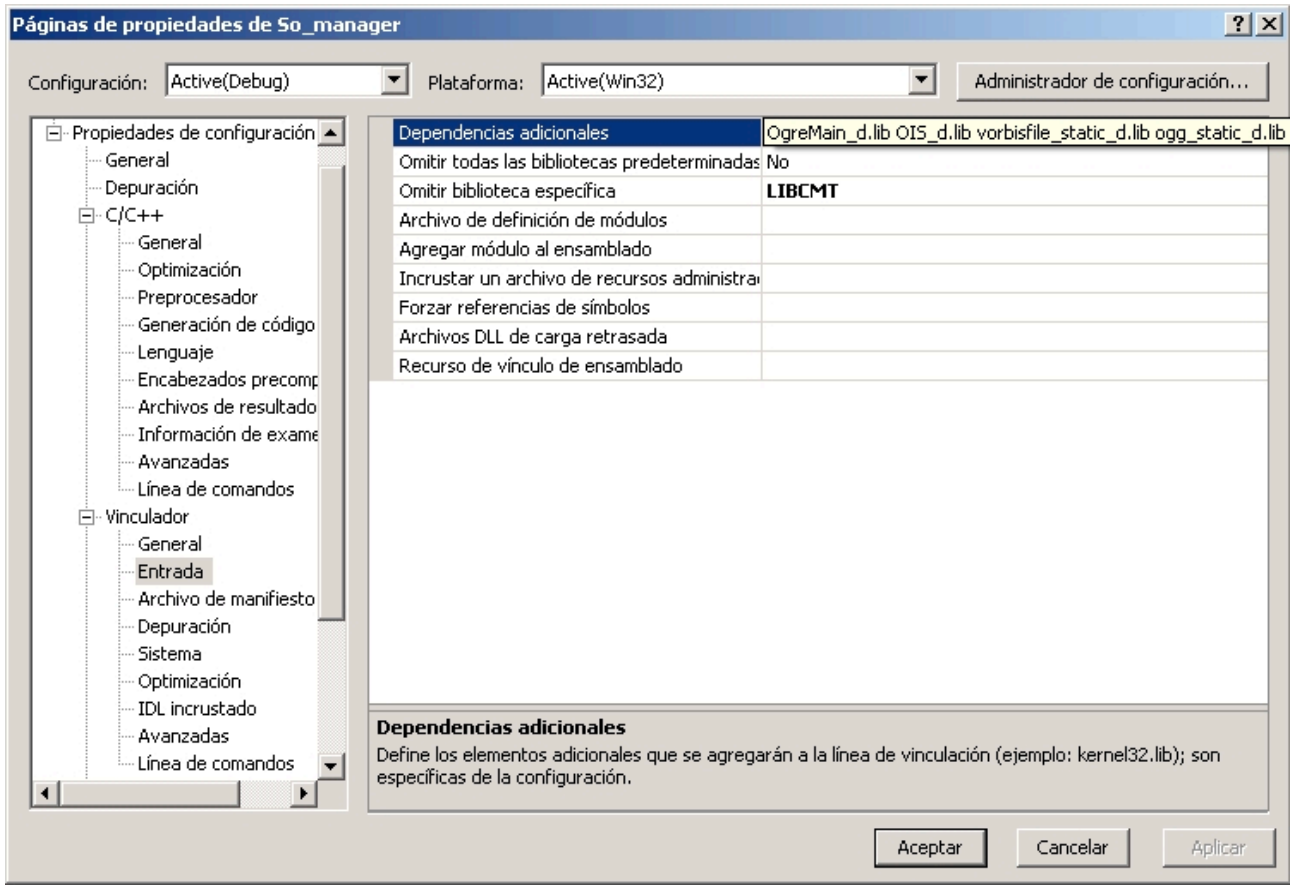


Fig.26 Propietats Vinculador/Entrada de l'aplicació.

```
vorbisfile_static_d.lib
ogg_static_d.lib
vorbis_static_d.lib
OpenAL32.lib
so_vs2005.lib
alut.lib
dsound.lib
dxerr.lib
comctl32.lib
winmm.lib
d3dx9d.lib
dxguid.lib
odbc32.lib
odbccp32.lib
DXUT.lib
DXUTOpt.lib
EFX-Util.lib
```

4. Seguidament s'ha de canviar el joc de caràcters per Unicode (Proyecto->Propiedades->Propiedades de configuración->General->Juego de caracteres).

5. Si intentem compilar i ens dona l'error " error C2664: 'MessageBoxW' : no se puede convertir el parámetro 2 de 'const char *' a 'LPCWSTR' " vol dir que hem de canviar la funció MessageBox(...) per MessageBoxA(...)

6. Finalment posem al directori de l'aplicació l'arxiu "SoundSystemOptions.cfg" i arranquem l'aplicació.

7. Nota: necessitarem que les .dll següents estiguin dins la carperta de l'executable o del directori System32 pel que respecta al mòdul de so:

    1. alut.dll
    2. OpenAL32.dll
    3. d3d9d.dll

8.3 Tutorial d'iniciació

Aquí deixo un petit tutorial per fer servir les classes :

1. **Inicialitzar el GT_SoundManager:**

```
#include "GT_SoundManager.h"

class TutorialApplication : public ExampleApplication
void TutorialApplication::createScene(void)
{
...
// Creating the manager. This may only be called once!
new GT_SoundManager( mSceneMgr, mHinst, mCamera );

// Obtaining a reference to the manager instance
GT_SoundManager& mgr = GT_SoundManager::getSingleton();
...

}
```

```
void createFrameListener(void)
{
GT_SoundManager *sm = GT_SoundManager::getSingletonPtr();
mRoot->addFrameListener(sm);
...
}
```

2. **Creació d'un so WAV**

```
void TutorialApplication::createScene(void)
{
...

//"So1" és el nom que li volem donar al so i ha de ser únic
//"roar.wav" és el nom de l'arxiu ha de ser un wav mono grabat a 44100Hz i
16bits
//headNode és un node previament creat
//amb true li indiquem que volem que es repeteixi
GT_Sound *so1 = mgr.createSound( "So1", "roar.wav", headNode, true );

so1->setInnerConeAngle( 90 );
so1->setOuterConeAngle ( 360 );
so1->setOuterConeGain ( 0.1 );
so1->setSourceRelative( false );
so1->play();
so1->setAutoCalculateVelocity( false );
...
}
```

8.4 Crear noves extensions:

8.4.1 Nous drivers de la tarja de so:

Per la creació d'extensions, per afegir un nou driver d'un diferent fabricant, simplement han d'implementar la classe abstracta GT_Driver.h, s'han d'implementar les funcions que es creguin necessàries, però s'ha de tenir en compte que les funcions que no s'implementin, no s'utilitzaran.

8.4.2 Nous efectes de so EAX:

Per a la creació de nous efectes de so s'ha:

- Implementar una nova classe que derivi de la classe FXStruct.

- S'ha d'afegir una nova branca al switch de la funció void GT_Sound::_initSound() que cridi mDriver->NomDeLaFuncio(mSource, effectsValues).

- I s'ha d'implementar una nova funció en tots els driver (encara que la resta no l'implementin), passant el source i la classe de paràmetres de l'efecte corresponent.

8.5 Cronologia d'importants bandes sonores del

gènere

A continuació una cronologia dividida en les diferents generacions on s'inclou bandes sonores que són importants en el gènere. La llista es basa en els llançaments japonesos, americans i europeus dels videojocs.

8 Bits(1983-1994)	16 Bits(1988-1997)	32/64 Bits(1994-2003)
1985 Super Mario Bros	**1989** Super Mario Land	**1997** GoldenEye 007
1987 Final Fantasy	**1991** Street Fighter II	**1998** The Legend of Zelda: Ocarina of Time
1987 Megaman	**1991** Sonic the Hedgehog	**1998** Resident Evil 2
	1993 Dragon Ball Z Super Butoden 2	**2001** Metal Gear Solid 2: Sons of Liberty

128 Bits(1998-2007)	256 Bits(2004-actualitat)
2002 Final Fantasy X	**2005** Castlevania Dawn of Sorrow
2003 The Legend of Zelda: The Wind Waker	
2006 Kingdom Hearts II	
2006 Dragon Quest VIII	

8.6 Taules amb les correspondències entre els atributs dels efectes de so:

AL_EFFECT_CHORUS	DSFXChorus	Descripció:	Rang de Valors:
AL_CHORUS_WAVEFORM	LONG lWaveform	Forma de la senyal que controla el temps d'espera de la senyal.	OpenAL: 0-sin 1-triangle default: 1 DirecSound: DSFXCHORUS_WAVE_TRIANGLE o DSFXCHORUS_WAVE_SIN default sin
AL_CHORUS_PHASE	LONG lPhase	Fase diferencial entre les LFO's esquerra i dret.	OpenAL: -180 fins 180 default:90 DirecSound: DSFXCHORUS_PHASE_MIN a DSFXCHORUS_PHASE_MAX. Default DSFXCHORUS_PHASE_90
AL_CHORUS_DELAY	FLOAT fDelay	Temps d'espera fins que es torna a reproduir el so.	OpenAL: 0.0 fins 0.016 default: 0.016 DirecSound:

AL_EFFECT_CHORUS	DSFXChorus	Descripció:	Rang de Valors:
			DSFXCHORUS_DELAY_MIN a DSFXCHORUS_DELAY_MAX default 16ms
AL_CHORUS_DEPTH	FLOAT fDepht	Percentatge entre en el que el temps d'espera és modulat.	OpenAL: 0 fins 1 default: 0.1 DirecSound: DSFXCHORUS_DEPTH_MIN a DSFXCHORUS_DEPTH_MAX default 10
AL_CHORUS_FEEDBACK	FLOAT fFeedback	Controla la quantitat de senyal que alimenta el chorus.	OpenAL: -1 fins 1 default: 0.25 DirecSound: DSFXCHORUS_FEEDBACK_MIN a DSFXCHORUS_FEEDBACK_MAX default 25

AL_EFFECT_DISTORTION	DSFXDistortion	Descripció:	Rang de Valors:
AL_DISTORTION_EDGE	DWORD fEdge	Control de la forma de la distorsió.	OpenAL: 0 fins 1.0 default 0.2 DirecSound: DSFXDISTORTION_EDGE_MIN a DSFXDISTORTION_EDGE_MAX default 15%
AL_DISTORTION_GAIN	DWORD fGain	Quantitat de guany després del canvi de la distorsió.	OpenAL: 0.1 fins 1 default: 0.05 DirecSound: DSFXDISTORTION_GAIN_MIN a DSFXDISTORTION_GAIN_MAX default -18dB
AL_DISTORTION_LOW_PASS_CUTOFF	DWORD fPreLowPassCutoff	Filtre de tall d'altes freqüències dels harmònics.	OpenAL: 80 fins 24000 default: 8000 DirecSound: DSFXDISTORTION_PRELOWPASSCUTOFF_MIN a DSFXDISTORTION_PRELOWPASSCUTOFF_MAX default 8000Hz
AL_DISTORTION_EQCENTER	DWORD fPostEQCenterFrequency	Control de la freqüència en la post-distorsióo atenuació.	OpenAL: 80 fins 24000 default: 3600 DirecSound: DSFXDISTORTION_POSTEQCENTERFREQUENCY_MIN a DSFXDISTORTION_POSTEQCENTERFREQUENCY_MAX default 2400Hz
AL_DISTORTION_EQBANDWIDTH	DWORD fPostEQBandwidth	Controla l'amplada de la freqüència en la post-distorsió atenuació.	OpenAL: 80 fins 24000 default: 3600 DirecSound: DSFXDISTORTION_POSTEQBANDWIDTH_MIN a DSFXDISTORTION_POSTEQBANDWIDTH_MAX default 2400Hz

AL_EFFECT_ECHO	*DSFXEcho*	*Descripció:*	*Rang de Valors:*
AL_ECHO_FEEDBACK	FLOAT fFeedback	Amplitud de la sortida.	OpenAL: 0 fins 1 default 0.5 DirecSound: DSFXECHO_FEEDBACK_MIN DSFXECHO_FEEDBACK_MAX default 50
AL_ECHO_LRDELAY	FLOAT fLeftDelay FLOAT fRightDelay	Controla la espera entre el primer 'tap' i el segon.	OpenAL: 0 fins 0.404 default 0.1 DirecSound: DSFXECHO_LEFTDELAY_MIN DSFXECHO_LEFTDELAY_MAX DSFXECHO_RIGHTDELAY_MIN DSFXECHO_RIGHTDELAY_MAX default 50

AL_EFFECT_FLANGER	*DSFXFlanger*	*Descripció:*	*Rang de Valors:*
AL_FLANGER_WAVEFORM	LONG lWaveform	Forma de la ona (sinusoïdal o triangular).	OpenAL: 0-sin 1-triangle default: 1 DirecSound: DSFXFLANGER_WAVE_TRIANGLE DSFXFLANGER_WAVE_SIN default sin
AL_FLANGER_PHASE	LONG lPhase	Desfase entre la LFO esquerra i la dreta. 0º vol dir que estan sincronitzades.	OpenAL: -180 fins 180 default 0 DirecSound: DSFXFLANGER_PHASE_MIN DSFXFLANGER_PHASE_MAX default zero
AL_FLANGER_DEPHT	FLOAT fDepth	Quantitat de ona original barrejada amb la ona generada.	OpenAL: 0 fins 1 default 1 DirecSound: DSFXFLANGER_DEPTH_MIN a DSFXFLANGER_DEPTH_MAX default 100
AL_FLANGER_FEEDBACK	FLOAT fFeedback	Nivell de senyal de la ona generada.	OpenAL: -1 fins 1 default -0.5 DirecSound: DSFXFLANGER_FEEDBACK_MIN a DSFXFLANGER_FEEDBACK_MAX default -50
AL_FLANGER_DELAY	FLOAT fDelay	El temps d'espera entre la ona reproduïda i la ona generada.	OpenAL: 0 fins 0.004 default 0.002 DirecSound: DSFXFLANGER_DELAY_MIN a DSFXFLANGER_DELAY_MAX default 2ms
AL_FLANGER_RATE	FLOAT fFrequency	S'utilitza per incrementar la freqüència de la ona generada.	OpenAL: 0 fins 10 default 0.27 DirecSound: DSFXFLANGER_FREQUENCY_MIN a DSFXFLANGER_FREQUENCY_MAX default 0.25

AL_EFFECT_RING_MODULATOR	*DSFXGargle*	*Descripció:*	*Rang de Valors:*
AL_RING_MODULATOR_FREQUENCY	DWORD dwRateHz	Freqüència de la senyal portadora	OpenAL: 0 fins 8000 default 440 DirecSound: DSFXGARGLE_RATEHZ_MIN DSFXGARGLE_RATEHZ_MAX default 20
AL_RING_MODULATOR_WAVEFORM	DWORD dwWaveShape	Forma de la ona portadora. Pot ser quadrada o triangular.	OpenAL: 0 sin 1 saw 2 square default 0 DirecSound: DSFXGARGLE_WAVE_TRIANGLE DSFXGARGLE_WAVE_SQUARE default triangle

AL_EFFECT_EAXREVERB	*DSFXI3DL2Reverb*	*Descripció:*	*Rang de Valors:*
AL_EAXREVERB_ROOM_ROLLOFF_FACTOR	flRoomRolloffFactor	Forma com s'atenua el so.	OpenAL: 0 a 10 default: 0 DirecSound: DSFX_I3DL2REVERB_ROOMROLLOFFFACTOR_MIN DSFX_I3DL2REVERB_ROOMROLLOFFFACTOR_MAX default: DSFX_I3DL2REVERB_ROOMROLLOFFFACTOR_DEFAULT
AL_EAXREVERB_DECAY_TIME	flDecayTime	Temps que es demora les ones reflexades.	OpenAL: 0.1 a 20 default: 1.49 DirecSound: DSFX_I3DL2REVERB_DECAYTIME_MIN DSFX_I3DL2REVERB_DECAYTIME_MAX default: DSFX_I3DL2REVERB_DECAYTIME_DEFAULT
AL_EAXREVERB_DECAY_HFRATIO	flDecayHFRatio	Retard de les freqüències més altes.	OpenAL: 0.1 a 20.0 default: 0.83 DirecSound: DSFX_I3DL2REVERB_DECAYHFRATIO_MIN DSFX_I3DL2REVERB_DECAYHFRATIO_MAX default: DSFX_I3DL2REVERB_DECAYHFRATIO_DEFAULT
AL_EAXREVERB_REFLECTIONS_GAIN	lReflections	Guany de les reflexions del so inicial.	OpenAL: 0 a 3.16 (-100db a +10db) default: 0.05 DirecSound: DSFX_I3DL2REVERB_REFLECTIONS_MIN DSFX_I3DL2REVERB_REFLECTIONS_MAX default: DSFX_I3DL2REVERB_REFLECTIONS_DEFAULT
AL_EAXREVERB_REFLECTIONS_DELAY	flReflectionsDelay	Temps d'espera entre l'arribada del so directe i les reflexions.	OpenAL: 0.0 a 0.3 (0 a 300ms) default: 0.007 DirecSound: DSFX_I3DL2REVERB_REFLECTIONSDELAY_MIN DSFX_I3DL2REVERB_REFLECTIONSDELAY_MAX DSFX_I3DL2REVERB_REFLECTIONSDELAY_DEFAULT
AL_EAXREVERB_LATE_REVERB_GAIN	lReverb	Guany de les segones	OpenAL: 0 a 10 (-100db a +20db)

		reverberacions.	default: 1.26 DirecSound: DSFX_I3DL2REVERB_REVERB_MIN DSFX_I3DL2REVERB_REVERB_MAX DSFX_I3DL2REVERB_REVERB_DEFAULT
AL_EAXREVERB_LATE_REVERB_DELAY	flReverbDelay	Espera de les segones reflexions.	OpenAL: 0.0 a 0.1 (0 a 100ms) default 0.011 DirecSound: DSFX_I3DL2REVERB_REVERBDELAY_MIN DSFX_I3DL2REVERB_REVERBDELAY_MAX DSFX_I3DL2REVERB_REVERBDELAY_DEFAULT
AL_EAXREVERB_DIFFUSION	flDiffusion	Difusió i degradació de les reflexions. Amb un valor de 0 proporciona més granularitat i el so pot arribar a sonar com diferents ecos.	OpenAL: 0 a 1 default: 1 DirecSound: DSFX_I3DL2REVERB_DIFFUSION_MIN DSFX_I3DL2REVERB_DIFFUSION_MAX DSFX_I3DL2REVERB_DIFFUSION_DEFAULT
AL_EAXREVERB_DENSITY	flDensity	Controla la coloració de les successions reverberacions.	OpenAL: 0 a 1 default:1 DirecSound: DSFX_I3DL2REVERB_DENSITY_MIN DSFX_I3DL2REVERB_DENSITY_MAX DSFX_I3DL2REVERB_DENSITY_DEFAULT
AL_EAXREVERB_HFREFERENCE	flHFReference	Valor de tall de les altes freqüències.	OpenAL: 1000.0 to 20000.0 default: 5000.0 DirecSound: DSFX_I3DL2REVERB_HFREFERENCE_MIN DSFX_I3DL2REVERB_HFREFERENCE_MAX DSFX_I3DL2REVERB_HFREFERENCE_DEFAULT

8.7 Glossari:

-LFO: Low-Frequency-Oscillator.

-EAX: Environmental audio extensions.

-DSP: Digital Signal Processor.

-Framerate és el nombre de “fotografies” per segon que genera el motor de render.

-Memòria X-RAM

Aquesta memòria de la placa descarrega la memòria RAM del sistema i permet un accés més ràpid a les dades per part de la tarja.

En les targetes amb memòria integrada (X-RAM) hi ha dues maneres d’interactuar amb elles.

Una d’elles és dient-li que carregui automàticament el sons a mesura que es van carregant (si hi ha prou espai).

La segona forma és dient-li so a so quins volem carregar a la tarja manualment, d’aquesta forma s’aconsegueix una millor optimització.

Eco: L’eco o tornaveu és un fenomen relacionat amb la reflexió del so. El senyal acústic original s’ha extingit, però encara torna so en forma d’ona reflectida.

Reverberació: La reverberació és un fenomen derivat de la reflexió del so. Consistent en una lleugera prolongació del so una vegada s'ha extingit l'original, a causa de les ones reflectides

MIDI: és l'acrònim de Musical Instrument Digital Interface. És un codi utilitzat en informàtica musical que permet la comunicació entre els diversos equips musicals electrònics. L'avantatge d'aquest format és el poc espai que ocupa en una memòria, però te com a gran inconvenient una mala qualitat de so.

PCM: De l'anglès, Pulse-Code Modulation. Un cop que els avenços tecnològics del segle XX van portar a l'ésser humà a tenir la capacitat de digitalitzar la informació d'àudio, aquest va ser un dels primers mètodes per aconseguir tal objectiu. El PCM és un procediment de modulació on un senyal analògic és mostrejat cada cert període regular de temps a través d'impulsos de manera que s'aconsegueix tenir tot una sèrie de valors del senyal en diversos moments.