

Títol: *Allegro PSP*

Volum: *I*

Alumne: *Diego Delgado Roda*

Director/Ponent: *Jesús Alonso Alonso*

Departament: *Llenguatges i Sistemes Informàtics*

Data:

DADES DEL PROJECTE

Títol del Projecte: Anàlisi de la llibreria Allegro per a la programació d'aplicacions multimèdia i la seva implantació a la consola portàtil PSP

Nom de l'estudiant: Diego Delgado Roda

Titulació: Enginyeria Informàtica

Crèdits: 37,5

Director/Ponent: Jesús Alonso Alonso

Departament: Llenguatges i Sistemes Informàtics

MEMBRES DEL TRIBUNAL (nom i signatura)

President: LUÍS SOLANO ALBAJES

Vocal: NARCÍS NABONA FRANCISCO

Secretari: JESÚS ALONSO ALONSO

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

Índex

Capítol 1. Introducció.....	3
1.1 Motivació.....	3
1.2 Objectius.....	4
1.3 Definicions.....	4
1.3.1 Què és la PSP.....	4
1.3.2 Què és el homebrew.....	5
1.3.3 Què és Allegro.....	6
Capítol 2. Estudi d'antecedents.....	9
2.1 Homebrew a les diferents consoles.....	9
2.1.1 Sega Dreamcast.....	9
2.1.2 Microsoft Xbox.....	9
2.1.3 GP2X.....	10
2.2 Homebrew a la PSP.....	10
Capítol 3. Entorn del PFC.....	11
3.1 La llibreria Allegro.....	11
3.1.1 Característiques d'Allegro.....	11
3.1.2 Estructura d'Allegro.....	12
3.2 La consola portàtil PSP.....	13
3.2.1 El maquinari de la PSP.....	13
3.2.2 El compilador de la PSP.....	14
3.2.3 PSPSDK.....	15
3.2.4 El kernel de la PSP.....	15
3.2.5 El XMB.....	16
3.3 La conversió d'Allegro a la PSP.....	16
Capítol 4. Anàlisi de requisits.....	19
4.1 Introducció.....	19
4.2 Requisits funcionals.....	19
4.2.1 Interfície drivers d'Allegro.....	20
4.2.2 Llista de requisits.....	23
4.2.3 Descripció dels requisits.....	24
4.3 Requisits no funcionals.....	33
4.3.1 Eficiència.....	33
4.3.2 Portabilitat.....	33
4.3.3 Usabilitat.....	34
4.3.4 Requisits de la llibreria Allegro.....	34
Capítol 5. Planificació i costos.....	37
5.1 Planificació.....	37
5.1.1 Identificació de les etapes del projecte.....	37
5.1.2 Tasques i dedicació estimada.....	38
5.1.3 Dedicació real.....	40
5.1.4 Formació.....	45
5.2 Anàlisi econòmic.....	47
5.2.1 Cost de personal.....	47
5.2.2 Cost de maquinari.....	48
5.2.3 Cost de programari.....	49

5.2.4 Cost d'ocupació.....	49
5.2.5 Cost total del projecte.....	49
Capítol 6. Especificació.....	51
6.1 Casos d'ús.....	51
6.1.1 Definició d'actors.....	51
6.1.2 Inicialització d'Allegro i gestió de la pantalla.....	52
6.1.3 Gestió de bitmaps.....	56
6.1.4 Transferència d'imatges i sprites.....	58
6.1.5 Funcions de so.....	60
6.1.6 Funcions del temporitzador.....	62
6.1.7 Gestió dels perifèrics d'entrada.....	63
6.2 Diagrama de classes d'alt nivell.....	65
Capítol 7. Disseny.....	67
7.1 Arquitectura d'Allegro.....	67
7.2 Mòdul gràfic.....	68
7.3 Mòdul so digital.....	70
7.4 Mòdul del temporitzador.....	72
Capítol 8. Implementació.....	75
8.1 Tecnologies utilitzades a Allegro PSP.....	75
8.1.1 C.....	75
8.1.2 OpenGL.....	75
8.1.3 Threads.....	76
8.1.4 GNU make.....	78
8.2 La cache de la PSP.....	79
8.3 Emulació 8 bits de color.....	83
8.4 Suport resolucions majors de 480x272.....	85
8.5 El gestor de VRAM.....	88
8.6 Estructura fitxers llibreria Allegro.....	90
8.7 Entorn de desenvolupament.....	92
8.7.1 Geany.....	92
8.7.2 PSPLink.....	92
Capítol 9. Resultats.....	93
9.1 Jocs de proves.....	93
9.1.1 Humphrey Remake.....	93
9.1.2 Exemples oficials d'Allegro.....	96
9.2 Conclusions.....	100
9.3 Perspectives de futur.....	101
Bibliografia i enllaços.....	103
Llibres.....	103
Llocs web.....	103
Annex A: Glossari.....	107

Capítol 1. Introducció

1.1 Motivació

Sempre he estat un gran aficionat a la tecnologia: grans ordinadors, portàtils, PDAs, DVDs, televisors LCD, etc. Així no és d'estranyar que el juny del 2008 em decidís a adquirir l'últim aparell portàtil de Sony: la PSP o PlayStation Portable. Una petita obra d'enginyeria capaç de reproduir MP3, vídeo MP4, amb la possibilitat de navegar per Internet amb connectivitat sense fils, amb funcions de disc USB, amb potència gràfica similar a màquines "majors" no gaire més antigues i amb tot un món desconegut per descobrir diferent dels a vegades avorrits i previsibles ordinadors personals.

Dies després vaig descobrir una forta i assentada comunitat de programadors de *homebrew* per a la PSP. Des de feia temps, potser des dels 80 amb el Commodore Amiga, coneixia l'existència de les *scenes* o comunitats de programadors que es dediquen a treballar, aprendre i compartir coneixements sobre la màquina que els apassiona. No ha estat, però, fins ara que m'he apropiat realment a aquest món.

Paral·lelament a tots aquests fets vaig descobrir el *remake* per a PC d'un joc de la època dels 80 programat per la companyia Made in Spain a l'anomenada època daurada dels videojocs a Espanya. Aquest joc estava programat amb la llibreria Allegro i estava adaptat a múltiples plataformes (Windows, Mac OS X, Unix, etc.) gràcies a la portabilitat que Allegro proporciona. Em va estranyar que no existís una versió d'aquest *remake* per a la PSP per dos motius: perquè el seu *homebrew* estava molt desenvolupat i perquè aquest ja acumulava 3 anys d'experiència (des del 2005), fins i tot existia una versió de la llibreria OpenGL (l'estàndard per a la programació de gràfics 2D i 3D desenvolupat per Silicon Graphics) per a la PSP. La idea del meu projecte, doncs, va sorgir precisament d'aquí: convertir una API (Application Programming Interface) de pes i història com és Allegro a una plataforma totalment desconeguda com és la PSP. Des d'aquell moment em vaig proposar adquirir nous coneixements relacionats amb aquest projecte i millorar o refrescar tot el que ja sabia: Allegro, C, C++, gràfics 2D, 3D, àudio, hardware PSP i el seu entorn de desenvolupament, programari lliure, portabilitat, etc.

Vaig pensar que segur que hi havia gent més preparada i amb més coneixements sobre aquest tema que jo però tenia un especial interès en desenvolupar aquest projecte per diferents raons. Personalment era una oportunitat i un repte per a aprendre i desenvolupar des d'una òptica d'enginyer informàtic sobre programació de gràfics i so, programació de hardware poc comú (sistemes portàtils o embedded systems, processador MIPS R4000, custom chips, ...), programació de drivers, ... I en general per a poder desenvolupar programari lliure mitjançant l'ús d'eines de programari lliure. Finalment, l'altra raó a més de la personal és que aquest projecte aportarà un gra de sorra a la comunitat *open source* en crear centenars de potencials conversions de jocs i d'aplicacions multimèdia al sistema PSP i animarà a futurs programadors a treballar més encara amb Allegro: <http://www.allegro.cc>

En el moment d'escriure aquestes línies i amb l'experiència acumulada que m'ha aportat aquest projecte he descobert un camí que se m'ha obert per, per què no?, poder enfocar la meua carrera professional a la programació multimèdia i de videojocs.

1.2 Objectius

L'objectiu central d'aquest PFC consisteix en l'ampliació d'un programari ja existent, una llibreria de programació d'aplicacions multimèdia. Concretament, consisteix en la realització de la implementació d'una part de la llibreria Allegro. Entre d'altres coses això vol dir que es parteix d'una especificació i d'un disseny previs de més de 10 anys de vida. És aquí on entra a escena un segon objectiu del projecte, l'anàlisi en profunditat de la llibreria Allegro. Aquest segon objectiu pretén extreure aquesta especificació i aquest disseny d'una manera clara i convertir-los en el punt de partida de l'objectiu central.

Per altra banda, la plataforma destinatària d'aquest PFC és la PSP (PlayStation Portable) una plataforma atípica en quant a la seva arquitectura i en quant al seu entorn de desenvolupament. Aquesta plataforma no és un PC, ni tan sols un ordinador o microordinador sinó un dispositiu electrònic de petita mida i d'ús específic anomenat en anglès *embedded system*. El tercer objectiu és l'anàlisi del maquinari i entorn de desenvolupament de la PSP.

Finalment, hi ha un quart objectiu que consisteix en fer la conversió a la PSP del videojoc amateur *Humphrey Remake* de Ignacio Pérez Gil, programat usant com a llibreria principal precisament Allegro.

Enumerant, els objectius d'aquest PFC són:

1. Estudi i anàlisi de la llibreria de programació d'aplicacions multimèdia i videojocs Allegro.
2. Estudi i anàlisi de la consola portàtil PlayStation Portable a nivell de maquinari i programació.
3. Conversió de la llibreria Allegro a la consola PSP.
4. Conversió del videojoc *Humphrey Remake* programat amb Allegro.

1.3 Definicions

1.3.1 Què és la PSP



PSP són les sigles de PlayStation Portable, la primera incursió de Sony Computer Entertainment al món de les consoles de joc portàtils i la tercera de la línia PlayStation després de PlayStation (1994) i PlayStation 2 (2000). El seu llançament oficial al Japó es va produir el 2004, a Europa el 2005.

Algunes de les seves característiques tècniques són:

- Pantalla TFT LCD de 4,3 polzades, relació d'aspecte 16:9 o widescreen amb una resolució de 480x272 píxels i 16,7 milions de colors.
- Capacitat per reproduir so 3D amb 7.1 canals gràcies al Virtual Mobile Engine de Sony.
- Connexió Wi-Fi IEEE 802.11b que permet accés a Internet i interconnexió amb d'altres PSP.
- Conté una CPU de nom Allegrex de fins 333 Mhz basada en la CPU MIPS R4000 de 32 bits amb una unitat de coma flotant (FPU) i una unitat de càlcul vectorial (VFPU) necessàries per a la computació gràfica 3D. També conté una altra CPU de nom Media Engine (basada igualment en el MIPS R4000) dedicada al processament d'àudio i vídeo.
- Memòria RAM de 32 MiB (64 MB en les noves revisions de la PSP) i memòria de vídeo o VRAM de 2 MiB.
- Reproducció d'àudio ATRAC3 plus, AAC, MP3, WMA i reproducció de vídeo H.264/MPEG-4 AVC.
- Ús de discs òptics UMD (Universal Media Disk) i targetes de memòria (Memory Stick PRO DUO) per a la lectura de dades i aplicacions i per a la reproducció d'imatges, música i pel·lícules.
- Menú de navegació XMB (XrossMediaBar) de Sony com a interfície per a la comunicació amb l'usuari.
- S'alimenta amb una bateria recarregable d'ions de liti o directament de la xarxa elèctrica.

El 2007 es va produir el llançament de la PSP Slim & Lite, una revisió de la PSP original. Algunes de les diferències són un 33% menys de pes, un 19% més prima, connectivitat a TV i 64 MiB de memòria RAM (el doble). En aquest PFC es treballa amb aquest nou model.

1.3.2 Què és el homebrew

Es denomina *homebrew* a les aplicacions i jocs creats per programadors aficionats i/o experts per a qualsevol plataforma, generalment consoles de joc propietàries. El seu ús és molt comú en usuaris avançats i està molt lligat al món *open source* en quant a la seva distribució lliure i la compartició de codi que facilita l'aprenentatge i el coneixement de l'anàlisi, disseny i programació d'aplicacions.

Per poder executar *homebrew* a la PSP és necessari tenir un *firmware* (BIOS o programari de baix nivell que inicialitza i controla el hardware) alternatiu a l'oficial programat també per gent del cercle *homebrew*.

La posició de Sony respecte el *homebrew* és no donar-li suport, és a dir, la garantia de la PSP perd la seva validesa. Hi ha, però, algunes veus representatives de Sony a favor d'aquest moviment: <http://www.pspfanboy.com/2007/07/25/sony-looking-for-a-way-to-safely-support-homebrew/>

Existeix una gran varietat de *homebrew* per a la PSP. Podem trobar nous entorns gràfics o shells com iRShell (<http://irshell.org/site/>), gestors de correu electrònic com PSPoste (<http://www.psposte.org/news/current.html>), compressors/descompressors de fitxers, aplicacions per a l'edició d'imatges, editors de text, GPS, llibreries o APIS d'alt nivell per a programadors, emuladors de màquines antigues, jocs i fins i tot una versió de Linux (<http://jacksonm88.googlepages.com/linuxonpsp.htm>)

Totes aquestes aplicacions no serien possibles si no fos per l'existència d'eines i llibreries lliures com un compilador de C per al processador MIPS (recordem que és la CPU en la qual es basa la PSP) i la llibreria **newlib** (<http://sourceware.org/newlib/>), una versió de la llibreria estàndard de C (**libc**) per a sistemes portàtils o *embedded systems* en anglès.

1.3.3 Què és Allegro



Allegro és una llibreria per a programadors de C/C++ orientada al desenvolupament d'aplicacions multimèdia i videojocs, distribuïda lliurement, que funciona a les següents plataformes: DOS, Unix (Linux, FreeBSD, Irix, Solaris), Windows, QNX, BeOS, Mac OS X, AmigaOS, Nintendo DS, Dreamcast i GP2X. Té moltes funcions de gràfics, so, entrada de l'usuari (teclat, ratolí i joystick) i temporitzadors. També té funcions matemàtiques en punt fix i coma flotant, funcions 3D, funcions per a tractar fitxers, fitxers de dades comprimits i una interfície gràfica. Va ser originalment escrita per Shawn Hargreaves a principis dels 90. Inicialment va ser concebuda al fantàstic Atari ST per passar ràpidament al Borland C dels PC i d'aquí a la plataforma de desenvolupament DJGPP de DOS on va madurar. Actualment és mantinguda per una comunitat de programadors com Peter Wang, Peter Hull o Trent Gamblin.

Allegro té dues branques principals: la versió estable 4.2.2 i la versió en desenvolupament o WIP (work in progress) 4.9.x. Per a la realització d'aquest PFC s'ha començant treballant amb la versió estable 4.2.2 i s'ha acabat amb la versió 4.3.10. Aquesta versió és similar a la 4.2.2, excepte que s'han corregit alguns petits *bugs* i que s'han incorporat en el paquet oficial de distribució *addons* o llibreries extra que estenen la funcionalitat d'Allegro com *loadpng*, per a suportar fitxers gràfics PNG, o *logg*, per a suportar fitxers d'àudio OGG/Vorbis.

A la web oficial d'Allegro es troben totes les novetats en el desenvolupament de la llibreria, així com la documentació, les descàrregues, llistes de correu, fòrums i molt més: <http://www.liballeg.org/>.

Dintre del món de desenvolupament de videojocs es poden destacar dos grans grups: les llibreries genèriques orientades al disseny de videojocs des d'un nivell relativament baix i que consisteixen bàsicament en una API o un conjunt de funcions que abstraen el maquinari; i els *game engines* o motors de joc, sistemes i/o entorns de desenvolupament de més alt nivell on es troben elements més especialitzats per a la construcció d'un videojoc tals com un motor 3D, un motor de col·lisions, intel·ligència artificial, etc.

Dintre del primer grup trobem llibreries com SDL (<http://www.libsdl.org/>), DirectX o la mateixa Allegro; i en el segon grup trobem *game engines* com OGRE (<http://www.ogre3d.org/>) o Crystal Space (http://www.crystalspace3d.org/main/Main_Page).

Per altra banda les llibreries i *game engines* poden ser de codi lliure o poden ser propietàries (ja siguin gratuïtes o de pagament). En el primer grup trobem SDL, Allegro, OGRE, Crystal Space o ClanLib (<http://www.clanlib.org/>). I en el grup de les propietàries o privades trobem el *game engine* Adventure Game Studio (<http://www.adventuregamestudio.co.uk/>), la llibreria DirectX de Microsoft, o Gamebryo (<http://www.emergent.net/en/Products/Gamebryo/>) de Emergent Game Technologies.

Capítol 2. Estudi d'antecedents

És ben conegut per tothom que una consola de videojocs, lògicament, serveix per jugar. Però les persones que anem una mica més enllà sabem que darrera d'una gran màquina s'acostuma a amagar una gran *scene*. Milers de programadors anònims treballant desinteressadament amb l'objectiu de compartir els seus treballs amb els demés, i fent que una consola sigui molt més que oci. Cada màquina té la seva història.

2.1 Homebrew a les diferents consoles

Actualment a més de la Sega Dreamcast, Nintendo o la PlayStation Portable, les plataformes més utilitzades per al desenvolupament de *homebrew* són consoles de generacions anteriors com la Atari 2600 i la Nintendo Entertainment System (NES). L'Atari 2600 i la NES són especialment interessants perquè utilitzen un microprocessador basat en el MOS 6502, resultant familiar a gent que programava microordinadors de 8 bits molt populars a la dècada dels 80 com el Commodore 64 o l'Apple II. En general les consoles portables i antigues són més utilitzades per al desenvolupament de *homebrew* degut a la seva simplicitat i a la menor necessitat de recursos en relació a les consoles modernes.

2.1.1 Sega Dreamcast

La Sega Dreamcast és una consola del 1998-1999 i va sorgir amb l'objectiu de recuperar les vendes perdudes de Sega per culpa de la primera PlayStation de Sony. La Dreamcast (DC) és, en molts aspectes, una de les pioneres en la incorporació de *homebrew*.

- **Aplicacions:** reproductors MP3 i de vídeo.
- **Sistemes operatius:** conversions de Linux, NetBSD i QNX.
- **Llibreries gràfiques i multimèdia:** conversions de SDL, OpenGL i Allegro.
- **Emuladors i jocs:** a destacar les conversions del Doom o el Quake.

2.1.2 Microsoft Xbox

La Xbox ha estat una de les plataformes més prolífiques en *homebrew*. Degut a que la Xbox té una arquitectura Intel x86 i utilitza llibreries estàndard de PC com DirectX, programar *homebrew* és molt senzill.

- **Sistemes operatius:** conversions de Linux o FreeBSD i la possibilitat d'executar Windows 98 o Mac OS X mitjançant emuladors.
- **Emuladors:** s'han programat emuladors de tot tipus per a aquesta consola i segueixen tenint un desenvolupament molt actiu.

2.1.3 GP2X

GP2X és una consola portàtil creada el 2005 per l'empresa sud-coreana Gamepark Holdings. És una consola molt especial ja que està orientada al programari lliure i està basada en GNU/Linux. Pensada per als desenvolupadors de *homebrew* però també per als desenvolupadors comercials.

- **Aplicacions:** conversió del reproductor multimèdia FFPlay per a la reproducció de formats no suportats per la pròpia consola com els de Windows Media o RealMedia.
- **Llibreries gràfiques i multimèdia:** conversions d'Allegro i SDL.
- **Emuladors i jocs:** degut al poc seguiment que té aquesta consola si el comparem amb la Sony PSP o la Nintendo DS, hi ha pocs jocs comercials. En contraposició hi ha una gran quantitat de conversions de jocs d'altres plataformes. Hi ha també una gran quantitat de jocs *freeware* realitzats exclusivament per a la GP2X.

2.2 Homebrew a la PSP

Com ja s'ha comentat al capítol d'Introducció existeix una gran quantitat de *homebrew* per a la PSP.

- **Aplicacions:** gràcies a la seva connectivitat a Internet s'han programat per a la PSP una gran quantitat d'aplicacions Internet com navegadors o programes de missatgeria.
- **Sistemes operatius:** la conversió de uCLinux (Linux/Microcontroller), és una versió de Linux per a plataformes sense MMU (Memory Management Unit) com és el cas de la PSP.
- **Llibreries gràfiques i multimèdia:**
 - Conversió de libmad, per descodificar àudio MPEG.
 - Conversió de libmikmod, permet reproduir fitxers d'àudio MOD, S3M, XM o IT.
 - pspGL, la versió PSP de l'estàndard OpenGL.
 - Conversió de la llibreria SDL, l'alternativa a Allegro però actualment més popular i utilitzada.
 - Conversió de la llibreria Allegro realitzada per l'autor d'aquest PFC.
- **Emuladors i jocs:** com a la majoria de plataformes amb *homebrew*, existeix una gran quantitat i varietat.

Per a la realització d'aquest PFC ha estat de gran ajuda el coneixement i l'experiència d'altres desenvolupadors a la PSP. També han estat d'ajuda altres projectes *homebrew* concrets com la versió PSP de l'emulador del Macintosh Basilisk II, o un *game engine* exclusiu per a la PSP anomenat triEngine.

Capítol 3. Entorn del PFC

3.1 La llibreria Allegro

Allegro es pot veure com un *framework* o entorn de programació pensat per facilitar el desenvolupament de programari multimèdia i de videojocs permetent als dissenyadors i programadors d'aquest tipus d'aplicacions passar més temps identificant requeriments de programari que tractant amb els detalls de baix nivell.

A més, i gràcies a la seva condició de multi plataforma, el programari desenvolupat amb Allegro funciona en DOS, Microsoft Windows, Unix, BeOS, QNX i Mac OS X gràcies a la interfície de programació d'aplicacions (API) portable que Allegro proporciona, aïllant la interfície de programació d'aplicacions específica de cada plataforma hardware i/o sistema operatiu.

La llibreria està escrita en el llenguatge de programació C i dissenyada per ser utilitzada per programadors de C i C++.

3.1.1 Característiques d'Allegro

L'entitat més important i representativa d'Allegro és el BITMAP. Totes les rutines gràfiques d'Allegro dibuixen a les entitats BITMAP, àrees de memòria que contenen imatges rectangulars. Fins i tot el *display* o pantalla física és considerada un BITMAP.

Aquesta és una relació de les funcions més significatives d'Allegro:

- Funcions per a dibuixar primitives gràfiques incloent píxels, línies, rectangles, polígons, arcs, cercles, el·lipses, text amb fonts proporcionals, etc. tant a la pantalla com a memòria de vídeo (VRAM) o a memòria RAM.
- Funcions per treballar amb sprites, funcions per *blitting* o transferència d'imatges, escalament d'imatges i rotació amb suport per *clipping* o retallament. Suport natiu per fitxers d'imatge BMP, LBM, PCX i TGA.
- Funcions per manipulació de paletes de colors. Funcions per conversió entre els diferents formats de color suportats: mode paleta o indexat, RGB i HSV o Matís – Saturació – Valor.
- Pantalles virtuals i hardware *scrolling* o desplaçament de la imatge, *page flipping* i *triple buffering* (suportat en algunes plataformes).
- Reprodueix vídeo FLI/FLC.
- Reprodueix música MIDI i so digital suportant fins a 64 veus simultànies. Suporta efectes de so com *looping*, reproducció cap enrere, efectes de canvis de volum, *pan* i freqüència. Pot llegir taules d'ones d'instruments des de fitxers externs. Reconeix fitxers d'àudio amb extensió WAV i VOC.
- Accés senzill al ratolí, teclat, joystick i rellotge programable o temporitzador per a programar interrupcions d'usuari.

- Fitxers propis per emmagatzemar els objectes d'Allegro (DATAFILES).
- Funcions matemàtiques incloent aritmètica de punt fix i manipulació de vectors, matrius i quaternions.
- Funcions per fabricar GUI (Graphical User Interface) o interfícies gràfiques d'usuari.
- Suport per text Ascii i Unicode UTF-8.
- Gràcies als *addons* o llibreries de tercers que actuen com extensions d'Allegro s'amplia el suport a fitxers d'imatge PNG, GIF, JPEG; vídeo MPEG; música Ogg, MP3, IT, S3M, XM; fonts TTF (TrueType), etc. Fins i tot existeix una llibreria que permet estendre la funcionalitat d'Allegro a OpenGL: AllegroGL.

3.1.2 Estructura d'Allegro

Allegro consisteix bàsicament en dos grans blocs o capes: la capa independent de la plataforma i la capa dependent de la plataforma.

La capa independent de la plataforma consisteix en la llibreria en sí mateixa o el conjunt de funcionalitats. Però perquè aquesta capa independent funcioni es necessita la funcionalitat que cada plataforma (Windows, Linux, PSP, ...) ofereix.

És precisament l'altra capa, la capa dependent de la plataforma, la que s'encarrega de comunicar-se amb les diferents plataformes i oferir aquesta funcionalitat. En la terminologia Allegro s'anomena conjunt de *drivers* o controladors de dispositiu. Existeixen tants conjunts de drivers com plataformes reals suportades i cada conjunt conté vuit classes de *drivers* que són:

- System driver o controlador de sistema.
- Graphic driver o controlador gràfic.
- Digital sound driver o controlador de so digital.
- MIDI driver o controlador MIDI.
- Timer driver o controlador del temporitzador.
- Keyboard driver o controlador de teclat.
- Mouse driver o controlador de ratolí.
- Joystick driver o controlador de joystick.

Aquesta capa dependent, doncs, actua com a capa d'abstracció del maquinari amagant les diferències entre cadascuna de les plataformes suportades per Allegro.

La figura 3.1 mostra Allegro com una caixa negra amb l'habilitat de funcionar en qualsevol plataforma:

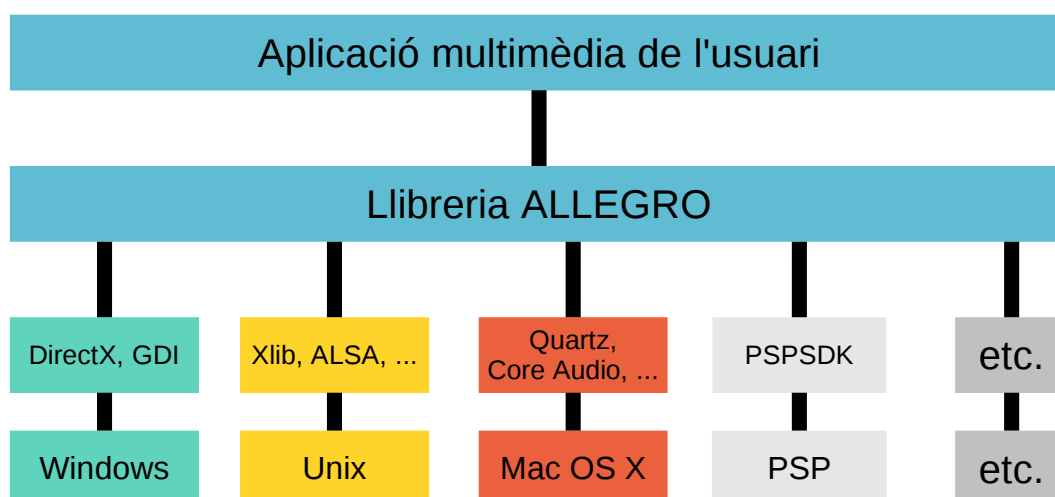


Figura 3.1 Allegro multi plataforma

3.2 La consola portàtil PSP

3.2.1 El maquinari de la PSP

ALLEGREX és el nom del processador central de la PSP. Està format pels següents components:

- El seu nucli és un processador RISC MIPS de 32 bits little endian basat en el model R4000 i amb una velocitat de rellotge variable entre 33 Mhz i 333 Mhz.
- Un processador de coma flotant o FPU de 32 bits per operar amb reals de precisió simple i un co-processador vectorial/matricial o VFPU.
- Una unitat hardware per fer *debug* (depuració).
- Un *profiler* per monitoritzar l'execució dels programes.
- Una memòria cache de dades de nivell 1 de 32 KiB amb una mida de línia de 64 bytes. I una altra cache d'instruccions.

El *Graphics Engine* (GE) és el nucli de la potència gràfica de la PSP:

- Està modelat seguint l'especificació OpenGL, l'estàndard en gràfics per computador. El programador coneixedor de l'API OpenGL pot treballar de manera similar a la PSP.
- Per dibuixar s'utilitzen *display lists*, una sèrie d'instruccions gràfiques que defineixen una imatge o escena a visualitzar. El processador gràfic executa aquestes instruccions per renderitzar la imatge.

- Té disponibles 2 MiB de memòria de vídeo per emmagatzemar les àrees de dibuix o *frame buffers*, les textures i les *display lists*.
- Té la capacitat de llegir textures i *display lists* directament de la memòria de sistema o memòria RAM. Les àrees de dibuix o *frame buffers*, però, han de ser a memòria de vídeo o VRAM.
- Suporta múltiples formats de textures (4, 8, 15, 16, 32 bits per píxel) i compressió de textures.

La PSP disposa de 32 MiB de memòria RAM i es divideix en 4 MiB pel kernel o nucli controlador de la PSP, 4 MiB de memòria volàtil i 24 MiB per l'usuari. Aquest és el mapa de memòria de la PSP que inclou la memòria de vídeo:

Direcció d'inici	Mida	Descripció
0x04000000	2MiB	VRAM
0x44000000	2MiB	VRAM sense cache
0x08800000	24MiB	RAM principal
0x48800000	24MiB	RAM principal sense cache
0x88000000	4MiB	RAM del kernel

Figura 3.2 Mapa de memòria de la PSP

Com es mostra a la figura, la VRAM i la RAM tenen dos modes d'adreçament: el mode habitual, fent ús de la cache del processador i per tant més ràpid; i un mode sense cache que obliga el processador a escriure i llegir sempre a memòria, un mode més lent però necessari en alguns casos. Aquesta característica es detallarà posteriorment al capítol d'Implementació.

La PSP Slim, una revisió posterior de la PSP inicial, conté entre d'altres millores el doble de memòria RAM (64 MiB).

3.2.2 El compilador de la PSP

Per poder generar codi orientat al maquinari de la PSP primer es necessita un compilador per l'ALLEGREX. El treball de desenvolupament de *homebrew* per a la PSP es fa principalment amb els potents i contrastats llenguatges C i C++ i el compilador utilitzat es basa en un altre pesat en el món *open source*: el gcc (GNU Compiler Collection). La funcionalitat necessària de la llibreria estàndard de C la dona Newlib, una llibreria de C lleugera pensada per a sistemes portàtils i d'ús específic o *embedded systems*.

3.2.3 PSPSDK

Fa falta, però, més que un compilador de C per a generar programari que tingui en compte les capacitats específiques de la PSP.

El PSPSDK (PSP Software Development Kit) és un conjunt de llibreries i eines que inclou entre les més importants:

- Llibreries prototipus o interfícies per accedir al sistema operatiu de la PSP. Per exemple per a la gestió de *threads* o fils d'execució, per a la gestió de fitxers o per a la programació del display o pantalla LCD.
- Les llibreries libGU (Graphics Utility) i libGUM (Graphics Utility Matrix) per a la programació del *Graphics Engine* (GE).
- User Audio Library per a la programació del so a baix nivell codificat en PCM (Pulse code modulation)
- Eines per a la construcció dels executables i formats de fitxer de la PSP: ELF, PRX, EBOOT.PBP, ...
- Documentació i un conjunt de *samples* o programes d'exemple per il·lustrar la funcionalitat de les diferents llibreries.

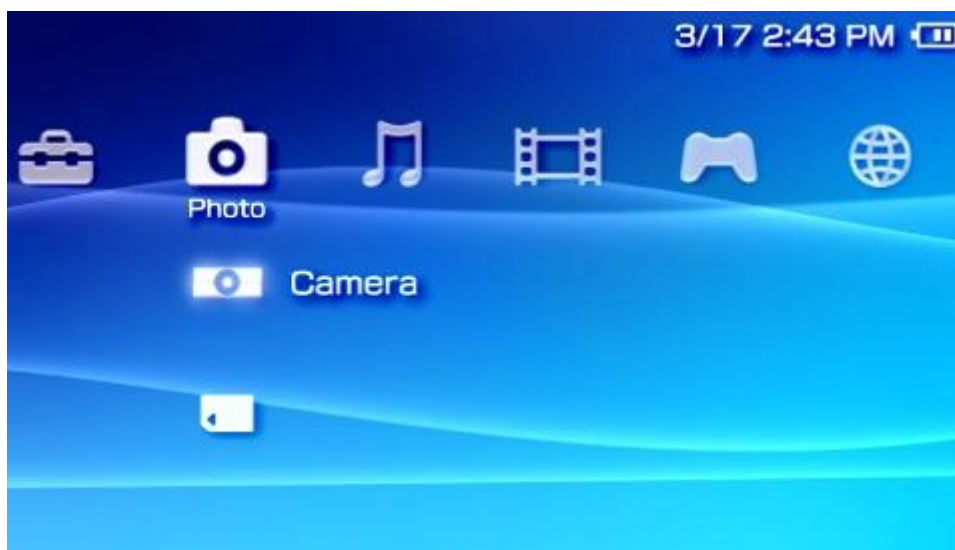
Finalment cal fer notar que tot aquest programari té llicències de programari lliure i està dissenyat per a ser utilitzat en sistemes operatius tipus Unix. És per això que tot el desenvolupament del projecte s'ha fet sota Linux.

3.2.4 El kernel de la PSP

El nucli (també conegut per l'anglicisme *kernel*) és el programari responsable de facilitar a les aplicacions del sistema un accés segur al maquinari i de gestionar els recursos (memòria, perifèrics, etc.). A diferència del nucli del sistemes operatius més coneguts com GNU/Linux o Windows, el nucli de la PSP no és monolític sinó que està format per mòduls. Les aplicacions d'usuari també es consideren mòduls. La comunicació entre els diferents mòduls es fa mitjançant l'exportació i importació de funcions entre aquests.

Els fitxers executables que suporta són el ELF (Executable and Linking Format), un format de fitxer estàndard per a executables, codi objecte i llibreries compartides; i el PRX (Relocatable eXecutable) similar al ELF. El format de fitxer ELF és utilitzat en els sistemes operatius tipus Unix i a molts *embedded systems* com la PlayStation, la Wii, alguns mòbils Sony Ericsson i naturalment la PSP.

3.2.5 El XMB



El XMB o XrossMediaBar és una interfície gràfica d'usuari (GUI) desenvolupada per Sony Computer Entertainment. Aquesta és la interfície principal de comunicació amb l'usuari a la PSP i d'altres productes de Sony.

Per poder llançar aplicacions des d'aquest sistema es necessita un fitxer EBOOT.PBP per a cada aplicació *homebrew* guardada a la Memory Stick. Aquest fitxer seria l'equivalent al .EXE del Windows i no és més que un contenidor amb l'executable principal i d'altres fitxers com un .PNG que representa la icona de l'aplicació al XMB.

Com s'ha explicat abans, es disposen de les eines necessàries per crear i gestionar aquests tipus de fitxers.

3.3 La conversió d'Allegro a la PSP

Per acabar amb aquest capítol a la figura 3.3 es presenta un diagrama detallat que explica en què consisteix la conversió de la llibreria Allegro a la consola portàtil PSP. Es marquen amb un quadre més gruix les parts que constitueixen l'objectiu central d'aquest PFC: la implementació dels *drivers* d'Allegro per a la PSP aconseguint d'aquesta manera fer totalment funcional a Allegro en aquesta plataforma.

El quadre més gran i principal representa a la llibreria Allegro i conté dos quadres clarament diferenciats: el primer engloba els mòduls funcionals que ofereix Allegro al programador d'aplicacions multimèdia (representat per un ninot), i el segon engloba els controladors de dispositiu o *drivers*, que fan de pont entre Allegro i la PSP. La comunicació amb la PSP es fa a través de les interfícies d'accés i abstracció del maquinari de la consola que proporciona el PSPSDK.

Tota aquesta estructura interna és completament transparent al programador, que només veu la interfície de la funcionalitat de la llibreria que Allegro li proporciona.

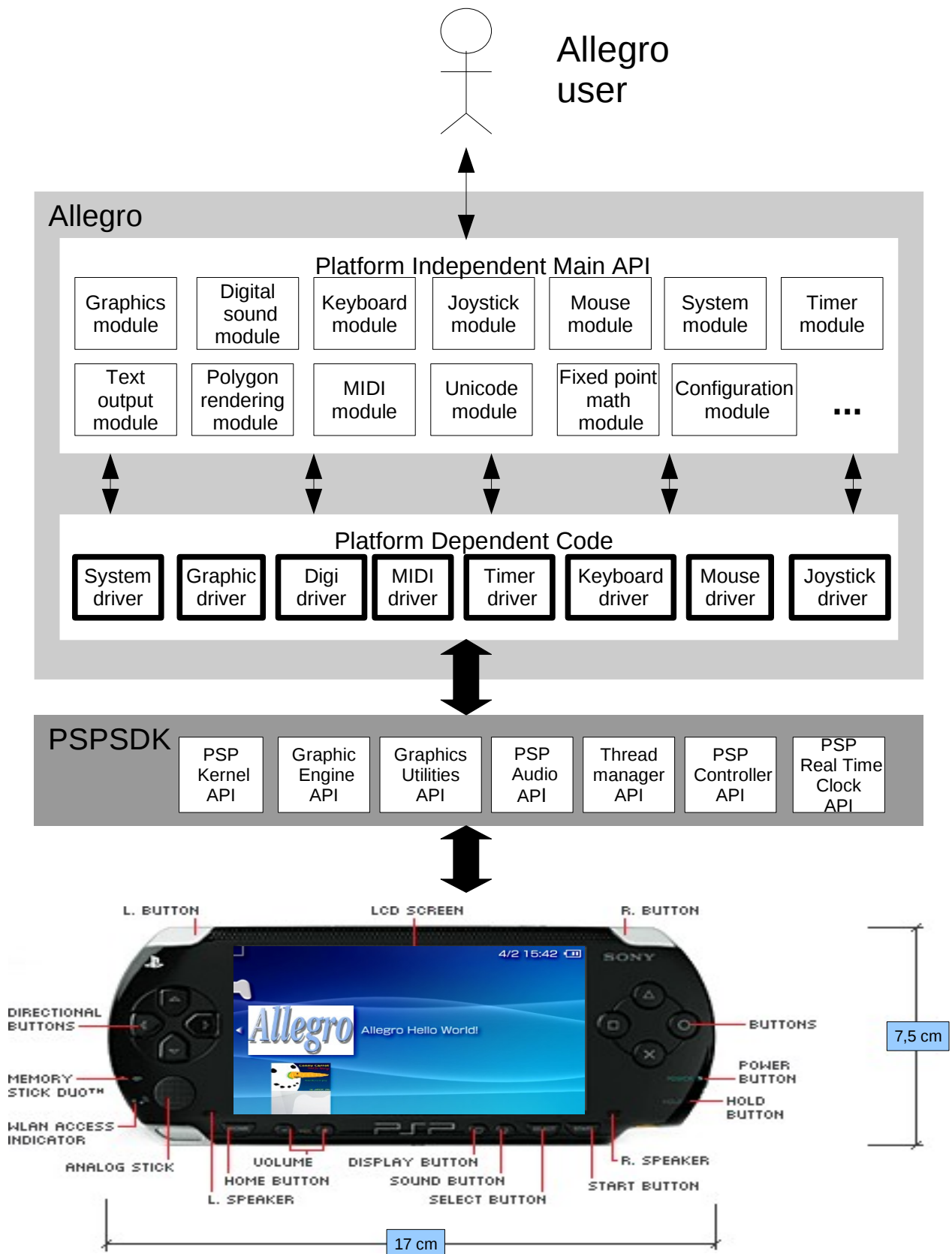


Figura 3.3 Allegro vs PSP

Capítol 4. Anàlisi de requisits

4.1 Introducció

L'anàlisi de requisits, aplicat a un sistema software, és una disciplina que pretén identificar i classificar les condicions que ha de complir aquest sistema. Aquestes condicions són els requisits.

Un cas habitual és quan els requisits es decideixen entre els clients d'un futur sistema software i l'empresa que realitzarà el sistema. Aquest PFC, però, no compleix aquest cas habitual ja que no s'està desenvolupant cap aplicació d'alt nivell que interaccionarà amb persones sinó que consisteix en desenvolupar una capa interna d'una llibreria de programació. En conseqüència els requisits, almenys una gran part, venen imposats i perfectament delimitats pel propi disseny de la llibreria Allegro.

Existeixen dos tipus de requisits, els funcionals i els no funcionals.

- Els requisits funcionals descriuen cadascuna de les funcions que ha d'efectuar el sistema. Així com cadascuna de les sortides que han de poder obtenir-se.
- Els requisits no funcionals no es refereixen directament a les funcions específiques que brinda el sistema, sinó a característiques com la l'eficiència, la portabilitat, etc.

4.2 Requisits funcionals

Com ja s'ha indicat anteriorment és Allegro qui imposa pràcticament tots els requisits. Concretament, per fer operativa la llibreria Allegro a la PSP s'ha d'implementar un conjunt de *drivers*, cadascun d'ells amb una funcionalitat ben definida i seguint un protocol de comunicació també ben definit.

Per a la realització d'aquest PFC s'ha implementat el següent:

- **Funcions del *driver* del sistema:** posada en marxa i acabament d'Allegro, gestió de les imatges a RAM, comunicació via consola amb l'usuari, retorn del mode gràfic segur de la plataforma perquè pugui Allegro pugui inicialitzar-se sempre, i enumeració de la resta de *drivers*.
- **Funcions del *driver* gràfic:** inicialització de modes gràfics amb diferents resolucions i profunditats de color, centrat de la pantalla, pantalles virtuals, *scrolling*, tècniques per a la generació d'animacions gràfiques com *double buffering*, *page flipping* i *triple buffering*, gestió de les imatges a la memòria de vídeo, i gestió de paletes de colors.
- **Funcions del *driver* del teclat:** emulació d'un teclat traduint els botons de la consola PSP a les tecles reals més comuns d'un teclat estàndard com l'*Enter* o les tecles del cursor; i lectura de pulsacions i alliberaments de tecles.
- **Funcions del *driver* del joystick o controlador:** lectura de la posició i de les pulsacions i alliberaments dels botons d'un *joystick* digital estàndard de quatre direccions i 8 botons.

- **Funcions del *driver* del temporitzador:** inicialització del temporitzador, mesura del temps amb una precisió de microsegon, funció de pausa.
- **Funcions del *driver* de so digital:** inicialització del mòdul de so segons el mode (mono o stereo), la resolució per mostra (de 8 o 16 bits) i la freqüència de mostreig; i reproducció d'àudio digital en format PCM amb fins a 64 veus.
- **Funcions gràfiques accelerades:** *blit* RAM/VRAM-RAM/VRAM, *blit* RAM/VRAM-Pantalla.

Aquest PFC no ha implementat tota la funcionalitat de tots els *drivers* principalment per dues raons. La primera és que hi ha funcions que només tenen sentit en ordinadors, com la gestió de finestres o la captura i restauració del context d'una aplicació. I la segona és per impossibilitat tècnica, ja siguin funcions per a la gravació de so o algunes funcions de teclat com la configuració dels LEDs. Així i tot, i malgrat aquesta impossibilitat, en molts casos sí que s'ha pogut implementar una funcionalitat gràcies a l'emulació, com el suport per resolucions amb 8 bits de color per píxel.

Hi ha un altre conjunt de requisits opcionals que són la implementació de les versions accelerades de les primitives gràfiques mitjançant el maquinari gràfic de la plataforma. En aquest PFC s'ha implementat la família de funcions relatives a la transferència d'imatges ja que és una operació molt comú i crítica.

4.2.1 Interfície drivers d'Allegro

Allegro defineix per a cada *driver* el conjunt de funcions que ha d'implementar el desenvolupador que vulgui convertir al llibreria a una altra plataforma. Aquest conjunt de funcions forma el que s'anomena una interfície o contracte.

A continuació s'exposa detalladament la interfície oficial dels *drivers* de la llibreria Allegro. En aquesta llista s'indiquen les funcions que no es poden implementar i les que es van considerar menys prioritàries.

Driver Sistema

- Inicialització i tancament.
- [No] Funcions varies gestió finestres.
- Mostra missatges de text.
- [No] Captura i restaura l'estat del mode consola.
- [No] Consulta profunditat de color i resolució de l'escriptori.
- Creació i destrucció de bitmaps a RAM.
- Funcions per a la gestió de sub bitmaps a RAM.
- [No] Funcions per l'intercanvi d'aplicacions en entorns multitasca.
- Retorna el mode gràfic segur de la plataforma (SAFE MODE).
- [Opcional] Gestió de mutex per a la sincronització de threads o fils d'execució.
- Retorna una llista dels drivers de cada tipus disponibles a la plataforma: gràfics, so, etc.

Driver Gràfics

- Inicialització mode gràfic i tancament.
- Suport resolucions estàndard menors o iguals a la pantalla de la PSP.
- Suport resolucions estàndard majors que la pantalla de la PSP.
- Suport per a 8, 15, 16 i 32 bits per píxel.
- [No] Suport per a 24 bits per píxel.
- Suport pantalles virtuals majors que la pantalla física i scrolling.
- Suport per a la sincronització vertical o vsync.
- Suport per a paletes de colors.
- Creació i destrucció de bitmaps de vídeo.
- [No] Creació i destrucció de bitmaps de sistema.
- Funcions per page flipping i triple buffering.
- [No] Funcions per a la gestió del punter del ratolí de la plataforma.
- [No] Funcions per a la captura i restauració del mode gràfic.

Driver Teclat

- Inicialització i desconnexió teclat.
- Lectura teclat per enquesta.
- [No] Lectura teclat per interrupcions.
- [No] Configuració LEDS teclat (Num Lock, Caps, etc.)
- Mapa botons PSP a scancodes.
- [Opcional] Ús d'un OSK (On screen keyboard) per emular un teclat real complet.

Driver Joystick

- Inicialització i sortida.
- Lectura per enquesta.
- Suport mode digital.
- [Opcional] Suport mode anàlogic.
- [Opcional] Calibra el joystick anàlogic.
- [Opcional] Guarda i carrega informació de la calibració.

Driver Temporitzador

- Inicialització i sortida.
- Mesura el temps real.
- Espera un temps determinat sense fer res.

Driver So digital

- Inicialització i sortida.
- Reproducció de so digital en format WAV PCM, 8 o 16 bits i mono o stereo.
- Funcions per a la gestió del volum, la freqüència i la distribució de canals o panning.
- [No] Funcions per a la gravació de so.

Driver MIDI

- [Opcional] Inicialització i sortida.
- [Opcional] Reproducció música en format MIDI.
- [Opcional] Llegeix instruments o patches d'un fitxer.

Driver Ratolí

- [Opcional] Inicialització i sortida.
- [Opcional] Lectura per enquesta.
- [No] Lectura per interrupcions.
- [No] Selecciona cursor de sistema.

Primitives gràfiques accelerades

- Transferència imatges RAM/VRAM – RAM/VRAM.
- Transferència d'imatges amb escalament RAM/VRAM – VRAM.
- [Opcional] Gestió sprites o imatges amb màscara.
- [Opcional] Altres primitives comuns com línies, triangles, polígons, etc.

4.2.2 Llista de requisits

Tot seguit la taula 4.1 fa una relació exhaustiva dels requisits funcionals per a la conversió d'Allegro a la PSP.

Cada requisit es representa per un codi i un títol. Aquest codi ajuda a la identificació del conjunt de requisits al qual pertany.

Aquests conjunts són els següents:

- SYS: Driver del sistema.
- GFX: Driver gràfic.
- KBD: Driver del teclat.
- CTRL: Driver del controlador o joystick.
- TMR: Driver del timer o temporitzador.
- DIGI: Driver per a la reproducció del so digital.
- MIDI: Driver per a la reproducció MIDI.
- MOU: Driver del ratolí.
- ACCEL: Primitives gràfiques accelerades.

Taula 4.1 Llista de requisits funcionals

Codi	Requisit
SYS-001	Inicialització sistema
SYS-002	Tancament sistema
SYS-003	Impressió missatges en mode text
SYS-004	Creació bitmap a RAM
SYS-005	Alliberació bitmap a RAM
SYS-006	Gestió sub bitmaps RAM
SYS-007	Inicialització de mode gràfic segur (safe mode)
SYS-008	[Opcional] Gestió objectes mutex
SYS-009	Creació llistes dels drivers disponibles a la plataforma
GFX-001	Inicialització del mòdul gràfic
GFX-002	Tancament del mòdul gràfic
GFX-003	Creació pantalla virtual
GFX-004	Suport de múltiples resolucions i profunditats de color
GFX-005	Creació de modes gràfics de DOS i Windows
GFX-006	Scroll de pantalla
GFX-007	Page flipping
GFX-008	Sincronització vertical
GFX-009	Creació bitmap a VRAM
GFX-010	Alliberació bitmap a VRAM
GFX-011	Triple buffering
GFX-012	Creació i modificació de paletes de colors

Codi	Requisit
KBD-001	Inicialització teclat virtual
KBD-002	Desconnexió teclat virtual
KBD-003	Gestió teclat per enquesta (polling mode)
KBD-004	Mapa botons PSP a scancodes.
KBD-005	[Opcional] Emulació teclat amb un OSK
CTRL-001	Instal·lació joystick
CTRL-002	Desconnexió joystick
CTRL-003	Joystick polling
CTRL-004	Suport mode digital
CTRL-005	[Opcional] Suport mode analògic
CTRL-006	[Opcional] Calibració joystick analògic
CTRL-007	[Opcional] Guardar/carregar informació calibració
TMR-001	Inicialització del temporitzador
TMR-002	Desconnexió del temporitzador
TMR-003	Mesura del temps transcorregut
TMR-004	Pausa del temporitzador
DIGI-001	Inicialització del mòdul de so digital
DIGI-002	Tancament del mòdul de so
DIGI-003	Reproducció de so digital en format PCM
DIGI-004	Creació efectes bàsics de so
MIDI-001	[Opcional] Inicialització del mòdul MIDI
MIDI-002	[Opcional] Tancament del mòdul MIDI
MIDI-003	[Opcional] Reproducció de música en format MIDI
MIDI-004	[Opcional] Llegeix instruments MIDI
MOU-001	[Opcional] Inicialització del ratolí virtual
MOU-002	[Opcional] Desconnexió del ratolí virtual
MOU-003	[Opcional] Lectura de l'estat del ratolí virtual
ACCEL-001	Blit per hardware
ACCEL-002	Blit amb escalament
ACCEL-003	[Opcional] Gestió sprites accelerada
ACCEL-004	[Opcional] Primitives gràfiques bàsiques amb acceleració

4.2.3 Descripció dels requisits

A continuació es fa una descripció textual de cadascun d'aquests requisits, juntament amb les seves entrades i sortides de manera que el lector entengui l'objectiu de cada requisit i en pugui fer un seguiment durant la vida del projecte.

SYS-001: Inicialització sistema

Descripció: Funció d'inicialització d'Allegro, tot comença a partir d'aquí i per tant aquesta funció és obligada.

Entrada: Res.

Sortida: Informació que identifica la plataforma PSP.

SYS-002: Tancament sistema

Descripció: Torna el sistema al mode inicial (normalment al XMB).

Entrada: Res.

Sortida: Surt de l'aplicació d'usuari de la forma especificada a la plataforma PSP.

SYS-003: Impressió missatges en mode text

Descripció: Allegro basa tota la seva comunicació amb l'usuari en mode gràfic i per tant específica de la plataforma. En cas de no poder inicialitzar un mode gràfic aquesta funcionalitat permet imprimir missatges de text de forma dependent de la plataforma.

Entrada: Missatge de text.

Sortida: Impressió a la pantalla del missatge de text en mode consola.

SYS-004: Creació bitmap a RAM

Descripció: Creació d'un bitmap o imatge a memòria RAM compatible amb la representació interna de la PSP.

Entrada: Dimensions de la imatge i profunditat de color.

Sortida: Imatge amb un format compatible amb Allegro i la PSP.

SYS-005: Alliberació bitmap a RAM

Descripció: Gestió de la memòria per alliberar una imatge creada a RAM.

Entrada: Una imatge existent.

Sortida: Imatge destruïda a memòria RAM.

SYS-006: Gestió sub bitmaps RAM

Descripció: Fer el tractament necessari quan es creïn sub imatges contingudes en d'altres imatges.

Entrada: Una sub imatge tot just creada i la seva imatge pare.

Sortida: La nova sub imatge transformada al format intern de la PSP.

SYS-007: Inicialització de mode gràfic segur (safe mode)

Descripció: En cas de no poder inicialitzar el mode gràfic sol·licitat per l'usuari (resolució i/o profunditat de color no suportades) necessitem proporcionar un mode segur al qual Allegro pugui recórrer.

Entrada: Res.

Sortida: Driver gràfic, resolució i profunditat de color que garanteixen la correcta inicialització del mòdul gràfic a la PSP.

SYS-008: [Opcional] Gestió objectes mutex

Descripció: Els objectes mutex (Mutual exclusion) permeten la compartició sincronitzada d'un recurs global entre més d'un procés o thread. La PSP opera amb threads i els mutex són necessaris.

Entrada: En la creació res. En les restants operacions un objecte mutex.

Sortida: Un mutex tot just creat, destruït, bloquejat o desbloquejat.

SYS-009: Creació llistes dels drivers disponibles a la plataforma

Descripció: Informa a Allegro, i per a cada tipus (gràfics, so, etc.), dels drivers disponibles a la plataforma. En el cas de la PSP, en general, hi ha un únic driver per tipus. En el cas de Unix, per exemple, hi ha més d'un driver de so digital (ALSA, OSS, ESD, etc.)

Entrada: Res.

Sortida: Per a cada tipus de driver retorna una llista en un format intern d'Allegro amb els drivers disponibles.

GFX-001: Inicialització del mòdul gràfic

Descripció: Entra al mode gràfic. Inicialitza la pantalla física segons el mode gràfic escollit (veure GFX-004).

Entrada: Resolució i profunditat de color.

Sortida: El bitmap d'Allegro que representa la pantalla física, l'emmagatzemament intern físic, el format de píxel, la velocitat de refresc del LCD i les característiques suportades del maquinari gràfic. A la PSP inicialitza el *Graphics Engine* (GE).

GFX-002: Tancament del mòdul gràfic

Descripció: Termina el mode gràfic.

Entrada: Res.

Sortida: Es termina el *Graphics Engine* (GE).

GFX-003: Creació pantalla virtual

Descripció: En el mateix moment que s'inicia el mòdul gràfic es crea una pantalla de dimensions majors que la pantalla física si així es vol. D'aquesta manera l'usuari pot fer scrolling i/o page flipping.

Entrada: Resolució de la pantalla virtual.

Sortida: El bitmap d'Allegro que representa la pantalla física passa a representar la pantalla virtual.

GFX-004: Suport de múltiples resolucions i profunditats de color

Descripció: Les resolucions que aprofiten millor la pantalla de la PSP són 480x272 (aspect ratio aprox. 16:9) i 368x272 centrat (aspect ratio aprox 4:3). També es soportada qualsevol resolució menor, 320x200, etc. Les profunditats de color suportades per Allegro són 8, 15, 16, 24 i 32 bits per píxel però la PSP només suporta de forma nativa 15, 16 i 32 bpp. Es pot suportar 8 bpp a la PSP mitjançant pseudoemulació. En quant al mode 24 bpp es pot substituir pel mode 32 bpp sense fer ús del canal alfa del color encara que es consumeix més memòria.

Entrada: Resolució i profunditat de color dintre del rang físic suportat.

Sortida: S'activa la resolució i profunditat demanades

GFX-005: Creació de modes gràfics de DOS i Windows

Descripció: Pensant en plataformes amb resolucions grans, es suporten resolucions de 640x480, 800x600, etc.

Entrada: Resolució estàndard de DOS o Windows i profunditat de color.

Sortida: S'activa la resolució demanada i es prepara el sistema d'escalament de primitives gràfiques.

GFX-006: Scroll de pantalla

Descripció: Mostrar a la pantalla física qualsevol àrea de la pantalla virtual activa. Permet fer tant scroll vertical com horitzontal.

Entrada: Coordenades dins de la pantalla virtual.

Sortida: La pantalla física mostra l'àrea demanada de la pantalla virtual.

GFX-007: Page flipping

Descripció: Mostrar a la pantalla física un altre bitmap de vídeo amb la mateixa resolució.

Entrada: Bitmap de vídeo amb la resolució gràfica actual.

Sortida: La pantalla física mostra el nou bitmap de vídeo.

GFX-008: Sincronització vertical

Descripció: Esperar al VBLANK, l'interval de temps al qual una pantalla no està refrescant la imatge. Serveix per a generar gràfics a una velocitat fixa i per aconseguir una transició suau entre successius quadres d'animació.

Entrada: Res.

Sortida: El VBLANK acaba de començar.

GFX-009: Creació bitmap a VRAM

Descripció: Igual que SYS-004 però a memòria de vídeo.

Entrada: Dimensions de la imatge i profunditat de color.

Sortida: Imatge a memòria de vídeo amb un format compatible amb Allegro i la PSP.

GFX-010: Alliberació bitmap a VRAM

Descripció: Igual que SYS-005 però a memòria de vídeo.

Entrada: Una imatge existent a VRAM.

Sortida: Imatge destruïda a VRAM.

GFX-011: Triple buffering

Descripció: Allegro ofereix al programador la funcionalitat de *triple buffering* si la plataforma subjacent la suporta. Aquest és el cas de la PSP. El *triple buffering* és una tècnica d'animació per a dibuixar gràfics de forma suau i suposa una millora de velocitat respecte el *double buffering*. Veure GFX-007.

Entrada: Bitmap de vídeo amb la resolució gràfica actual.

Sortida: La pantalla física mostrarà el nou bitmap de vídeo quan arribi el VBLANK. A diferència del *page flipping* la CPU delega el canvi d'imatge en el processador gràfic i té més temps disponible per dibuixar el següent marc d'animació, això explica per què teòricament és més ràpid.

GFX-012: Creació i modificació de paletes de colors

Descripció: Actualitza la paleta activa de 256 colors de la PSP.

Entrada: Un rang de colors de la paleta que es vol crear o modificar.

Sortida: La paleta de colors de la PSP s'ha actualitzat de la forma demanada.

KBD-001: Inicialització teclat virtual

Descripció: Inicialitza un mini teclat el més pràctic i funcional possible emulat pels botons de la PSP.

Entrada: Res.

Sortida: Es poden llegir pulsacions i alliberaments de tecles.

KBD-002: Desconnexió teclat virtual

Descripció: Desconnecta el teclat virtual.

Entrada: Res.

Sortida: No es pot usar el teclat.

KBD-003: Gestió teclat per enquesta (polling mode)

Descripció: Gestiona els esdeveniments de teclat per enquesta. El més adient seria fer-ho per interrupcions però a la PSP no és possible. Allegro contempla aquest mode atípic de llegir el teclat.

Entrada: Res.

Sortida: Informa de l'estat del teclat.

KBD-004: Mapa botons PSP a scancodes

Descripció: Relaciona els botons de la PSP a codis de teclat d'Allegro.

Entrada: La llista de botons de la PSP.

Sortida: La relació botó PSP – scancode d'Allegro independent de la plataforma.

KBD-005: [Opcional] Emulació teclat amb un OSK

Descripció: Emula un teclat complet mitjançant un teclat a pantalla (On screen keyboard).

Entrada: Res.

Sortida: Un teclat a la pantalla de la PSP controlat pels botons amb el mateix comportament d'un teclat real.

CTRL-001: Instal·lació joystick

Descripció: Inicialitza els joysticks.

Entrada: Res.

Sortida: Allegro té disponibles els joysticks de la plataforma.

CTRL-002: Desconnexió joystick

Descripció: Desconnecta el joysticks.

Entrada: Res.

Sortida: No es poden usar el joysticks.

CTRL-003: Joystick polling

Descripció: Gestiona els esdeveniments del joystick per enquesta.

Entrada: Res.

Sortida: Informa de l'estat del joystick.

CTRL-004: Suport mode digital

Descripció: Allegro suporta dos tipus de joystick: digital i analògic. En aquest cas es permet llegir un joystick digital.

Entrada: Els botons digitals de la PSP.

Sortida: Relaciona els botons digitals de la PSP amb un joystick digital independent de la plataforma.

CTRL-005: [Opcional] Suport mode analògic

Descripció: Permet Allegro llegir un joystick analògic.

Entrada: El controlador analògic de la PSP.

Sortida: Tradueix el controlador analògic de la PSP a un joystick analògic independent de la plataforma.

CTRL-006: [Opcional] Calibració joystick analògic

Descripció: La majoria de joysticks analògics necessiten ser calibrats abans de poder-se utilitzar.

Entrada: El joystick a calibrar.

Sortida: En cas d'èxit el joystick analògic està llest per ser utilitzat.

CTRL-007: [Opcional] Guardar/carregar informació calibració

Descripció: Guarda i carrega en un fitxer la informació de calibració del joystick analògic.

Entrada: Un fitxer qualsevol o el fitxer de configuració d'Allegro.

Sortida: El fitxer conté la informació de l'última calibració del joystick.

TMR-001: Inicialització del temporitzador

Descripció: Inicialitza el temporitzador.

Entrada: Res.

Sortida: S'ha donat d'alta un temporitzador que comença a mesurar el temps real.

TMR-002: Desconnexió del temporitzador

Descripció: Elimina el temporitzador.

Entrada: Res.

Sortida: S'ha eliminat el temporitzador.

TMR-003: Mesura del temps transcorregut

Descripció: Mesura el temps real periòdicament.

Entrada: Res.

Sortida: Informa Allegro del temps real transcorregut.

TMR-004: Pausa del temporitzador

Descripció: Durant un cert temps el temporitzador deixa de funcionar alliberant la CPU.

Entrada: Temps prefixat.

Sortida: El temporitzador deixa de mesurar el temps durant el temps prefixat.

DIGI-001: Inicialització del mòdul de so digital

Descripció: Inicialitza el l'àudio de la PSP.

Entrada: Nombre de veus de so.

Sortida: L'àudio de la PSP s'ha inicialitzat.

DIGI-002: Tancament del mòdul de so

Descripció: Allibera l'àudio de la PSP i el mesclador d'Allegro.

Entrada: Res.

Sortida: La PSP ja no té la capacitat de reproduir so.

DIGI-003: Reproducció de so digital en format PCM

Descripció: Reprodueix les veus de so actives.

Entrada: Veus de so actives i que contenen informació sonora.

Sortida: Es reproduïx per la PSP la informació sonora actual.

DIGI-004: Creació efectes bàsics de so

Descripció: Es llegeix i modifica el volum, la freqüència i el pan (distribució de canals) d'una veu.

Entrada: Veu de so activa i el seu volum, pitch i pan.

Sortida: El volum, pitch i pan de la veu llegit o modificat.

MIDI-001: [Opcional] Inicialització del mòdul MIDI

Descripció: Inicialitza la capacitat de la PSP per a reproduir música MIDI.

Entrada: Res.

Sortida: Ja es pot reproduir so MIDI.

MIDI-002: [Opcional] Tancament del mòdul MIDI

Descripció: Tanca el mòdul MIDI.

Entrada: Res.

Sortida: No es pot reproduir so MIDI.

MIDI-003: [Opcional] Reproducció de música en format MIDI

Descripció: Reprodueix fitxers de música MIDI.

Entrada: Fitxer MIDI a reproduir.

Sortida: Es comença a reproduir el fitxer amb música MIDI.

MIDI-004: [Opcional] Llegeix instruments MIDI

Descripció: Llegeix els instruments requerits per un fitxer MIDI.

Entrada: Fitxer MIDI i fitxer de patches o instruments.

Sortida: Els patches o instruments disponibles per a començar la reproducció.

MOU-001: [Opcional] Inicialització del ratolí virtual

Descripció: Inicialitza el mòdul del ratolí a la PSP potser mitjançant el controlador analògic.

Entrada: Res.

Sortida: El ratolí virtual està llest per ser llegit.

MOU-002: [Opcional] Desconnexió del ratolí virtual

Descripció: Elimina el ratolí virtual.

Entrada: Res.

Sortida: El ratolí virtual s'ha desconnectat.

MOU-003: [Opcional] Lectura de l'estat del ratolí virtual

Descripció: Llegeix per enquesta el moviment i els botons del ratolí. Veure KBD-003.

Entrada: Res.

Sortida: Informa de l'estat del ratolí.

ACCEL-001: Blit per hardware

Descripció: Copia una àrea rectangular d'una imatge origen a una imatge destí sense l'ús de la CPU.

Entrada: Imatges origen i destí, coordenades origen i destí i mida de l'àrea a copiar.

Sortida: S'ha copiat una part de la imatge origen a la imatge destí en funció de les coordenades.

ACCEL-002: Blit per hardware amb escalament

Descripció: Com ACCEL-001 excepte que pot escalar imatges.

Entrada: Imatges origen i destí, coordenades origen i destí, mida de l'àrea a copiar i mida de l'àrea destí on copiar-hi possiblement de mida diferent.

Sortida: S'ha copiat una àrea de la imatge origen a una àrea de la imatge destí. L'àrea copiada s'ha ampliat o reduït en funció de la mida de l'àrea destí.

ACCEL-003: [Opcional] Gestió sprites accelerada

Descripció: Dibuixa sprites fent ús de les capacitats gràfiques de la PSP.

Entrada: Un sprite i la imatge i les coordenades destí on dibuixar-lo.

Sortida: S'ha copiat l'sprite a la imatge i coordenades destí.

ACCEL-004: [Opcional] Primitives gràfiques bàsiques amb acceleració

Descripció: Dibuixa línies, polígons, etc. fent ús de les capacitats gràfiques de la PSP.

Entrada: L'objecte gràfic d'Allegro a dibuixar i la imatge i coordenades destí.

Sortida: S'ha dibuixat l'objecte gràfic al lloc indicat.

4.3 Requisits no funcionals

4.3.1 Eficiència

Una llibreria orientada a aplicacions multimèdia on la velocitat i fluïdesa és molt important requereix que sigui eficient. Allegro delega molta d'aquesta eficiència a la capa dependent o plataforma específica. Segons quina sigui aquesta plataforma i les seves capacitats aquest requisit s'aconsegueix fent ús de funcions gràfiques accelerades, d'execució concurrent o d'un sistema d'interrupcions.

A la PSP aquest requisit no funcional pren més protagonisme ja que Allegro està dissenyada per ser usada en ordinadors i la PSP per la seva naturalesa d'*embedded system* té carències de CPU i memòria si la comparem amb aquests ordinadors.

Afortunadament, el punt fort de la PSP es la capacitat gràfica i consegüentment aquesta desitjada eficiència d'Allegro s'aconsegueix delegant funcions com la transferència d'imatges al *Graphics Engine* (GE). Concretament la transferència d'imatges és una operació crítica molt comú en la gestió de gràfics i en la que intervé una gran quantitat de dades. Aquesta eficiència s'aconsegueix també, i com es veurà més endavant, dissenyant i/o implementant la resta de *drivers* mitjançant programació concurrent.

Satisfer en una alt grau aquest requisit és molt important ja que en cas contrari Allegro PSP seria de poca utilitat pràctica.

4.3.2 Portabilitat

La portabilitat és la capacitat que té un programari per executar-se en diferents plataformes, el codi font del programari pot tornar-se a utilitzar quan aquest programari passa d'una plataforma a una altra i no es necessari crear un altre codi font.

Com ja s'ha vist aquesta és una de les característiques que tenen les aplicacions escrites en Allegro. Algunes plataformes, però, no segueixen l'estàndard C al 100% i necessiten una codificació especial del programa principal o `main()` com passa en el cas de Windows o de la PSP. Allegro té en compte aquesta particularitat i ofereix el que anomena el "magic main" per convertir en temps de compilació l'aplicació de l'usuari en una aplicació compatible amb la plataforma específica.

Per altra banda el desenvolupament d'aplicacions Allegro per a la PSP ha de funcionar tant si es programa en C com si es fa amb C++. Per aconseguir això es segueixen unes convencions per mesclar codi C i C++.

4.3.3 Usabilitat

Aquest requisit no funcional té a veure amb la facilitat que es dona al programador d'aplicacions multimèdia que vol utilitzar Allegro. En quant a la programació l'API d'Allegro és senzilla i intuïtiva. I en quant a la integració amb l'entorn de desenvolupament de l'usuari és feina de la plataforma específica que sigui més o menys satisfactòria.

En el cas de la PSP s'han creat fitxers d'ajuda a l'usuari per a la compilació de les seves aplicacions, tenint en compte que l'entorn de desenvolupament serà tipus Unix, ja sigui a GNU/Linux o a Windows mitjançant un emulador POSIX com Cygwin.

4.3.4 Requisits de la llibreria Allegro

A continuació es detallen els requisits que tenen a veure amb la construcció, configuració, ús, etc. de la pròpia llibreria Allegro tenint en compte els requisits no funcionals:

Codi	Requisit
LIB-001	Creació d'un “magic main”
LIB-002	Configuració estàtica de la llibreria
LIB-003	Creació del <i>makefile</i> de la llibreria
LIB-004	Creació <i>scripts</i> de suport a l'usuari d'Allegro

Taula 4.2 Llista de requisits de la llibreria Allegro

LIB-001: Creació d'un *magic main*

Descripció: S'adapta el programa principal de l'usuari als requeriments d'una aplicació compatible amb PSP. Aquest *magic main* transforma una aplicació C estàndard de forma transparent a l'usuari i permet la portabilitat d'aplicacions Allegro ja existents.

Entrada: Un aplicació programada en Allegro.

Sortida: L'aplicació és compatible amb la PSP.

LIB-002: Configuració estàtica de la llibreria

Descripció: Definir quines llibreries i característiques són disponibles per a la construcció de la llibreria Allegro a la PSP (per exemple si el compilador té unes extensions concretes, si la plataforma es little o big endian, etc.)

Entrada: Els fitxers font que conformen la llibreria Allegro.

Sortida: La llibreria Allegro construïda per a la PSP.

LIB-003: Creació del *makefile* de la llibreria

Descripció: Definir un *makefile* de l'eina *GNU make* per a la PSP adaptant-lo al sistema de construcció d'Allegro.

Entrada: Res.

Sortida: Un *makefile* que conté les instruccions exactes per a construir i instal·lar la llibreria Allegro PSP.

LIB-004: Creació *scripts* de suport a l'usuari d'Allegro

Descripció: Definir *shell scripts* d'ajut a la compilació d'aplicacions Allegro. Per exemple per generar automàticament paràmetres pel compilador.

Entrada: L'entorn de desenvolupament específic de l'usuari.

Sortida: Un o més *shell scripts* que l'usuari d'Allegro pot utilitzar per automatitzar el procés de construcció de la seva aplicació.

Capítol 5. Planificació i costos

En aquest capítol s'enumeren i descriuen les tasques necessàries per a desenvolupar el projecte. A partir d'aquí es fa una previsió del temps i cost necessaris per a dur-lo a terme.

5.1 Planificació

L'estimació inicial consistia en determinar el temps necessari d'estudi i anàlisi per a obtenir, si no tots els requisits, si els principals. Tot seguit, i durant la immersió en el projecte i la determinació dels primers requisits es va completar aquesta estimació tenint ja en compte el desenvolupament principal. Tanmateix, donada la poca experiència en la gestió d'aquest tipus de projectes, aquestes estimacions poden resultar poc ajustades a la realitat. Així i tot, aquestes estimacions van permetre fitar la durada probable del projecte.

Durant tot el procés d'anàlisi i desenvolupament s'ha portat un autocontrol en forma de diari personal. Cada dia es redactava un petit informe amb els descobriments i avenços aconseguits. Aquest diari ha resultat ésser una eina molt valuosa per a la organització del coneixement i per a l'avaluació de la planificació i objectius aconseguits.

5.1.1 Identificació de les etapes del projecte

Uns dels avantatges de realitzar aquest projecte és que no ha estat necessari fer reunions interminables amb un hipotètic client per obtenir els requisits ja que es tractava d'ampliar un programari ja existent seguint, teòricament, unes regles clares.

Per començar, la nostra presa de requisits va consistir en determinar què significava exactament convertir la llibreria Allegro a la PSP. Degut a la naturalesa dels requisits, per aconseguir això era imprescindible obtenir una especificació i un disseny d'Allegro. Es va iniciar una presa de contacte amb la documentació oficial de l'API per a obtenir l'esquelet de l'especificació.

Seguint el fil de l'especificació el següent pas era obtenir una visió i comprensió del disseny d'Allegro analitzant l'estructura de directoris i fitxers de la llibreria, el codi font i la comunicació entre els diferents mòduls i capes. S'havia d'afegir el fet de que per entendre aquest funcionament intern era indispensable l'aprenentatge o repàs de molts conceptes com la programació de *drivers*, construcció de llibreries, gràfics, so, etc. i programació de baix nivell en general.

Una vegada assolits els coneixements mínims relacionats amb el bloc d'Allegro faltava la immersió en l'altre gran caixa negra: la PSP. En aquest cas, el coneixement previ era totalment inexistent. Es va seguir un altre procés de formació començant amb l'entorn de desenvolupament amb Linux i eines de programari lliure i acabant amb l'exploració de la PSP centrada en la visió del maquinari que el PSPSDK (PSP Software Development Kit) proporcionava. Hauria estat molt desitjable una documentació més completa i professional però no s'ha d'oblidar que en la programació de *homebrew* en general i en el PSPSDK en particular existeix un fort component d'enginyeria inversa i d'artesanía. Aquesta falta de documentació s'ha substituït amb molta

experimentació individual, amb l'anàlisi d'altre *homebrew* i amb la recerca de coneixement a webs i fòrums de discussió sobre programació a la PSP.

A partir d'aquí es van començar a implementar els *drivers* d'Allegro per a la PSP o el que és el mateix: a traduir la funcionalitat de l'antiga llibreria Allegro a la moderna consola portàtil Sony PlayStation Portable.

Aconseguit un primer prototipus dels *drivers* més importants, entrava en escena el gran darrer objectiu d'aquest PFC: la conversió del videojoc *Humphrey Remake* de Ignacio Pérez Gil programat principalment amb Allegro. Gràcies a la seva relativa complexitat i per estar orientat als recursos d'un PC aquest videojoc es va convertir en una primera prova de foc excel·lent per afinar l'eficiència i mesurar les possibilitats d'Allegro a la PSP. Aquesta exigent avaluació va fer que estigués obligat a descobrir formes més eficients de resoldre els problemes, per exemple usant l'acceleració gràfica de la PSP. Per altra banda, i a mesura que s'aprofundia en el coneixement d'Allegro i la PSP, també apareixien nous requisits.

Finalitzada la conversió del *Humphrey Remake* i amb l'experiència, el coneixement i els nous requisits obtinguts es va procedir a implementar la resta de drivers i la versió final dels ja implementats.

Finalment es van passar els jocs de proves. Aquests jocs de proves consistien en els petits programes d'exemple oficials de la llibreria que abracen totes les funcionalitats d'Allegro i en diferents videojocs existents de diversa complexitat.

L'últim pas ha estat el manual d'usuari d'Allegro PSP i aquesta memòria. Algunes seccions ja estaven mig elaborades amb anterioritat com els capítols d'introducció o anàlisi de requisits i alguns diagrames del capítol de disseny.

5.1.2 Tasques i dedicació estimada

La planificació inicial tenia com a objectiu finalitzar el projecte en un quadrimestre tenint en compte que es va començar al mes d'agost i que a la fi del quadrimestre de tardor hi ha un temps de vacances abans de l'inici de les classes del següent. Aproximadament 26 setmanes des de principis d'agost fins a principis de febrer.

Tenint en ment aquest termini i suposant que els caps de setmana no es treballaria en el projecte es va planificar una cadència de treball molt assequible i realista de 6 hores diàries de mitjana que suposava un total de 780 hores. En cas de no complir els terminis es tenia marge per afegir més hores. Amb aquesta planificació es complia el requisit mínim de 750 hores per a la realització d'un PFC de 37,5 crèdits.

Passem a veure les diferents tasques identificades i el temps que es pensava dedicar-hi:

Tasca	Setmanes	Hores
Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada	4	120
La PSP, el seu entorn de desenvolupament i el PSPSDK	3	90
Programació <i>scripts</i> per a la construcció de la llibreria	1	30
Implementació <i>drivers</i>	9	270
Sistema i gràfics	4	120
Teclat, joystick i ratolí	2	60
Temporitzador	1	30
So digital i MIDI	2	60
Port del <i>Humphrey Remake</i> , acceleració gràfica i nous probables requisits.	4	120
Proves	2	60
Documentació	3	90
TOTAL	26	780

Taula 5.1 Planificació inicial PFC: Tasques i duració

La idea inicial consistia en realitzar escalonadament l'estudi de la llibreria Allegro, la formació relacionada, l'anàlisi de la PSP i el procediment per a la construcció de la llibreria tenint així un punt de partida que determinaria la possibilitat o no de continuar amb el projecte.

A partir d'aquí es pensava començar amb el desenvolupament dels *drivers* més importants, el de sistema i el gràfic. Una vegada estigués assentat el desenvolupament la idea era iniciar l'anàlisi i conversió del *Humphrey Remake*. D'aquí sorgiria per necessitat el més que probable desenvolupament de l'acceleració gràfica.

Acabat ja el driver gràfic, almenys la part directament relacionada amb el *Humphrey Remake*, es procediria al desenvolupament de la resta de *drivers* menys el de so i el MIDI ja que la llibreria i la majoria d'aplicacions funcionarien perfectament sense àudio. Durant aquest procés seria necessari l'estudi de les llibreries relacionades del PSPSDK. Paral·lelament s'anirien incorporant i verificant aquests *drivers* al *Humphrey Remake*.

Tenint ja les primeres implementacions operatives dels *drivers*, la idea era desenvolupar amb exclusivitat els difícils, almenys per mi degut a la meva ignorància, *driver* de so digital i *driver* MIDI. Una vegada passada aquesta etapa d'àudio el pas final era integrar-ho tot al *Humphrey Remake* per finalitzar així la seva conversió.

A partir d'aquí es dedicaria la resta del temps a concloure tots els requisits no implementats possibles i a optimitzar els *drivers* en general, obtenint així la conversió Allegro PSP.

Finalment s'efectuarien tots els jocs de proves possibles i la documentació exigida.

El següent diagrama mostra aquesta distribució inicial de les tasques:

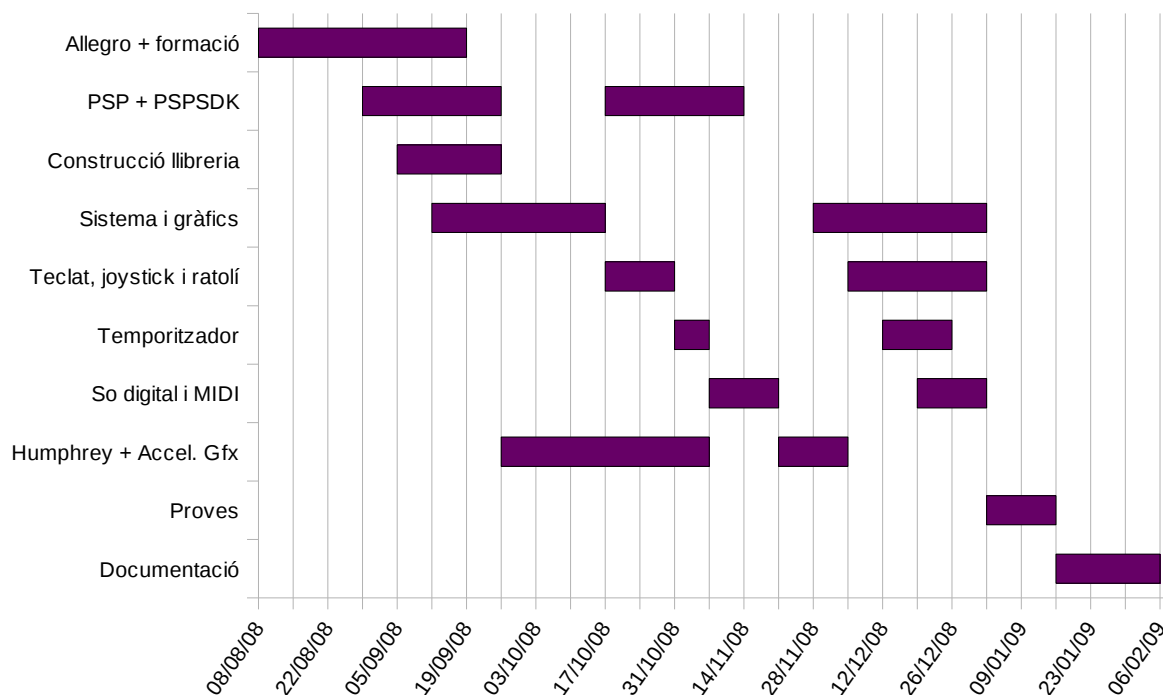


Figura 5.1 Planificació inicial PFC: Diagrama de Gantt

5.1.3 Dedicació real

Tot seguit s'intenta exposar amb bastant fidelitat la feina desenvolupada durant tot el transcurs del projecte.

- **Setmanes 1, 2, 3:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

- Manual API Allegro: estructures de dades i funcions públiques.
- Presa de contacte amb eines per analitzar grans projectes: GNU Global, GNU cflow, ...
- Estructura interna directoris i fitxers d'Allegro, anàlisi global del codi font per a obtenir la primera visió del disseny i funcionament.
- Inici i final d'Allegro: allegro_init() - allegro_exit(). Capa independent, capa dependent i llur comunicació. Drivers d'Allegro. Taules de mètodes virtuals.
- Lectura "The C Programming Language – Ritchie & Kernighan". Lectura "Sistemas Operativos – Tanenbaum", capítol E/S. Anàlisi drivers VESA, lectura programació baix nivell en DOS i VGA. Barreja de codi C i C++.
- Lectura "Linkers & Loaders – John R. Levine" i teoria sobre les llibreries i els seus diversos formats: estàtiques, compartides, dinàmiques, etc.

- **Setmanes 4, 5:**

La PSP, el seu entorn de desenvolupament i el PSPSDK.

Programació scripts per a la construcció de la llibreria.

- Compilació i instal·lació del PSP Toolchain i el PSPSDK.
- Familiarització amb l'entorn de desenvolupament PC- PSP.
- Anàlisi i execució dels exemples del PSPSDK.
- Anàlisi estructura de directoris i fitxers del PSPSDK i format dels executables a la PSP.
- Lectura a les webs més importants sobre desenvolupament de *homebrew* a la PSP.
- Lectura “Managing Projects with GNU Make – Robert Mecklenburg”
- Lectura “Beginning Linux Programming – Neil Matthew, Richard Stones”, capítol Shell programming.
- Primera versió del makefile.psp i configuració estàtica de la llibreria Allegro PSP.
- Primera compilació de la capa independent d'Allegro.
- Inici anàlisi llibreries PSPSDK.
- Presa de contacte amb l'aplicació PSPLink: eina d'ajuda al desenvolupament PC-PSP.

- **Setmana 6:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

La PSP, el seu entorn de desenvolupament i el PSPSDK.

Implementació drivers sistema i gràfics.

- Allegro: objectes BITMAP, representació del color i modes gràfics.
- PSPSDK: llibreries sceDisplay, sceGe i rutines d'inicialització d'executables (crt0).
- Primers prototipus del driver de sistema i del driver gràfic: 480x272x16 bpp
- Implementació magic main d'Allegro.

- **Setmana 7:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

La PSP, el seu entorn de desenvolupament i el PSPSDK.

Implementació driver teclat.

- Allegro: el mòdul de teclat, KEY_ scancodes.
- PSP controller: llibreria sceCtrl.
- Primera versió driver teclat.

- **Setmanes 8, 9:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

La PSP, el seu entorn de desenvolupament i el PSPSDK.

Port del *Humphrey Remake*, acceleració gràfica i nous probables requisits.

- Allegro: blitting i sprites.
- PSP: Cache, textures, alineació de memòria, codis d'instrucció del *Graphics Engine*.
- Inici port *Humphrey Remake*, mesures de velocitat d'execució i profiling.
- Aparició i anàlisi nou requisit per suportar resolucions de Windows.
- Primeres versions de les operacions de transferència accelerada d'imatges.
- Lectura “Beginning Linux Programming – Neil Matthew, Richard Stones”, capítol Debugging.
- Aprenentatge bàsic GNU gdb.
- Anàlisi d'altre *homebrew* de la PSP.

- **Setmana 10:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

La PSP, el seu entorn de desenvolupament i el PSPSDK.

Implementació driver so digital.

- Anàlisi llibreries *psaudio* i *psaudiolib* del PSPSDK.
- Anàlisi del mòdul del so digital i el mesclador o mixer d'Allegro.
- Lectura bàsica de www.dspguide.com i del capítol de so de “Física – Paul A. Tipler”.
- Primera versió driver so digital.

- **Setmana 11:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

La PSP, el seu entorn de desenvolupament i el PSPSDK.

Implementació driver temporitzador.

- El rellotge del IBM PC.
- Allegro: el temporitzador i les interrupcions d'usuari.
- Programació amb threads.
- Implementat el driver del temporitzador i verificada la seva precisió.

- **Setmanes 12, 13, 14:**

Implementació driver so digital.

Port del *Humphrey Remake*, acceleració gràfica i nous probables requisits.

- Implementació final driver so digital.
- Suport resolucions Windows: escalament primitives gràfiques d'Allegro.
- Anàlisi i construcció de la llibreria DUMB de reproducció de música per a la PSP.
- Primera versió d'Allegro PSP amb la integració de tots els drivers desenvolupats fins el moment: sistema, prototipus gràfics, teclat, temporitzador i so digital.
- Conversió *Humphrey Remake* amb la primera versió d'Allegro PSP, noves mesures, optimitzacions, correcció de bugs i proves finals.

- **Setmanes 15, 16:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

Documentació.

- Allegro: paletes de colors i bitmaps de vídeo.
- Lectura “Sistemas Operativos – Tanenbaum”, capítol Gestió de memòria.
- Inici documentació: diagrames varis arquitectura Allegro.
- Eines programari lliure: OpenOffice Draw i Gnuplot.

- **Setmanes 17, 18, 19, 20, 21:**

Implementació driver gràfic.

- Anàlisi, disseny i implementació de l'emulació del mode 8 bits de color: reescriptura primitives gràfiques per aquest mode.
- Suport per paletes de colors.
- Suport per resolucions estàndards menors de 480x272. Centrat de la imatge.
- Suport per resolucions amb 15 i 32 bits per píxel.
- Suport pantalles virtuals.
- Implementats l'scrolling, el triple buffering i el page flipping.
- Anàlisi, disseny i implementació d'un gestor de memòria de vídeo pel driver gràfic.

- **Setmana 22:**

Programació scripts per a la construcció de la llibreria.

Implementació driver MIDI.

- Creació scripts de suport al programador.
- Suport per música MIDI a partir del driver DIGMID d'Allegro basat en síntesi per taula d'ones.

- **Setmanes 23, 24:**

Anàlisi Allegro per obtenir especificació i disseny + Formació relacionada.

Implementació driver sistema.

Implementació driver teclat.

Implementació driver joystick.

Proves.

Documentació.

- Allegro: estructures de dades i visió del joystick.
- Versió final driver sistema.
- Versió final driver teclat.
- Implementat el driver de joystick digital.
- Proves amb els exemples oficials distribuïts amb Allegro.
- Documentació instruccions de construcció i ús d'Allegro PSP.
- Llançament de la versió final d'Allegro PSP.

- **Setmanes 25, 26:**

Proves.

- Integració amb C++: inici conversió del joc Ghouls'n Ghosts Remix.
- Aparició i anàlisi de nous requisits: draw_sprite() i clear_bitmap() accelerats.
- Correcció bugs i nova versió Allegro PSP.
- Conversió joc senzill: Conny Carrot.

- **Setmanes 27, 28, 29:**

Documentació.

- Eines programari lliure: OpenOffice Writer i Calc.
- Documentació memòria.
- Presentació.

Com es pot veure al següent diagrama la dedicació real és globalment semblant a la dedicació de la planificació inicial.

L'anàlisi d'Allegro, el PSPSDK i la formació ha durat una mica més de l'esperat i principalment s'ha fet en paral·lel amb les tasques d'implementació dels *drivers*. Ha estat el més lògic i pràctic.

Per altra banda l'elaboració dels *drivers* té dues etapes ben diferenciades. A la primera etapa es va implementar el necessari per tenir una versió d'Allegro capaç d'executar el joc Humphrey. A la segona etapa es va implementar el *driver* gràfic completament (amb diferència el *driver* més important i que més feina ha donat) i quasi tots els *drivers* restants.

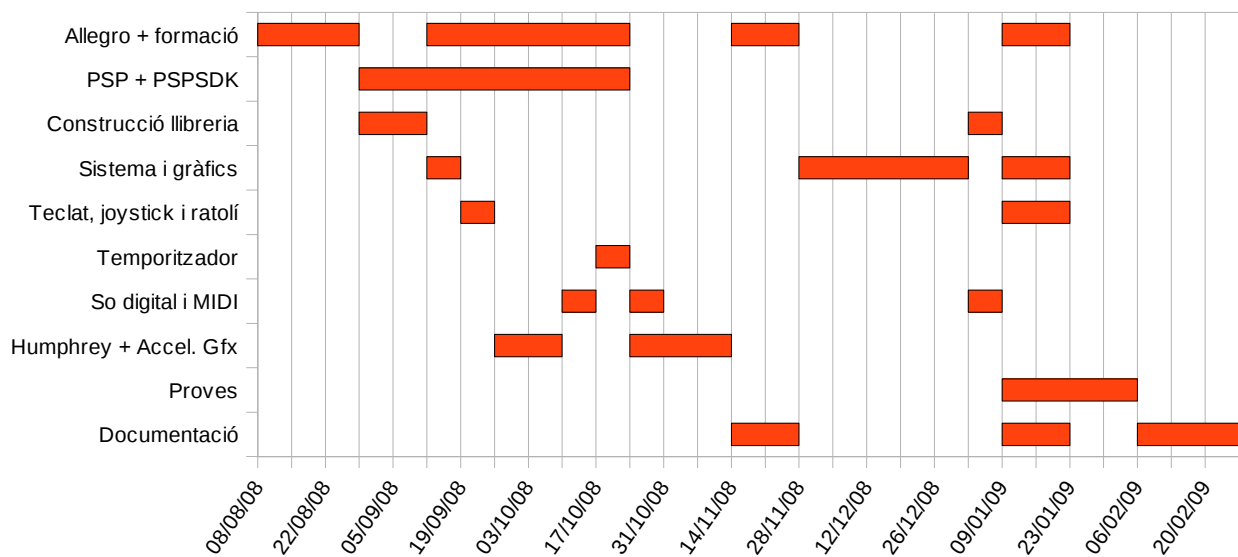


Figura 5.2 Dedicació real PFC: Diagrama de Gantt

5.1.4 Formació

Les múltiples àrees de coneixement abraçades per aquest PFC era un dels factors més atractius per al seu desenvolupament. En moltes d'aquestes àrees el meu coneixement era limitat o quasi nul, i per tant era necessari fer una bona inversió en formació.

A continuació es detallen les principals disciplines de coneixement amb els aspectes més aprofundits i assimilats amb èxit:

- **Llenguatge C/C++**
 - **Ús de funcions *inline*:** funcions que volem que s'executin més ràpidament estalviant el procés costós de crida i retorn de subrutina.
 - **Vtables o taules de mètodes virtuals:** permet en temps d'execució executar un mètode real o un altre a partir d'un mètode virtual genèric.

- **Name mangling:** tècnica usada pel compilador de C++ per fer únics els identificadors del codi d'una aplicació o llibreria ja que en aquest llenguatge el programador pot definir, per exemple, diferents funcions amb el mateix identificador. Això s'anomena polimorfisme.
- **Macros definides pels compiladors de C estàndards:** aquestes macros defineixen variables d'entorn com la versió, el dialecte o característiques del compilador i sistema operatiu subjacent.
- **Dialectes i estàndards del llenguatge C:** C90, C99, ... determinen quines característiques estan definides o no al llenguatge.
- **Àmbit de variables i funcions:** domini de l'ús dels identificadors *static* i *extern* per determinar la visibilitat de variables i funcions.

- **Llibreries i executables**
 - **Tipus de llibreries:** estàtiques, dinàmiques, compartides i carregades dinàmicament. Formats de llibreries a Linux i Windows amb les seves semblances i diferències.
 - **Teoria de enllaçadors o *linkers* i carregadors de codi.**
 - **Formats de fitxers executables:** EXE, ELF, PBP, PRX.
 - **Ús del GNU make per a la construcció de llibreries.**

- **Programació directa del maquinari en el PC-DOS**
 - VGA BIOS pels gràfics, programació del temporitzador, el teclat, la Sound Blaster, ...

- **Drivers i sistemes operatius**
 - **Teoria de drivers:** comunicació amb els controladors del maquinari i situació en l'arquitectura d'un sistema operatiu.
 - **Gestió dels drivers:** programació asíncrona amb l'ús d'interrupcions o programació per enquesta o *polling* amb l'ús de *threads* o fils d'execució.
 - **Exemples de disseny de drivers gràfics:** VESA 2.0, VESA 3.0, VESA VBE/AF, dispositius framebuffer de Linux, ...

- **Linux i eines de desenvolupament**
 - **Bash:** com analitzar i escriure fitxers *scripts* o de comandes grans. Ús d'expressions regulars.
 - **El compilador GCC:** ús d'opcions avançades i opcions particulars pel processador MIPS.
 - **GNU make:** com analitzar i escriure makefiles complexos.

- **Gràfics**
 - **Bitmaps:** representacions internes, *pitch o stride*.
 - **Colors:** representacions RGB, BGR, HSV, indexada.
 - **Textures:** ús i representació. Mapatge UV i tècniques avançades de mapatge de textures.
 - **Tècniques de programació:** *double buffering, triple buffering, scrolling, page flipping, vertical sync*.

- **Àudio**
 - **Formats:** representacions PCM, WAV, VOC.
 - **Teoria bàsica tractament digital del senyal:** principis físics, procés de mostreig.
 - **Algorismes d'àudio avançats:** mescla de veus o *mixing* i *streaming* per a la reproducció de grans fitxers d'àudio.

- **Gestió de la memòria**
 - **Memory Management Unit (MMU):** memòria virtual, paginació.
 - **Memòria cache:** tipus de cache i la seva gestió.
 - **Alineació de dades a memòria.**
 - **Algorismes de gestió de memòria.**

5.2 Anàlisi econòmic

S'ha realitzat l'anàlisi del cost total d'aquest PFC tenint en compte el cost de personal, de recursos materials consumits (maquinari i programari) i d'ocupació d'espai (tenint en compte els serveis associats a aquest espai com llum, Internet, etc.) Per a obtenir el cost de personal s'ha tingut en compte l'estimació inicial del temps de dedicació del projecte.

5.2.1 Cost de personal

En el desenvolupament d'aquest PFC són necessaris al menys dos perfils: el d'analista i el de programador. A més, en tot projecte és molt comú la presència d'un cap de projecte qui dirigeix les operacions de la resta de l'equip.

En l'actualitat aquests perfils cobren sous de l'ordre dels que es mostren a la taula 5.2, on es suposa una mitjana de 240 dies laborables de 8 hores l'any (1920 hores en total).

Perfil	Sou anual brut	Cost/hora
Cap de projecte	134.400 €	70,00 €
Analista	96.000 €	50,00 €
Programador	48.000 €	25,00 €

Taula 5.2 Cost per perfil professional

La duració total del projecte estimada, tenint en compte la taula 5-1 de l'apartat de planificació, és de **780 hores**. D'aquest total s'estima un 10% de tasques per l'analista i la resta pel programador. Per últim s'hauria de sumar la feina del cap de projecte estimada en un 7% del total, aproximadament **55 hores més**.

Totes aquestes estimacions donen com a resultat el següent cost de personal per a la realització d'aquest projecte:

	Hores	Cost mig	Cost
Cap de projecte	55	70,00 €/hora	3.850,00 €
Analista	78	50,00 €/hora	3.900,00 €
Programador	702	25,00 €/hora	17.550,00 €
TOTAL			25.300,00 €

Taula 5.3 Cost total de personal

5.2.2 Cost de maquinari

Aquest seria l'equip de treball per a la realització del projecte:

- Un PC Intel Pentium 4 HT 3,20Ghz, 1 GiB de RAM, 160 GiB de disc dur i monitor de 19". L'equipament informàtic en ser un actiu de l'empresa i tenir una vida mitjana de 3 anys té un cost d'amortització per compensar aquesta depreciació. Durant els 6 mesos de duració del projecte s'ha de pagar la part proporcional d'aquest cost d'amortització.

Equip	Preu equip	Termini d'amortització	Duració projecte	% imputable	Cost
Ordinador sobretaula	1.600 €	36 mesos	6 mesos	$(6/36)*100=16,67\%$	266,67 €

- Una consola Sony PlayStation Portable Slim + perifèrics i cables necessaris: **200 €**

El cost total de maquinari és doncs:

Concepte	Cost
Equip informàtic	266,67 €
Consola PSP	200,00 €
TOTAL	466,67 €

Taula 5.4 Cost total de maquinari

5.2.3 Cost de programari

Per a la realització d'aquest projecte s'ha utilitzat exclusivament programari lliure i per tant amb cost 0 € :

- GNU/Linux Fedora 9
- Llibreria Allegro
- PSP Software Development Kit amb llicència BSD
- IDE Geany
- Altres eines de desenvolupament i anàlisi: GNU Make, GNU GLOBAL, PSPLink, etc.
- OpenOffice.org 3
- Navegador web Mozilla Firefox

5.2.4 Cost d'ocupació

S'ha estimat un cost de lloguer de 600 € mensuals per una oficina on desenvolupar el projecte, aquest preu inclou els serveis d'accés a Internet, telèfon i electricitat. El cost total d'ocupació és doncs $600 \text{ €} * 6 \text{ mesos} = 3.600 \text{ €}$

5.2.5 Cost total del projecte

Finalment, calculem el cost total del projecte:

Concepte	Cost
Cost de personal	25.300,00 €
Cost de maquinari	466,67 €
Cost de programari	0,00 €
Cost d'ocupació	3.600,00 €
TOTAL	29.366,67 €

Taula 5.5 Cost total del projecte

Capítol 6. Especificació

L'especificació pretén definir clarament un sistema de programari, però sense determinar com es farà. És una de les parts decisives en el seu desenvolupament.

L'especificació té bàsicament dues funcions, una servir de contracte entre el client i l'empresa sobre els límits del programari, i l'altre servir de base per a la resta d'etapes en el desenvolupament d'aquest programari.

Per a la realització del document de l'especificació s'utilitza la notació UML (Unified Modeling Language). UML és el llenguatge de modelat de sistemes d'informació més conegut i emprat actualment. Aquest llenguatge gràfic permet visualitzar, especificar, construir i documentar un sistema de programari, en el cas concret d'aquest PFC una llibreria de programació.

L'especificació completa d'Allegro és fora de l'abast d'aquest PFC. Tanmateix sí que s'especificarà el conjunt de mòduls i funcionalitats més significatives que tot programador d'Allegro hauria de conèixer. A més, aquesta especificació és necessària per contextualitzar l'objectiu principal del PFC de programar la capa dependent o *drivers* d'Allegro per a la plataforma PSP.

Es donarà una visió d'Allegro de dues formes:

- Una visió funcional que mostri què pot fer un programador d'aplicacions amb Allegro.
- Una visió estructural on es mostren les entitats que pertanyen al domini d'Allegro i les relacions entre elles.

Al capítol següent, dedicat al disseny, apareixeran els *drivers* i es donarà una visió de la interacció i el flux d'informació existent entre la capa independent i la capa dependent.

Per donar la visió funcional s'utilitzarà el diagrama de casos d'ús d'UML i la seva especificació textual detallada. I per a la visió estructural s'utilitzarà el diagrama de classes d'alt nivell.

6.1 Casos d'ús

El diagrama de casos d'ús mostra la funcionalitat del sistema des d'un punt de vista extern. Hi intervenen uns actors que són els elements fora del sistema que interaccionen amb el sistema i hi intervenen els casos d'ús que són els procediments amb els quals els actors interaccionen amb el sistema.

6.1.1 Definició d'actors

Allegro només és utilitzat pel programador d'aplicacions multimèdia (o per la pròpia aplicació multimèdia segons com es miri). Aquest tindrà accés a tota la funcionalitat d'Allegro.

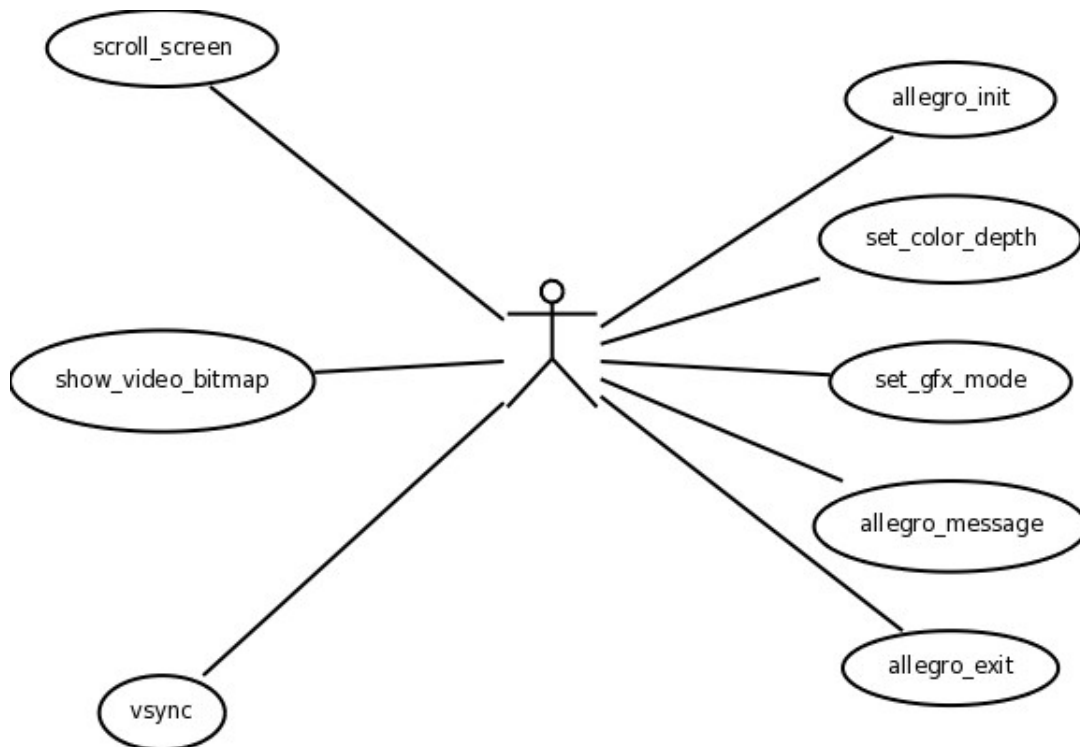


A qualsevol part d'aquest capítol on aparegui aquest actor es fa referència al programador usuari de la llibreria Allegro.

Els casos d'ús que es mostren a continuació s'han agrupat en diferents blocs funcionals per facilitar la seva lectura i comprensió. Aquests són:

- Inicialització d'Allegro i gestió de la pantalla
- Gestió de bitmaps
- Transferència d'imatges i sprites
- Funcions de so
- Funcions del temporitzador
- Gestió dels perifèrics d'entrada

6.1.2 Inicialització d'Allegro i gestió de la pantalla



Cas d'ús: allegro_init

Descripció: Inicialitza la llibreria Allegro. S'ha de cridar aquesta funció abans de fer-hi res.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador inicia la seva aplicació Allegro.	2. Allegro detecta la plataforma dependent on s'està executant i la posa en marxa. A partir d'aquest moment el programador ja podrà cridar la resta de funcions de la llibreria.

Cursos alternatius:

2. En cas de no detectar la plataforma subjacent Allegro avisa de l'error al programador.

Cas d'ús: set_color_depth

Descripció: Estableix el format de píxel pels següents modes gràfics i bitmaps.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol treballar amb una de les profunditats de color disponibles a Allegro: 8, 15, 16, 24 i 32.	2. A partir d'aquest moment l'establiment d'un mode gràfic i la creació d'un bitmap prendrà aquest format de píxel escollit pel programador.

Cas d'ús: set_gfx_mode

Descripció: Activa el mode gràfic en funció del driver gràfic (normalment, i per portabilitat, el driver gràfic per defecte de la plataforma subjacent), la resolució de la pantalla física i, eventualment, la resolució de la pantalla virtual escollits.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol establir un mode gràfic indicant un driver gràfic determinat i una resolució de pantalla física determinada. Indica també la resolució de la pantalla virtual en cas de necessitar la funcionalitat d'scrolling o page flipping,	2. Inicialitza el driver gràfic corresponent i activa la pantalla física amb la resolució demanada i la profunditat de color per defecte (8 bits per píxel) o l'establida en una crida prèvia a set_color_depth(). Eventualment s'activa la pantalla virtual estenent les dimensions de la pantalla física.

Cursos alternatius:

2. Retorna error al programador en cas de no poder establir el mode gràfic desitjat, ja sigui per no poder inicialitzar el driver gràfic demanat i/o per resolució no suportada (tant física com virtual) i/o per profunditat de color actual no suportada.

Cas d'ús: allegro_message

Descripció: Mostra missatges d'error en mode text als usuaris de l'aplicació multimèdia.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol mostrar un missatge a l'usuari d'una forma independent de la plataforma i en mode text. Típicament quan algun mòdul d'Allegro no es pot inicialitzar.	2. Allegro mostra el missatge de text usant el format de la funció estàndard printf().

Cursos alternatius:

2. En cas de no haver inicialitzat Allegro correctament el missatge pot no mostrar-se.

Cas d'ús: allegro_exit

Descripció: Tanca la llibreria Allegro. Normalment es crida de forma automàtica aquesta funció en sortir de l'aplicació.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador finalitza la seva aplicació Allegro.	2. Allegro retorna la plataforma al mode text o consola i desconnecta els dispositius que estaven instal·lats: el ratolí, el teclat, el temporitzador, etc.

Cas d'ús: scroll_screen

Descripció: Intenta moure la pantalla física perquè mostri una altra àrea de la pantalla virtual. Inicialment la pantalla física està posicionada a la cantonada superior esquerra de la pantalla virtual.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
-------------------	----------------------

1. El programador demana un desplaçament de pantalla indicant les coordenades destí dins de la pantalla virtual on vol posicionar la cantonada superior esquerra de la pantalla física.	2. Allegro mou la pantalla física en funció de les coordenades demanades i sempre que quedi completament dins de la pantalla virtual.
---	---

Cursos alternatius:

2. Retorna error al programador si el driver gràfic actual de la plataforma subjacent no suporta scrolling o en cas que el desplaçament demanat de la pantalla física ultrapassi les dimensions de la pantalla virtual.

Cas d'ús: show_video_bitmap

Descripció: Canvia la pantalla física perquè mostri el bitmap de vídeo especificat.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol efectuar un page flipping indicant el nou bitmap de vídeo a visualitzar.	2. La pantalla física passa a visualitzar el nou bitmap de vídeo només si aquest té les mateixes dimensions de la pantalla.

Cursos alternatius:

2. Es retorna error al programador si el bitmap indicat no és de vídeo i/o les seves dimensions no coincideixen amb les dimensions de la pantalla física.

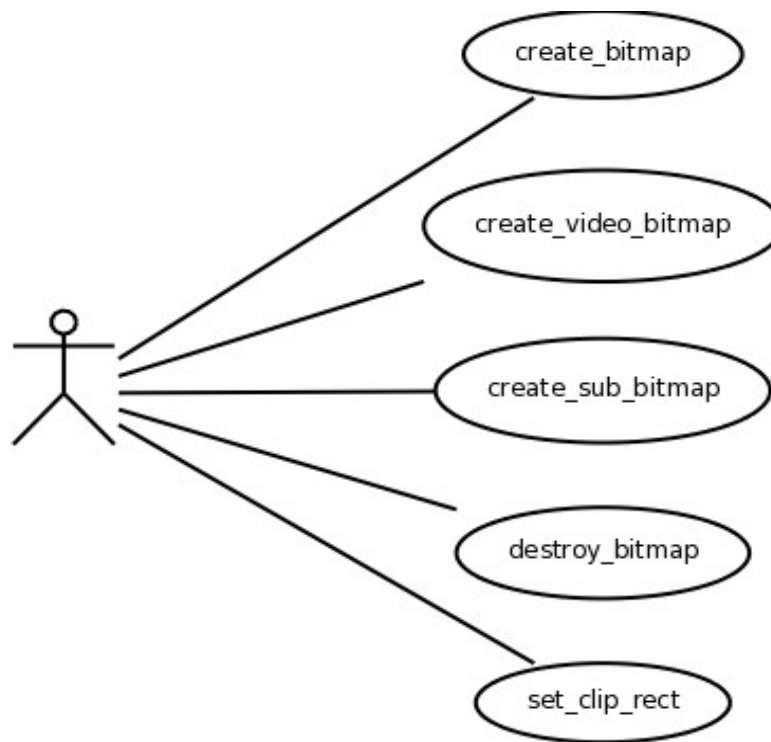
Cas d'ús: vsync

Descripció: Espera que comenci un VBLANK. Aquesta funció es crida automàticament abans de, per exemple, canviar la paleta de colors o de fer un desplaçament de pantalla.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol sincronitzar-se amb la interrupció vertical del monitor.	2. Allegro espera l'inici del següent VBLANK per a continuació retornar el control al programador.

6.1.3 Gestió de bitmaps



Cas d'ús: create_bitmap

Descripció: Crea un bitmap a la RAM del sistema. Totes les funcions gràfiques d'Allegro dibuixen a objectes bitmap, àrees de memòria que representen imatges rectangulars. Aquest tipus de bitmap es sol utilitzar per emmagatzemar gràfics o com a espais temporals de dibuix per fer double buffering. A més tot bitmap té associat un rectangle de clipping o retallament que indica l'àrea de la imatge on és permès dibuixar.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol crear un objecte bitmap d'unes dimensions determinades a la memòria estàndard del sistema.	2. Allegro crea un bitmap amb les dimensions desitjades i amb la profunditat de color per defecte o la indicada prèviament amb set_color_depth(). L'àrea del clipping s'estableix a la mida total del bitmap.

Cursos alternatius:

2. Es retorna error al programador en cas que el bitmap no es pugui crear, normalment per falta de RAM.

Cas d'ús: create_video_bitmap

Descripció: Crea un bitmap a memòria de vídeo del sistema. Aquest tipus de bitmap es pot utilitzar per a aprofitar les operacions accelerades de vídeo a la plataforma subjacent, o per crear múltiples pàgines de vídeo que poden ser visualitzades amb les funcions de page flipping i triple buffering.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol crear un objecte bitmap d'unes dimensions determinades a la memòria de vídeo.	2. Allegro crea un bitmap a memòria de vídeo amb les dimensions desitjades i amb la profunditat de color per defecte o la indicada prèviament amb <code>set_color_depth()</code> . L'àrea del clipping s'estableix a la mida total del bitmap.

Cursos alternatius:

2. Es retorna error al programador en cas que el bitmap no es pugui crear, normalment per falta de memòria de vídeo.

Cas d'ús: create_sub_bitmap

Descripció: Crea un un sub-bitmap, un bitmap que comparteix àrea de dibuix amb un bitmap ja existent però amb la seva pròpia mida i el seu propi rectangle de clipping. Aquest tipus de bitmap és útil per dividir un altre bitmap en diferents seccions.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol crear un sub-bitmap posicionat a un punt determinat d'un bitmap pare i amb una mida determinada.	2. Allegro crea un sub-bitmap inclòs dins del bitmap pare indicat i amb la mida desitjada (retallant si ultrapassa els límits del bitmap pare).

Cursos alternatius:

2. Es retorna error al programador si no es pot crear el sub-bitmap.

Cas d'ús: destroy_bitmap

Descripció: Elimina un bitmap a RAM, bitmap de vídeo o sub-bitmap.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
-------------------	----------------------

1. El programador ja no necessita el bitmap i el vol destruir.	2. Allegro allibera la memòria que conté aquest bitmap, ja sigui un bitmap de memòria, de vídeo o sub-bitmap.
--	---

Cas d'ús: set_clip_rect

Descripció: Estableix el rectangle de clipping o retallament d'un bitmap. Res es dibuixarà fora d'aquest rectangle.

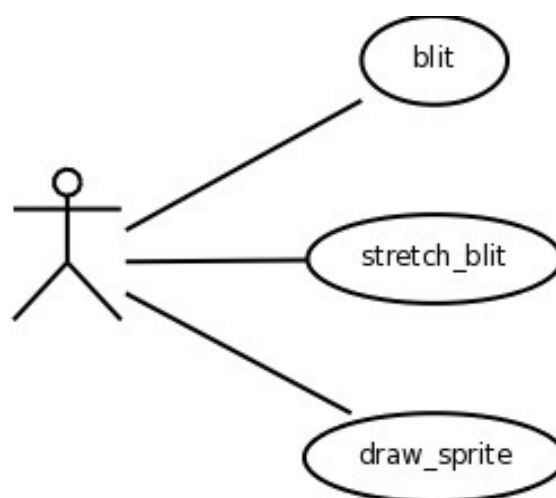
Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol delimitar l'àrea de dibuix dins d'un bitmap indicant les cantonades superior esquerra i inferior dreta d'un rectangle que representa aquesta àrea.	2. Si les coordenades indicades representen un rectangle dins el bitmap s'estableix l'àrea de clipping del bitmap.

Cursos alternatius:

2. Si les coordenades indicades pel programador no representen un rectangle o si la intersecció del rectangle amb el bitmap és buida, es desactiva la possibilitat de dibuixar al bitmap en qüestió.

6.1.4 Transferència d'imatges i sprites



Cas d'ús: blit

Descripció: Copia una àrea rectangular d'un bitmap origen a un bitmap destí. Normalment ambdós bitmaps tindran el mateix format de píxel. En cas contrari aquesta funció pot utilitzar-se per convertir imatges d'un format de píxel a un altre.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol copiar una àrea d'una imatge origen a una imatge destí. Aquesta àrea s'indica amb la seva cantonada superior esquerra dins de la imatge origen i la seva mida.	<p>2. Allegro retalla l'àrea origen a copiar si ultrapassa els límits del bitmap origen que la conté.</p> <p>3. Es fa la intersecció d'aquesta àrea eventualment retallada amb el rectangle de clipping del bitmap destí.</p> <p>4. Si el resultat d'aquesta intersecció és una àrea no buida llavors es copia a la zona desitjada del bitmap destí.</p>

Cursos alternatius:

4. Si el bitmap origen i el bitmap destí tenen diferents formats de píxel es copia l'àrea resultant fent la conversió de píxel origen-destí en funció del mode de conversió de color establert per defecte per Allegro o pel programador.

Cas d'ús: stretch_blit

Descripció: Escala una àrea rectangular d'un bitmap origen a un bitmap destí. Igual que blit(), excepte que pot escalar imatges. En aquest cas els bitmaps origen i destí han de tenir el mateix format de píxel.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol copiar una àrea d'una imatge origen a una àrea d'una imatge destí. Es defineix un rectangle origen a partir de la seva cantonada superior esquerra dins la imatge origen i la seva mida. Es defineix de la mateixa manera un rectangle destí al bitmap destí normalment de diferent mida.	2. Allegro copia, estirant o reduint en cas necessari, el rectangle origen al rectangle destí i tenint en compte el rectangle de clipping o retallament del bitmap destí.

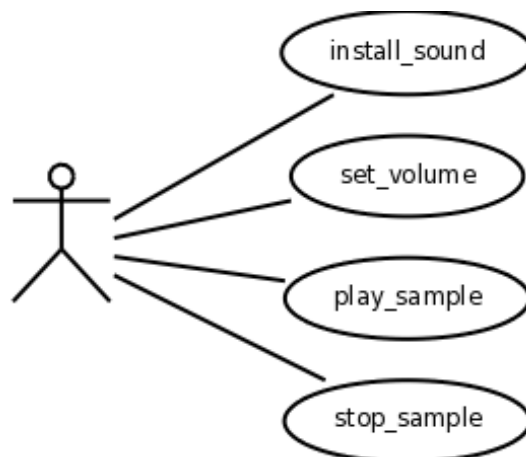
Cas d'ús: draw_sprite

Descripció: Dibuixa una còpia d'un sprite a un bitmap destí. Usa un mode de dibuixat on els píxels transparents no es mostren de manera que l'sprite s'integra amb la imatge de fons que representa el bitmap.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol dibuixar un sprite a un bitmap destí a la posició indicada.	2. Allegro dibuixa l'sprite al bitmap destí saltant-se els píxels transparents i tenint en compte l'àrea de clipping del bitmap destí.

6.1.5 Funcions de so



Cas d'ús: install_sound

Descripció: Inicialitza el mòdul de so. Se li indica tant el driver de so digital com el driver de so MIDI desitjats. Normalment, i per portabilitat, s'indiquen els drivers per defecte de la plataforma subjacent.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador té intenció de reproduir so a la seva aplicació multimèdia i usant uns drivers determinats.	2. Allegro inicialitza el mòdul de so correctament i a partir d'aquest moment es poden usar funcions les funcions per a la gestió de so.

Cursos alternatius:

2. Si no s'ha pogut inicialitzar el mòdul de so s'avisarà de l'error al programador.

Cas d'ús: set_volume

Descripció: Modifica el volum de so global de l'aplicació multimèdia.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol modificar el volum de reproducció dels samples digitals i/o el volum de reproducció MIDI.	2. Allegro modifica els volums globals corresponents només dins l'aplicació del programador i entre els marges possibles.

Cas d'ús: play_sample

Descripció: Reprodueix un sample de so digital. Un sample és un objecte que representa la informació de so a reproduir per Allegro. Per defecte es reproduïx un sample una única vegada des del principi fins al final, però es pot reproduir indefinidament en un bucle en qualsevol direcció.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol reproduir una peça de so digital amb un volum, una distribució de canals, una freqüència i un mode de reproducció determinats.	2. Allegro llança el sample de so digital amb les característiques desitjades i retorna al programador a quina veu de so es reproduïx.

Cursos alternatius:

2. Si no existeixen veus disponibles per a reproduir el sample es retorna error al programador.

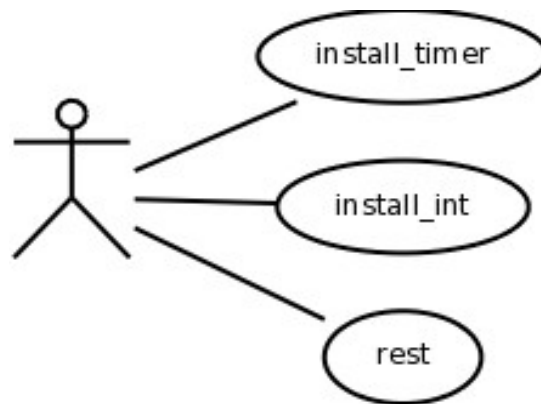
Cas d'ús: stop_sample

Descripció: Para la reproducció d'un sample determinat.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol deixar de reproduir una peça de so digital determinada.	2. Allegro deixa de reproduir el sample a totes les veus on hi sigui i allibera aquestes veus.

6.1.6 Funcions del temporitzador



Cas d'ús: install_timer

Descripció: Instal·la el gestor d'interrupcions del temporitzador. Amb aquest mòdul es poden instal·lar rutines d'interrupció que posteriorment s'executaran cadascuna amb diferents freqüències temporals.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol inicialitzar el gestor d'interrupcions d'Allegro.	2. Allegro instal·la el mòdul gestor d'interrupcions i a partir d'aquest moment es pot utilitzar la seva funcionalitat.

Cursos alternatius:

2. En cas d'error s'avisava al programador.

Cas d'ús: install_int

Descripció: Programa una rutina d'interrupció d'usuari per executar-se cada cert temps.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol executar una rutina amb una certa freqüència fixa.	2. Allegro executarà la rutina del programador amb la velocitat desitjada fins que no s'indiqui el contrari o fins a la fi de l'aplicació.

Cursos alternatius:

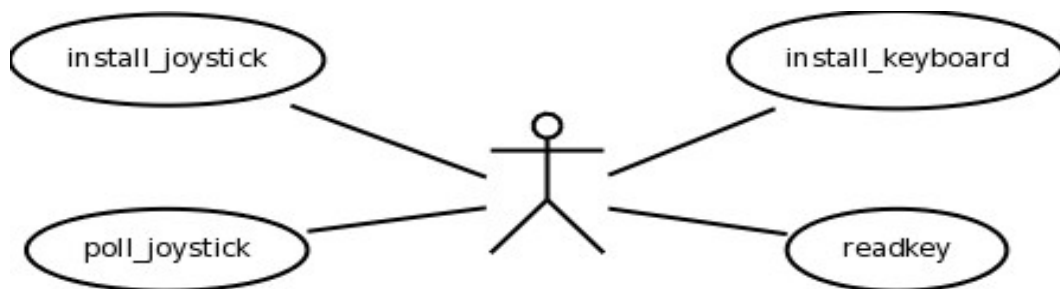
2. En retorna error en cas de superar el límit màxim de rutines d'interrupció instal·lades.

Cas d'ús: rest

Descripció: Espera un cert temps sense fer res. En plataformes multitasca aquesta funció es pot utilitzar per cedir (yield en anglès) el temps de CPU assignat a l'aplicació permetent així l'execució d'altres processos.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol introduir una pausa en l'execució de la seva aplicació de forma independent de la plataforma.	2. Allegro espera el temps especificat sense executar l'aplicació per continuar amb normalitat tot seguit.

6.1.7 Gestió dels perifèrics d'entrada**Cas d'ús: install_keyboard**

Descripció: Inicialitza el teclat d'Allegro permetent la comunicació amb l'usuari mitjançant aquest perifèric.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol instal·lar el gestor del teclat independent de la plataforma.	2. A partir d'aquest moment el programador pot accedir al teclat utilitzant les rutines i estructures de dades d'Allegro.

Cursos alternatius:

2. En cas d'error s'avisava al programador.

Cas d'ús: readkey

Descripció: Retorna el següent caràcter ASCII del buffer de teclat (existeix també la versió Unicode d'aquesta funció). A la majoria de les plataformes el teclat és un perifèric d'entrada que es llegeix per interrupció. En aquest cas és la rutina de servei de la interrupció qui s'encarrega d'omplir el buffer de teclat.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol llegir el caràcter de teclat introduït per l'usuari de l'aplicació multimèdia.	2. Allegro retorna el següent caràcter del buffer de teclat i el seu scancode associat independent de la plataforma (codi de baix nivell que identifica una tecla en funció de la seva posició al teclat). Si el buffer és buit Allegro s'espera fins que es premi una tecla.

Cas d'ús: install_joystick

Descripció: Inicialitza el gestor de joysticks d'Allegro. S'indica el tipus de joystick desitjat i específic de la plataforma o, per portabilitat, s'indica el tipus de joystick per defecte de la plataforma subjacent.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
1. El programador vol instal·lar un joystick d'un tipus determinat pel seu ús a la aplicació multimèdia.	2. Allegro inicialitza el joystick desitjat i calibra la seva posició central (per tant és necessari que el joystick estigui en posició neutra).

Cursos alternatius:

2. En cas d'error s'avisava al programador.

Cas d'ús: poll_joystick

Descripció: Llegeix les variables d'estat del joystick. A la majoria de les plataformes el joystick és un perifèric d'entrada que es llegeix per enquesta i per tant Allegro només suporta aquest mode de lectura.

Curs típic d'esdeveniments:

Acció dels actors	Resposta del sistema
-------------------	----------------------

1. El programador vol llegir la posició actual del joystick i l'estat dels seus botons.

2. Allegro actualitza les variables d'estat del joystick.

3. El programador ja pot llegir la informació actual del joystick.

Cursos alternatius:

2. Si no s'ha instal·lat cap joystick es retorna error al programador.

6.2 Diagrama de classes d'alt nivell

El diagrama de classes és un tipus de diagrama estàtic que descriu l'estructura d'un sistema mostrant les seves classes i les relacions entre elles.

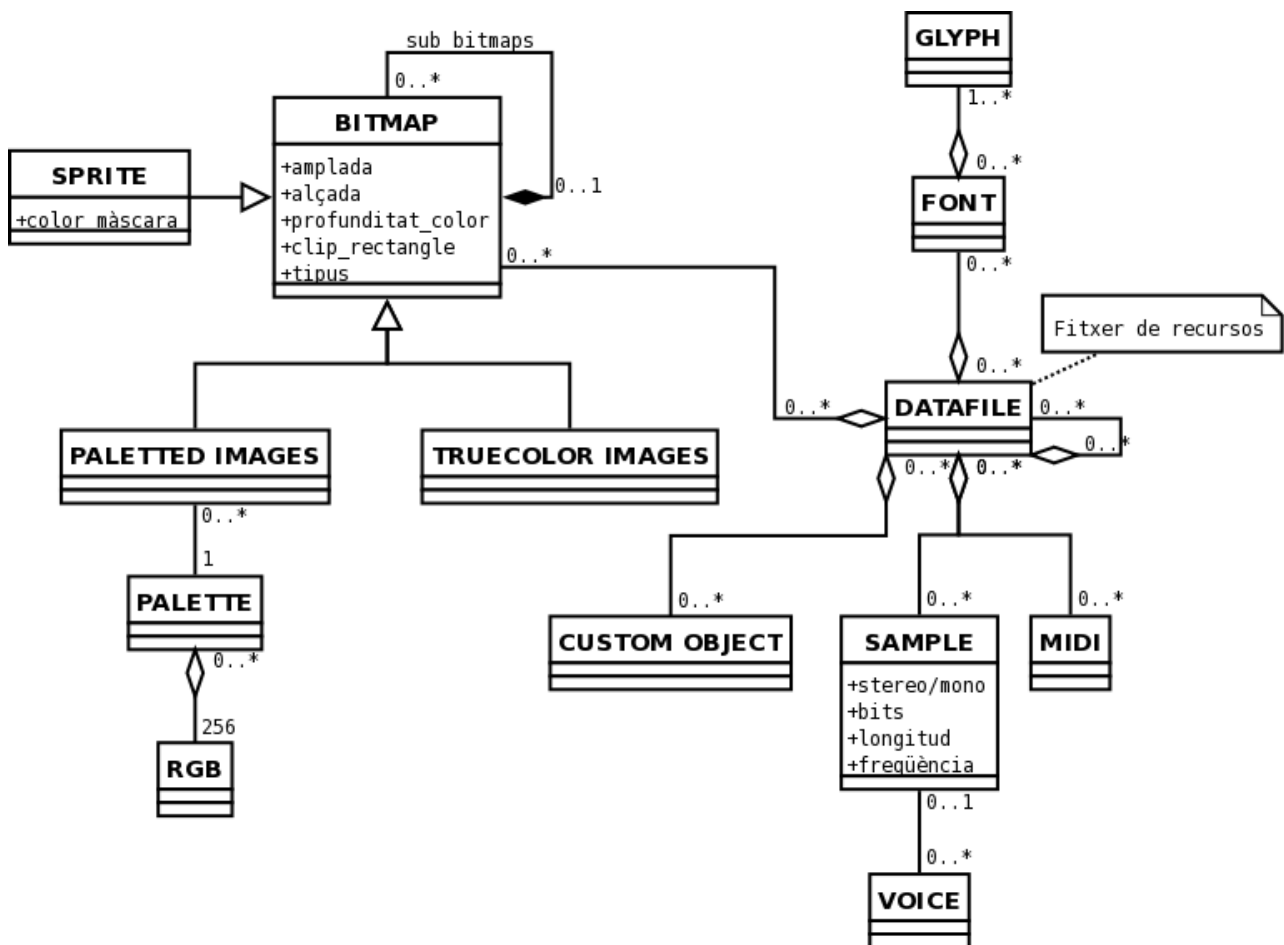


Figura 6.1 Principals entitats multimèdia d'Allegro

A l'anterior diagrama de classes es mostren les entitats més importants que formen part del domini de la llibreria multimèdia Allegro i que tot programador hauria de conèixer.

Com s'ha comentat anteriorment el BITMAP és l'entitat estrella ja que tota la visualització gràfica es basa en aquesta. Per altra banda un bitmap es pot dividir en sub bitmaps que comparteixen la mateixa àrea de dibuix del bitmap pare.

Finalment comentar que Allegro permet el programador emmagatzemar els diferents recursos multimèdia en fitxers anomenats DATAFILES. Aquests fitxers poden contenir *bitmaps*, fonts de caràcters, mostres de so digital, fitxers MIDI, objectes definits pel propi programador (custom objects) i, fins i tot, d'altres *datafiles*.

Capítol 7. Disseny

Una vegada finalitzada l'etapa d'especificació ja coneixem el què: l'abast d'Allegro en quant el seu domini i la seva funcionalitat. Ara ens falta conèixer el com: quina és la seva arquitectura, com funciona la comunicació amb les plataformes/sistemes reals i com es dissenyen les classes fonamentals.

7.1 Arquitectura d'Allegro

Un dels objectius principals d'Allegro és la portabilitat o independència de la plataforma. El mateix codi font, sense canviar ni tan sols un caràcter, hauria de compilar i córrer a totes les plataformes suportades.

Un altre objectiu principal és la facilitat d'ús: el temps necessari per a tenir operativa una aplicació multimèdia o videojoc ha de ser curt, i fent que la feina d'inicialització i configuració sigui mínima.

Finalment és molt important l'eficiència i velocitat d'execució d'Allegro, encara que és en el procés d'implementació i optimització, i no tant en l'elecció de l'arquitectura, on s'assoleix aquest objectiu.

Per aquestes i per altres raons que només l'autor coneix s'ha acabat determinant que Allegro tingui una arquitectura monolítica formada per mòduls entrelaçats. En la pràctica, això significa que Allegro es construeix i s'executa com un sol bloc on els diferents mòduls funcionals (sistema, gràfics, so, ...) es comuniquen entre ells i amb l'aplicació multimèdia seguint el mètode senzill i poc costós de crida a subrutina. Un dels inconvenients d'aquesta arquitectura és la seva baixa mantenibilitat ja que les modificacions a un component poden afectar a tot el conjunt. En contraposició existeix l'arquitectura client-servidor o multicapa on els mòduls són altament independents, fet que facilita la mantenibilitat i intercanviabilitat; i on la comunicació es realitza a través de missatges dins d'una infraestructura.

Dins d'aquest gran bloc monolític que és Allegro, i com ja s'ha parlat en els capítols anteriors, es poden distingir dues capes: la capa independent i els *drivers* o controladors de dispositiu que formen la capa dependent. Aquests s'encarreguen de que els diferents mòduls d'Allegro funcionin a les diferents plataformes reals. Cada driver es comunica amb el perifèric d'una plataforma (pantalla, teclat, ratolí, rellotge, ...) ja sigui directament (DOS) o via el sistema operatiu o kernel (Unix, Windows, PSP, ...), i tradueix les particularitats del maquinari a un llenguatge independent i comú.

Allegro defineix cada driver com una interfície o conjunt de mètodes abstractes que ha d'implementar tota plataforma per a ser suportada. En total hi ha 8 drivers, cadascun d'ells amb la seva interfície.

En el següent diagrama de classes UML es mostra l'exposat fins ara.

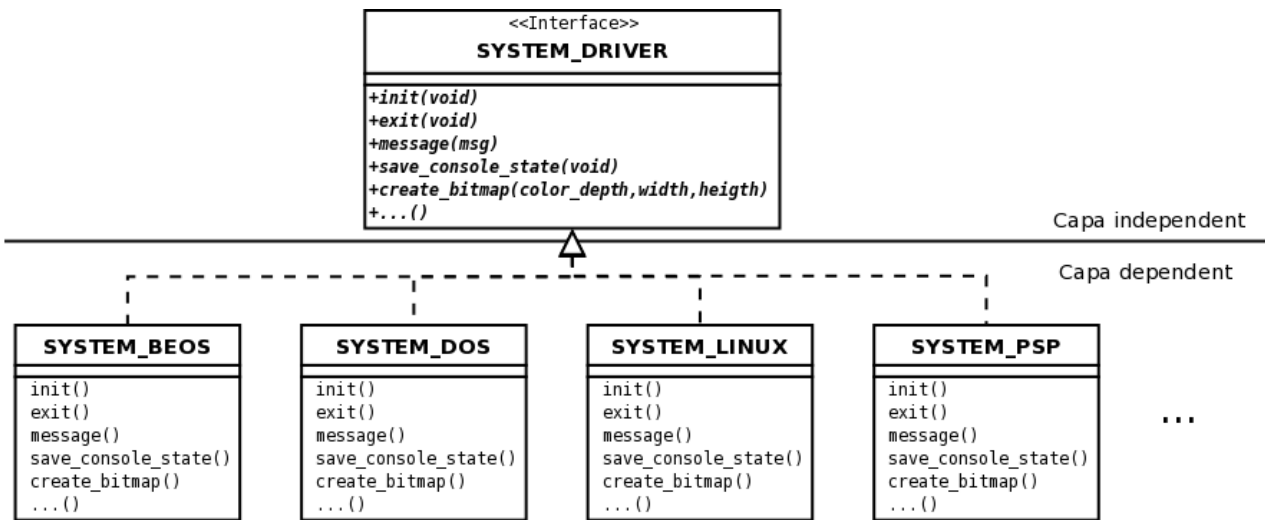


Figura 7.1 Arquitectura drivers d'Allegro

Aquesta figura mostra de forma simbòlica la frontera entre Allegro i les plataformes reals. En aquest cas la interfície **SYSTEM_DRIVER**, el driver de sistema, indica als implementadors quines funcions s'han d'implementar i amb quins paràmetres, perquè Allegro pugui comunicar-se amb la plataforma desitjada. És en el moment de construcció i/o compilació de la pròpia llibreria Allegro quan es selecciona una classe implementadora o una altra en funció de quina sigui la plataforma de desenvolupament. Per exemple, en el cas de la Sony PSP, la classe **SYSTEM_PSP** es converteix en la implementació del **SYSTEM_DRIVER** d'Allegro en el moment que es construeix i compila Allegro per a la PSP. En la pràctica això s'aconsegueix programant de forma adient els *scripts* o fitxers d'instruccions per a la construcció de la llibreria Allegro (veure el capítol Anàlisi de requisits, apartat Requisits no funcionals, subapartat Requisits de la llibreria Allegro).

Gràcies als drivers o controladors de dispositiu, per tant, Allegro opera de forma totalment transparent a qualsevol plataforma real i amb total portabilitat.

7.2 Mòdul gràfic

Aquesta arquitectura utilitzada en el disseny dels drivers també és utilitzada en el disseny dels bitmaps, l'entitat central del mòdul gràfic d'Allegro. En aquest cas cada bitmap s'associa a una interfície de mètodes gràfics que Allegro suporta, ja siguin pel dibuix de primitives gràfiques (line, putpixel, ...), transferències d'imatges (blit_to_memory, masked_blit, ...), etc. De manera anàloga als drivers aquesta interfície entraria dins la capa independent.

És a la capa dependent on s'implementen aquests mètodes gràfics. Aquesta implementació depèn d'una sèrie de factors com són la profunditat de color del bitmap (8, 16, ... bits per píxel), la seva representació interna (normalment un bitmap és representa com una matriu lineal però pot ser un conjunt de plans), el tipus de memòria on resideix (sistema o vídeo bàsicament) i la plataforma específica que els gestiona (Windows, Linux, etc.)

Aquesta separació clara entre un bitmap i els seus mètodes gràfics dona molta flexibilitat i permet la implementació de mètodes gràfics accelerats que són altament dependents de cada maquinari gràfic com són la transferència d'imatges, l'emplenat d'un polígon, el dibuix d'un triangle, etc. Així doncs, el mòdul gràfic d'Allegro no només opera de forma totalment transparent a qualsevol plataforma específica i maquinari gràfic sinó també independentment del tipus de bitmap o imatge a tractar.

Un altre aspecte que cal fer notar, i com ja s'ha deixat veure en anteriors capítols, és que la pantalla física que interacciona amb l'usuari final es representa com un bitmap de vídeo. Aquí es demostra que Allegro és molt ortogonal ja que tant el concepte d'imatge com el concepte de pantalla física es programen i es veuen com a conceptes pràcticament idèntics.

Com una imatge (mai millor dit) val més que mil paraules passem a veure el següent diagrama que resumeix tot l'explicat fins ara.

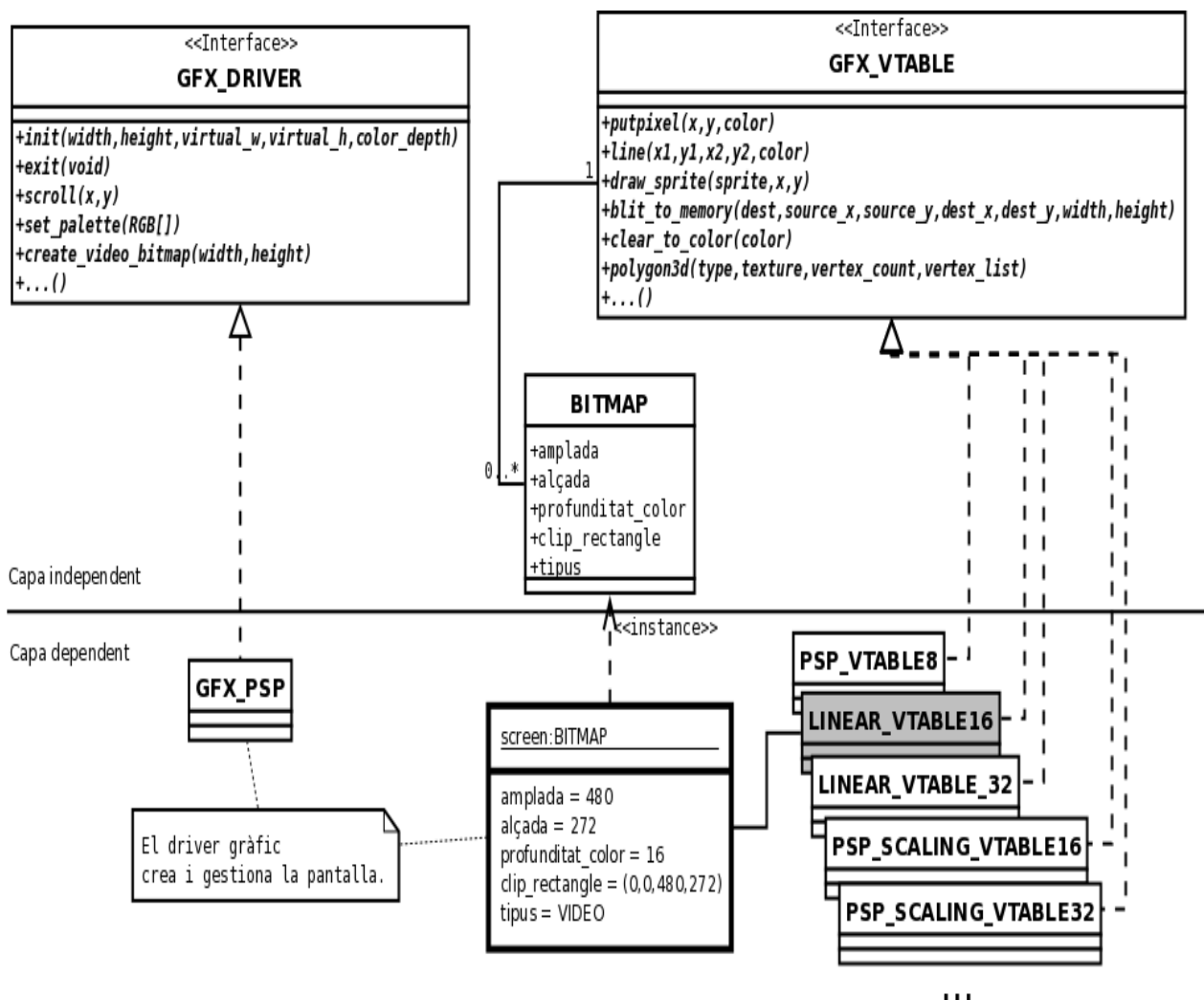


Figura 7.2 Arquitectura mòdul gràfic

Aquest diagrama mostra l'estat d'Allegro després d'haver inicialitzat el mode gràfic de la PSP amb una resolució de 480x272 i una profunditat de color de 16 bits per píxel. Com es pot veure la pantalla és un objecte BITMAP i té associada la taula de funcions per dibuixar en aquesta profunditat. Existeixen d'altres implementacions que s'han creat exclusivament per a la PSP, com són la PSP_VTABLE8 que emula el mode de 8 bits per píxel; i les taules PSP_SCALING_XX que permeten a la PSP suportar resolucions majors de 480x272. En el capítol següent d'implementació es veuran amb més detall.

7.3 Mòdul so digital

Per a representar la informació sonora s'utilitza la classe SAMPLE. Aquesta classe conté la informació sonora en format digital PCM (de l'anglès Pulse Code Modulation). Aquesta representació del so real o analògic consisteix en una sèrie de mostres obtingudes a una freqüència determinada. Si veiem la longitud d'ona com una funció $x \rightarrow y$ on la x és el temps i la y és l'amplitud d'ona, les mostres es correspondrien amb la variable y . El nombre de bits utilitzats per a representar aquestes mostres determinen la resolució sonora i típicament és igual a 8 o 16 bits. Amb 8 bits es poden representar $2^8 = 256$ valors i $2^{16} = 65536$ amb 16 bits. Segons quin sigui el rang d'aquests valors parlem de mostres amb signe ($2^{bits-1} \geq mostra \geq 2^{bits-1} - 1$) o sense signe ($0 \geq mostra \geq 2^{bits} - 1$).

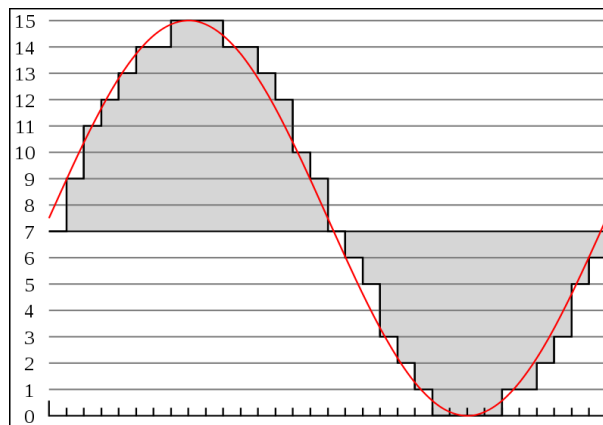


Figura 7.3 Representació PCM sense signe d'un senyal analògic amb 4 bits per mostra.

Qualsevol fitxer d'àudio ja sigui WAV, MP3, MOD, OGG, etc. es reproduïble per Allegro en quant existeixin els descodificadors corresponents ja que tots aquests formats es poden reduir a l'essència de la representació plana PCM.

Per a la reproducció de so Allegro té una capacitat de fins 256 veus virtuals. Aquestes veus actuen com una capa d'abstracció del maquinari de so real permetent al programador una gestió del so independent de la plataforma. A cada objecte SAMPLE se li assigna una d'aquestes veus virtuals en el moment de la reproducció.

Tanmateix, el maquinari de so té un límit en el nombre de samples que pot reproduir simultàniament que són les veus reals. Normalment aquest nombre de veus és molt més petit que el nombre de veus virtuals que suporta Allegro. Es presenta un problema, doncs, si el nombre de samples simultanis excedeix aquest límit. Allegro ho resol mitjançant una política de prioritats on, en cas necessari, un sample que estigui sonant per una veu virtual determinada es silencia per donar pas a un altre sample més prioritari.

La PSP té 8 veus de so, un nombre que en condicions normals sembla més que suficient. Però per una màquina amb pocs recursos com la PSP (principalment manca de CPU i memòria), i degut a la forma de programar l'àudio en aquesta consola, és molt costós suportar més d'una o dues veus simultànies. Per superar aquest important obstacle i per a una programació eficient del driver de so digital, es va prendre la decisió d'utilitzar la funcionalitat del mesclador o mixer d'Allegro, un mòdul de la llibreria que permet la reproducció per programari de fins a 64 veus simultànies en només una única veu real de so. Malgrat aquesta solució pot suposar un excés d'utilització de CPU segueix sent molt menys costosa que la primera com es va comprovar en la pràctica.

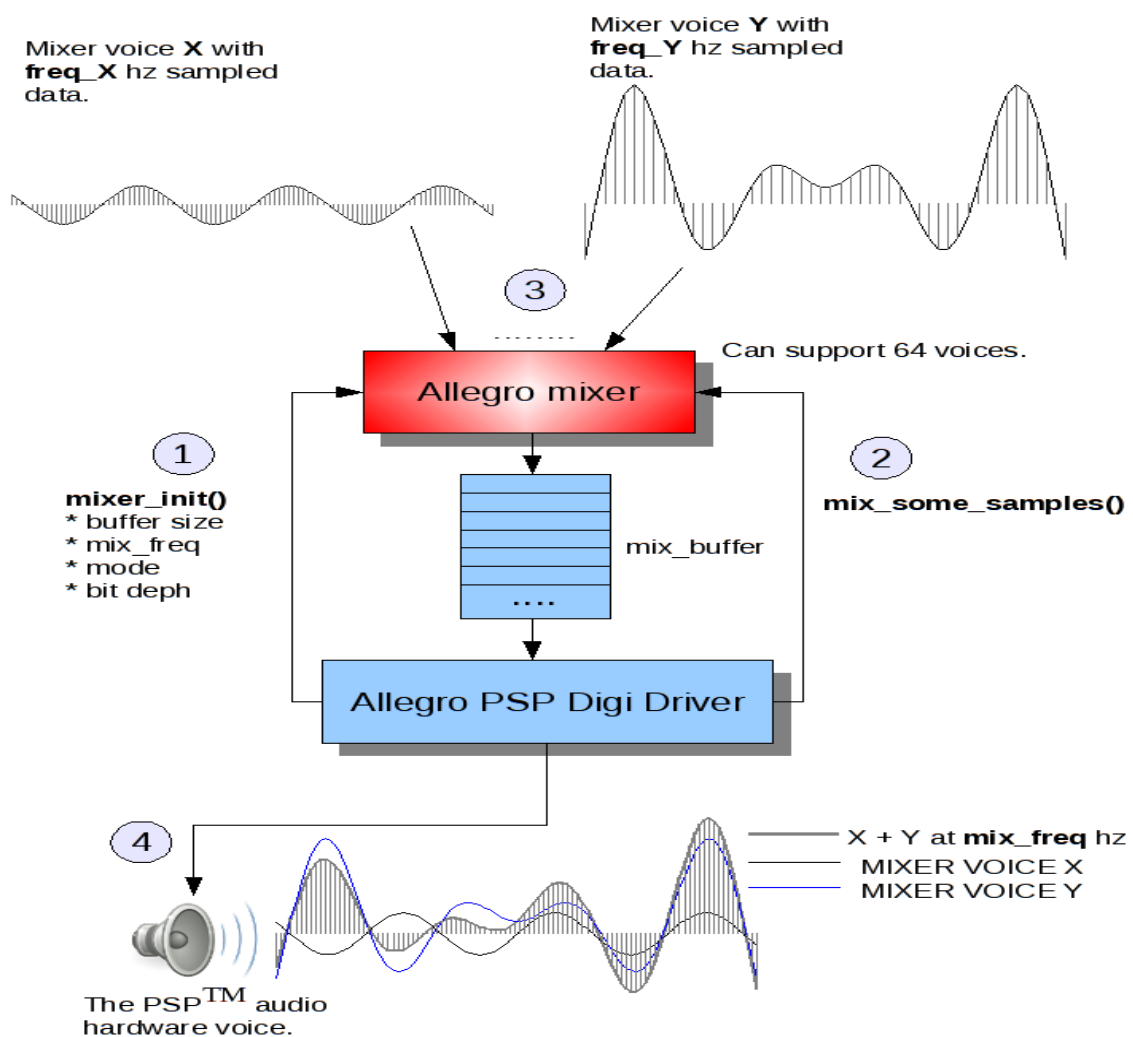


Figura 7.4 Arquitectura mòdul de so digital a la PSP

A l'anterior diagrama es mostra el funcionament del mòdul de so a la PSP com una seqüència d'interaccions entre el mesclador d'Allegro i el driver de so digital de la PSP. Cada sample a reproduir ocupa una veu del mesclador o veu real de les 64 disponibles i potencialment simultànies. Com la capa independent d'Allegro té la visió del maquinari que el driver li mostra, aquest disseny en concret li fa veure que el maquinari de so de la PSP té capacitat per a 64 veus, fet que és fals ja que, com es pot comprovar, només s'activa una veu real a la PSP.

Tot comença quan el driver informa al mixer de com es vol la mescla indicant uns valors com la freqüència de mostreig, el mode (estèreo o mono) i els bits de resolució de les mostres. A partir d'aquest moment, i periòdicament, es van demanant mescles que el mixer deixa a un buffer intermedi. Aquestes mescles contenen el so audible final que servirà per alimentar el processador digital del senyal o DSP (en anglès digital signal processor) de la PSP. Com es pot comprovar el procés de mescla de samples és teòricament molt simple: es sumen o resten les mostres coincidents en el temps de cadascuna de les ones.

7.4 Mòdul del temporitzador

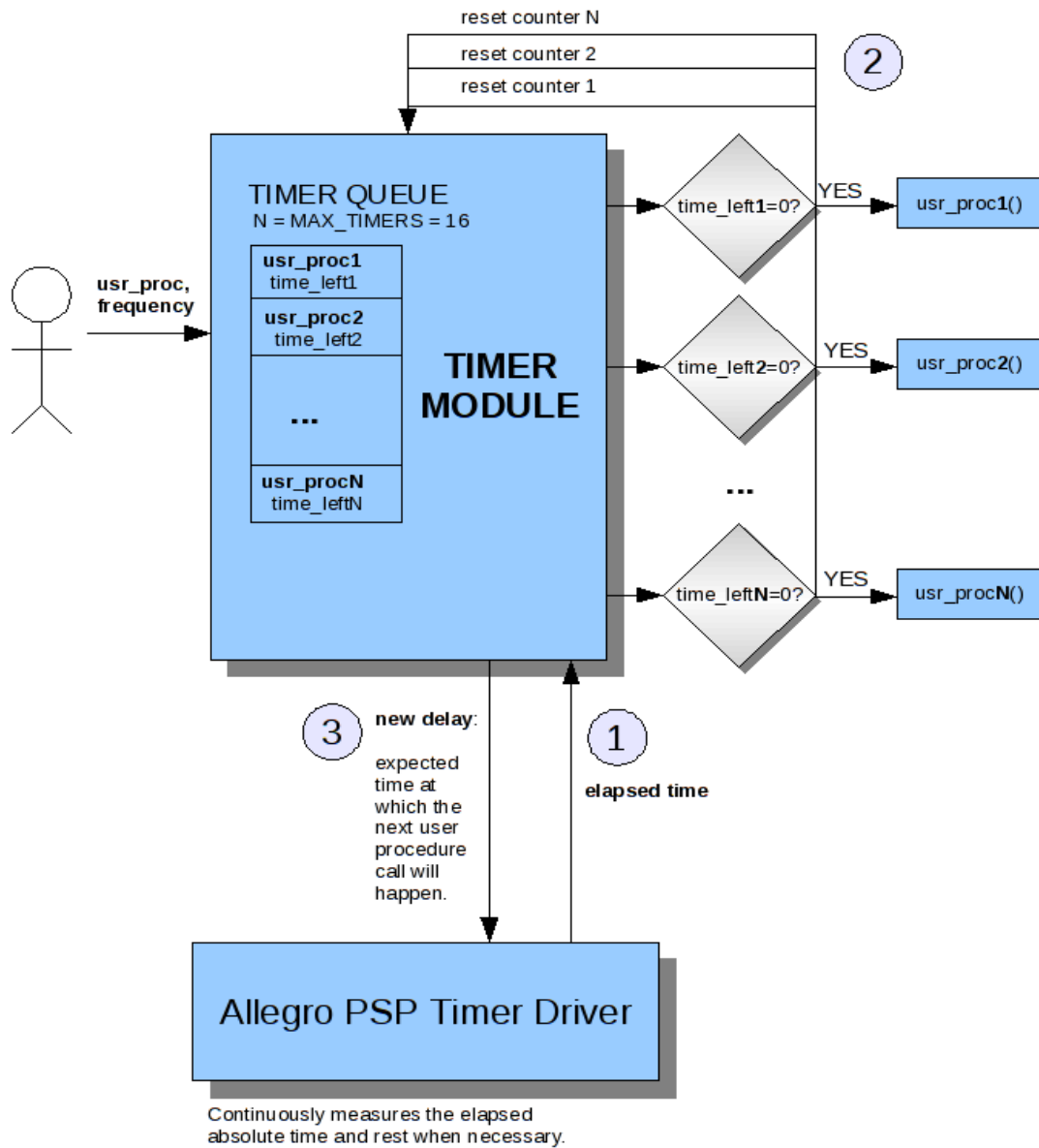
Allegro permet instal·lar funcions que s'executen periòdicament gràcies als temporitzadors. Segons el Diccionari de la llengua catalana de l'Institut d'Estudis Catalans (DIEC) un temporitzador és un dispositiu que comanda una determinada operació en un moment o durant un temps predeterminat. Per aconseguir això és indispensable mesurar en tot moment el temps real. Aquest procés és clarament dependent de la plataforma i per tant és funció principal del driver del temporitzador.

En el llenguatge independent i abstracte d'Allegro la unitat mínima de temps és el tick, dels quals considera que n'hi ha exactament 1.193.181 en un segon. Aquest és el nombre de ticks del temporitzador Intel 8253 que porten els PCs i és una herència de les primeres versions d'Allegro on aquesta llibreria només funcionava a la plataforma PC-DOS. Aquest valor també ens indica que la precisió del rellotge d'Allegro és aproximadament 1 microsegon.

Si observem en detall l'arquitectura distingim clarament una cua de temporitzadors. Cada temporitzador comanda una funció determinada en un moment determinat prèviament decidit pel programador o usuari. A mesura que va passant el temps i cada vegada que el comptador d'un temporitzador arriba a zero es crida a la seva funció associada i es fa una reprogramació del comptador per tornar a repetir el procés.

Com ja hem dit la funció principal del driver consisteix a mesurar el temps i traduir-lo al temps independent de la plataforma d'Allegro. Una qüestió que sorgeix és: amb quina freqüència s'ha d'avisar al mòdul del temporitzador del temps real transcorregut? La resposta ens la dona la pròpia arquitectura del mòdul del temporitzador i depèn del temps mínim que ha de passar per cridar a una funció instal·lada qualsevol.

Per acabar, el següent esquema mostra clarament tot l'exposat fins ara.



NOTE

All the time values are represented in PC hardware *clock ticks*, of which there are 1.193.181 a second.

Figura 7.5 Arquitectura mòdul del temporitzador

Capítol 8. Implementació

8.1 Tecnologies utilitzades a Allegro PSP

Iniciem aquest capítol amb una breu introducció a cadascuna de les tecnologies que s'han hagut d'investigar i aplicar a aquest projecte, sense les quals no hauria estat possible la seva realització.

8.1.1 C

El llenguatge de programació C va ser creat per Dennis Ritchie i Ken Thompson als laboratoris Bell d'AT&T, a principis de la dècada dels 70. El llenguatge C es va crear per la necessitat de tenir-ne un que fos més flexible que l'assemblador a l'hora de programar, però que mantingués la característica de ser un llenguatge proper a la màquina. És doncs un llenguatge ideal per a la creació de programari de sistema, encara que també s'utilitza per a crear aplicacions de més alt nivell. Existeix un compilador d'aquest llenguatge per a un ampli ventall de plataformes, des de microcontroladors a supercomputadors per la qual cosa C és altament portable.

Amb aquestes característiques no és d'estranyar que C hagi estat el llenguatge de creació d'Allegro, encara que també s'utilitzin fragments de codi en assemblador i386 i PowerPC.

Per a la implementació dels drivers d'Allegro per a la PSP s'ha utilitzat exclusivament el llenguatge C.

8.1.2 OpenGL

OpenGL és una especificació estàndard que defineix una API multillenguatge i multiplataforma per a escriure aplicacions que produeixen gràfics 3D. Desenvolupada originalment per Silicon Graphics Incorporated (SGI). OpenGL significa Open Graphics Library, que traduït és "biblioteca de gràfics oberta".

Com ja es va introduir al capítol d'entorn del PFC, el PSPSDK conté la llibreria gràfica libGU que proporciona l'accés a les funcions accelerades 2D i 3D del Graphics Engine (GE) de la PSP. Aquesta llibreria és d'una aparença molt similar a OpenGL.

Per a la realització de part del driver gràfic de la PSP s'ha utilitzat el mapatge de textures gràfiques utilitzant la filosofia OpenGL. El mapatge de textures és un mètode per a crear una visualització més realista d'objectes 3D afegint detall, textura superficial o color. Als següents apartats es veurà un cas real on s'ha aplicat aquesta tècnica.

8.1.3 Threads

Un fil d'execució (*thread* en anglès) és una característica dels sistemes operatius que permet a un procés executar diferents tasques al mateix temps. Els diferents fils d'execució comparteixen una sèrie de recursos com l'espai de memòria, els fitxers oberts, etc. Per tant, i en contraposició al procés un fil d'execució és molt més lleuger. Aquesta característica simplifica el disseny d'un programa que executa diferents funcions concurrentment.

En una màquina amb un únic processador, l'execució concurrent d'aquests threads es produeix assignant un temps d'execució a cadascun. Aquest intercanvi de threads es produeix en el que s'anomena canvi de context i succeeix suficientment ràpid de tal manera que l'usuari té la percepció de que les diferents tasques associades a cada thread s'executen simultàniament.

En una aplicació multimèdia d'Allegro es distingeixen clarament 3 tasques que s'han d'executar concurrentment:

1. El programa principal que s'encarrega entre d'altres de la gestió gràfica i la gestió dels esdeveniments d'entrada (teclat, joystick, ...)
2. La gestió l'àudio i comunicació amb el DSP.
3. La gestió del temps.

Per implementar aquest escenari que ens presenta Allegro l'alternativa clàssica a aquest model d'execució anomenat multithreading és el model d'interrupcions. En aquest model els diferents perifèrics i dispositius envien una senyal al processador indicant que s'ha d'interrompre el curs d'execució actual i passar a executar codi específic per respondre a aquesta senyal (entrada de teclat, tic de rellotge, notificació de buffer buit del DSP, etc.)

Al kernel de la PSP, de la mateixa manera que a molts dels sistemes operatius actuals, no és possible programar interrupcions (almenys no fàcilment) i l'alternativa és el multithreading. Aquest és, per tant, el model escollit per implementar els drivers d'Allegro a la PSP.

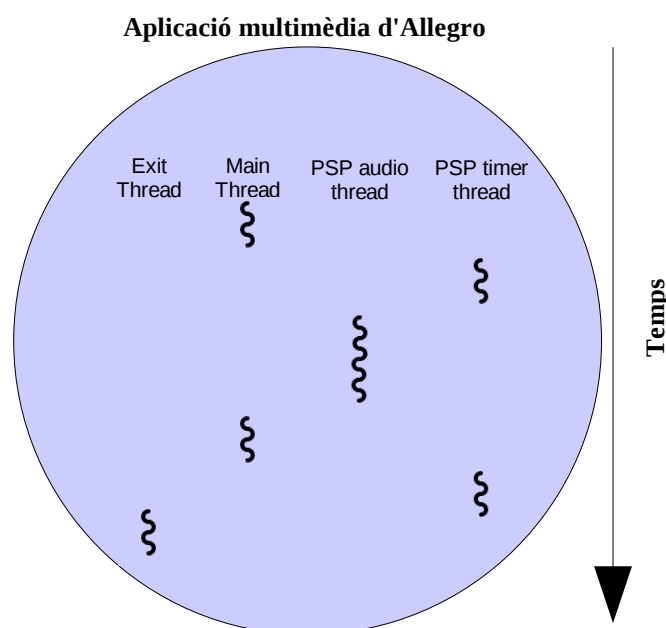


Figura 8.1 Allegro PSP multithreading

A l'anterior figura es poden identificar els threads en execució dins d'un procés Allegro a la PSP. L'*Exit thread* és un fil d'execució molt específic i obligatori a tot el *homebrew* de la PSP: s'encarrega de capturar la pulsació del botó HOME de la consola, avortar l'aplicació i sortir al XMB o menú principal de la PSP.

D'altra banda, els sistemes operatius implementen el multithreading de dues maneres:

- 1) Multifil apropiatiu o *preemptive multithreading*: permet al sistema operatiu determinar quan s'ha de fer un canvi de context.
- 2) Multifil cooperatiu o *cooperative multithreading*: depèn del propi fil abandonar l'execució quan arriba a un punt de detenció com l'espera d'una operació d'E/S.

A la PSP els threads són cooperatius això vol dir que ens hem d'assegurar que qualsevol d'ells alliberi el processador amb una certa freqüència.

- L'*Exit thread* durant tota l'execució de l'aplicació està “adormit” o en mode *sleep* a l'espera de que el kernel el desperti i per tant no consumeix CPU.
- El *Main thread* periòdicament llegeix per enquesta els esdeveniments d'entrada abandonant la CPU.
- El *PSP audio thread* espera que el DSP li demani dades d'àudio abandonant la CPU.
- El *PSP timer thread* deixa d'executar-se petits intervals de temps (veure arquitectura del mòdul del temporitzador al capítol de Disseny).

Quan qualsevol thread abandona la CPU i el kernel ha de fer el canvi de context i escollir un altre thread per executar, es segueix una política de prioritats. Cada thread té assignat un nombre que representa la seva prioritat, quant més baix és el nombre més alta és la seva prioritat. Dels 3 threads actius: *main*, *audio* i *timer* he decidit assignar les prioritats seguint el mètode d'experimentació en l'execució de les proves reals: primer el fil del temporitzador per garantir que la mesura del temps sigui precisa, segon el fil de l'àudio per garantir una reproducció sonora sense interrupcions i finalment el fil principal.

Per acabar amb aquest apartat, es mostra com es crea el *PSP timer thread*:

```
static int psp_timer_thread() {
    /* Codi del fil del temporitzador */
}

static SceUID timer_thread_UID;
/* Crea el thread */
timer_thread_UID = sceKernelCreateThread("psp_timer_thread", // Nom
                                        (void *)&psp_timer_thread, // Punt d'entrada
                                        0x18, // Prioritat
                                        0x10000, // Mida de la pila
                                        0, // Atributs
                                        NULL); // Opcions
/* Inicia l'execució del thread (sense arguments)*/
sceKernelStartThread(timer_thread_UID, 0, NULL);
```

8.1.4 GNU make

GNU make és una utilitat molt important i utilitzada en el desenvolupament de programari sota Linux. Defineix un llenguatge per automatitzar el procés de construcció d'executables i llibreries a partir de fitxers font, el procés d'instal·lació, la neteja de fitxers temporals, i en general qualsevol procés que es pugui automatitzar. Tot gira a partir del fitxer *makefile* on s'especifica una sèrie de regles. En aquestes regles es detallen els objectius a aconseguir (un executable, una llibreria, ...), la seva llista de dependències (fitxers font, altres objectius previs, ...) i la llista d'accions a executar per assolir aquests objectius.

El llenguatge de GNU make és pot veure com un híbrid entre declaratiu i imperatiu. L'especificació de l'objectiu i les seves dependències segueix el model declaratiu en quant es detalla el què es vol aconseguir. Per altra banda, la llista d'accions a executar segueix clarament el model imperatiu ja que es detalla el com.

Aquesta és la sintaxi d'una regla del *makefile*:

```
objectiu: dependència1, dependència2, ...
        llista d'accions
```

La llibreria Allegro es construeix amb l'eina GNU make i cada plataforma suportada ha d'escriure el seu *makefile* corresponent. Allegro defineix regles per a la construcció de la llibreria en tres versions (debug, profile i optimitzada), la generació de les demos i exemples oficials, la generació d'utilitats extra, la generació de documentació en una gran quantitat de formats, la instal·lació/desinstal·lació de la llibreria i la neteja dels directoris de treball.

A continuació es mostra una part del contingut del fitxer *makefile.psp* on es veuen les regles, i algunes de les variables que hi intervenen, per a instal·lar els arxius de la llibreria Allegro PSP:

```
# ----- define some variables that the primary makefile will use -----
PSP_PREFIX = $(shell psp-config --psp-prefix)
RANLIB = psp-ranlib

# ----- link as a static library -----
LIB_NAME = lib/psp/lib$(VERSION).a
LIB_MAIN_NAME = lib/psp/lib$(VERSION)-main.a

# ----- rules for installing and removing the library files -----
DESTDIR =
INSTALLDIR = $(DESTDIR)$(PSP_PREFIX)
LIBDIR = lib

$(INSTALLDIR)/lib/lib$(VERSION).a: $(LIB_NAME)
    cp $(LIB_NAME) $(INSTALLDIR)/lib
    $(RANLIB) $(INSTALLDIR)/lib/lib$(VERSION).a
$(INSTALLDIR)/lib/lib$(VERSION)-main.a: $(LIB_MAIN_NAME)
    cp $(LIB_MAIN_NAME) $(INSTALLDIR)/lib
    $(RANLIB) $(INSTALLDIR)/lib/lib$(VERSION)-main.a
```

La primera regla té com objectiu l'arxiu principal d'Allegro al nostre sistema i s'indica amb el seu nom absolut, típicament */usr/local/pspdev/psp/lib/liballeg.a* per a la versió optimitzada o final de la llibreria. Per aconseguir aquest objectiu s'ha d'executar l'acció principal que consisteix a copiar aquest mateix arxiu des del directori de treball corresponent al directori d'instal·lació final.

8.2 La cache de la PSP

La CPU de la PSP, l'ALLEGREX, incorpora una memòria cache per accelerar l'accés a la memòria principal o de vídeo. En la programació habitual d'altres plataformes normalment aquesta cache es pot ignorar però en programació a baix nivell a la PSP s'ha de tenir en compte. Ara veurem per què.

La cache del processador és una petita peça de memòria ràpida on es guarden còpies de blocs de la memòria que s'accedeixen amb més freqüència en un moment determinat de l'execució d'un programa. D'aquesta manera s'intenten reduir els lents accessos a memòria i per tant accelerar l'execució dels programes.

Quan la CPU llegeix un bloc de memòria, primer comprova si hi ha una còpia d'aquest bloc a la cache. Si hi és, diem que s'ha produït un *cache hit*, la CPU pot accedir a les dades molt ràpidament. En cas contrari, diem que s'ha produït un *cache miss*, ha d'anar a memòria per llegir les dades, però a més es copia aquest bloc a la cache per a futurs accessos que probablement es produiran.

Les escriptures actuen de manera semblant. En cas de *cache hit* es modifiquen les dades al bloc de la cache. Però a diferència del procés de lectura, el bloc corresponent a memòria queda lògicament desactualitzat. Aquesta diferència entre les dades d'un bloc de memòria i la seva corresponent còpia a cache dura un temps no determinat. El temps que trigui la CPU en substituir aquest bloc de cache per un altre: es retorna a la memòria el bloc de la cache substituït i llavors la memòria passa a estar actualitzada.

Tot això passa de forma totalment transparent des del punt de vista del programador. Sense comptar la millora de velocitat que es produeix en l'execució del nostre programa, no hi ha manera de saber si aquesta gestió de la cache s'està produint.

Com es pot entreveure a partir d'aquesta petita exposició el problema es produeix a les escriptures i amb els blocs de la cache no actualitzats a memòria. Si la CPU fos l'únic usuari de la memòria no hi hauria cap problema, però la PSP té altres unitats funcionals (per exemple el *Graphics Engine*) que també utilitzen la memòria i que es comuniquen amb la CPU precisament via memòria. En aquest context, i perquè aquesta comunicació funcioni correctament, és necessari que cada usuari de memòria tingui una visió consistent i coherent d'aquesta.

Malauradament, i a diferència d'altres plataformes, no existeix cap maquinari auxiliar que s'encarregui de mantenir aquesta coherència de memòria a la PSP. Si el nostre programa escriu a memòria comandes perquè executi el GE o si el nostre programa escriu píxels a la memòria de vídeo, no hi ha cap garantia de que realment s'hagin escrit a memòria en el moment que el GE intenta accedir-hi. Això té conseqüències desagradables pels programadors que veuen gràfics corruptes, vèrtex desapareguts, textures amb píxels invisibles, ... i tot de manera no determinista.

L'ALLEGREX ofereix una solució a aquest problema que consisteix en accedir a la memòria especificant que no es gestioni la cache. Això s'aconsegueix fent un OR bit a bit d'un punter a memòria amb 0x40000000, ja sigui un punter a memòria principal o un punter a memòria de vídeo.

Però aquesta solució no és completament satisfactòria ja que es disminueix el rendiment global de les aplicacions. El PSPSDK ens ofereix una alternativa: un conjunt de funcions per gestionar explícitament la cache a les nostres aplicacions. Aquestes funcions permeten al programador mantenir la coherència entre memòria i cache de manera que tots els dispositius (CPU i GE principalment) tinguin exactament la mateixa visió.

Aquest és un extracte del driver gràfic que il·lustra la gestió explícita de la cache:

```

/* La paleta de colors o Color LookUp Table de la PSP */
static unsigned int __attribute__((aligned(16))) clut[256];

/* Implementa el mètode set_palette() de la interfície GFX_DRIVER */
static void psp_set_palette(AL_CONST RGB *p, int from, int to, int retracesync)
{
    ...

    /* Actualitza l'estructura de dades de la paleta */
    for (c=from; c<=to; c++) {
        clut[c] = makecol32(_rgb_scale_6[p[c].r],
                           _rgb_scale_6[p[c].g],
                           _rgb_scale_6[p[c].b]);
    }

    /* Abans de cridar al GE perquè actualitzi la paleta ens assegurem de que les
       dades tot just escrites estiguin a memòria principal fent un write-back de la
       cache del processador */
    sceKernelDcacheWritebackAll();

    ...

    /* Cridem a la interfície del maquinari gràfic perquè actualitzi la paleta */
    ...
    sceGuClutMode(GU_PSM_8888,0,0xff,0);
    sceGuClutLoad((256/8),clut);
    ...
}

```

Per altra banda i per resoldre el problema de la cache quan la CPU escriu informació gràfica a la memòria de vídeo visible, és a dir l'*screen BITMAP* d'Allegro (veure l'arquitectura del mòdul gràfic) s'ha decidit fer ús de la característica d'adreçament sense cache que ofereix l'ALLEGREX. Implementar la gestió de la cache per a cada operació gràfica, de manera semblant a l'exemple anterior, hauria estat l'alternativa.

Malgrat l'adreçament sense cache és un mètode poc eficient d'accés a memòria de vídeo per part del processador, en la majoria dels casos no ens trobarem en aquesta situació ja que en la metodologia de programació de les aplicacions gràfiques no és comú que la CPU escrigui directament a pantalla: normalment es fa ús de la tècnica del *double/triple buffering o page flipping* on el processador gràfic s'encarrega de dibuixar a la pantalla fent una transferència d'imatges o *blitting*. Per tant només ens hem d'assegurar que la memòria estigui actualitzada abans d'indicar al GE que faci la transferència d'una imatge a la pantalla.

Com es pot veure aquesta última solució és molt millor que reescriure totes les operacions gràfiques d'Allegro només perquè augmenti el rendiment de la seva execució directament a pantalla. A més, adoptar aquesta solució és l'única manera de suportar la característica de l'especificació d'Allegro de permetre que el programador tingui accés directe a la memòria que representa la pantalla.

Suposem que volem executar a la PSP l'aplicació gràfica següent:

```

#include <allegro.h>

int main(void)
{
    ...

    /* Inicialitza el mode gràfic a 480x272x16 */
    set_color_depth(16);
    set_gfx_mode(GFX_AUTODETECT, 480, 272, 0, 0);

    ...

    /* Accés directe a pantalla: esborra la pantalla a blanc */
    clear_to_color(screen, makecol(255, 255, 255));

    ...

    /* Accés directe a pantalla: dibuixa una cadena de text de color negre
       a la pantalla */
    textout_centre_ex(screen, font, "Hello, PFC!", SCREEN_W/2, SCREEN_H/2,
                      makecol(0,0,0), -1);

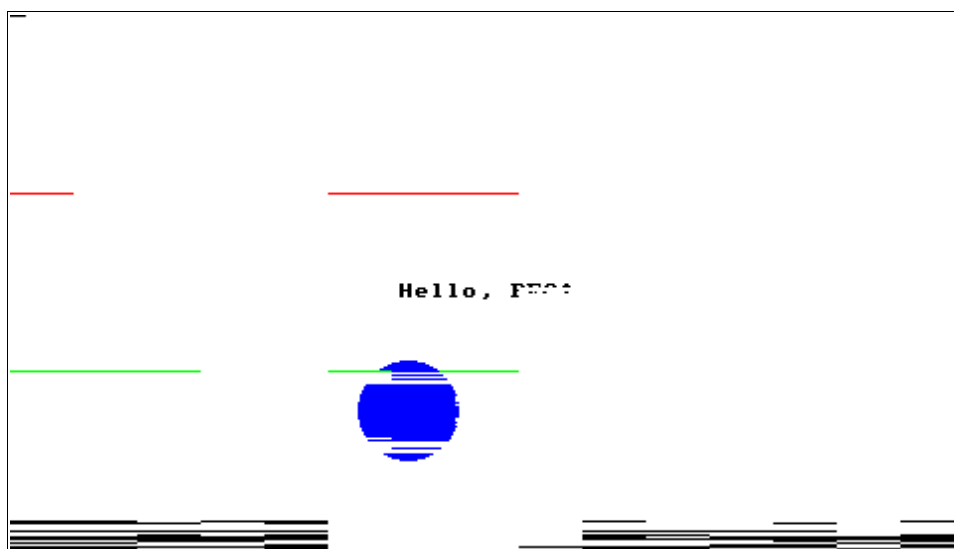
    /* Accés directe a la memòria que representa la pantalla usant l'estructura
       interna del BITMAP: dibuixa dues línies horitzontals que divideixen la pantalla
       en tres */
    for (i=0; i<SCREEN_W; i++) {
        ((short *)screen->line[90])[i] = makecol(255,0,0);
        ((short *)screen->line[180])[i] = makecol(0,255,0);
    }

    /* Accés directe a pantalla: dibuixa un cercle blau */
    circlefill(screen, 200, 200, 25, makecol(0,0,255));

    ...
}
END_OF_MAIN()

```

Aquest seria el resultat havent implementat el driver gràfic sense tenir en compte la cache de la PSP:



Per solucionar aquest problema i com ja hem dit més a dalt s'ha fet el següent en el driver gràfic:

```
/* Implementa el mètode init() de la interfície GFX_DRIVER */
static BITMAP *psp_display_init(int w, int h, int v_w, int v_h, int color_depth)
{
    BITMAP *psp_screen;
    uintptr_t vram_start, screen_start;

    ...

    /* Obté l'adreça base de la memòria de vídeo de la PSP */
    vram_start = (uintptr_t)sceGeEdramGetAddr();

    ...

    screen_start = vram_start + ... ;

    /* La lectura i escriptura de la VRAM corresponent a la pantalla es farà sense
       utilitzar la cache de l'ALLEGREX */
    screen_start |= 0x40000000;

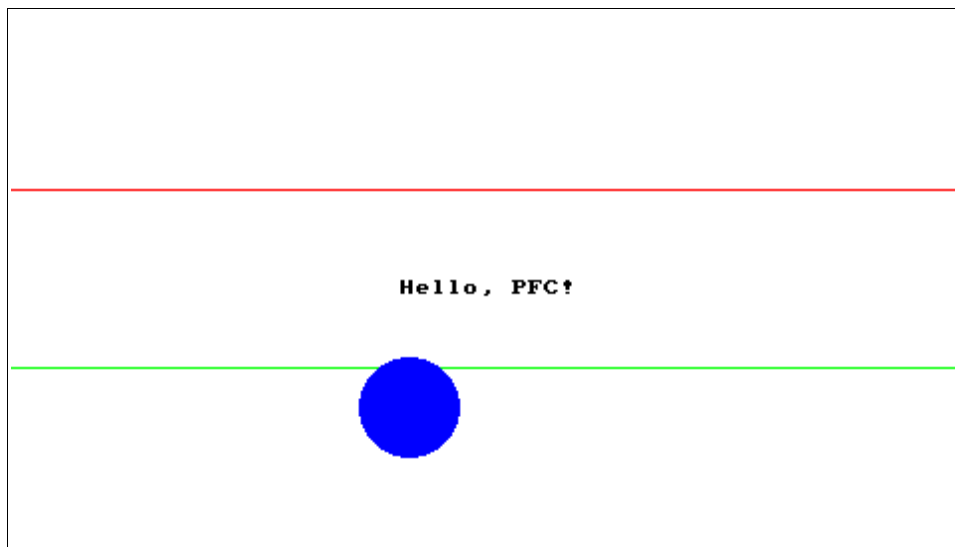
    ...

    /* Crea l'screen BITMAP d'Allegro */
    psp_screen = _make_bitmap(v_w, v_h, screen_start, &gfx_psp, color_depth,
    bytes_per_line);

    ...
}

/* A partir d'aquí l'screen BITMAP d'Allegro serà el psp_screen i les operacions
directes a pantalla es comportaran de la forma esperada */
```

Gràcies a que ara tenim en compte la cache de la PSP la nostra aplicació anterior es comportarà de la manera esperada:



8.3 Emulació 8 bits de color

Allegro suporta profunditats de color de 8, 15, 16, 24 i 32 bits per píxel. A més d'aquesta diferència entre el nombre de bits per representar un píxel a pantalla, la profunditat de 8 bits per píxel que suporta Allegro té la particularitat de que aquests 8 bits no representen una codificació d'un color en format RGB com sí succeeix a les altres profunditats (anomenades *truecolor* per Allegro). Aquests 8 bits representen un índex a una paleta de colors de 256 valors. En el passat quan la memòria era molt més reduïda aquest mètode de representació del color era quasi obligatori, com succeïa a les primeres targetes gràfiques pel PC per exemple. Actualment hi ha memòria en abundància com per a representar un píxel amb fins a 24 o 32 bits. Amb aquesta mida per píxel s'aconsegueix representar més de la totalitat dels colors que pot percebre l'ull humà aconseguint per tant imatges fotogràfiques.

La PSP suporta 15,16 i 32 bits per píxel i semblaria suficient si no fos perquè els 8 bits de color és una profunditat important a Allegro. En primer lloc és la profunditat per defecte quan s'inicia el mode gràfic, això significa que hi ha una gran quantitat d'aplicacions ja existents que no funcionarien directament a la PSP si no es suportés aquesta profunditat, fent a Allegro PSP coixa de bon començament. Per altra banda té els avantatges de que es consumeix poca memòria (ideal per a una màquina amb pocs recursos com la PSP) i es pot utilitzar la funcionalitat d'Allegro per crear efectes gràfics modificant la paleta de colors.

Malgrat la PSP no suporta el format de píxel de 8 bits sí que pot treballar amb textures d'aquest format. Les textures són imatges que s'apliquen a la superfície de models 3D. Gràcies a aquesta característica de la consola es pot emular amb relativa facilitat i amb eficiència els modes gràfics amb 8 bits per píxel d'Allegro com es veurà a continuació.

Les textures s'utilitzen en el processament 3D per la qual cosa és necessari treballar en aquest mode. La idea consisteix en suposar que la pantalla o *screen BITMAP* d'Allegro és una textura de 8 bits. Totes les operacions gràfiques dirigides a la pantalla modifiquen aquesta textura i s'executen de la manera habitual però a una zona de la VRAM no visible. Per tant, i per fer l'*screen BITMAP* "visible", al final de cada operació gràfica s'ha de crear un model 3D que representa el rectangle de la pantalla física i mapar-li aquesta textura.

Tot això passa de manera totalment transparent al programador i/o usuari de l'aplicació que creu que està treballant amb una profunditat de 8 bits de color de forma nativa. L'inconvenient d'aquest sistema és lògicament la velocitat. Per a cada operació gràfica, des de un *putpixel()* fins a un *blit()* passant per un *line()* o un *draw_sprite()* s'ha de fer una operació 3D de mapatge de textura. Afortunadament el processament 3D és un dels punts forts de la consola i, per altra banda, les textures de 8 bits no ocupen gairebé espai si les comparem amb textures RGB. En les proves que s'han fet la pèrdua de velocitat és imperceptible i, per tant, l'emulació és pràctica i usable.

Per fer el mapatge de textures s'utilitza el mapatge UV o UV mapping que consisteix en associar els punts d'un espai 2D d'eixos UV a un espai 3D d'eixos XYZ. En el nostre cas, però la Z o tercera dimensió sempre serà zero ja que una pantalla és plana.

A la següent figura es pot veure un exemple general d'aquesta tècnica on es mapa una textura a l'hemisferi superior d'una esfera:

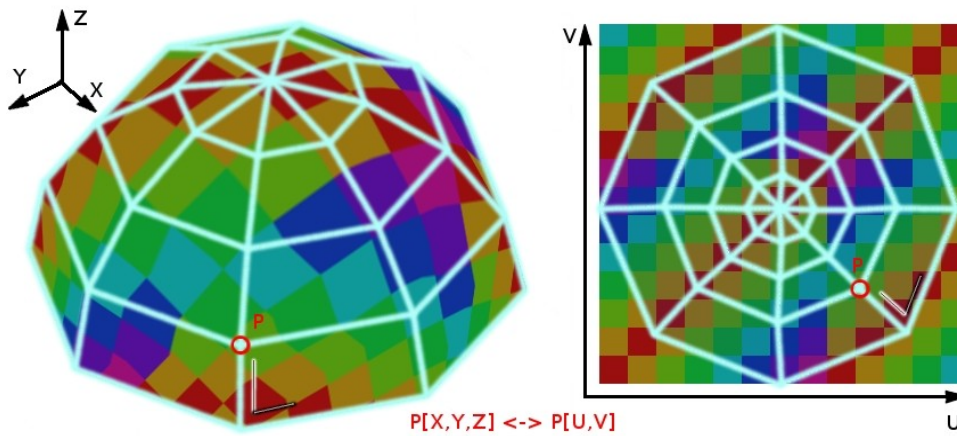


Figura 8.2 UV mapping

Quan s'inicia un mode gràfic a Allegro indicant la resolució i la profunditat de color, bàsicament es crea l'*screen* BITMAP i s'inicialitza el maquinari gràfic de la plataforma subjacent. A partir d'aquest moment tota operació gràfica que tingui com destí la pantalla es veurà automàticament ja que existeix una associació directa entre l'*screen* BITMAP i la VRAM “visible” o de pantalla de la plataforma subjacent. A la PSP també funciona de la mateixa manera però només amb les profunditats de color suportades per la consola: 15, 16 i 32 bits per píxel.

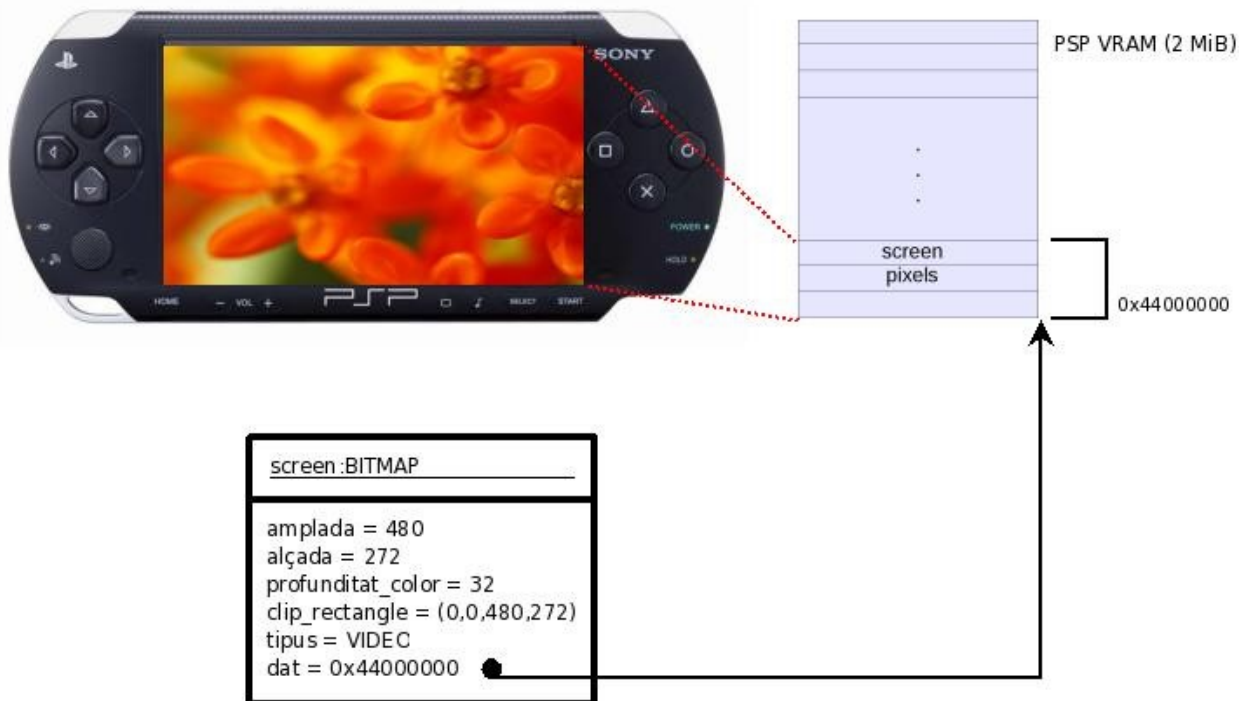


Figura 8.3 Representació estàndard de l'Allegro *screen*

En canvi, quan s'ha iniciat un mode gràfic amb 8 bits de color el funcionament és el següent:

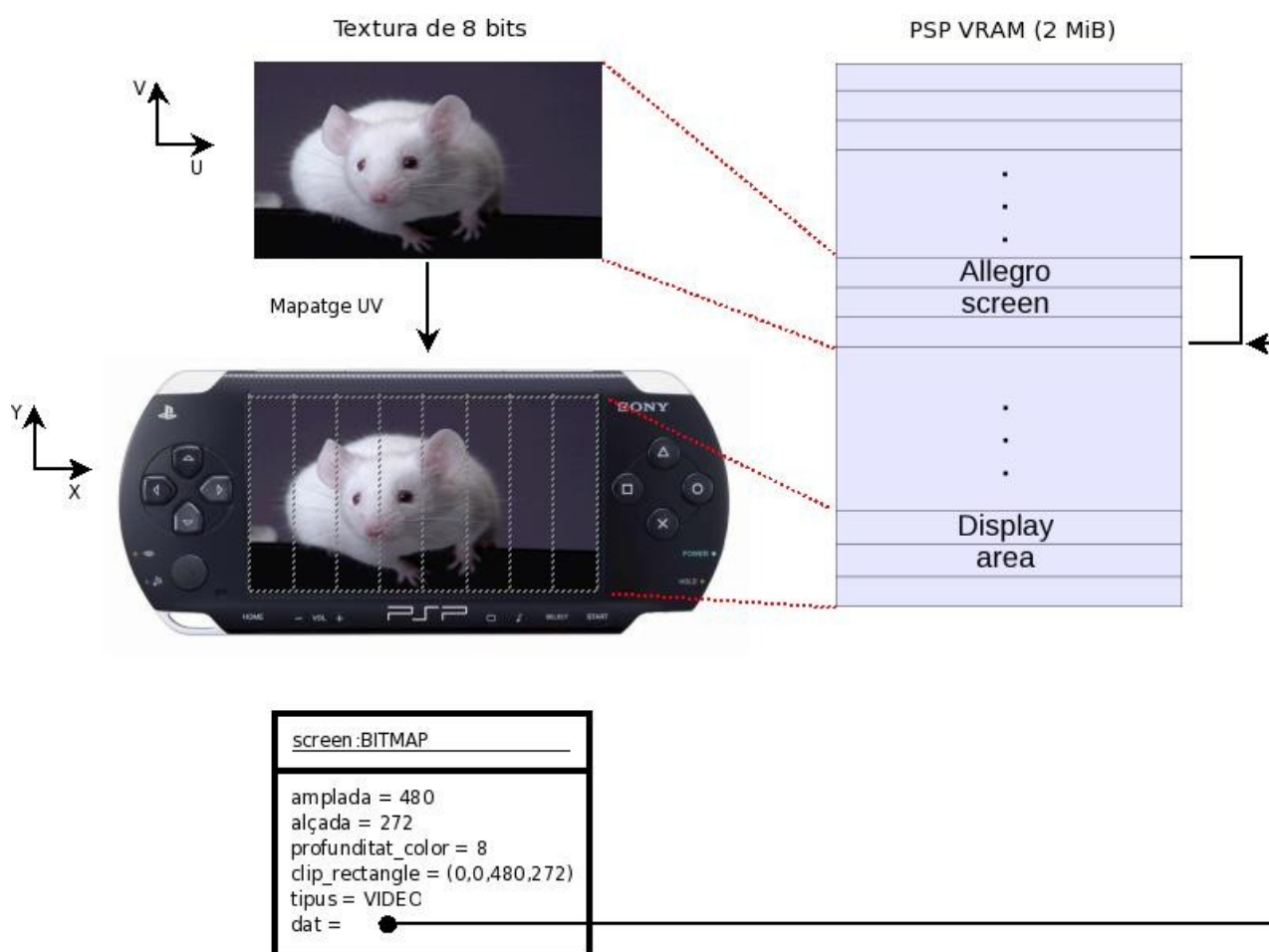


Figura 8.4 Emulació 8 bpp

8.4 Suport resolucions majors de 480x272

La resolució màxima del LCD de la PSP és de 480x272 píxels amb un relació d'aspecte aproximada de 16:9. Aquesta resolució és menor que les resolucions habituals dels ordinadors: en Windows típicament 640x480, 800x600, ... i fins i tot una resolució molt comú de DOS com 400x300 és massa gran per a la PSP.

Un dels objectius en la conversió de la llibreria Allegro a la PSP, a més de fer-la plenament funcional, era aconseguir la portabilitat directe de centenars d'aplicacions ja existents. Moltes d'aquestes aplicacions utilitzen aquestes resolucions majors i per tant era quasi obligat suportar-les d'alguna manera.

El principi en el qual es basa el suport per a resolucions majors és l'escalament de dimensions i coordenades. Per una banda les dimensions que intervenen tant en la creació de la pantalla com en

la creació de bitmaps o el dibuix de figures geomètriques (radi d'un cercle, ...). I per l'altre les coordenades respecte a la pantalla o d'altres bitmaps que intervenen en el dibuix de primitives gràfiques en general (origen i final d'una línia, centre d'un cercle, ...).

Aquest escalament depèn de dos factors, un per a cada dimensió: un factor de reducció horitzontal i un factor de reducció vertical. És en el moment de l'establiment del mode gràfic on, si la resolució demanada supera la resolució màxima de la PSP, es selecciona el parell de factors de reducció adients i a partir d'aquest moment es fa l'escalament de les successives operacions gràfiques de l'aplicació en execució.

Com s'ha explicat al capítol de disseny, tot bitmap té associat una interfície de mètodes per a treballar amb ell. Fent una determinada implementació d'aquesta interfície podem, en aquest cas, realitzar aquest procés d'escalament gràfic de forma totalment transparent. La taula de mètodes que implementa la interfície GFX_VTABLE s'anomena PSP_SCALING_VTABLE, una per a cada profunditat de color.

Com a exemple il·lustratiu imaginem que volem dibuixar un rectangle de coordenades (160,120) i (480,360) a un bitmap BMP de mida 640x480 píxels:

```
BITMAP *BMP;

/* S'estableix un mode gràfic amb una resolució comú de Windows */
set_color_depth(16);
1 set_gfx_mode(GFX_AUTODETECT, 640, 480, 0, 0);

/* Es crea un bitmap a memòria on dibuixar */
2 BMP = create_bitmap(640, 480);

/* Es dibuixa un rectangle al bitmap tot just creat */
3 rectfill(BMP, 160, 120, 480, 360, makecol(255,128,0));
```

Sense el suport de resolucions majors de 480x272 el resultat de dibuixar el bitmap BMP a la pantalla de la PSP seria el següent (l'origen (0,0) es troba a la cantonada superior esquerra):



En canvi amb el suport de resolucions majors de 480x272 el driver gràfic, per començar, detectaria que és una resolució superior a la física. Per una resolució de 640x480 es seleccionaria un factor de reducció d'amplada de 0.75 i un factor de reducció d'alçada de 0.5:

```

1 /* Implementa el mètode init() de la interfície GFX_DRIVER */
static BITMAP *psp_display_init(int w, int h, int v_w, int v_h, int color_depth)
{
    ...
    if (w > 480 || h > 272) {
        ...
1     if (set_scale_factors(w, h) != 0) {
            ...
        }
        ...
    }
    ...
}

```

En la creació del bitmap a memòria RAM, el driver de sistema en aquest cas li aplicaria l'escalament establert anteriorment:

```

/* Crea un bitmap a RAM */
BITMAP * psp_create_bitmap_ex(int color_depth, int width, int height, int
                               with_scaling)
{
    GFX_VTABLE *vtable;
    BITMAP *bitmap;
    ...

    /* Associa al bitmap la implementació PSP_SCALING_VTABLEXX corresponent */
2 if (with_scaling) {
        vtable = _psp_get_vtable(color_depth);
        ...
    }
    ...

    /* Redueix les dimensions del rectangle de retallament i les dimensions del bitmap
       a crear */
2 if (with_scaling) {
        ...
        CALCULATE_SCALED_CLIP_RECT(bitmap);
        ...
        CALCULATE_SCALED_BMP_SIZE(bitmap);
    }
    ...
}

```

Finalment en dibuixar el rectangle al bitmap es modificarien, segons els factors d'escalament establerts, les coordenades dels seus punts:

```
/* Implementa el mètode rectfill() de la interfície GFX_VTABLE */
void scaled_rectfill(BITMAP *bmp, int x1, int y_1, int x2, int y2, int color)
{
    ...
    3 CALL_ORIG_FUNCTION(bmp, rectfill, (bmp, SCALE_W(x1), SCALE_H(y_1), SCALE_W(x2),
        SCALE_H(y2), color));
    ...
}
```

I el resultat a la pantalla de la PSP seria aquest:



8.5 El gestor de VRAM

Allegro permet crear bitmaps a la memòria de vídeo o VRAM de la plataforma subjacent. Aquests bitmaps es solen utilitzar en operacions gràfiques accelerades ja que normalment el processador gràfic només té accés a la memòria de vídeo.

En la majoria de plataformes suportades per Allegro, com Windows o MacOS X, els drivers gràfics respectius utilitzen llibreries que gestionen la seva memòria de vídeo (per exemple, al MacOS X s'utilitzen els Offscreen Graphics Worlds per a crear bitmaps de vídeo).

Les llibreries del PSPSDK són de molt més baix nivell en aquest aspecte i ha estat necessari implementar un gestor de VRAM per a donar suport a la creació i destrucció de bitmaps de vídeo en Allegro: `create_vídeo_bitmap()` i `destroy_vídeo_bitmap()`. Cal fer notar que aquesta particularitat no la trobem en els bitmaps de RAM ja que són les funcions de la llibreria estàndard de C `malloc()` i `free()` les que s'encarreguen de la gestió de la RAM a la PSP.

No tenia com a objectiu tornar a inventar la roda i fent memòria (mai millor dit) de temari d'assignatures de sistemes operatius, vaig prendre com a base el gestor de memòria del MINIX, un sistema operatiu amb aparença UNIX que es va implementar d'inici als IBM PC i és utilitzat per a ensenyar el funcionament intern dels sistemes operatius.

Adaptat a la PSP, aquest gestor de memòria està format per una llista enllaçada de forats o *holes list* no contigus a memòria. Cadascun d'aquests elements de la llista apunta a un bloc de la VRAM disponible per a la creació d'un o més bitmaps de vídeo. Hi ha funcions per a la inicialització del gestor, reserva de VRAM, alliberament de VRAM i fusió de blocs buits o forats contigus per a la compactació de la memòria.

A la figura següent es mostra el gestor de VRAM en un moment determinat de la seva execució. Es poden distingir dos bitmaps de vídeo i tres blocs lliures de diferent mida disponibles:

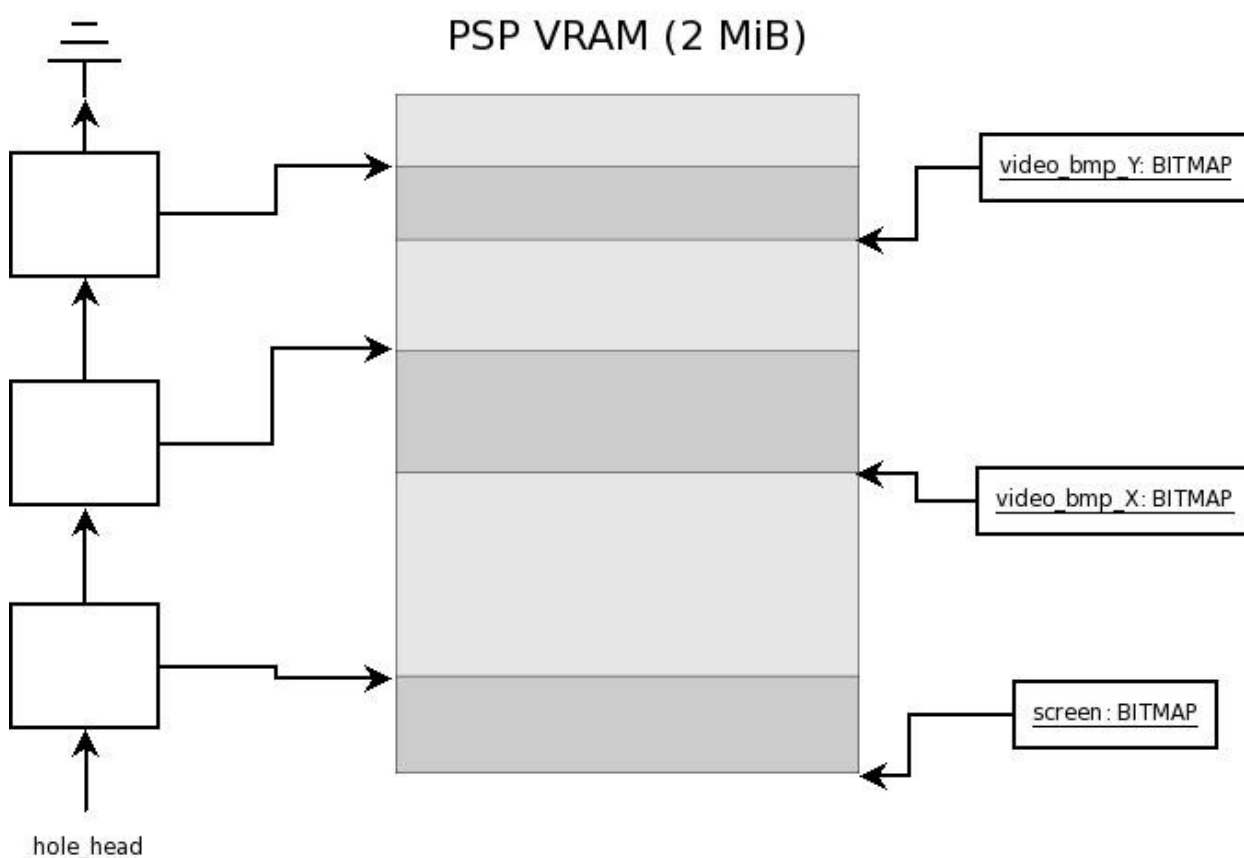


Figura 8.5 PSP video memory manager

8.6 Estructura fitxers llibreria Allegro

Degut a la seva naturalesa de llibreria es fa necessari tenir ben clar l'arbre de directoris i fitxers que contindran tant els fitxers de capçalera com el codi font d'Allegro. Des del punt de vista del desenvolupador d'aplicacions multimèdia que utilitzi una llibreria complexa com Allegro resulta de molta utilitat conèixer la distribució dels seus fitxers de capçalera o *headers*. Això li dóna una visió directa dels diferents mòduls funcionals de la llibreria amb les seves variables i tipus de dades públics. Per altra banda, conèixer la distribució del codi font és imprescindible pel desenvolupador de la pròpia llibreria.

A partir de la capçalera principal `include/<allegro.h>` el programador té accés a la llibreria sencera. El directori `include/allegro/` conté les capçaleres que donen accés als mòduls individuals: `sound.h`, `color.h`, `keyboard.h`, etc. Cada fitxer capçalera del mòdul conté les seves declaracions i definicions de tipus.

Les diferents plataformes suportades per Allegro es localitzen a `include/platform/`, que conté els fitxers de capçalera de totes les plataformes amb informació específica per a la construcció i configuració de la llibreria o informació dels drivers. A la figura 7.7 es mostra l'estructura i el contingut en detall dels fitxers de capçalera d'Allegro.

En quant a la distribució del codi font de la llibreria l'estructura és similar: el directori `src/` conté el codi font dels diferents mòduls d'Allegro (la part independent de la plataforma): `sound.c`, `graphics.c`, `text.c`, etc., i els directoris `src/dos/`, `src/win/`, `src/unix/`, `src/psp/`, etc. contenen el codi dels drivers (part dependent de la plataforma).

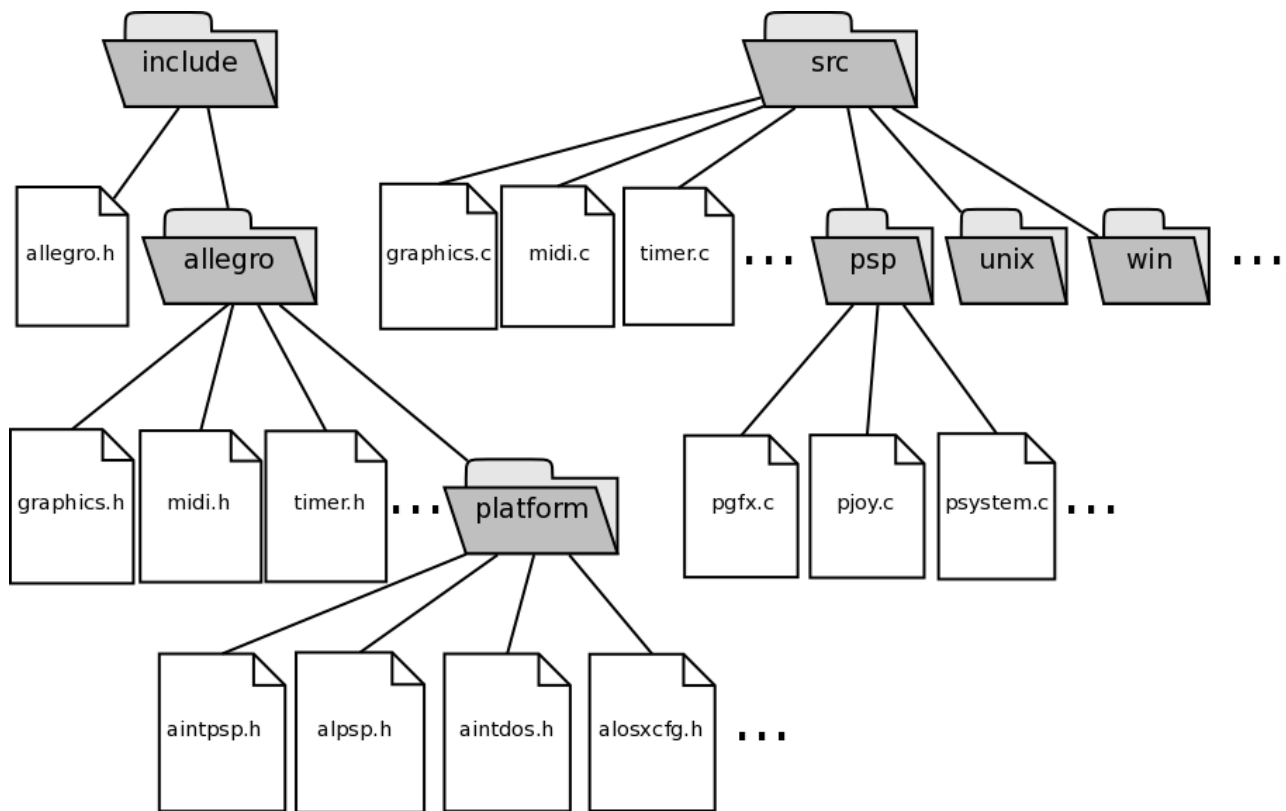


Figura 8.6 Estructura fitxers font i capçalera

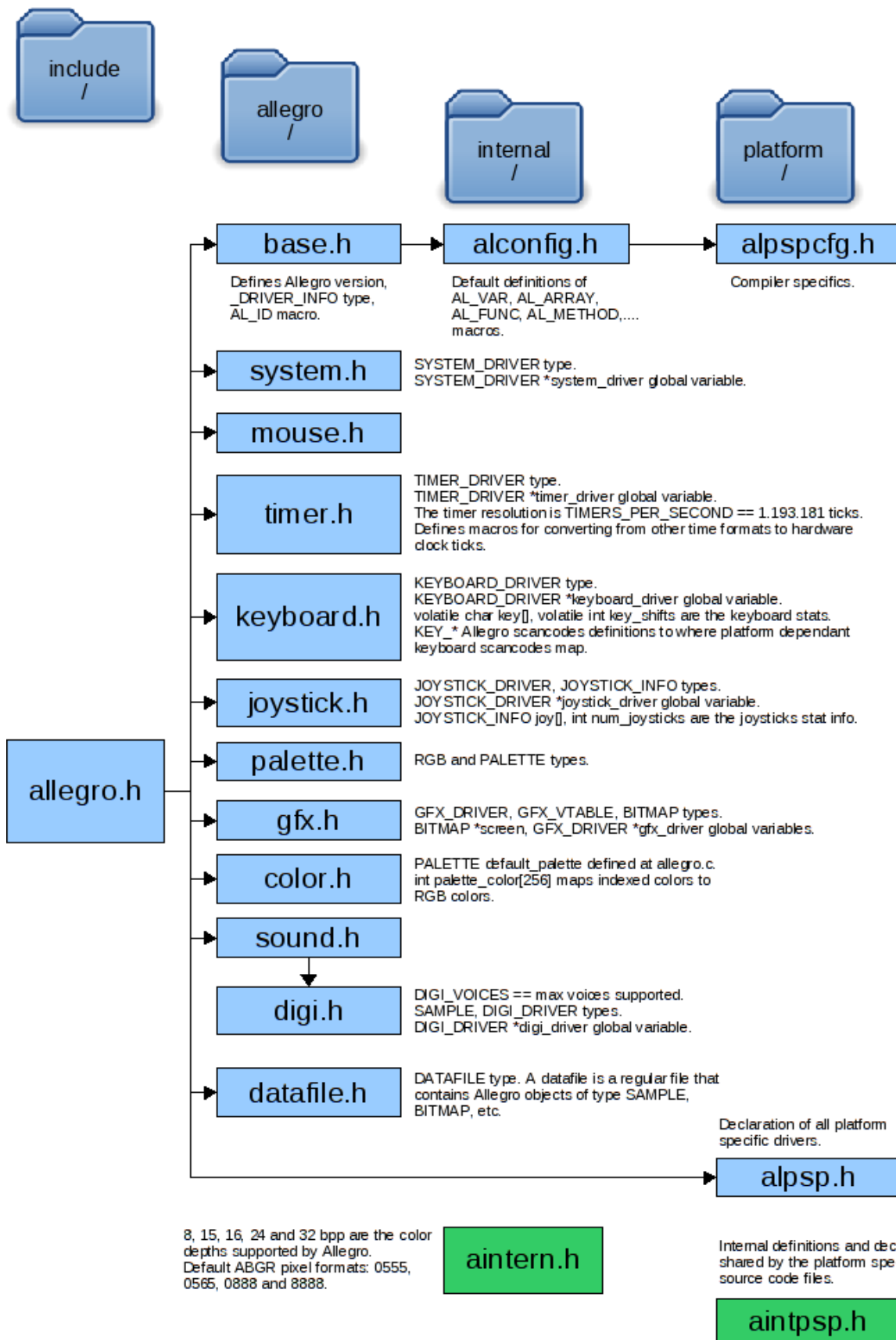


Figura 8.7 Fitxers capçalera de la llibreria Allegro

8.7 Entorn de desenvolupament

Les eines que més s'han utilitzat en el desenvolupament d'aquest projecte han estat el Geany i el PSPLink. Aquest últim gairebé imprescindible en un entorn cross-compiling com era el nostre cas, on es desenvolupa programari en un plataforma diferent (PC) a la plataforma objectiu (PSP).

8.7.1 Geany



Geany és un editor de text amb característiques bàsiques d'un IDE (Entorn integrat de desenvolupament). Per tant es tracta d'una eina lleugera que facilita l'edició i la compilació de programari. Suporta un ampli ventall de llenguatges entre ells C, shell scripts i Makefiles. És de codi obert i molt ràpid, ideal per entorns Linux.

8.7.2 PSPLink

PSPLink és una aplicació desenvolupada per TyRaNiD (un dels pesos pesats dintre de l'*scene* de la PSP) i té com objectiu facilitar l'execució d'aplicacions a la PSP mitjançant una línia de comandes sense necessitat d'anar al XMB o shell gràfic de la consola.

Sense aquesta utilitat un desenvolupador de *homebrew* a la PSP hauria de seguir el següent procediment:

- 1) Compilar l'aplicació utilitzant el PSP toolchain (psp-gcc, ...) sota Linux.
- 2) Moure el fitxer resultant a la Memory Stick de la consola mitjançant la connexió USB amb l'ordinador.
- 3) Navegar pel XrossMediaBar de la PSP i llançar l'execució de l'aplicació tot just compilada.

Tenint en compte que els passos 2) i 3) es prenen el seu temps, seguir aquest mètode d'execució de programari a la consola es torna molt lent i ineficient.

És aquí on entra a escena PSPLink, proporcionant un shell sobre USB des d'on es poden llançar i depurar aplicacions, capturar imatges o consultar informació del sistema com l'estat de la memòria o els mòduls i threads carregats.

Només és necessari connectar la consola al PC via USB, llançar el servidor PSPLink a la PSP, obrir el shell del PSPLink a un terminal de Linux i a partir d'aquest moment el procés anteriorment descrit quedaria així:

- 1) Compilar l'aplicació utilitzant el PSP toolchain (psp-gcc, ...) sota Linux.
- 2) Llançar l'execució de l'aplicació tot just compilada.

En aquest cas el pas 2) és immediat creant així la il·lusió d'estar executant programari al mateix PC.

Capítol 9. Resultats

9.1 Jocs de proves

9.1.1 Humphrey Remake

En aquest capítol entra a escena el quart objectiu d'aquest PFC: la conversió a la PSP del videojoc amateur *Humphrey Remake*, de Ignacio Pérez Gil.



Com ja s'ha explicat anteriorment, durant el desenvolupament de la conversió d'Allegro aquest videojoc ha estat el centre de totes les proves i la referència per a mesurar el rendiment d'Allegro. Principalment per dues raons, per nostàlgia i per ser un videojoc complet dissenyat per a PC, fet que ens indica moltes vegades (i així ha estat en aquest cas) que no hi ha una bona optimització de recursos: CPU i memòria principalment.

Tècnicament el *Humphrey Remake* funciona amb una resolució de 640x480 amb una profunditat de color de 16 o 32 bits per píxel i utilitza la tècnica del *double buffering* per a la reproducció dels successius quadres o *frames* d'animació. Aquesta animació transcorre a una velocitat de 50 *frames* per segon. El videojoc es controla amb teclat o joystick. En quant al so, per als efectes sonors s'utilitzen *samples* mono de 8/16 bits per mostra i 11025/22050 Hz de freqüència; i per a la música de fons s'utilitza el format .IT (Impulse Tracker) que descodifica la llibreria externa DUMB (Dynamic Universal Music Bibliotheqe).

El primer obstacle a superar va ser la resolució, és clar que 640x480 és massa gran per als 480x272 de la PSP. En la tècnica del *double buffering*, però, es fabrica un *frame* d'animació a un bitmap de memòria i només quan està llest es mostra a la pantalla. Per tant semblava evident que tan sols era necessari fer l'escalament d'un *frame* d'animació just abans de mostrar-lo per pantalla i no de totes les operacions gràfiques. És a dir, internament es treballaria amb 640x480 però just al final es reduiria la imatge.

Com es va comprovar el rendiment era molt baix i insuficient degut a la gran quantitat d'operacions que intervenen en la fabricació d'un quadre d'animació, una gran quantitat d'sprites que es mouen contínuament, la decodificació i reproducció de música, etc. i tot això responsabilitat exclusiva de la CPU. Als 2,40 o 3 Ghz d'un PC va perfecte, però els 333 Mhz de l'ALLEGREX sembla que no eren suficients.

Per superar aquest segon obstacle era necessari accelerar el procés d'alguna manera. Després de fer un anàlisi de rendiment o *profiling* en anglès, vaig comprovar que les funcions `draw_sprite()` i `blit()` d'Allegro eren les que més temps consumien en l'execució.

Amb la premissa bàsica de no tocar el codi original del *Humphrey Remake* només quedava l'opció d'accelerar la funció `blit()`, tant els blits interns de RAM-RAM com el blit final de RAM a pantalla. La funció `draw_sprite()` implica la gestió d'un color transparent o màscara que només és possible de fer-la acceleradament mitjançant el Graphics Engine de la PSP si es treballa a la VRAM. Però el *Humphrey* treballa exclusivament a RAM.

Una vegada aconseguides les versions accelerades dels dos tipus de `blit()` es va comprovar que la millora era pràcticament nul·la. Fent un anàlisi amb molta més profunditat es va veure que el coll d'ampolla era una transferència interna d'una imatge molt gran usant `draw_sprite()`. Arribats a aquest punt era necessari pensar en una altra solució.

El problema real era que s'estava treballant amb dades de l'ordre de 640x480 píxels cada *frame*. Per tant calia reduir aquest volum fent escalament des de bon començament implementant el suport per a resolucions majors de 480x272. Finalment es va aconseguir el resultat esperat i la conversió va ser tot un èxit.



El següent quadre mostra les diferents mesures de rendiment realitzades durant tot el desenvolupament de la conversió del *Humphrey Remake*.

El rendiment s'expressa amb *Frames/s* i *MiB/s*. Els *Frames/s* expressen el nombre de marcs complets d'animació que el sistema pot generar en un segon, com més alt aquest valor més velocitat i fluïdesa tindrà l'aplicació. Normalment els videojocs generen *50 frames/s* si estan en format PAL i *60 frames/s* en format NTSC.

Per altra banda, els *MiB/s* són una unitat derivada dels *Frames/s* i la mida d'un marc o imatge, significa mebibytes (2^{20} bytes) per segon i representa la quantitat de dades transferides per segon a la pantalla.

Per exemple, per calcular els *MiB/s* en el cas de “Sense suport per 640x480” i usant exclusivament la CPU, Allegro generaria internament marcs de 640x480 píxels on cada píxel està representat amb dos *bytes*:

$$640*480*2 \text{ bytes/frame} * 27 \text{ frames/s} = 16.588.800 \text{ bytes/s} / 2^{20} \text{ bytes/MiB} = 15 \text{ MiB/s}$$

	Blit RAM → screen	Blit RAM → RAM	Frames/s	MiB/s
	CPU: mètode stretch_blit() d'Allegro	CPU: mètode blit() d'Allegro	27	15
Sense suport per 640x480	CPU: mètode stretch_blit() d'Allegro	GE: blit accelerat	30	17
	GE: mapatge i escalament de textura	CPU: mètode blit() d'Allegro	28	16
	GE: mapatge i escalament de textura	GE: blit accelerat	31	18
Amb suport per 640x480: escalament intern	CPU: mètode blit() d'Allegro	CPU: mètode blit() d'Allegro	35	8
	GE: blit accelerat	GE: blit accelerat	70	17

Figura 9.1 Rendiment *Humphrey Remake* a la PSP

Sense el suport per 640x480 i amb acceleració la millora és pràcticament inexistent. També es pot veure que la reducció del volum de dades amb el suport per 640x480 té una important penalització i això explicaria perquè la versió per CPU té un rendiment tan baix si la comparem amb la versió accelerada final (el doble de rendiment).

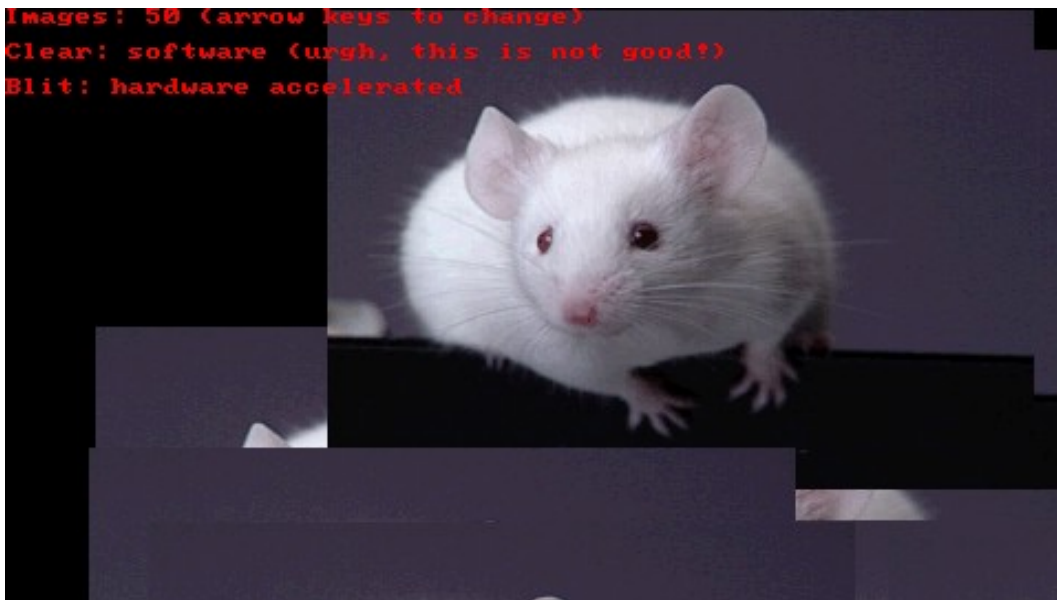
Finalment cal fer notar que sense el suport per 640x480 calia fer l'escalament del marc d'animació abans de mostrar-lo per pantalla, per aquesta raó les tècniques utilitzades a cada tipus de *blit* són diferents.

9.1.2 Exemples oficials d'Allegro

La llibreria Allegro es distribueix públicament com un paquet que a més d'incloure el codi font, instruccions i scripts per a la seva construcció, conté un conjunt de petites aplicacions o exemples que serveixen com a joc de proves de totes les funcionalitats d'Allegro.

Tot seguit s'explicaran els resultats d'alguns d'aquests exemples com una mesura de l'èxit de la conversió d'Allegro a la PSP. Concretament es veuran els exemples `exaccel.c`, `exjoy.c`, `expal.c`, `exmid.c`, `exscroll.c` i `extimer.c`.

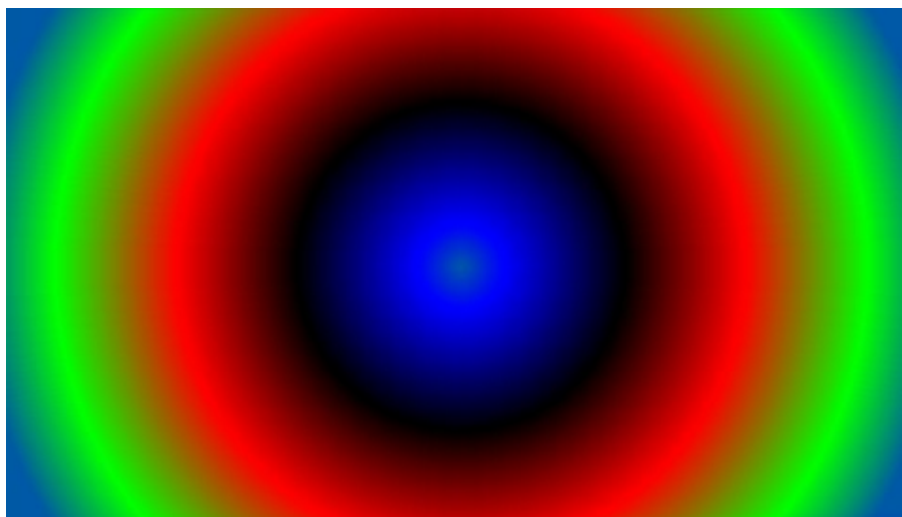
- ◆ **exaccel.c:** Aquest programa mostra com utilitzar memòria VRAM per a emmagatzemar gràfics que després seran transferits a pantalla fent page flipping. Es carrega una imatge de 320x200x32 i es fan fins a 50 blits d'aquesta a pantalla. El rendiment amb blit VRAM-VRAM accelerat arriba fins als 75 *frames/s*. Aquí es demostra que les característiques del driver gràfic de la PSP com el gestor de memòria de vídeo, el page flipping i les transferències VRAM-VRAM accelerades funcionen molt bé.

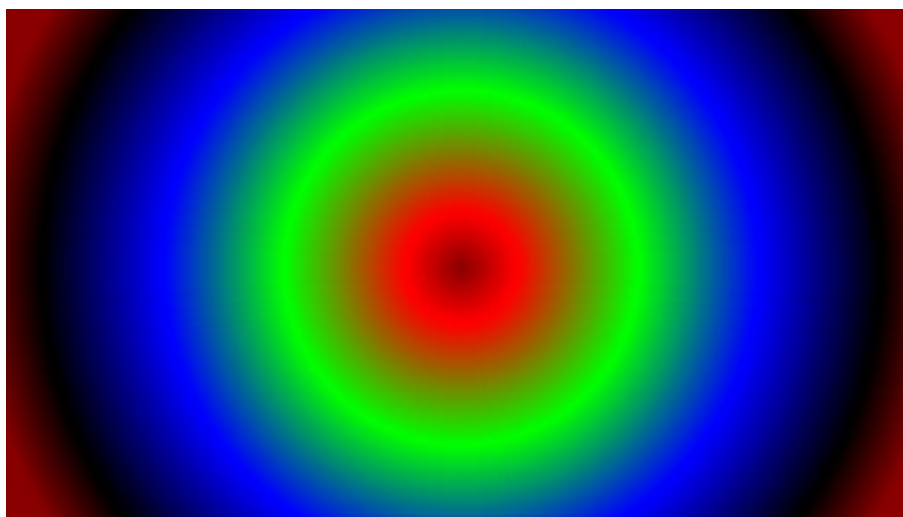


- ◆ **exjoy.c:** Aquest programa detecta i llegeix l'entrada del joystick. S'informa l'usuari dels botons actius.

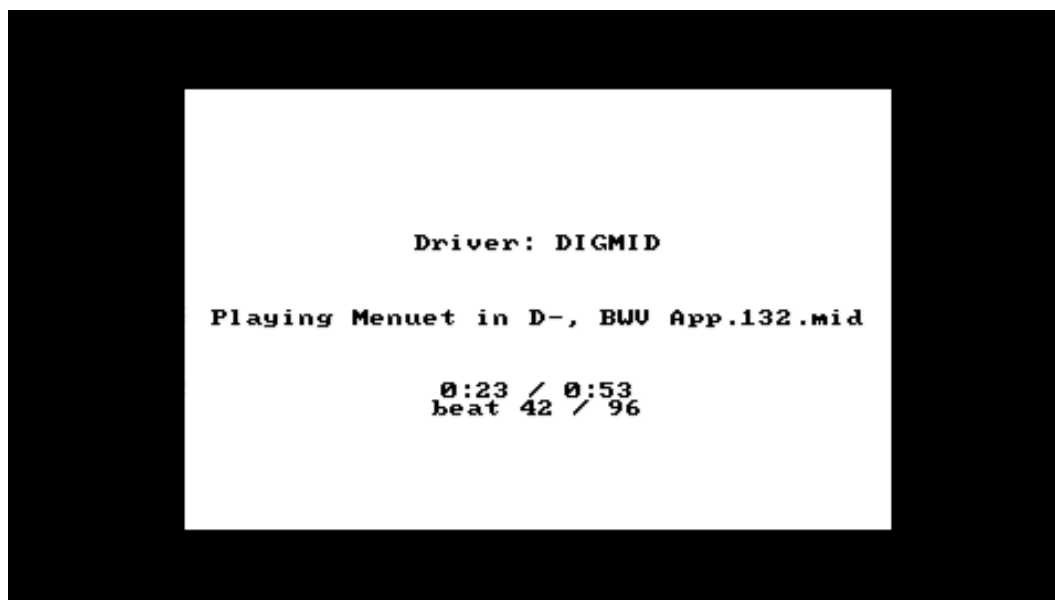


- ◆ **expal.c:** Aquest programa mostra com manipular la paleta de colors fent que una imatge estàtica tingui moviment. Un exemple on es comprova que l'emulació de 8 bits de color i la gestió de paletes a la PSP funciona perfectament.





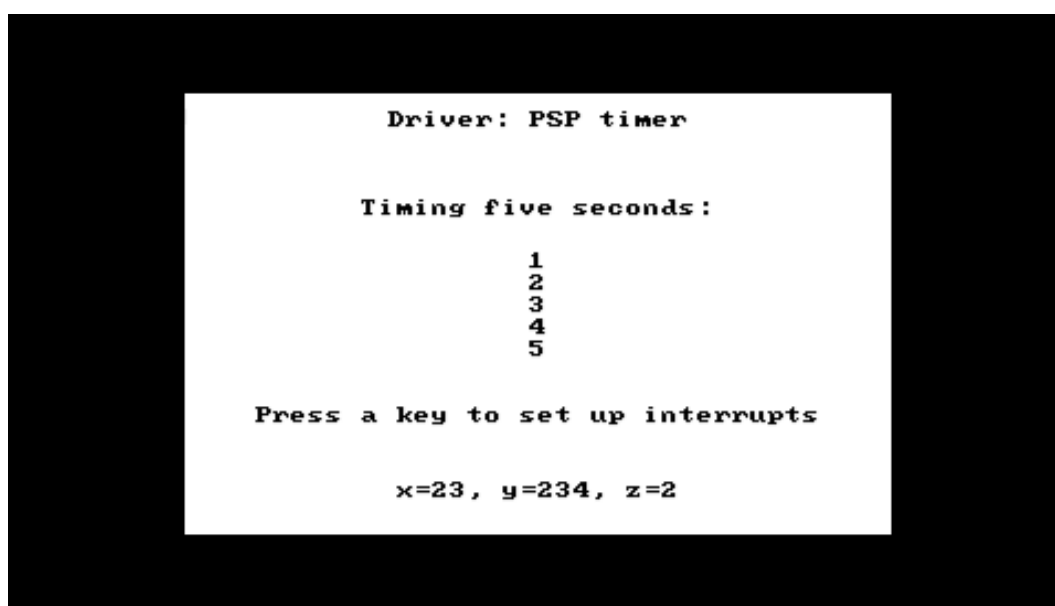
- ◆ **exmid.c:** Aquest programa mostra com reproduir fitxers MIDI. La PSP no té suport per MIDI però Allegro ofereix el driver DIGMID que l'emula per CPU. Només són necessaris els instruments en un o més fitxers externs. Cal fer notar que aquesta funcionalitat disminueix enormement el rendiment d'Allegro ja que fa un ús intens dels threads del temporitzador i so digital de la PSP, penalitzant el thread principal (veure capítol Implementació, apartat Threads). Per tant no és gaire recomanable en aplicacions on es requereixi eficiència.



- ◆ **exscroll.c:** Aquest programa mostra l'*scrolling* o desplaçament a través de la pantalla virtual. S'inicia el mode gràfic de la PSP amb una pantalla virtual de 960x272 o el que és el mateix, una pantalla el doble d'ample que la física (480x272), i es va movent de dreta a esquerra un senzill paisatge. Aquí es demostra la correcta implementació a la PSP del suport per a pantalles virtuals i l'*scroll* o desplaçament per aquesta pantalla virtual.



- ◆ **extimer.c:** Aquest programa mostra l'ús del temporitzador. S'instal·len tres rutines que incrementen cadascuna un comptador x, y i z amb una freqüència diferent per simular un cronòmetre. La primera s'executa cada segon, la segona rutina s'executa cada dècima de segon i la tercera s'executa cada 10 segons. Per comprovar la precisió es va comparar l'execució d'aquest exemple a la PSP amb el rellotge del sistema operatiu i amb alguns cronòmetres online com <http://www.online-stopwatch.com/full-screen-stopwatch/>.



9.2 Conclusions

Després de comprovar el rendiment d'Allegro puc assegurar que la conversió compleix els mínims exigibles i els ultrapassa. Aquest rendiment s'ha aconseguit majoritàriament gràcies a que la PSP es pot programar a molt baix nivell a diferència, per exemple, d'un PC amb Windows o un Mac amb Mac OS X.

Un dels objectius era que aquesta conversió fos capaç d'executar videojocs i aplicacions ja existents i pensades per ordinadors de més potència. En el moment d'escriure aquestes línies ja s'han convertit fins a 4 videojocs complets de diferent complexitat usant aquesta llibreria:

- **Humphrey Remake** de Ignacio Pérez Gil.
(<http://retrospec.sgn.net/users/ignacio/humpe.htm>)
- **Conny Carrot – Quest for Revenge** de Simon Parzer.
(<http://www.allegro.cc/depot/ConnyCarrot-QuestforRevenge/>)
- **Ghouls'n Ghosts Remix** de Ramuso.
(<http://www.valarsoft.com/ramuso/index.php?dpage=pagine&page=pagine&pagID=8>)
- **Alex the Allegator 4** de Johan Peitz
(<http://www.freelunchdesign.com/games.php?id=7>)

Aquest fet demostra que Allegro PSP és portable, eficaç i eficient. En quant a la usabilitat, qualsevol desenvolupador amb un mínim d'experiència amb programació a la PSP pot crear aplicacions o fer conversions amb Allegro amb un cost molt baix de temps i recursos.

En quant a les funcionalitats suportades, Allegro PSP ofereix al programador la majoria de les especificacions de la llibreria, convertint a Allegro en una eina alternativa molt potent per a la programació de videojocs a la PSP. Des del 2005 l'única opció equivalent possible era la llibreria SDL (<http://www.libsdl.org/>), per aquesta raó m'omple de satisfacció comprovar que aquest projecte serà útil a una gran quantitat de desenvolupadors.

Finalment voldria comentar a títol personal, que gràcies al coneixement avançat de programació de baix en nivell de C que he adquirit, he estat seleccionat per una empresa dedicada a la programació de microcontroladors. Actualment participo en un projecte de programació d'un sistema operatiu per a una targeta intel·ligent o *SmartCard*, un dels requisits d'aquest projecte és precisament la programació dels *drivers* o *middleware* per comunicar la targeta amb diferents sistemes operatius i/o plataformes.

9.3 Perspectives de futur

Malgrat sembli el contrari hi ha molta feina a fer encara. Aquest serien els punts a destacar:

- Implementació del *driver* del ratolí mitjançant el controlador analògic de la PSP.
- Suport per al *joystick* analògic d'Allegro utilitzant lògicament també el controlador analògic.
- Donar suport a un teclat complet mitjançant un OSK (On Screen Keyboard) el qual simula un teclat virtual a la PSP utilitzant com a mètode d'entrada només els botons de la PSP. Es pot utilitzar la llibreria que proporciona el PSPSDK, bàsicament una interfície al OSK de Sony present al *kernel* de la PSP. També es pot utilitzar un OSK alternatiu anomenat Danzeff OSK el qual ofereix un millor mètode d'introducció de caràcters (<http://pspupdates.qj.net/Danzeff-OSK-Source-Released/pg/49/aid/23876>).
- Implementar versions accelerades de moltes de les funcions gràfiques d'Allegro com *draw_sprite()* i *clear_bitmap()* quan el destí es VRAM o memòria de vídeo. Aquestes dues operacions són molt utilitzades juntament amb el *blit()*.
- Determinar si seria interessant implementar part dels *drivers* o de les funcions gràfiques accelerades usant el llenguatge assemblador del MIPS. La versió DOS d'Allegro, per exemple, té molt de codi en assemblador del i386.

Actualment els desenvolupadors d'Allegro estan treballant pel que serà Allegro 5. Aquesta versió d'Allegro ha sofert una reescriptura de tota la seva API perquè es pugui integrar fàcilment amb d'altres llibreries externes. Algunes de les noves funcionalitats que incorpora són suport per multi-pantalla, reproducció i codificació de vídeo, suport natiu per OpenGL i MP3 o codificació de les coordenades gràfiques amb nombres reals per a suportar sub-píxels.

Seria desitjable i perfectament raonable estudiar la viabilitat de reconvertir Allegro PSP (actualment en la versió 4.3.10) a aquesta nova versió d'Allegro més adaptada a les exigències de programació actuals.

Bibliografia i enllaços

Llibres

Aquests han estat els llibres consultats amb més o menys profunditat:

- Matthew, Neil i Stones, Richard** [2008] *Beginning Linux Programming, 4th edition*, Wrox.
- Kernighan, Brian W. i Ritchie, Dennis M.** [1988] *The C Programming Language, Second Edition*, Prentice Hall.
- Tanenbaum, Andrew S.** [1993] *Sistemas Operativos: Diseño e Implementación*, Prentice Hall
- Levine, John R.** [2000] *Linkers & Loaders*, Morgan Kaufmann.
- Mecklenburg, Robert** [2005] *Managing Projects with GNU Make, Third Edition*, O'Reilly.
- Von Hagen, William** [2006] *The Definitive Guide to GCC, Second Edition*, Apress.
- Tipler, Paul A.** [1993] *Física**, Tercera Edición, Editorial Reverté.

Llocs web

Per a la realització d'aquest PFC i en el procés de recerca i investigació s'ha visitat un ampli nombre de llocs web. Els més importants a destacar són:

- **Allegro - A game programming library.** URL: <http://alleg.sourceforge.net/>
El lloc web oficial d'Allegro amb les últimes notícies sobre la llibreria, documentació, tutorials, llistes de correu, etc.
- **PS2DEV.ORG: Playstation Programming - ;.** URL: <http://ps2dev.org/>
Dedicat a la programació de les consoles Sony PS2, PSP i PS3 amb programari lliure. Des de fa uns quants anys existeixen uns fòrums per desenvolupadors on es pot consultar i extreure molta informació.
- **Google.** URL: <http://www.google.es/>
Res a dir que no s'hagi dit ja sobre aquest gran cercador d'Internet.

Altres webs consultades:

- Allegro
 - **allegro.cc – game developing community network.** URL: <http://www.allegro.cc/>
 - **Allegro 4 development – Allegro wiki.**
URL: http://wiki.allegro.cc/index.php?title=Allegro_4_development

- **Allegro Hacker's Guide.** URL: <http://alleg.sourceforge.net/latestdocs/en/ahack.html>

- PSP i *homebrew*
 - **An introduction to development on the Sony PSP.**
URL: <http://www.osix.net/modules/article/?id=713>
 - **psDevWiki.** URL: <http://wiki.pspdev.org/>
 - **PSP-Programming.com.** URL: <http://www.psp-programming.com/>
 - **Homebrew Illuminati.** URL: <http://yaustar.juliusparishy.com/cu/>
 - **PSP Cache HOWTO.** URL: <http://www.goop.org/psp/cache-howto.html>
 - **Ghoti.nl: PSP tutorials, games.** URL: <http://www.ghoti.nl/PSPtutorials.php>
 - **Home of the Hitmen.** URL: <http://hitmen.c02.at/html/psp.html>
 - **Yet another Playstation Portable Documentation.**
URL: http://hitmen.c02.at/files/yaspsd/psp_doc/frames.html

- Programació multimèdia i en baix nivell
 - **256-Color VGA Programming in C.** URL: <http://www.brackeen.com/vga/>
 - **Welcome to Inverse Reality!.**
URL: <http://www.inversereality.org/tutorials/tutorial.html>
 - **The Scientist and Engineer's Guide to Digital Signal Processing.**
URL: <http://www.dspguide.com/>
 - **VESA BIOS Extension/Accelerator Functions (VBE/AF).**
URL: <http://www.vesa.org/public/VBE/VBE-AF07.pdf>
 - **Program Library HOWTO.**
URL: <http://www.dwheeler.com/program-library/Program-Library-HOWTO/index.html>
 - **Intel ® 64 and IA-32 Architectures Software Developer's Manuals.**
URL: <http://developer.intel.com/products/processor/manuals/index.htm>
 - **Data alignment: Straighten up and fly right.**
URL: <http://www.ibm.com/developerworks/library/pa-dalign/>
 - **CS 551/851: Big Data in Computer Graphics (Fall 2002) - Texture**
URL: <http://www.cs.virginia.edu/~gfx/Courses/2002/BigData/papers/Texturing/>

- Entorn i eines de desenvolupament
 - **FedoraForum.org – Fedora Support Forums and Community.**
URL: <http://forums.fedoraforum.org/>

- **GNU GLOBAL source code tag system.** URL: <http://www.gnu.org/software/global/>
- **Geany.** URL: <http://www.geany.org/>

- Altres
 - **[32] How to mix C and C++, C++ FAQ Lite.**
URL: <http://www.parashift.com/c++-faq-lite/mixing-c-and-cpp.html>
 - **Name mangling – Wikipedia, the free encyclopedia.**
URL: http://en.wikipedia.org/wiki/Name_mangling
 - **Performance analysis on Playstation Portable.**
URL: <http://zytek.nuxi.pl/psp/pspprofilngV1.pdf>

Annex A: Glossari

Allegrex	És el nom de la CPU principal de la PSP. Aquesta CPU està basada en un processador RISC MIPS32 R4000 i té una velocitat programable que oscil·la entre 33 Mhz i 333 Mhz.
Allegro	Una llibreria de programari gratuïta i de codi obert per al desenvolupament de videojocs i aplicacions multimèdia.
Blit	En informàtica gràfica, BIT BLIT (bitblt, blitting, etc) és una primitiva gràfica que consisteix a combinar dos mapes de bits en un de sol. Es tracta d'una de les primitives gràfiques més bàsiques i per tant més utilitzades en gràfics 2D. El nom deriva de la la instrucció BitBLT (bit block transfer, transferència de bloc de bits) de l'ordinador Xerox Alto.
C	Un llenguatge de programació d'ordinadors de propòsit general desenvolupat el 1972 per Dennis Ritchie a Bell Telephone Laboratories, per ser utilitzat amb el sistema operatiu Unix. Malgrat C va ser dissenyat per implementar programari de sistema, és també molt popular per a desenvolupar programari portable ja que existeix un compilador per aquest llenguatge a pràcticament qualsevol plataforma/sistema.
C++	Un llenguatge de programació dissenyat a mitjans dels 80 per Bjarne Stroustrup. La intenció de la seva creació va ser estendre el reeixit C i convertir-lo en un llenguatge orientat a objectes.
Cache	La memòria cau, o memòria <i>cache</i> és aquella memòria d'alta velocitat instal·lada en el mateix processador i en la qual s'emmagatzemen les dades que el processador necessita utilitzar més freqüentment.
Clipping rectangle	En informàtica gràfica de vegades és desitjable restringir l'efecte de primitives gràfiques a una regió d'un àrea major i protegir altres regions contigües. Totes les primitives gràfiques són acotades a l'espai d'aquest clipping rectangle i res es dibuixa fora d'aquest espai.
Debugging	És el procés d'identificar i corregir errors de programació.
Double buffering	En informàtica gràfica, és una tècnica per a disminuir o eliminar efectes visuals no desitjats del procés del dibuixat d'imatges. El seu nom prové del fet que s'utilitzen dos buffers: el buffer actiu i visible o front buffer, i el buffer de treball o en fabricació anomenat back buffer.

Embedded system	En català sistema incrustat (o sistema integrat) és un sistema informàtic d'ús específic, que és encapsulat totalment pel dispositiu que controla. Té requisits específics i realitza tasques ben definides, a diferència d'un ordinador personal d'ús general.
Firmware	Fa referència al conjunt d'instruccions de programa per a propòsits específics, gravat en una memòria ROM, que estableix la lògica de més baix nivell que controla els circuits electrònics d'un dispositiu de qualsevol tipus.
Frame	Fa referència al contingut d'una pantalla o d'una imatge i forma part d'una seqüència animada. També s'anomena marc d'animació. El nombre de <i>frames</i> per segon en una aplicació gràfica es sol utilitzar per a mesurar el seu rendiment.
GE	Veure <i>Graphics Engine</i> .
Graphics Engine	El cor de la capacitat gràfica de la PSP modelat segons una API similar a OpenGL. Pot operar en 2D i 3D. Soporta, entre d'altres, les següents característiques: textures, retallament 3D, morphing, il·luminació, alpha blending, etc.
Homebrew	És el conjunt d'aplicacions informàtiques realitzades per aficionats de plataformes de videojocs propietàries.
Indexed format	És una forma de representar imatges en informàtica on cada píxel es representa com un índex a una paleta prefixada de colors (normalment 8 bits per accedir 256 colors).
Llibreria	En informàtica, és una col·lecció de subrutines o classes utilitzades per al desenvolupament de programari.
Mapatge UV	És un procés de modelatge 3D que consisteix a “embolicar” una imatge 2D, coneguda com a textura, en un model 3D. UV és el nom dels eixos de referència de la imatge 2D.
Memory stick	Un format de tarja de memòria flash extraïble desenvolupada per Sony. És un dels mètodes d'emmagatzemament a la PSP i permet la càrrega de <i>homebrew</i> a la consola.
MIPS	Les sigles de <i>Microprocessor without Interlocked Pipeline Stages</i> que identifiquen tota una família de microprocessadors d'arquitectura RISC desenvolupats per MIPS Technologies. Són utilitzats a molts dispositius i concretament a les consoles de Sony incloent la PSP.

Mixer	És el component analògic d'una tarja de so que s'encarrega principalment de gestionar i mesclar les senyals d'àudio de diferents fonts. Un software mixer s'encarrega de “simular” aquest comportament permetent reproduir múltiples veus sonores en només una veu real.
newlib	És una implementació de la llibreria estàndard de C dissenyada per a embedded systems o sistemes incrustats. Actualment és mantinguda per desenvolupadors de Red Hat.
OpenGL	OpenGL és una especificació estàndard que defineix una API multilinguatge i multiplataforma per a escriure aplicacions que produeixen gràfics 3D. Desenvolupada originalment per Silicon Graphics Incorporated (SGI), OpenGL significa Open Graphics Library.
Page flipping	Es pot dir que consisteix en la versió “hardware” del double buffering on l'intercanvi de buffers (en aquest cas són a VRAM) es fa instantàniament pel processador gràfic.
PCM	Pulse-code modulation és una representació digital d'un senyal analògic on la magnitud del senyal és mostrejada regularment a una freqüència determinada, i codificada numèricament.
Pixel format	Especifica com es representarà el color d'un píxel en una imatge. Per exemple ABGR 8888 indica un valor de quatre bytes on s'utilitzen 8 bits per a representar el canal alpha, blau, verd i vermell del color d'un píxel.
Polling	En informàtica normalment fa referència a una operació de consulta constant, generalment cap a un dispositiu o perifèric, per a crear una activitat sincrònica sense l'ús d'interrupcions per part d'aquest dispositiu o perifèric. Per exemple per a consultar l'estat del joystick a la PSP.
Profiling	És el procés d'investigació del comportament d'un programa utilitzant informació extreta mentre aquest s'està executant. L'objectiu principal d'aquest procés és determinar les seccions d'un programa que requereixen optimització.
PSP	La PlayStation Portable (PSP) és una videoconsola portàtil de Sony Computer Entertainment. Tercera consola de la línia PlayStation i primera incursió de Sony al mercat de les portàtils, fou presentada oficialment el 13 de maig de 2003 a la fira d'entreteniment electrònic E3. Fou comercialitzada el 12 de desembre de 2004 al Japó, el 24 de març de 2005 a Amèrica del Nord i l'1 de setembre del mateix any a Europa, Austràlia i Nova Zelanda.
PSPSDK	Les sigles de PSP Software Development Kit. És una col·lecció d'eines de programari lliure i llibreries escrites per a la PSP. Inclou també documentació i d'altres recursos que els desenvolupadors poden utilitzar per dissenyar

programari per a la PSP.

RGB	La descripció RGB (de l'anglès <i>Red, Green, Blue</i>) d'un color fa referència a la composició del color en termes de la intensitat dels colors primaris amb els quals es forma: el vermell, el verd i el blau.
Sample	És una veu anglesa que es tradueix com <i>mostra</i> i fa referència al so gravat en qualsevol tipus de suport.
Scrolling	Consisteix en el moviment dels gràfics que conformen l'escenari d'una aplicació gràfica o videojoc a través de la pantalla.
Sprite	En informàtica gràfica, és una imatge o animació normalment bidimensional que està integrada en una escena major i per on es sol moure-hi lliurement i amb freqüència.
Textura	És una imatge utilitzada per a cobrir la superfície d'un objecte virtual, ja sigui tridimensional o bidimensional. L'objectiu principal és donar a aquest objecte virtual una aparença realista.
Thread	En informàtica, un fil d'execució (<i>thread</i> en anglès) és una característica dels sistemes operatius que permet a un procés executar diferents tasques al mateix temps. Aquesta característica dona la possibilitat al programador de dissenyar un programa que executa diferents funcions concurrentment.
Timer	Un temporitzador en català, és un tipus de rellotge especialitzat que compta cap endarrere a partir d'un temps fixat, com un rellotge de sorra. S'utilitzen per al control i la sincronització d'esdeveniments.
Toolchain	En informàtica, és un conjunt de programes informàtics (<i>tools</i>) que s'utilitzen per a crear un determinat producte (normalment un altre programa o sistema informàtic). Els diferents programes es solen usar en seqüència (d'aquí la paraula <i>chain</i> que significa cadena en anglès) de tal manera que la sortida d'un sigui l'entrada del següent. Normalment fa referència a qualsevol conjunt d'eines de desenvolupament com el PSP toolchain que es compon de compilador, assemblador, enllaçador de codi, etc.
Triple buffering	En informàtica gràfica, consisteix en una variant del double buffering. És més ràpid que l'anterior ja que s'intenta minimitzar el temps de sincronització entre la CPU i el monitor fent que la CPU estigui més ocupada.
Vblank	És la diferència de temps entre la darrera línia d'una imatge "dibuixada" per una pantalla o monitor, i l'inici de la següent. En els televisors o monitors CRT, durant aquest temps el canó d'electrons torna a la seva posició inicial.

- Vsync** La sincronització vertical fa referència a la sincronització de canvis de *frames* amb el Vblank. S'utilitza per limitar la producció de *frames* per part de l'aplicació a una velocitat fixa marcada pel monitor fent una transició suau entre aquests.
- XMB** El XMediaBar és una GUI o interfície gràfica d'usuari desenvolupada per Sony Computer Entertainment. És utilitzada, no només a la PSP, també a la Playstation 3 o als nous televisors BRAVIA.