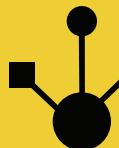




Escuela  
Politécnica  
Superior

# Demo técnica para PC



Grado en Ingeniería Multimedia

## Trabajo Fin de Grado

Autor:

Luis González Aracil

Tutor/es:

Francisco José Gallego Durán

Abril 2019



Universitat d'Alacant  
Universidad de Alicante



# Demo técnica para PC

---

## La demoscene

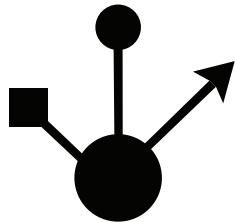
### Autor

Luis González Aracil

### Tutor/es

Francisco José Gallego Durán

*Ciencia de la Computación e Inteligencia Artificial*



Grado en Ingeniería Multimedia



Escuela  
Politécnica  
Superior



Universitat d'Alacant  
Universidad de Alicante

ALICANTE, Abril 2019



# Resumen

Para poder usar una herramienta en toda su capacidad y funcionalidad, es necesario cono-  
cerla a fondo. Con la complejidad creciente de los ordenadores, está apareciendo una tendencia  
por despreocuparse de los detalles del bajo nivel, considerando que los detalles de implemen-  
tación no son o deberían ser parte del problema.

Este trabajo es, en primer lugar, una reivindicación de la importancia del bajo nivel. A lo  
largo del mismo, se tratará la subcultura informática de la *demoscene*.

El *demoscening* se originó a primeros de los años 80, con el auge de los primeros ordena-  
dores personales, y su principal objetivo era el de crear demostraciones técnicas de gráficos  
y sonido por computador en tiempo real, buscando siempre resultados sorprendentes que  
consiguieran sobrepasar las limitaciones técnicas de las máquinas de la época. Eran por tanto  
demostraciones artísticas de ingenio y de conocimiento del bajo nivel.

A lo largo de este trabajo se explorarán algunas de las técnicas gráficas más comúnmente  
usadas en el mundo de la *demoscene*, revisadas desde la actualidad. Se hará un análisis previo  
de cada una de estas técnicas para a continuación proceder a su implementación y posterior  
optimización. Además, para concluir, se hará una propuesta de una demo aplicando todos los  
conocimientos explorados previamente.



# Abstract

In order to be able to use a tool at its maximum capability and functionality, it's important to have an in-depth knowledge of it. Given the always increasing complexity of computers, there's a trend in software development to look away from low level implementation details, considering that these details should not be part of the problem's solution.

This essay pretends to emphasise the importance of the low level. Throughout this document, I will be writing about a computer's subculture called *demoscene*.

The *demoscene* originated back in the 80's, when personal computers' popularity was in a crest. *Demosceners*'s main goal was to create technical graphical demos in real time. These demos always tried to show off the abilities of the programmers, who tried to overcome the technical limitations of the machines at the time, with surprising results.

This document will explore and revise some of the graphical techniques that were commonly used by *demosceners*. Before the implementation of every technique, an in-depth analysis will take place. After it, the demo will be revised and optimized if possible. Furthermore, there will be a last proposal consisting on a final demo gathering all the knowledge previously exposed.



# Motivación y objetivo general

Hablando con un compañero del trabajo sobre el lenguaje ensamblador, yo estaba intentando argumentarle su utilidad, a lo que él me contestó *"Hoy en día, saber ensamblador es como saber latín"*. Su afirmación zanjó el tema, pues a partir de ese momento no tuve ganas de seguir discutiendo, pero me hizo reflexionar. ¿Es inútil el ensamblador? ¿Sirve para algo el bajo nivel?

Para mí, la pregunta *"¿Para qué sirve saber ensamblador?"* es perfectamente equiparable a la pregunta *"¿Para qué le sirve a un arquitecto conocer las herramientas y materiales con los que va a construir una casa?"*.

Si una edificación cayera por una mala elección del material de los cimientos por parte del arquitecto, no habría duda en a quién culpar. Nadie abogaría que la culpa no es del arquitecto porque no es su responsabilidad conocer las bases. Sin embargo, hoy en día hay una enorme tendencia en el mundo del desarrollo software por menospreciar o infravalorar los cimientos de la programación, considerándolo algo arcaico y de carácter puramente didáctico, pero no práctico.

Yo me opongo radicalmente a esta visión, no sólo porque estoy convencido de la importancia de conocer el bajo nivel, si no que también encuentro cierta belleza en él. Cómo instrucciones en apariencia tan simples pueden construir sistemas tan complejos. A ello, se suma una gran curiosidad por saber cómo las cosas están hechas, desde el principio.

Una de las cosas que encuentro más apasionantes de la computación es la capacidad de los ordenadores, máquinas inertes y carentes de inteligencia real -por el momento- para reproducir nuestra realidad a partir de modelos matemáticos.

Los gráficos por computador son, por lo general, complejos. Sin embargo, hoy en día es posible crear con un ordenador imágenes que parecen fotografías y son capaces de engañar al ojo humano.

El objetivo principal de este trabajo es ir a las raíces, y revisar algunas de las técnicas que se usaban en los orígenes de los gráficos por computador para, a partir de operaciones con bajo coste computacional, generar escenas complejas.



*A mis padres, por estar ahí, siempre.*

*A mi hermana, por ser mi recordio y mi alegría.*

*A mi familia y amigos, por apoyarme y alegrarme los días.*

*A mi tutor,*

*por la visión que me ha dado sobre el mundo de la programación  
y que tan valiosa es.*

*A Lola, por haber marcado la dirección cuando estábamos perdidos*



*If you give people  
the choice of writing  
good code or fast code,  
there's something wrong.  
Good code should be fast*

Bjarne Stroustrup

*When the whole world is silent,  
even one voice becomes powerful.*

Malala Yousafzai



# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
<b>2</b>	<b>Estado del arte</b>	<b>3</b>
2.1	La demoscene . . . . .	3
2.1.1	Qué es la demoscene . . . . .	3
2.1.2	Orígenes de la demoscene . . . . .	4
2.1.3	Composición y cultura de la demoscene . . . . .	4
2.1.4	La demoscene en la actualidad . . . . .	5
2.2	Eventos de demoscening . . . . .	6
2.3	Grupos de demoscening . . . . .	7
2.3.1	Farbrausch . . . . .	7
2.3.2	Future Crew . . . . .	9
2.3.3	PoPsY TeAm . . . . .	10
2.3.4	Equinox . . . . .	11
2.3.5	Fairlight . . . . .	11
2.3.6	RGBA . . . . .	12
2.3.7	Batman Group . . . . .	13
2.4	Portales de demoscening . . . . .	14
2.5	Demos destacables . . . . .	14
2.6	Efectos gráficos más comunes . . . . .	14
2.7	Influencia de la demoscene en la industria . . . . .	14
<b>3</b>	<b>Objetivos</b>	<b>17</b>
<b>4</b>	<b>Metodología</b>	<b>19</b>
4.1	Software . . . . .	19
4.2	Tests de rendimiento . . . . .	20
4.3	Entorno: motor gráfico . . . . .	20
4.4	Las demos . . . . .	21
4.4.1	Búsqueda de información . . . . .	21
4.4.2	Planteamiento formal . . . . .	21
4.4.3	Implementación . . . . .	21
4.4.4	Refinamiento . . . . .	21
<b>5</b>	<b>Tests de rendimiento</b>	<b>23</b>
5.1	Implementación . . . . .	23
5.2	Resultados . . . . .	23

<b>6 El motor gráfico</b>	<b>25</b>
6.1 Investigación inicial . . . . .	25
6.2 Características . . . . .	25
6.2.1 Detectar input . . . . .	25
6.2.2 Dibujar texto . . . . .	25
6.2.3 Dibujar líneas . . . . .	25
<b>7 Demos clásicas</b>	<b>27</b>
7.1 Fuego . . . . .	28
7.1.1 Investigación inicial . . . . .	28
7.1.2 Planteamiento formal . . . . .	28
7.1.3 Implementación . . . . .	28
7.1.4 Refinamiento . . . . .	28
7.1.5 Resultado . . . . .	28
7.2 Geometría . . . . .	28
7.2.1 Investigación inicial . . . . .	28
7.2.2 Planteamiento formal . . . . .	28
7.2.3 Implementación . . . . .	28
7.2.4 Refinamiento . . . . .	28
7.2.5 Resultado . . . . .	28
7.3 Planos infinitos . . . . .	28
7.3.1 Investigación inicial . . . . .	28
7.3.2 Planteamiento formal . . . . .	28
7.3.3 Implementación . . . . .	28
7.3.4 Refinamiento . . . . .	28
7.3.5 Resultado . . . . .	28
7.4 Plasma . . . . .	28
7.4.1 Investigación inicial . . . . .	28
7.4.2 Planteamiento formal . . . . .	28
7.4.3 Implementación . . . . .	28
7.4.4 Refinamiento . . . . .	28
7.4.5 Resultado . . . . .	28
7.5 RotoZoom . . . . .	28
7.5.1 Investigación inicial . . . . .	28
7.5.2 Planteamiento formal . . . . .	28
7.5.3 Implementación . . . . .	28
7.5.4 Refinamiento . . . . .	28
7.5.5 Resultado . . . . .	28
7.6 Deformaciones de imagen . . . . .	28
7.6.1 Investigación inicial . . . . .	28
7.6.2 Planteamiento formal . . . . .	28
7.6.3 Implementación . . . . .	28
7.6.4 Refinamiento . . . . .	28
7.6.5 Resultado . . . . .	28

7.7 Túnel de puntos . . . . .	28
7.7.1 Investigación inicial . . . . .	28
7.7.2 Planteamiento formal . . . . .	28
7.7.3 Implementación . . . . .	28
7.7.4 Refinamiento . . . . .	28
7.7.5 Resultado . . . . .	28
<b>8 Demo final</b>	<b>29</b>
8.0.1 Investigación inicial . . . . .	29
8.0.2 Planteamiento formal . . . . .	29
8.0.3 Implementación . . . . .	29
8.0.4 Refinamiento . . . . .	29
8.0.5 Resultado . . . . .	29
<b>9 Conclusiones</b>	<b>31</b>
<b>Bibliografía</b>	<b>33</b>



# Índice de figuras

2.1	Assembly 2004 - Fuente: Wikipedia . . . . .	8
2.2	Farbrausch 41: Debris - Fuente: YouTube . . . . .	8
2.3	Videojuego de 96kB: .kkrieger - Fuente: YouTube . . . . .	9
2.4	Farbrausch 63: Magellan - Fuente: YouTube . . . . .	9
2.5	Second Reality - Fuente: YouTube - En esta captura se puede ver un efecto de reflexión en dos esferas en tiempo real, mediante <i>raytracing</i> . . . . .	10
2.6	VIP2 - Fuente: YouTube . . . . .	11
2.8	Dead Ringer (por FairLight) - Fuente: YouTube - Demo 64k ganadora de Assembly 2006 . . . . .	12
2.9	Elevated - Fuente: YouTube - Intro 4K ganadora en Breakpoint 2009 . . . . .	13
2.10	Batman Forever - Fuente: YouTube . . . . .	13



# **Índice de tablas**



# **Índice de Códigos**



# 1 Introducción

Sin saber muy bien cómo, hemos llegado a un punto en el que conocer las bases del funcionamiento de un computador nos parece algo obsoleto, e incluso arcaico. Tecnologías de hace 20 años se tachan de reliquias en un mundo que aún no cuenta un siglo de antigüedad.

El irrefrenable avance de la tecnología y velocidad de evolución es innegable, pero a menudo, cuando se avanza muy rápido, también se pierde muy rápido.

La abstracción en el mundo de la computación ha sido un factor clave, de hecho es el factor que ha permitido que un set reducido de instrucciones como el que tienen los ordenadores sea capaz de imitar la realidad. Abstraer el software y llevarlo hacia modelos más cercanos al ser humano ha permitido pensar más en términos de nuestro día a día y menos en términos de mover memoria y realizar sumas y restas. Y esto es bueno, si para realizar cualquier mínima tarea tuviéramos que preocuparnos de hasta el más mínimo detalle de implementación, la curva de aprendizaje sería demasiado inclinada, y la eficiencia de la producción del software caería en picado.

Sin embargo, a más nos alejamos del hardware y más capas de abstracción añadimos, las instrucciones que escribimos se alejan más y más del reducido set de instrucciones que nuestro ordenador puede ejecutar. Como dijo una vez David J. Wheeler, *"Todo problema en computación puede resolverse con otra capa de indirección, excepto el problema de tener demasiada indirección"*<sup>1</sup>.

Esta frase, además de tener un punto cómico, plantea un problema más serio del que muchas veces nos damos cuenta. Hoy en día hay aplicaciones construidas dentro de webs. Para ejecutar código en el cliente de una web se usa *JavaScript*, un lenguaje interpretado. Esto significa que para ejecutar una instrucción de código máquina proveniente de *JavaScript* es necesario, primero, interpretar la línea de código, compilarla y traducirla al lenguaje de la máquina virtual de *JavaScript* que integra el navegador y que esta máquina virtual interactúe con el sistema operativo para ejecutar la orden necesaria. Esto obviando una gran cantidad de pasos intermedios e ignorando las propias capas de abstracción de la memoria, el funcionamiento del procesador, los posibles fallos de la caché del procesador... Y si a todo este proceso, ya de por sí complejo y con muchas capas de por medio, añadimos una aplicación web compleja que añade nuevas capas de abstracción sobre el propio código que se ejecuta en *JavaScript*... ¿No parece demasiado?

Y sin embargo hoy en día prácticas como esta son perfectamente aceptadas, e incluso a veces, son punteras en la industria. Se justifica el hecho de tener una gran capacidad de

---

<sup>1</sup>[http://www.stroustrup.com/bs\\_faq.html#really-say-that](http://www.stroustrup.com/bs_faq.html#really-say-that)

cómputo para poder añadir más y más carga computacional. Se habla de las ventajas en la velocidad de producción, o en la sencillez de manejo del alto nivel en comparación del bajo nivel. Y es cierto que en muchos casos se gana eficiencia o productividad, ¿pero y lo que estamos perdiendo a cambio?

Podemos estar reduciendo la velocidad de nuestro programa cientos de veces, y aún así muchas veces no importa, porque la diferencia entre que algo tarde en ejecutarse 0,001s a que tarde 0,1s se nota, pero tampoco importa tanto. Sin embargo, pensamientos como este son peligrosos, y son los que han llevado a que programas aparentemente sencillos y ligeros incluyan tiempos de espera al iniciarlos, tal y como argumenta Mike Acton<sup>2</sup>.

Y lo que es más, cabe preguntarse, ¿de verdad tanta abstracción simplifica el problema?

La realidad es que hay software donde la gran cantidad de capas que lo forman no solo reduce su tiempo de ejecución, si no que aumenta su complejidad de forma innecesaria, hecho que al que se llama popularmente *overengineering*.

De hecho, hoy en día existen hasta aplicaciones gráficas y juegos dentro de la web. Capas de abstracción dentro de capas de abstracción. Pero en aplicaciones tan computacionalmente costosas como aquellas que manejan gráficos y/o modelos matemáticos, toda esta abstracción tiene un coste que pasa factura. Quizá es precisamente por este motivo, que empiezan a surgir iniciativas interesantes, como Wasm<sup>3</sup>.

Los gráficos siempre han requerido grandes capacidades de cómputo, aumentar la resolución de pantalla supone un coste cuadrático. Es por ello, que en el terreno de los gráficos, la simplicidad, la sencillez y la eficiencia priman. En una web, la diferencia entre 10 o 100 fotogramas por segundo puede no ser relevante, pero en una aplicación gráfica, en una reproducción de vídeo o en un videojuego, es un factor clave.

Y sin embargo, a pesar de su altísimo coste computacional, los gráficos por computador nos acompañan desde principios de los años 50, (dónde los ordenadores eran miles de veces menos potentes) en forma de hacks rudimentarios que permitían generar gráficos a partir de ficheros de texto<sup>4</sup>. No obstante, el boom de los gráficos por computador se produciría en los años 80, con la aparición y popularización de los primeros ordenadores personales, así como del videojuego. Es en este marco en el que se originaría la *demoscene*.

El propósito de este estudio es, pues, visitar el arte del *demoscening* e investigar y exponer algunas de las técnicas gráficas que más comúnmente se usaban en los orígenes de la *demoscene*. Pretende ser una vuelta a los orígenes, donde se exploren los gráficos desde una perspectiva actual pero cercana al bajo nivel.

---

<sup>2</sup><https://www.youtube.com/watch?v=rX0ItVEVjHc&t=4620s>

<sup>3</sup><https://webassembly.org>

<sup>4</sup><http://www.catb.org/jargon/html/D/display-hack.html>

## 2 Estado del arte

### 2.1 La demoscene

#### 2.1.1 Qué es la demoscene

La *demoscene* es una subcultura informática cuyo principal objetivo es la creación de demostraciones técnicas llamadas *demoscenes*. Una *demoscene* es un programa autocontenido y por norma general de peso ligero que intenta explotar al máximo el *software* y el *hardware* de la máquina que la ejecuta, con el fin de generar efectos visuales y sonoros. El objetivo de una *demoscene* suele consistir en mostrar el ingenio y las habilidades del programador, así como tratar de impresionar al público.

Además, aunque el *demoscening* en sí mismo no se puede considerar una forma de arte, sí que es cierto que muchas demos poseen un cierto componente artístico.

Se distinguen principalmente dos tipos de demo<sup>1</sup>:

- **Demo:** programa que genera gráficos y sonido en tiempo real. Suele tener una extensión superior a 5 minutos y normalmente no tienen límite de tamaño. Una demo suele ser creada por un grupo de personas que incluye al menos un programador, un diseñador gráfico y un músico. Las demos actuales suelen estar realizadas en 3D y cuentan con aceleración gráfica por hardware. Las demos más antiguas o realizadas para plataformas más antiguas (conocidas como "demos *oldskool*") son procesadas de forma íntegra por la CPU (pues las plataformas para las que se desarrollan no poseen GPU) y suelen combinar ilusiones 3D con efectos gráficos en 2D.
- **Intro:** una demo de corta duración. Una intro suele ser temática (mientras que una demo suele compilar distintas escenas/temáticas). Además, las intros no suelen superar los 5 minutos de duración y su tamaño tiende a estar restringido. Las principales categorías de intros son 64K (65536 bytes), 4K (4096 bytes) y 1K (1024 bytes).

Existen otras categorías de demo, aunque son mucho menos comunes, como las **mega demos** (demos de gran duración/extensión, compuestas por múltiples partes) o las **dentros** (intros cuyo propósito es ofrecer un avance de una demo por llegar, todavía en desarrollo).

Además, existen muchas otras categorías derivadas o relacionadas con la *demoscene*, como la creación de gráficos procedimentales. Una de las subcategorías más populares dentro de esta son las **4K images**, imágenes complejas y de alta resolución generadas proceduralmente por programas de 4096 bytes. También es posible encontrar categorías similares relacionadas

---

<sup>1</sup>[http://www.oldskool.org/demos/explained/demo\\_reference.html](http://www.oldskool.org/demos/explained/demo_reference.html)

con la generación procedural de archivos de música o vídeo. Las mayores diferencias entre estas producciones y las demos son su tiempo de ejecución (no se ejecutan en tiempo real) y las técnicas que usan (al no ser el tiempo una limitación, pueden usar algoritmos computacionalmente más costosos, pero que generan resultados más complejos).

### 2.1.2 Orígenes de la demoscene

A principios de los años 80, con la popularización de los primeros ordenadores personales, la computación dejó de ser algo que sucedía en universidades para pasar a abrirse al gran mercado. Con ello, llegó también la distribución del software, aunque en aquella época no se producía por internet, si no tan sólo por medios físicos, como los disquetes. Estos programas venían con protecciones de copia por parte de los desarrolladores para evitar su distribución ilegal. Poco tardaron, no obstante, en aparecer los primeros *crackers*, personas que se dedicaban a eliminar las protecciones de copia del software para su distribución gratuita. Esto llevó a la creación de una subcultura informática basada en el *cracking* de videojuegos y otros tipos de software, al margen de la legalidad. Esto se hacía no solo con la intención de poder distribuir el software de forma gratuita, si no que también suponía una fuente de diversión y competición para los *crackers*<sup>2</sup>.

Es por ello que los denominados *crackers* empezaron a "firmar" el software que *crackeaban* con seudónimos que aparecían en los menús o en las intros de los juegos. Con el tiempo, la competición y la ambición de los *crackers* fue aumentando, y llegó un punto en el que no solo se limitaban a quitar las protecciones de copia del software, si no que también creaban sus propias intros para los programas.

Es en este punto cuando la *demoscene* empieza a tomar forma, cuando una parte de los *crackers* deciden retornar a la legalidad pero sin dejar atrás la competición y la diversión. De este modo, este nuevo sector se empieza a dedicar a la creación de intros y demos cuyo objetivo es mostrar sus habilidades al resto de *demosceners*<sup>3</sup>.

### 2.1.3 Composición y cultura de la demoscene

La *demoscene* es una subcultura informática muy centrada en el trabajo en equipo y en compartir. Con el desarrollo y popularización de la *demoscene*, a partir de principios de los 90 se popularizó y se estandarizó, con la creación de eventos y competiciones.

Heredado de las *copyparties*, eventos en los que *crackers* y *demosceners* se juntaban para conocerse y compartir software, al margen de la legalidad, nuevos eventos empezaron a crearse a principios de los noventa. Estos eventos sí eran legales y se centraban únicamente en el aspecto de las demos. Pasan a ser eventos sociales en los que los *demosceners* se conocen, comparten y compiten.

Estos eventos, conocidos como *demoparties*, pasan entonces a ser concursos y tener distintas categorías y premios. Para concursar, normalmente los competidores se juntaban en grupos,

<sup>2</sup><https://web.archive.org/web/20170726063815/http://tomaes.32x.de/text/faq.php#2.3>

<sup>3</sup><http://widescreen.fi/assets/reunanen-wider-1-2-2014.pdf>

---

usualmente compuestos por al menos un programador, un diseñador gráfico y un músico. Estos grupos tenían su propio nombre e identidad. A su vez, cada uno de los componentes del grupo también solía usar un seudónimo. El uso de un seudónimo es una herencia de los orígenes de la *demoscene* en el *cracking*, aunque el propósito de usar un alias cambia. Mientras que los *crackers* usaban un nombre falso para ocultar su identidad, pues realizaban actividades ilegales, los *demosceners* usan este alias como una forma de expresión.

#### 2.1.4 La demoscene en la actualidad

Si bien la *demoscene* siempre se ha mantenido como una subcultura y nunca ha llegado a tener una popularidad masiva, su auge se dio en los años noventa. En la actualidad, muchos de los eventos de *demoscening* que se crearon en los 90 han desaparecido, y otros tantos han derivado en eventos dedicados a los ordenadores de forma mucho más genérica, derivando en eventos de software o en *LAN-parties*.

Del mismo modo que muchas ferias y eventos han desaparecido, muchos otros también se han ido creando. No obstante, parece que hay una tendencia general hacia el olvido.

Las *demoparties* en la actualidad suelen ser eventos locales, normalmente humildes, donde participan apasionados y nostálgicos.

Es difícil atribuir las causas de la lenta caída en popularidad de la *demoscene*, aunque hay varios factores que pueden contribuir a ello, del mismo modo que hay una serie de factores que evitan que se pierda.

Por un lado, la *demoscene* es cada vez más pequeña debido a:

- Siempre ha sido una subcultura, y nunca ha destacado especialmente por encima de otras subculturas informáticas.
- Para poder participar en la *demoscene* hace falta una gran cantidad de conocimiento matemático y de programación de bajo nivel, cualidades que no abundan.
- Con el auge de los ordenadores, otro tipo de espacios más accesibles al público como las ferias tecnológicas, las ferias de videojuegos o las ferias de programación han eclipsado parcialmente a la *demoscene*.
- Una parte de los *demosceners* eran reacios a la incorporación de gente nueva e inexperta a la *demoscene*, pues si bien aportaban sangre nueva, sus metas y conocimiento se alejaban de los originales de la escena.

Todos estos factores han contribuido a colocar la *demoscene* como una práctica de nicho. Quizá una crítica válida sería que no ha sabido adaptarse de forma conveniente a los nuevos tiempos. El mundo de la *demoscene* no ha sabido publicitarse o venderse suficientemente bien como para llegar a ser conocido por un público mayor. Sin embargo, la *demoscene* sigue viva, y estos son algunos de los factores que contribuyen a ello:

---

- El auge de lo *retro*. En esta última década ha empezado a masivizarse un cierto reconocimiento y nostalgia hacia los orígenes de la computación y del videojuego. La popularización de los juegos *indie*, técnicas como el *pixel art* y tributos a videojuegos antiguos han llevado a destapar obras olvidadas y a suscitar un nuevo interés por todo lo *retro*.
- La pasión y dedicación de los *demosceners*, que siguen produciendo y compartiendo su obra, extendiendo así su pasión y abriéndola al público general.
- La masivización de la informática. Hoy en día hay muchísimos más informáticos que en los años 80 y 90. Si bien es cierto que hay una tendencia de abandono hacia el bajo nivel, también hay mucha más gente que se hace preguntas, investiga y se interesa por el mismo.

Como reflexión final me gustaría añadir que creo que es posible mantener el panorama de la *demoscene* vivo, pero pienso que esto va a ser muy complicado si no se intenta abrir al público, cosa que algunas *demoparties* ya están haciendo. Está claro que al abrir la *demoscene* al público general se pierden cosas por el camino, y se gana gente que tiene un interés mucho más casual y mucho menos pasional. Creo que en el panorama actual esa gente es necesaria. No todo el mundo tiene el tiempo o la capacidad como para meterse de lleno en la *demoscene*, pero hay muchas personas que sí pueden tener curiosidad por ella de una forma mucho más básica, y esta gente también cuenta. Cada vez es más común encontrar en ferias de videojuegos puestos *retro*, y hay incluso museos del videojuego. La nostalgia y la veneración por los orígenes se abre paso, y pienso que es en este lugar donde la *demoscene* puede buscar su camino.

## 2.2 Eventos de demoscening

A continuación se listan y describen brevemente algunas *demoparties* que aún se celebran en la actualidad, en el año 2019.

- **Revision:** tiene lugar en durante Pascua, en Alemania. Es la sucesora de la *demoparty* Breakpoint<sup>4</sup>. El evento se estableció en 2011, tras el fin de Breakpoint, y mantiene a muchos de los organizadores. El evento congrega a más de 800 personas de todo el mundo cada año, y es el mayor evento dedicado exclusivamente a la *demoscene* en el mundo<sup>5</sup>.
- **Assembly [2.1]:** tiene lugar en verano en Finlandia, y es el mayor evento de *demoscening* en el país. Además, es uno de las *demoparties* más antiguas que siguen en activo, cumpliendo 25 años el pasado 2017. Sin embargo, el evento no está dedicado exclusivamente a la *demoscene*, si no que es también un evento de videojuegos y *esports*. No obstante, la *demoscene* sigue siendo importante en él, y se celebran anualmente concursos en 5 categorías distintas (Demo, Oldskool demo, 64K intro, 4K intro y 1K intro)<sup>6</sup>.

---

<sup>4</sup><http://breakpoint.untergrund.net>

<sup>5</sup><https://2019.revision-party.net>

<sup>6</sup><https://www.assembly.org/summer19/demoscene>

- **VIP (Very Important Party)**: es la *demoparty* más longeva de Francia, cumpliendo este año su 20 aniversario. Congrega a personas de todas partes de Europa, aunque es una *demoparty* principalmente francesa<sup>7</sup>. Fue fundada por el grupo de *demosceners* PoPsY TeAm<sup>8</sup>.
- **Nova**: esta *demoparty* fue creada en 2017, por lo que este año se celebra su tercera edición. Es la mayor *demoparty* en Reino Unido, con una salón principal con capacidad para hasta cien *demosceners*<sup>9</sup>.
- **Alternative Party**: es uno de los eventos de *demoscening* más grandes de Finlandia. Es un festival bastante peculiar, (o alternativo, como su nombre indica) y su objetivo es motivar a los programadores y artistas a explorar su creatividad y nuevos puntos de vista. Suele mezclar ordenadores de distintas épocas y capacidades. Aunque mantiene una categoría constante a la mejor demo, el resto de categorías y premios cambian cada año, suponiendo así un reto aún mayor para los programadores. Ha tenido distintos invitados célebres dentro del mundo de la informática, como Al Lowe, creador del juego *Leisure Suit Larry*. Además, la primera noche del evento incluye un concierto de música en el que participan artistas del mundo de la *demoscene*. Este año se celebra su 20 aniversario<sup>10</sup>.
- **Chaos Constructions**: es el festival de *demoscenes* más longevo y popular que se celebra en Rusia. Tiene lugar en verano y se creó en 1995, aunque en aquel momento se llamada *Enlight*. La celebración de esta *demoparty* en el año 1997 congregó a más de 1200 personas. En la actualidad, su temática se ha ampliado, incluyendo nuevas áreas como exposiciones y charlas empresariales. En 2018, el festival tuvo lugar durante dos días sin interrupción, contó con participantes internacionales y se emitió de forma íntegra por *Twicht*<sup>11</sup>

## 2.3 Grupos de demoscening

A continuación se listan y describen brevemente algunos de los grupos de *demosceners* más populares.

### 2.3.1 Farbrausch

Farbraush es un grupo de *demosceners* de origen alemán que empezó a ser notado a partir de diciembre del 2000, con su octava producción, llamada *fr-08: .the .product*<sup>12</sup>.

El nombre del grupo se puede traducir como "éxtasis de color". Todos sus proyectos empiezan por "fr-número\_del\_proyecto", donde el número del proyecto se decide en el momento de

<sup>7</sup>[https://en.wikipedia.org/wiki/Very\\_Important\\_Party](https://en.wikipedia.org/wiki/Very_Important_Party)

<sup>8</sup><http://www.popsyteam.org>

<sup>9</sup><http://www.novaparty.org>

<sup>10</sup><http://www.altparty.org>

<sup>11</sup><https://chaosconstructions.ru/en/>

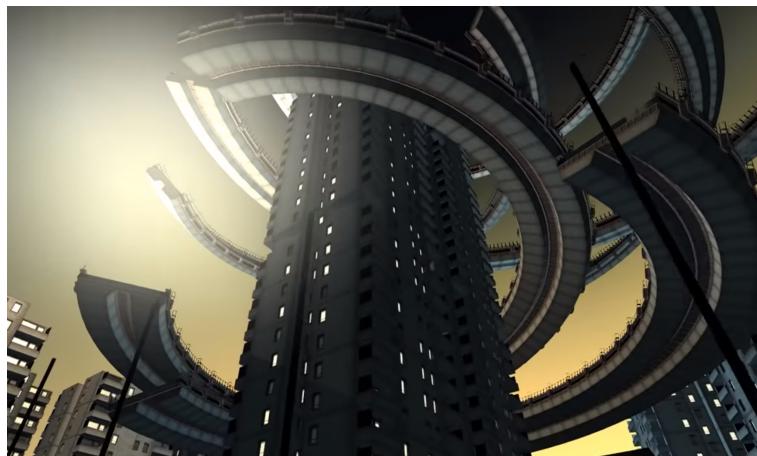
<sup>12</sup><http://www.farbrausch.de/prod&which=17.py>



**Figura 2.1:** Assembly 2004 - Fuente: Wikipedia

empezar a trabajar en el mismo, independientemente de cuándo se produzca su lanzamiento.

Farbraush tiene un gran cantidad de demos notorias, como Debris [2.2], que está considerada en el popular portal *demoscener* <http://www.pouet.net> como la mejor demo de todos los tiempos.



**Figura 2.2:** Farbrausch 41: Debris - Fuente: YouTube

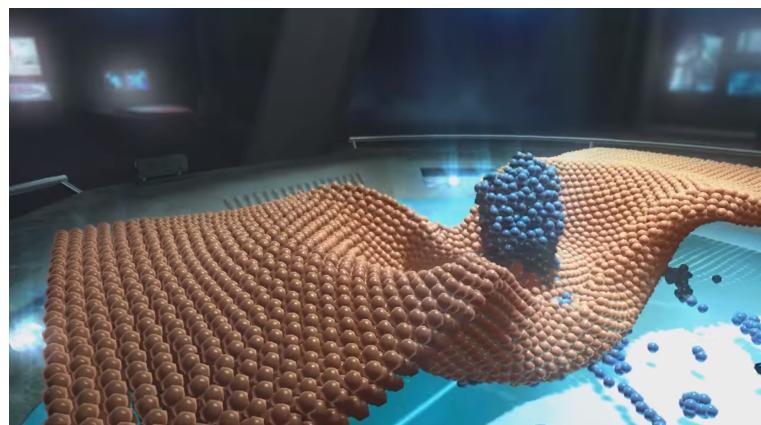
Además, en el año 2004 un subgrupo dentro de Farbrausch, denominado *.theprodukkt*, lanzó *.kkrieger*, un juego de disparos en primera persona que ocupaba tan sólo 96kB. Este pequeño tamaño se consiguió mediante el uso de generación procedural para las texturas y el uso de formas básicas (cubos, esferas...) combinados y deformados para los modelos. El juego [2.3] recibió distintos premios y fue alabado por la comunidad.

---



**Figura 2.3:** Videojuego de 96kB: .kkrieger - Fuente: YouTube

El grupo sigue en activo y siguen produciendo obras de gran calidad, contando con más de diez productos que han recibido primeros premios en distintas competiciones. En general, sus demos tienden a proponer una temática bastante urbana o robótica, con un cierto aire post-apocalíptico. Sin embargo, su capacidad, imaginación y variedad de contenido [2.4] nunca deja de sorprender.



**Figura 2.4:** Farbrausch 63: Magellan - Fuente: YouTube

### 2.3.2 Future Crew

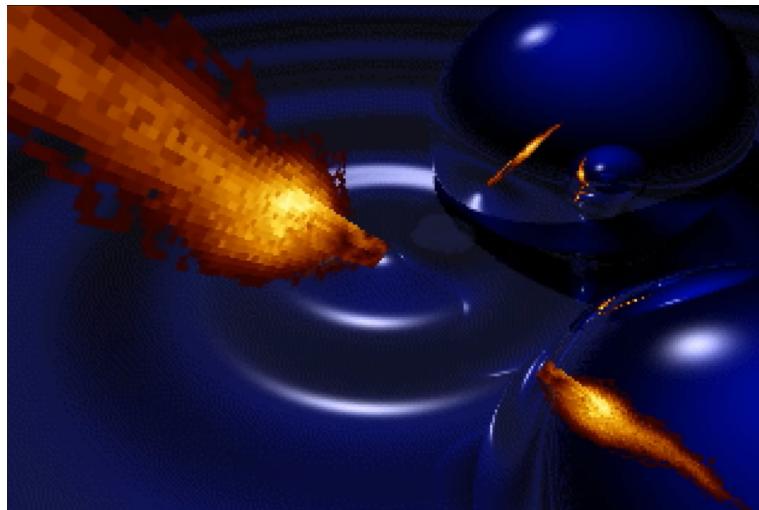
Future Crew<sup>13</sup> fue un grupo de *demosceners* finés, activo principalmente entre 1987 y 1994. Su obra y legado son ampliamente conocidos en el mundo de la *demoscene*. El grupo empezó creando demos para Commodore 64, aunque no tardó en pasar a PC.

Su trabajo es especialmente conocido no sólo por su calidad, si no también porque consiguieron resultados que en aquella época parecían imposibles. Su demo, Second Reality [2.5],

---

<sup>13</sup>[https://en.wikipedia.org/wiki/Future\\_Crew](https://en.wikipedia.org/wiki/Future_Crew)

publicada en julio de 1993, se considera una de las demos más influyentes en la historia de la *demoscene*. Además, el grupo fue coorganizador de la primera edición de la *demoparty Assembly*.



**Figura 2.5:** Second Reality - Fuente: YouTube - En esta captura se puede ver un efecto de reflexión en dos esferas en tiempo real, mediante *raytracing*

Si bien no se produjo una disolución oficial, el grupo se fue deshaciendo paulatinamente hacia la segunda parte de los 90. La mayoría de sus miembros pasaron a la industria del videojuego o de los gráficos por computador, muchos de ellos fundando sus propios estudios, con resultados exitosos.

### 2.3.3 PoPsY TeAm

PoPsY TeAm<sup>14</sup> es un grupo de *demosceners* franceses fundado en Lyon, en julio de 1996. Empezaron produciendo demos para Atari y posteriormente para PC.

Son los creadores y promotores de VIP (Very Important Party), la *demoparty* más relevante de Francia. Además, PoPsY TeAm se estableció en 2001 como una asociación legalmente registrada en Francia.

Su demo más conocida es VIP2 [2.6], una demo que presentaron en la *demoparty* holandesa TakeOver en el 2000, resultando ganadora. El objetivo de esta demo era también el de promover su propia *demoparty*, VIP. Esta demo, además, fue la primera de PoPsY TeAm en usar aceleración gráfica por hardware.

El grupo siempre ha intentado promover la *demoscene* y entre otras cosas, han llegado a organizar viajes en bus a diversas partes de Europa para hacer posible al resto de *demosceners*

---

<sup>14</sup><http://www.popsyteam.org>



**Figura 2.6:** VIP2 - Fuente: YouTube

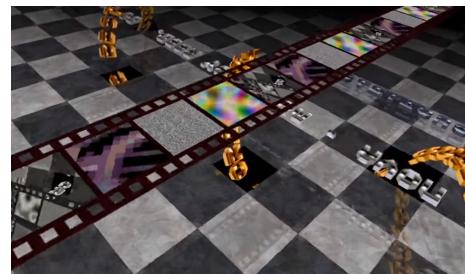
de la región de Lyon atender a *demoparties* europeas. Además, diversos miembros del equipo han participado en la industria del videojuego.

#### 2.3.4 Equinox

Equinox<sup>15</sup> es un grupo de *demosceners* francés que estuvo principalmente activo entre 1988 y 2007, siendo principalmente conocido por sus demos para Atari ST, aunque sus últimas demos, publicadas pasado el cambio de milenio, fueron lanzadas para PC.



(a) Pupul intro (1989) - Fuente : YouTube



(b) Sota 2004 invitation intro (64k intro) -  
Fuente: YouTube

#### 2.3.5 Fairlight

FairLight<sup>16</sup> es un grupo de *demosceners* de origen sueco, formado en 1987. FairLight empezó creando demos para Commodore, aunque ha creado también demos para Amiga, SNES y posteriormente en PC.

FairLight fue fundado en 1987 por dos *crackers* suecos, ex-miembros de un grupo llamado "West Coast Crackers". De hecho, FairLight no solo se dedicaba a la *demoscene*, si no tam-

<sup>15</sup> <https://equinox.planet-d.net>

<sup>16</sup> <http://www.fairlight.to>

bién al mundo del *cracking*, rompiendo juegos para su lanzamiento gratuito de forma ilegal. De hecho, llegaron a hacerse especialmente conocidos por la velocidad a la que eran capaces de lanzar juegos *crackeados*<sup>17</sup>. Tal fue su impacto que en Abril de 2004, varios miembros del grupo fueron tomados por el FBI en una operación antipiratería denominada *Operation FastLink*. Más de 120 personas fueron arrestadas en esta operación, en la que se consideraba a los *crackers* como una organización criminal.



**Figura 2.8:** Dead Ringer (por FairLight) - Fuente: YouTube - Demo 64k ganadora de Assembly 2006

Este es, quizás, un grupo en el que se reflejan y mezclan los orígenes de la *demoscene*, provenientes del mundo del *cracking*. A pesar de todo, el grupo volvió a estar en activo a partir de octubre de 2006.

### 2.3.6 RGBA

RGBA<sup>18</sup> es un grupo español de *demosceners* que estuvo activo entre 2001 y 2009. Todas sus producciones fueron lanzadas para PC, principalmente en Windows.

Son especialmente conocidos por su demo Elevated[2.9]. Esta demo, realizada en colaboración con TBC<sup>19</sup>, es especialmente conocida y celebrada por la comunidad *demoscener*, situándose como la segunda más popular en el portal <http://www.pouet.net>.

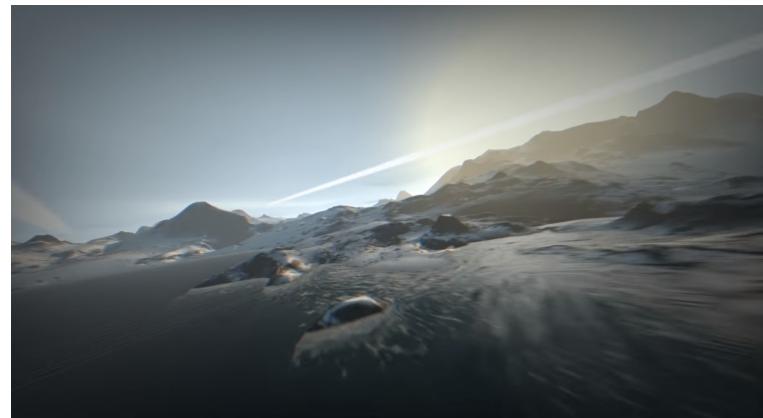
Los binarios de todas sus producciones se pueden encontrar en GitHub.

---

<sup>17</sup><https://computersweden.idg.se/2.2683/1.444716/we-might-be-old-but-were-still-the-elite>

<sup>18</sup><http://www.rgbaproject.org>

<sup>19</sup><https://demozoo.org/groups/641/>



**Figura 2.9:** Elevated - Fuente: YouTube - Intro 4K ganadora en Breakpoint 2009

### 2.3.7 Batman Group

Batman Group<sup>20</sup> es un modesto grupo de *demosceners* de origen español, activo desde 1993. Han producido juegos y demos para Amstrad CPC, ZX Spectrum, Amiga, Android e iOS.

Su demo más conocida es Batman Forever[2.10], para Amstrad CPC, lanzada en 2011 para la *demoparty* eslovaca Forever, quedando en primera posición.



**Figura 2.10:** Batman Forever - Fuente: YouTube

---

<sup>20</sup><https://demozoo.org/groups/18871/>

El grupo se vio envuelto en una polémica a finales de 2018, debido a que el grupo de desarrolladores retro *4MHz* había usado para sus producciones código cedido por Batman Group sin su correcta atribución. La polémica se resolvió con la disolución de todo tipo de relación entre Batman Group y *4MHz*<sup>21</sup>.

## 2.4 Portales de demoscening

Hablar de pouet específicamente y luego del alguna más

## 2.5 Demos destacables

juntar y hablar de las principales demos en la demoscene, hacer referencia al top de pouet, incluir elevated y otras mas e intentar explicarlas

## 2.6 Efectos gráficos más comunes

comentar algunos de los efectos de la demoscene más comunes y cómo se conseguían (comentar cómo se conseguían sólo por encima)

## 2.7 Influencia de la demoscene en la industria

La *demoscene* siempre se ha mantenido de forma discreta. Algunas de las razones de que esto sea así se han listado anteriormente, como el hecho de que hace una gran cantidad de conocimiento y pasión para poder participar de forma activa en ella. Sin embargo, esto no ha impedido dejar su huella en la industria informática, especialmente en la del videojuego.

La lista de personalidades que vienen del mundo de la *demoscene* o se han visto influidos por ella es extensa<sup>22</sup>. A continuación se listan algunas de las más destacables:

- **DICE**<sup>23</sup>: La compañía *Digital Illusions*, conocida por juegos como varios de los títulos de la saga *Battlefield* o *Mirror's Edge Catalyst*, cuenta con una gran plantilla proveniente de la *demoscene*, entre los que podemos contar miembros de FairLight.
- **Remedy**<sup>24</sup>: Esta compañía es especialmente conocida por la saga Max Payne. Fue cofundada por dos miembros de Future Crew. Además, esta compañía mantenía una estrecha relación con Futuremark, creadores de 3DMark, un software de pruebas de rendimiento (*benchmarks*). Esta última compañía también poseía una gran cantidad de miembros provenientes de la *demoscene*, contando con varios de Future Crew.
- **Starbreeze, Ascaron, 49Games, Techland, Lionhead Studios, Guerrilla Games**: Todas estas compañías también cuentan o han contado con miembros de la *demoscene*.

<sup>21</sup><http://www.amstrad.es/forum/viewtopic.php?t=5247>

<sup>22</sup><https://chipflip.wordpress.com/2015/06/12/famous-people-who-came-from-the-demoscene/>

<sup>23</sup><http://www.dice.se>

<sup>24</sup><https://www.remedygames.com/>

- Will Wright, creador del videojuego Spore, afirma que la *demoscene* fue una gran influencia para el juego, debido a que este está fundamentalmente basado en la generación procedural de contenido<sup>25</sup>.
- John Carmack, *lead programmer* de videojuegos como Wolfenstein 3D, Doom y Quake, afirmó en la QuakeCon de 2011 que tiene en alta consideración a aquellos programadores que desarrollan demos de 64K, pues tienen que hacer frente a grandes limitaciones y se obtiene mucho conocimiento en el proceso<sup>26</sup>.
- Jaakko Iisalo, principal diseñador de Angry Birds, fue un *demoscener* activo y reconocido durante los 90.

Además, hay algunas otras subculturas informáticas que están estrechamente relacionadas con la *demoscene* o derivan de la misma, como por ejemplo la música por *tracker* (hay toda una comunidad de músicos que crean producciones a través del uso de *trackers*, software para la producción de música).

---

<sup>25</sup> <http://www.gamespy.com/articles/595/595975p1.html>

<sup>26</sup> [https://www.youtube.com/watch?v=4zgYG-\\_ha28#t=4827s](https://www.youtube.com/watch?v=4zgYG-_ha28#t=4827s)

---



## 3 Objetivos

A continuación se listan los objetivos principales del siguiente trabajo:

- **Entender la *demoscene*:** entender y exponer la subcultura informática de la *demoscene*, sus orígenes y motivación, los rasgos culturales compartidos por sus participantes, sus principales grupos y eventos, así como su legado.
- **Entender la importancia de conocer el bajo nivel:** qué es el bajo nivel y su relevancia en el contexto actual. Cómo influye el conocimiento del bajo nivel y del hardware en el rendimiento de un programa informático.
- **Crear pruebas de rendimiento:** realización de pruebas con el fin de conocer qué operaciones son más costosas de realizar en un computador y por qué. Exponer posibles mejoras y optimizaciones en el código para mejorar el rendimiento de un programa.
- **Entender e implementar los efectos más usados por la *demoscene*:** recopilar las técnicas gráficas más comúnmente usadas en el origen de la *demoscene*. Ofrecer una comprensión profunda y detallada de las mismas, desde un punto de vista tanto teórico como práctico.
- **Crear una breve *demoscene* original:** entender y compilar todo el conocimiento aprendido en una sola demo, que combine todos los efectos y mejoras de rendimiento estudiadas.



# 4 Metodología

## 4.1 Software

- **Toggle<sup>1</sup>**: Se utilizará la herramienta online Toggl para contabilizar el tiempo dedicado a cada parte del proyecto. Esto permitirá poder analizar qué partes del trabajo han requerido más dedicación y por qué, ayudando a completar el estudio.
- **Git<sup>2</sup>**: Se utilizará Git para el control de versiones. El uso de Git nos permitirá, además, tener un registro detallado de la evolución del código. El código para este proyecto se aloja en GitHub<sup>3</sup>.
- **Make<sup>4</sup>**: El código de este proyecto será compilable tanto en las plataformas Windows como Linux. Para ello, el uso de la herramienta Make facilitará la compilación de los proyectos así como su portabilidad.
- **MinGW<sup>5</sup>**: MinGW es un entorno de desarrollo para Windows que ofrece un entorno similar al de GNU. Se usará para compilar tanto el código como las librerías en Windows, haciendo la portabilidad más consistente y sencilla.
- **GCC<sup>6</sup>**: GCC es una colección de compiladores con soporte para C++. Se usará para compilar el código de este proyecto, tanto en Windows (a través de MinGW) como en Linux (de forma nativa).
- **GLFW<sup>7</sup>**: GLFW es una librería multiplataforma para OpenGL que facilita los procesos de creación de ventana, generación de contexto y manejo de input.
- **OpenGL<sup>8</sup>**: OpenGL es una extendida librería para la creación y manipulación de gráficos bidimensionales y tridimensionales. En este proyecto, sin embargo, su uso será mínimo y restringido. Se utilizará tan solo para dibujar una textura en nuestra ventana. Será mediante la manipulación de esta textura que generaremos gráficos. OpenGL, por tanto, será un mero mediador, redibujando constantemente la misma textura en pantalla.

---

<sup>1</sup><https://toggl.com/>

<sup>2</sup><https://git-scm.com>

<sup>3</sup><https://github.com/donluispanis/TFG>

<sup>4</sup><https://www.gnu.org/software/make/>

<sup>5</sup><http://www.mingw.org>

<sup>6</sup><https://gcc.gnu.org>

<sup>7</sup><https://www.glfw.org>

<sup>8</sup><https://www.opengl.org>

- **Valgrind**<sup>9</sup>: Valgrind es una herramienta de depuración y perfilado. Se utilizará para realizar pruebas de rendimiento y para comprobar el correcto funcionamiento del programa.
- **Compiler Explorer**<sup>10</sup>: Compiler Explorer es una herramienta online que permite ver la salida en ensamblador del código escrito de forma instantánea. Resultará muy útil para analizar y entender mejor el código que se ejecuta.

## 4.2 Tests de rendimiento

Se pretende recopilar una serie de resultados cuantificables que muestren el coste de distintos tipos de operaciones computacionales, tanto a nivel de coste temporal como espacial (y cantidad de instrucciones).

Se realizará un análisis exhaustivo de los resultados obtenidos, exponiéndolos y razonando acerca de los mismos.

Para realizar estas pruebas, se procederá a la ejecución de pequeños programas que contengan pruebas concretas. Las pruebas que se propone realizar son: operaciones matemáticas con números enteros, operaciones matemáticas con números en coma flotante, coste de la reserva y liberación de memoria, coste del acceso a memoria y coste de los bucles y las operaciones condicionales.

Tras analizar los resultados, se elaborarán una serie de directrices para escribir código que sea generalmente más rápido y/o más fácilmente optimizable por el compilador. Para poder analizar correctamente los resultados obtenidos, se tendrá en cuenta el código ensamblador generado por el compilador.

## 4.3 Entorno: motor gráfico

El objetivo final de este proyecto es recopilar e implementar una serie de efectos gráficos. Sin embargo, para poder realizar esta tarea, es necesario disponer de un entorno que nos permita realizar labores básicas, como gestión de la ventana o de input.

Mediante la creación de un entorno se asegura un flujo de trabajo consistente entre todas las demos, así como la reutilización de código. Será necesaria, por tanto, la creación de un motor gráfico. Este motor deberá de ser lo más sencillo posible, pues debe limitarse a facilitar tareas básicas, pero no debe ofrecer soluciones a problemas complejos o específicos a una demo. Este motor debe ser conciso y ligero, pues debe influir lo mínimo posible en el rendimiento de la demo.

---

<sup>9</sup><http://valgrind.org>

<sup>10</sup><https://godbolt.org>

Este motor debe darnos soporte para: manejo de la ventana, acceso a un espacio de memoria donde sea posible manipular gráficos, manejo de entradas de teclado, dibujado básico en ventana (puntos, líneas, áreas rectangulares y texto)

## 4.4 Las demos

Para afrontar cada una de las demos, se seguirá un procedimiento común que se expone a continuación.

### 4.4.1 Búsqueda de información

En una primera fase, se procederá a la búsqueda de documentación e información sobre la demo. Esto incluye tanto vídeos como imágenes, además de tutoriales o explicaciones teóricas. En esta fase, se plantea recopilar tanta información acerca de la demo como sea posible, y lograr un modelo teórico básico acerca de cómo debería funcionar.

### 4.4.2 Planteamiento formal

Una vez se ha recopilado información sobre la demo a estudio y se posee un conocimiento suficiente sobre la misma, se procede a un planteamiento formal, previo a su implementación. Este planteamiento incluye entender en profundidad la base matemática de la demo, si la hay. Se debe realizar un análisis y explorar distintos puntos de vista desde los que se podría implementar la demo.

### 4.4.3 Implementación

Analizada la demo, y se habiendo razonado sobre el mejor modo de desarrollarla, se procede a la fase de implementación en código de la demo. Esta es una fase experimental y que permite flexibilidad. Es posible que sea necesario probar distintos acercamientos, buscando el que más se adecúe al resultado que se busca y que ha sido definido en el planteamiento formal de la demo.

### 4.4.4 Refinamiento

Cuando la demo ya está completada a nivel de funcionalidad, se procede a su refinamiento y refactorización. En esta fase se busca hacer el código más legible (nombres de variables y funciones explícitos, correcta documentación del código...) así como hacer el código más eficiente (identificar los factores críticos del programa y buscar e implementar soluciones más eficientes).

---



## **5 Tests de rendimiento**

### **5.1 Implementación**

### **5.2 Resultados**



# **6 El motor gráfico**

## **6.1 Investigación inicial**

## **6.2 Características**

### **6.2.1 Detectar input**

### **6.2.2 Dibujar texto**

### **6.2.3 Dibujar líneas**





## 7 Demos clásicas

### 7.1 Fuego

- 7.1.1 Investigación inicial
- 7.1.2 Planteamiento formal
- 7.1.3 Implementación
- 7.1.4 Refinamiento
- 7.1.5 Resultado

### 7.2 Geometría

- 7.2.1 Investigación inicial
- 7.2.2 Planteamiento formal
- 7.2.3 Implementación
- 7.2.4 Refinamiento
- 7.2.5 Resultado

### 7.3 Planos infinitos

- 7.3.1 Investigación inicial
- 7.3.2 Planteamiento formal
- 7.3.3 Implementación
- 7.3.4 Refinamiento
- 7.3.5 Resultado

### 7.4 Plasma

- 7.4.1 Investigación inicial
- 7.4.2 Planteamiento formal
- 7.4.3 Implementación
- 7.4.4 Refinamiento
- 7.4.5 Resultado

### 7.5 RotoZoom

- 7.5.1 Investigación inicial
  - 7.5.2 Planteamiento formal
  - 7.5.3 Implementación
  - 7.5.4 Refinamiento
  - 7.5.5 Resultado
- 

### 7.6 Deformaciones de imagen

## **8 Demo final**

- 8.0.1 Investigación inicial**
- 8.0.2 Planteamiento formal**
- 8.0.3 Implementación**
- 8.0.4 Refinamiento**
- 8.0.5 Resultado**



## 9 Conclusiones

El bajo nivel es importante, la demoscene no debe perderse, las matemáticas son fundamentales, avanzar hacia el futuro teniendo muy presente el pasado, a veces para avanzar hay que mirar atrás, entender lo que hicieron los que iban por detrás de nosotros y saber aplicarlo

A test for bibliography with XeLATEX.[1]



## Bibliografía

M. Alfonso, B. Bernardo, C. Carlos, and D. Domingo. El problema de los gatos y los perros. *Mascotas*, 50:112–115, 2010.

M. Alfonso, M. Marta, and N. Nuria. Mi viaje a EEUU. *Revista de viajes*, 14:50–56, 2010.