

Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

|  |
| --- |
| **Trabajo Fin de Grado** |
| **Herramienta docente para la prueba de Shaders glsl** |

|  |
| --- |
| **Alumno** |
| **Antonio Jesús Viso Sánchez** |
| **Tutores** |
| FRANCISCO DE ASÍS CONDE RODRÍGUEZ |
| (Departamento de Informática) |
| JOSE NEGRILLO CARDENAS |
| (Departamento de Informática) |

**07, 2019**



Don FRANCISCO DE ASÍS CONDE RODRÍGUEZ y Don JOSE NEGRILLO CARDENAS, tutores del Trabajo Fin de Grado titulado: ‘**Herramienta docente para la prueba de Shaders glsl**’, que presenta Don Antonio Jesús Viso Sánchez, otorgan el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, [mes] de 2019

|  |  |
| --- | --- |
| El alumno: | Los tutores: |
|  |  |
| Antonio Jesús Viso Sánchez | FRANCISCO DE ASÍS CONDE RODRÍGUEZ  JOSE NEGRILLO CARDENAS |

**Agradecimientos**

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

|  |  |
| --- | --- |
| Ficha del Trabajo Fin de Título | |
| Titulación | Grado en Ingeniería Informática |
| Modalidad | Proyecto de Ingeniería |
| Especialidad (solo TFG) | Tecnologías de la Información |
| Mención (solo TFG) | Sistemas Gráficos |
| Idioma | Español |
| Tipo | Específico |
| TFT en equipo | No |
| Autor/a | Antonio Jesús Viso Sánchez |
| Fecha de asignación | 21/11/2018 |
| Descripción corta | Herramienta para la edición, visualización y validación de shaders, adaptada a los requisitos de la asignatura de Programación de Aplicaciones Gráficas del Grado de Ingeniería Informática |

|  |  |
| --- | --- |
| normas aplicadas en este documento | |
| Locales | |
| TFT-UJA:2017 | Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén  (Normativa marco UJA aprobada en Consejo de Gobierno) |
| TFT-EPSJ:2017 | Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén  (Normativa EPSJ aprobada en Junta de Escuela) |
| TFT-EPSJ | Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén |
| Nacionales e Internacionales | |
| ISO 2145:1978 | Documentación - Numeración de divisiones y subdivisiones en documentos escritos |
| UNE 50132:1994 | Traducción de la ISO 2145 |
| APA 6ª edición | Estilo de referencias y citas de APA (American Psychological Association) |

|  |  |
| --- | --- |
| normas utilizadas como base o referencia | |
| Nacionales | |
| UNE 157001:2014 | Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico |
| UNE 157801:2007 | Criterios generales para la elaboración de proyectos de sistemas de información |
| *Estas normas se han utilizado como base o referencia para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como proyecto la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.* | |

Contenido

[1 Especificación del trabajo 14](#_Toc20479351)

[1.1 Introducción 14](#_Toc20479352)

[1.2 Antecedentes y estado del arte 15](#_Toc20479353)

[2 Objetivos del trabajo 17](#_Toc20479354)

[3 Tecnologías utilizadas 17](#_Toc20479355)

[3.1.1 Tecnologías 17](#_Toc20479356)

[3.1.1.1 QT 17](#_Toc20479357)

[3.1.1.2 OpenGL 18](#_Toc20479358)

[3.1.1.3 GLSL 18](#_Toc20479359)

[3.1.2 Software 18](#_Toc20479360)

[3.1.2.1 Trello 18](#_Toc20479361)

[3.1.2.2 Gantt Project 19](#_Toc20479362)

[3.1.2.3 Github 20](#_Toc20479363)

[3.1.2.4 Visual Studio 20](#_Toc20479364)

[3.1.2.5 Qt Creator 21](#_Toc20479365)

[3.1.2.6 Adobe Photoshop 21](#_Toc20479366)

[4 Análisis y diseño 22](#_Toc20479367)

[4.1 Casos de Uso 22](#_Toc20479368)

[4.2 Requisitos 23](#_Toc20479369)

[4.2.1 Requisitos funcionales 23](#_Toc20479370)

[4.2.2 Requisitos no funcionales 24](#_Toc20479371)

[4.3 Planificación temporal 24](#_Toc20479372)

[4.4 Metodología de desarrollo de software 29](#_Toc20479373)

[4.4.1 Recogida de requisitos 29](#_Toc20479374)

[4.4.2 Gestión de backlog 30](#_Toc20479375)

[4.4.3 Sprint Planning Meeting 30](#_Toc20479376)

[4.4.4 Ejecución de sprint 30](#_Toc20479377)

[4.4.5 Inspección e iteración 31](#_Toc20479378)

[4.5 Presupuesto 31](#_Toc20479379)

[4.5.1 Costes de materiales 32](#_Toc20479380)

[4.5.2 Costes de Software 32](#_Toc20479381)

[4.5.3 Costes de Hardware 33](#_Toc20479382)

[4.5.4 Coste de Personal 33](#_Toc20479383)

[4.5.5 Mecanismos de control de calidad 34](#_Toc20479384)

[5 Desarrollo / implementación 35](#_Toc20479385)

[6 Pruebas 35](#_Toc20479386)

[7 Conclusiones y trabajo futuro 36](#_Toc20479387)

[8 Apéndices 36](#_Toc20479388)

[8.1 Guía original del Trabajo Fin de Título 36](#_Toc20479389)

[8.2 Documentación de entrada 36](#_Toc20479390)

[8.3 Instalación y configuración del sistema 36](#_Toc20479391)

[8.4 Manuales de usuario 37](#_Toc20479392)

[9 Definiciones y abreviaturas 37](#_Toc20479393)

[10 Bibliografía 39](#_Toc20479394)

Índice de ilustraciones

[Ilustración 10.1 28](#_Toc1631674)

Índice de tablas

[Tabla 10.1 28](#_Toc1631675)

# Especificación del trabajo

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - “*Criterios Generales para la elaboración de proyectos de Sistemas de Información*”.

A lo largo del documento se utilizarán términos y acrónimos que estarán en negrita y cuya descripción aparecen en el **apartado 8** (*Definiciones y abreviaturas).*

## Introducción

La asignatura de Programación de Aplicaciones Gráficas del Grado de Ingeniería Informática, tiene una curva de aprendizaje muy pronunciada, debido que abarca todo el proceso para la generación de gráficos.

La creación de gráficos necesita de tres partes esenciales para la correcta creación de la imagen que se quiere mostrar. Hace falta una geometría correcta, qué esté correctamente cargada en la Unidad de procesamiento gráfico (**GPU**); un shader OpenGL Shading Language (**shader** **GLSL**) correcto, bien compilado y cargado en la GPU; y establecer correctamente los parámetros de Open Graphics Library (**OpenGL**). La dificultad reside en que no se puede probar cada pieza por separado antes de visualizar el resultado final ya que las tres piezas son imprescindibles para poder ejecutar la aplicación gráfica.

En este Trabajo de Fin de Grado se pretende desarrollar una herramienta multiplataforma que permita al usuario probar directamente los Shaders GLSL que esté desarrollando, sin necesidad de haber escrito las otras dos piezas de software necesarias. La herramienta proporciona un entorno donde poder escribir o cargar los Shaders que el usuario quiera probar y aplicarlos sobre geometrías predefinidas y con configuraciones de parámetros predefinidas, así como comprobar la validez sintáctica de los Shaders escritos.

## Antecedentes y estado del arte

Existen una serie de aplicaciones que resuelven parcialmente el problema, entre ellas:

* [Shdr](http://bkcore.com/blog/3d/shdr-online-glsl-shader-editor-viewer-validator.html): es un editor online ESSL (GLSL), funciona como editor ,visualizador y validador de shaders todo con **WebGL** creado porThibaut Despoulain.



Ilustración 1.3.1 Aplicación Shdr

* [Kick.js](http://www.kickjs.org/example/shader_editor/shader_editor.html): Es un motor gráfico montado en WebGL para navegadores modernos, que además posee un editor de shaders.

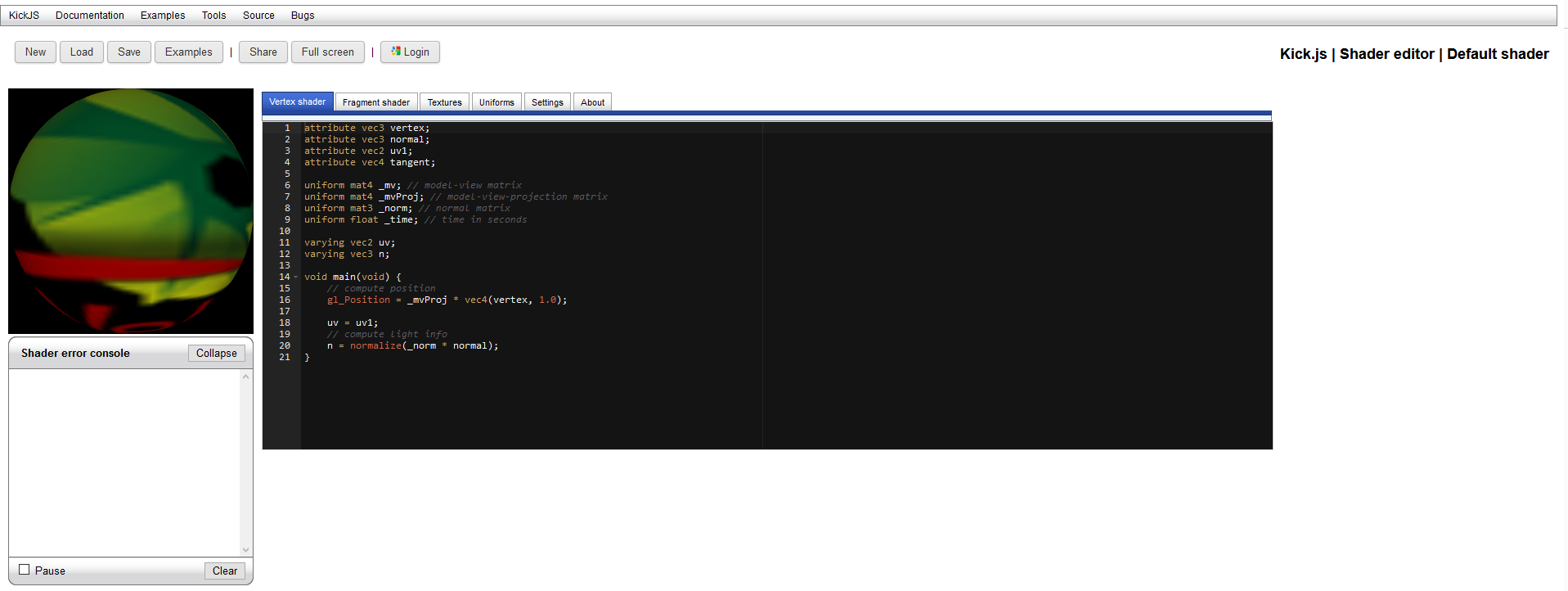


Ilustración 1.3.2 Aplicación Kick.js

Estas aplicaciones online presentan una serie de inconvenientes importantes:

* No están totalmente adaptadas a la asignatura PAG del Grado de Ingeniería Informática y no permiten usar características avanzadas de OpenGL, (no disponibles en WebGL).
* Son complicadas y poco intuitivas.

En este Trabajo de Fin de Grado se propone una herramienta basada en OpenGL 4.5, para la edición, visualización y validación de los shaders GLSL.

# Objetivos del trabajo

* Desarrollar un prototipo que permita cargar y compilar shaders GLSL desde el punto de vista sintáctico.
* Implementar algunas geometrías predefinidas, sobre las que se puedan aplicar los shaders cargados, para ver si el resultado obtenido es el que se buscaba.
* Visualizar el resultado de la ejecución de un shader program, sin necesidad de construir una aplicación gráfica completa
* Mostrar resultados intermedios en forma gráfica, para facilitar la prueba de los shaders, como por ejemplo los cube maps en el caso de shaders para simular reflexión o los shadow maps en el caso de shaders para simular sombras arrojadas.

# Tecnologías utilizadas

Para el desarrollo de este proyecto se han usado unas las tecnologías y un software que permite el desarrollo de las estas tecnologías.

### Tecnologías

En este apartado se detallan las tecnologías empleadas en el proyecto y la justificación de su uso.

#### QT

Es un framework multiplataforma orientado a objetos ampliamente usado para desarrollar programas que utilicen interfaz gráfica de usuario, así como también diferentes tipos de herramientas para la línea de comandos y consolas para servidores que no necesitan una interfaz gráfica de usuario.

Se ha usado este framework ya que es de código abierto, posee una comunidad activa, gran flexibilidad a la hora de generación de aplicaciones y una herramienta de generación de interfaces muy fluida y sencilla a la hora de trabajar.

#### OpenGL

Sera la API principal para la generación de gráficos 3D, igual que en la asignatura de Programación de Aplicaciones Gráficas del Grado de Ingeniería Informática.

#### GLSL

Lenguaje de alto nivel de sombreado con una sintaxis basada en el lenguaje de programación C y cuyo lenguaje es usado en los shaders creados por el usuario, en la herramienta que se propone.

### Software

Se detallará el software usado en la planificación, el diseño gráfico, desarrollo el producto, gestión de versiones, …

#### Trello

Aplicación online para administración de proyectos que permite el uso de metodología **Kanban**, sobretodo lo usare para gestionar las tareas diarias, dependiendo de la etapa del proyecto.

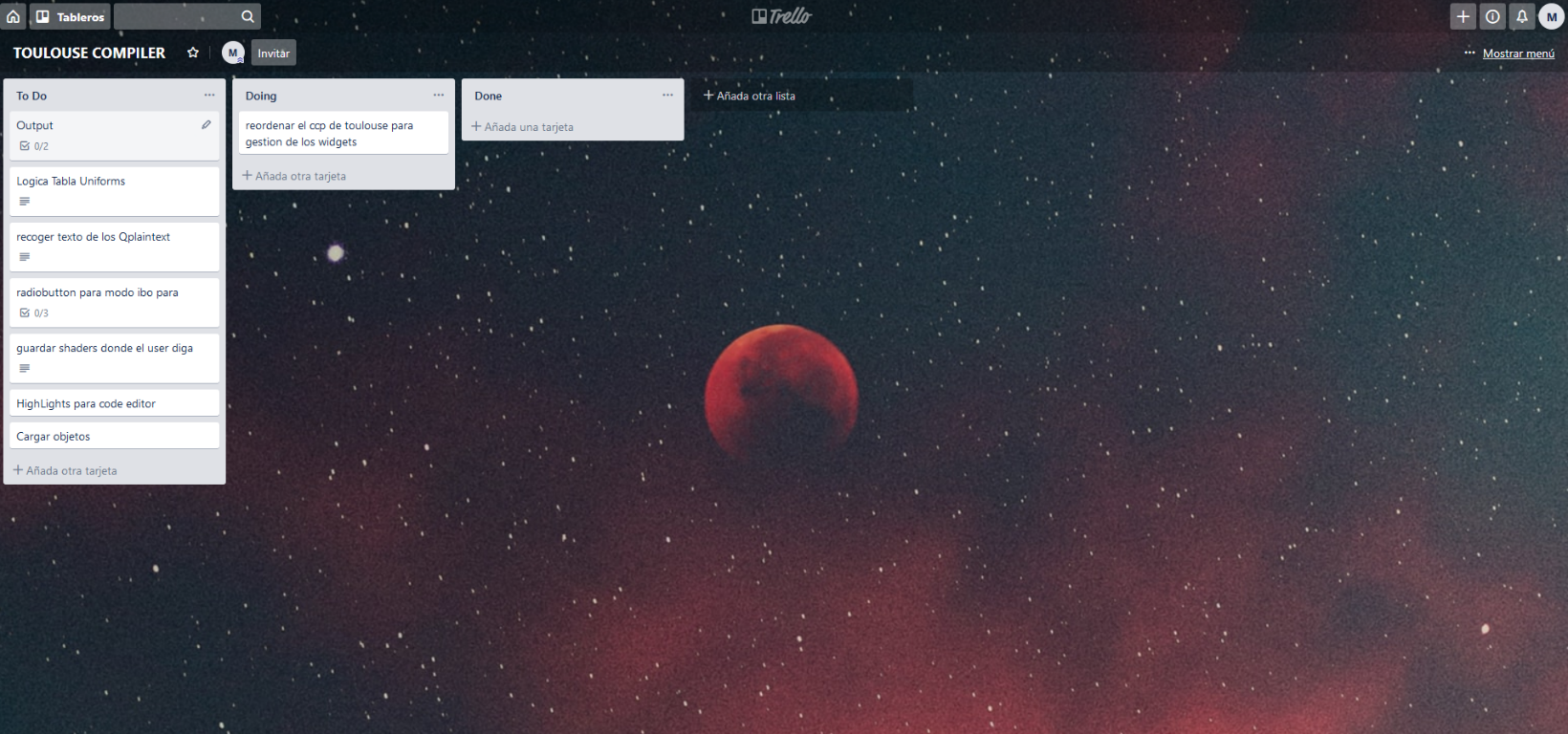


Ilustración 2.1.1 Trello

#### Gantt Project

Un software gratuito para gestión de tareas y proyectos. Permite generar diagramas de **Gantt**, que usare para especificar el tiempo de las etapas del desarrollo.

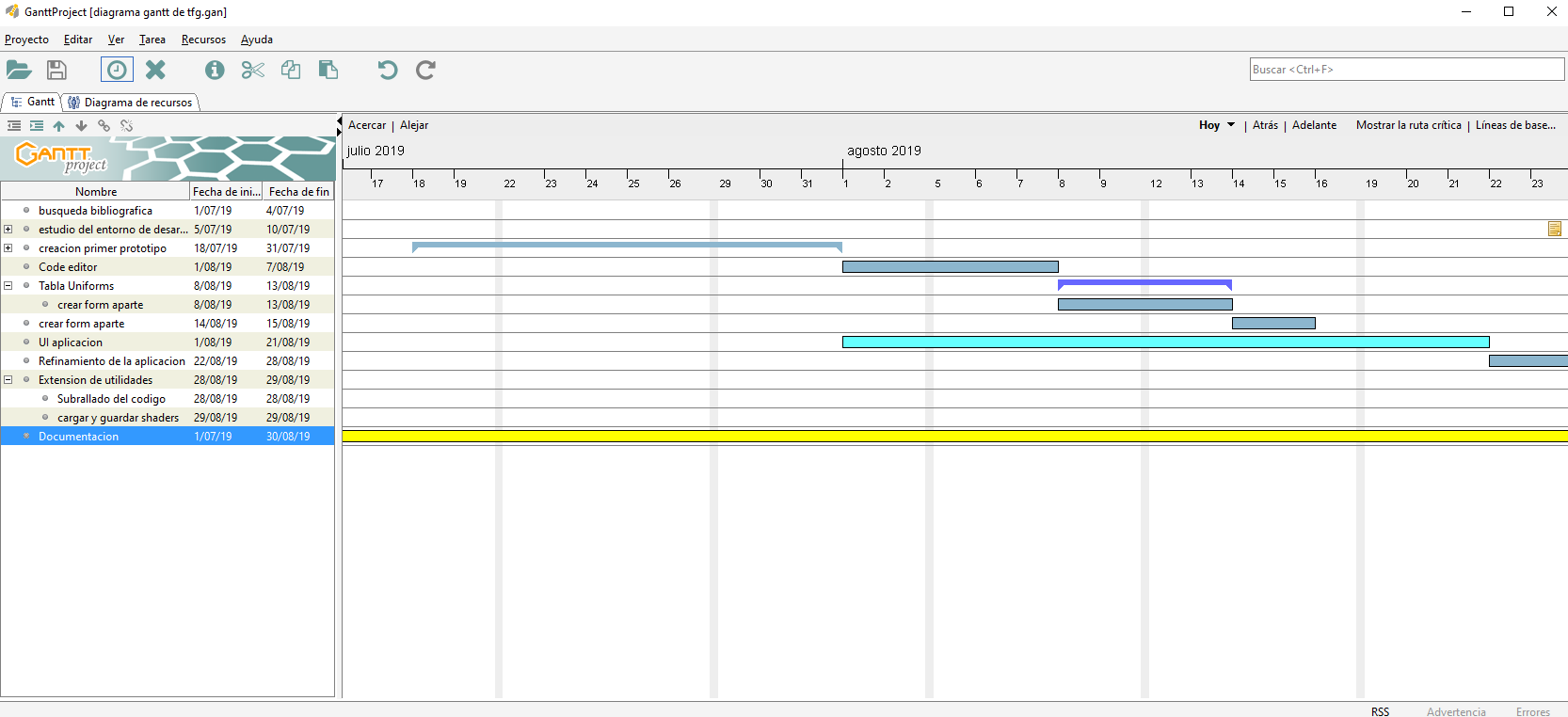


Ilustración 2.1.2 Gantt Project

#### Github

Es una plataforma online para alojar proyectos utilizando el sistema de control de versiones Git. Yo usare sobretodo la aplicación de escritorio ya que se integra con Visual Studio.

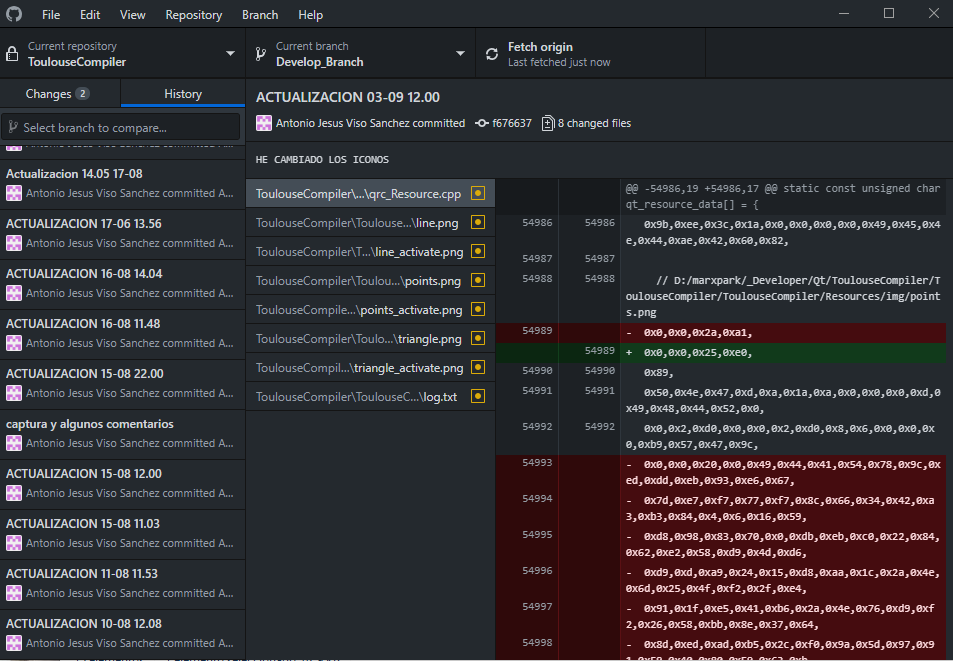


Ilustración 2.1.3 Github

#### Visual Studio

Entorno de desarrollo integrado para Windows, Linux y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic. Para este proyecto se usará la versión Enterprise 2017 será el compilador principal de la aplicación.

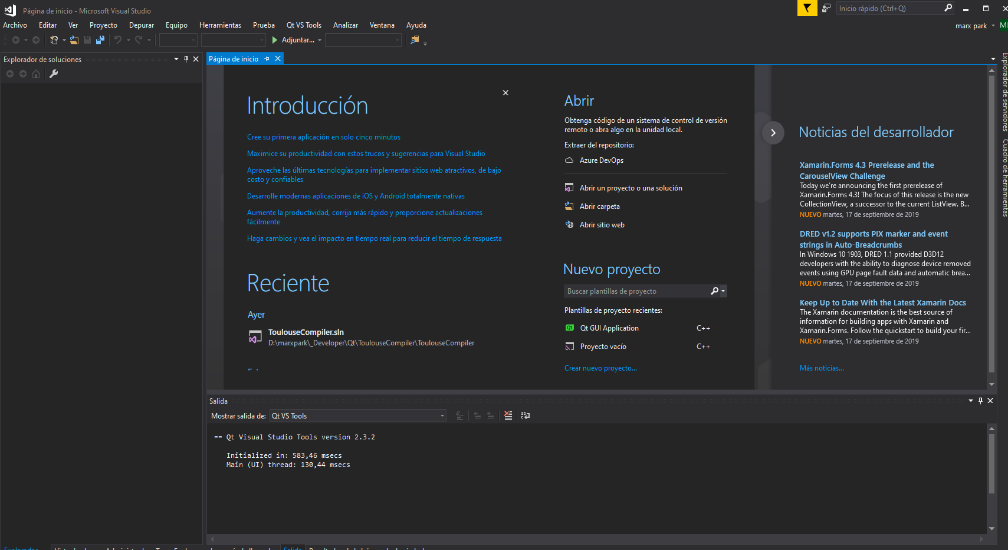


Ilustración 2.1.4 Visual Studio

#### Qt Creator

Qt Creator es un IDE multiplataforma programado en C++, JavaScript y QML para el desarrollo de aplicaciones con Interfaces Gráficas de Usuario.

De este programa usare la integración con Visual Studio y el gestor de creación de Interfaces Qt Designer.

Para este proyecto se ha usado la versión 4.8 de QT.

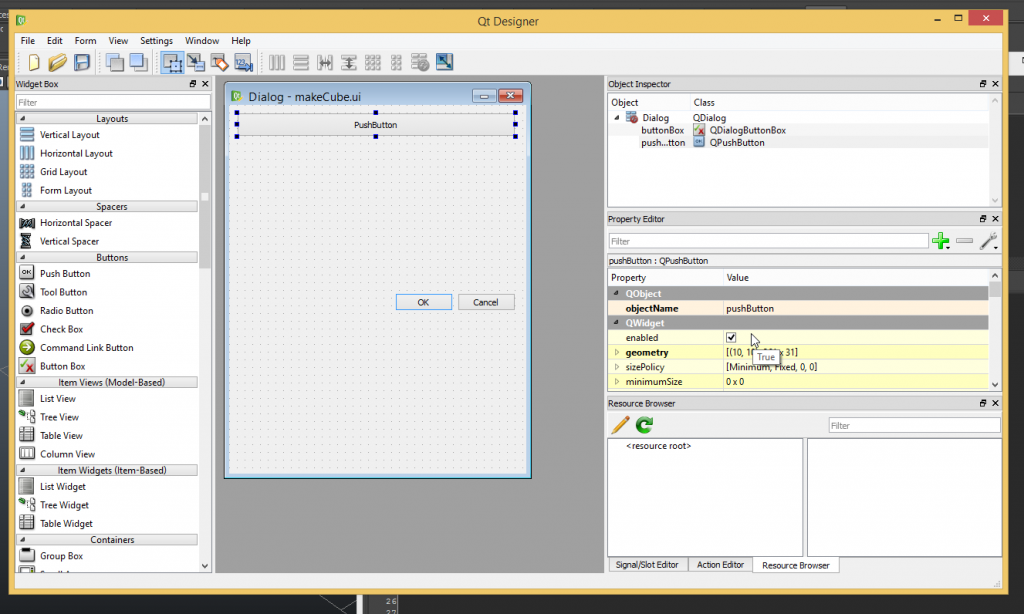


Ilustración 2.1.5 QT Designer

#### Adobe Photoshop

Es un editor de gráficos rasterizados usado principalmente para el retoque de fotografías y gráficos. En este proyecto se ha usado sobre todo para la creación de iconos y la creación de diagramas para la memoria.



Ilustración 2.1.6 Adobe Photoshop

# Análisis y diseño

## Casos de Uso

## Requisitos

Para el desarrollo de esta aplicación se hizo un estudio en alumnos de la asignatura de Programación de Aplicaciones Gráficas como también en gente que trabaja en el sector de la creación de software, concretándose los siguientes requisitos funcionales y no funcionales.

### Requisitos funcionales

Los requisitos funcionales representan las funcionalidades que el sistema debe ser capaz de realizar al final del desarrollo. A continuación, se enumeran los requisitos:

* Se debe tener una consola para mostrar los errores y el funcionamiento de la aplicación.
* Se debe poseer un espacio para escribir el código necesario.
* Se deberían marcar los errores en la línea correspondiente dentro del código.
* Se deben visualizar objetos en 3d para el posterior tratamiento con Shaders.
* Se debe poder y guardar Shaders.
* Se debería hacer un resaltado de palabras reservadas de GLSL.

### Requisitos no funcionales

Una vez definidos los requisitos funcionales se definirán los no funcionales. Se trata de requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema, sino a las propiedades del sistema: rendimiento, interfaz, seguridad, usabilidad, seguridad, ...

* La parte donde se escribe el código de los shaders, se verá afectada por el efecto de resaltado de las palabras reservadas (se destacan aquellas variables reconocidas por el sistema como palabras reservadas de GLSL).
* La interfaz será intuitiva, seguirá una guía de estilo uniforme, manteniendo la misma tipografía, …
* La interfaz del programa deberá tener dos idiomas, español e inglés, siendo posible añadir otros idiomas más delante de forma sencilla.
* Se podrá añadir la opción de cargar objetos del usuario de forma sencilla.

## Planificación temporal

La importancia de una buena planificación se hace presente en el éxito de todo proyecto más o menos grande. Se necesita dividir el problema en tareas más ligeras y con un tiempo de realización fijo.

Ya que el presente proyecto es un Trabajo de Fin de Grado, por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

Se tendrán en cuenta los días no hábiles, como los fines de semana (3 meses,6 horas diarias, en total aproximadamente 300 horas). Considerando el último de ellos como medio mes para reuniones con los tutores especificando las posibles modificaciones (de los parámetros del programa) resultantes de las reuniones. Y el otro medio para el testeo final y refinamiento del programa.

Para la parte de la planificación temporal del desarrollo del proyecto, se generó un diagrama de Gantt, el cual especificaba el tiempo necesario para cada parte del desarrollo.

A continuación, se muestran las tareas reflejadas en el diagrama (*ver ilustración 1.10.1*):

* Búsqueda bibliográfica. 4 días.
* Estudio del entorno de desarrollo QT. 4 días.
  + Creación de programa simple funcional con QT. 1 día.
  + Creación más compleja de aplicación con OpenGL. 3 días.
* Creación del primer prototipo.15 días.
  + Integración de los shader programs al prototipo.2 días.
  + Integración de las prácticas de PAG.8 días.
  + Implementación del Code Editor. 5 días.
* Diseño de la interfaz.7 días.
  + Diseño e Implementación de Tabla Uniforms.3 días.
  + Diseño e Implementación del Log.2 días.
  + Integración del Code Editor a la interfaz. 2 días.
* Implementación de utilidades del programa. 5 días
  + controles de cámara para OpenGL. 2 días.
  + Opciones selección distintos objetos. 3 días.
* Extensión de utilidades. 5 días.
  + Subrayado del código.1 día.
  + cargar y guardar shaders.1 día
  + Selección idioma. 3 días.
* Refinamiento de la aplicación. 1 día.
* Documentación.50 días

Destacando que los objetivos a realizar dentro de la tarea asignada se planearan con un diagrama de Kanban (*Ver ilustración 1.10.2*).

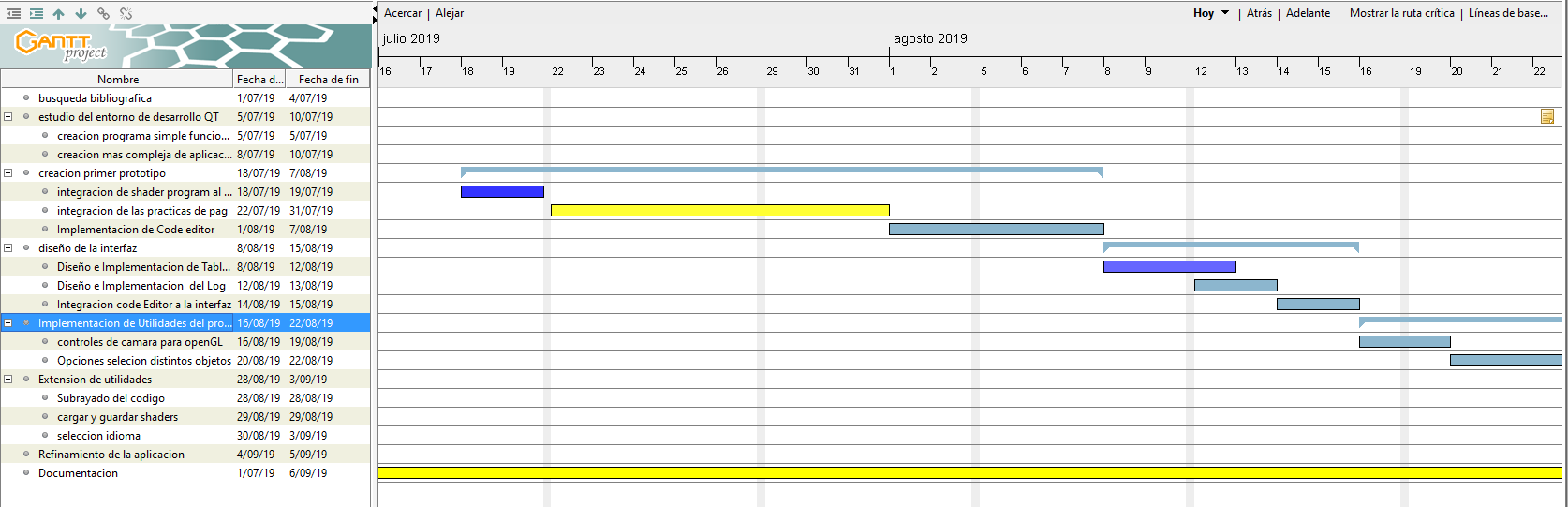


Ilustración 1.10.1 Diagrama de Gantt del Proyecto

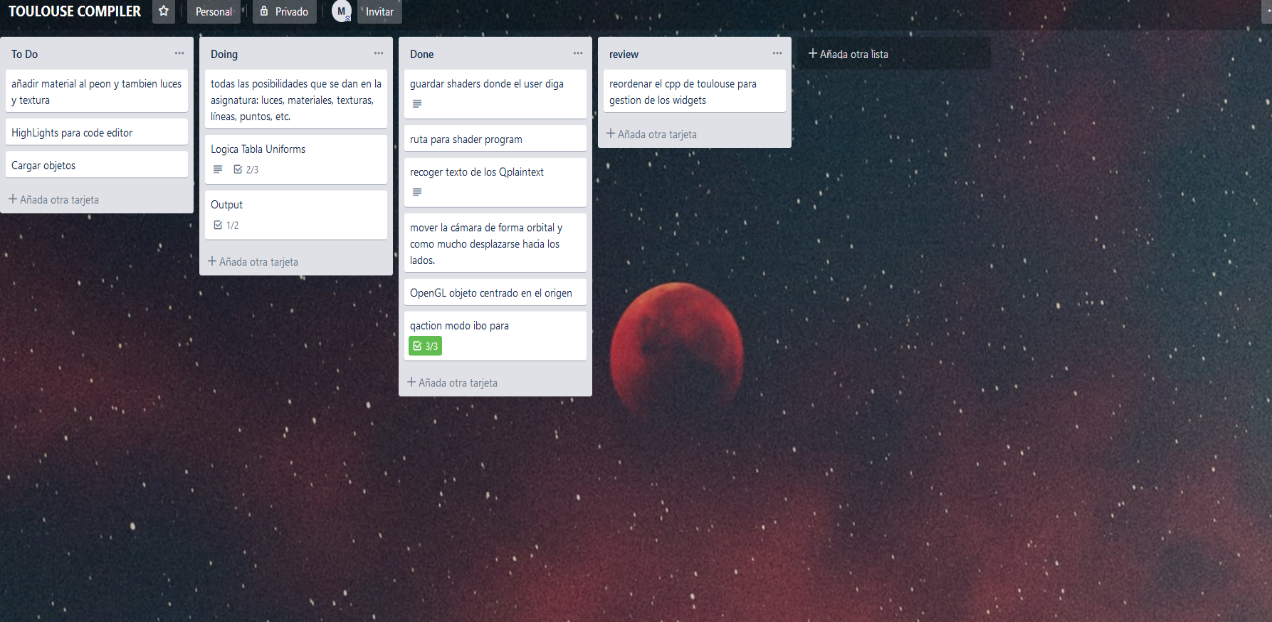


Ilustración 1.10.2 Diagrama de Kanban del Proyecto

## Metodología de desarrollo de software

Para el desarrollo de este proyecto se seguirá una metodología **SCRUM**, ya que de esta forma se puede ir especificando pequeños cambios, agregar nuevas funcionalidades en el proyecto, … Además, se puede obtener un feedback por parte del cliente (en este caso los tutores).

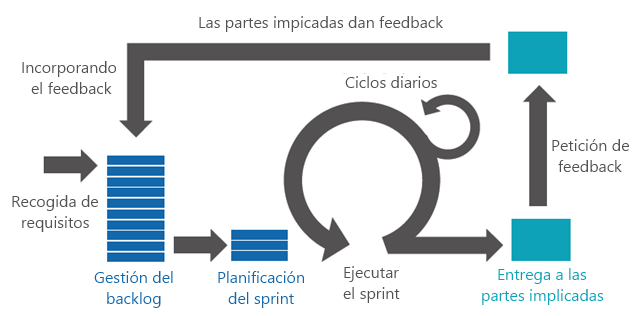


Ilustración 4.4.1 Diagrama fases SCRUM

Se necesitará de varias iteraciones dentro de la metodología para llegar a la fase final del proyecto.

A continuación, se explicará las fases dentro de la metodología SCRUM.

### Recogida de requisitos

El proceso comienza con la generación de la lista de objetivos o requisitos priorizada, que actúa como plan del proyecto y que es entregada por el cliente al equipo de desarrollo.

La lista de requisitos priorizada representa la visión y expectativas del cliente respecto a los objetivos y entregas del producto o proyecto.

### Gestión de backlog

Después de que se haya revisado la representación de los requisitos, para cada requisito se indica la importancia y el costo estimado de completarlo, que en este caso será tiempo.

### Sprint Planning Meeting

Un sprint es una unidad de trabajo que agrupa un conjunto de tareas en un periodo de tiempo. La primera iteración es de planificación y está compuesta por dos partes:

* **Selección de requisitos:** Es la iteración entre cliente y equipo, el momento en que el equipo pregunta al cliente las dudas que surgen y se seleccionan los requisitos más prioritarios que se comprometen a completar en la iteración.
* **Planificación de la iteración:** Se elabora la lista de tareas o acciones necesarias para desarrollar los requisitos a los que se han comprometido.

De esta forma siempre se mantiene el contacto con el cliente y este está más involucrado en el proyecto.

### Ejecución de sprint

En la metodología SCRUM un proyecto se ejecuta en bloques temporales cortos y fijos, llamados sprint, que son iteraciones de 2 semanas. Si se sobrepasa este tiempo, como máximo un sprint puede tomar 4 semanas.

De esta forma no se gasta más tiempo en una parte del proyecto y en otras menos.

### Inspección e iteración

El último día de la iteración se realiza la reunión de revisión de la iteración, y se compone de dos partes:

**Sprint Review:** El equipo desarrollador presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado. El cliente revisa el entregable y se adaptan las mejoras necesarias.

**Sprint Retrospective:** En esta fase el equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, enfocando el proceso a la mejora continua del equipo.

De esta forma se puede seguir evolucionando la aplicación y el equipo va mejorando con el desarrollo.

## Presupuesto

Una parte importante del proyecto es cuanto se está dispuesto a pagar para el desarrollo del producto, es decir el presupuesto del que se dispondrá. Se supondrá que estos gastos los asume una empresa ficticia, por lo tanto, no se dedicará únicamente a este proyecto. Esta suposición hará posible que se puedan repartir los recursos entre estos proyectos (se pensara que están desarrollando 4 proyectos).

Para poder especificar correctamente los costes hay que diferenciarlos en:

* Costes de materiales.
* Costes de software.
* Costes de hardware.
* Costes de personal.

Todos los costes llevan aplicado el 21% de los impuestos.

### Costes de materiales

Los costes de carácter material, son recursos de uso general y coste reducido y donde no va incluido los costes de hardware para el desarrollo. En la tabla 4.5.1 se muestran los costes materiales:

|  |  |  |
| --- | --- | --- |
| Nombre | Objetivo | Coste(€) |
| Post-It 47.6 mm x 47.6 mm | Destinado a señalar las tareas que se realizaran ese día y errores detectados. | 3,00 € |
| Caja de 50 unidades de bolis BIC | Para diseños en papel y apuntar tareas | 11,28 € |
| Coste Total | | 14,28 € |

Tabla 3.5.1 Desglose de costes en materiales

### Costes de Software

En este apartado se incluirán todos los programas utilizados en el proyecto. La mayoría del software es gratuito o se abona mensualmente, así que se tendrá en cuenta y no se amortizaran.

|  |  |  |
| --- | --- | --- |
| Nombre | Objetivo | Coste(€) |
| Adobe Photoshop CC | Destinado a la generación de los partes de la interfaz | 24,19 €/mes x 3 |
| Licencia de Windows 10 | Sistema Operativo del ordenador. | 7,58 € |
| Office 365 personal | Realización de la documentación | 7,00 €/mes x 3 |
| Visual Studio 2017 versión estudiante | Compilación y generación de código | 0 € |
| Github | Control de versiones | 0 € |
| Trello | Generador de diagramas Kanban | 0 € |
| QT Creator | Entorno para la generación de aplicaciones gráficas | 0 € |
| Coste Total (€) | | 101,15 € |

Tabla 3.5.2 Desglose de costes de software.

Los programas mostrados serán necesarios para el desarrollo del programa, pero puede darse el caso que se necesite de algún programa externo que no sea gratuito, que ira dentro del porcentaje adicional para costes imprevistos.

### Costes de Hardware

Estos costes engloban los equipos necesarios u otros dispositivos necesarios para el desarrollo del proyecto. Al ser solo una persona la que está trabajando en él, solo es necesario la compra de un ordenador.

Se estima que se necesitará un ordenador valorado en 1000 € , para el desarrollo de este proyecto.

### Coste de Personal

En este apartado se mostrarán los costes asociados a la parte de personal. Toca destacar que todo el trabajo de este proyecto recae sobre solo un alumno, por lo tanto, hará distintos roles dentro del desarrollo del proyecto.

El salario base de los trabajadores se ha extraído del *Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública. (*[*https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf*](https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf)*)*

Para la realización del proyecto se necesitará de los siguientes roles:

* **Analista/Diseñador**: Se encargará de todo el análisis y diseño del sistema.
* **Programador**: Se encargará de la implementación del sistema.
* **Supervisor de pruebas**: Encargado de la experimentación y pruebas de usuario.
* **Jefe del proyecto**: Se encargará de supervisar, generar la documentación y difundir los resultados obtenidos.

Al sueldo mensual de los trabajadores hay que aplicarle el 33% de impuestos a la empresa en concepto de gasto para la Seguridad Social.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | Sueldo base mensual | Retención mensual | Meses | Dedicación al proyecto | Coste(€) |
| Analista/diseñador | 1.535,22 € | 506,62 € | 2 | 25% | 1.020,92 € |
| Programador | 1.089,20 € | 359.43 € | 3 | 100% | 4.945,89 € |
| Supervisor de pruebas | 828,39 € | 273,36 € | 3 | 25% | 826.31 € |
| Jefe del proyecto | 1.637,50 € | 540,37 € | 3 | 25% | 1.606,41 € |
| Coste Total | | | | | 8.399,53 € |

Tabla 4.5.4 Desglose de costes de personal.

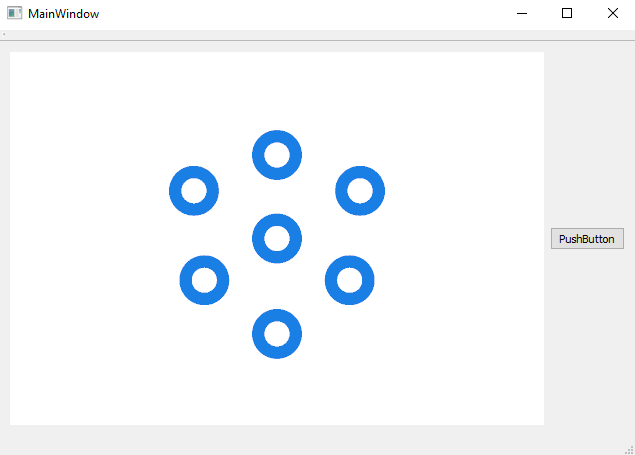
### Mecanismos de control de calidad

El aseguramiento de la calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por un/a tutor/a, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

# Desarrollo / implementación

La primera aplicación fue la que logró unir Qt y OpenGL, mostrando una imagen en 2d:



# Pruebas

# Conclusiones y trabajo futuro

# Apéndices

**Objetivo**: incluir la documentación adicional que pueda resultar de utilidad para comprender correctamente el proyecto.

**Contenido**: toda la documentación relevante que no aparezca anteriormente. A continuación se proponen algunos contenidos de entre los más habituales.

## Guía original del Trabajo Fin de Título

**Contenido**: incluir la propuesta o **guía original del TFG/TFM** (publicada en la web de la EPS en el momento de la convocatoria), así como el histórico de modificaciones que haya podido tener dicha propuesta (título, objetivos, etc.).

## Documentación de entrada

[Opcional]

**Contenido**: este anexo debe incluir la documentación de la que se dispone previa realización del trabajo, salvo la guía del Trabajo Fin de Título, que se especificará en su propio apartado. Esto puede incluir todo tipo de información tanto en papel (escaneados) como en formato electrónico. Se puede incluir la propuesta de un cliente, un pliego de condiciones, documentos de la empresa cliente o sistema anterior (incluso escaneados), información de otros proyectos, informes técnicos, planos, fotos, datasets científicos, etc.

## Instalación y configuración del sistema

[Opcional]

**Contenido**: en caso de desarrollar algún software o sistema, es conveniente incluir unas instrucciones que permitan a los lectores descargar, instalar, configurar e utilizar dicho sistema.

## Manuales de usuario

[Opcional]

**Contenido**: en caso de desarrollar algún software o sistema, es conveniente incluir unas instrucciones de uso que permitan probar todas las funcionalidades implementadas.

[

# Definiciones y abreviaturas

**Objetivo**: facilitar la comprensión del texto mediante la **descripción de la terminología empleada, cuando ésta sea muy específica** y requiera de aclaraciones. Se deben relacionar todas las definiciones, abreviaturas, etc. que se han utilizado y su significado.

**Contenido**: relación de la terminología específica, definiciones, abreviaturas, etc., que se han utilizado a lo largo del trabajo, así como su significado.

* **GPU**: Una **unidad de procesamiento gráfico** o **GPU** es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.
* **Shader GLSL**: **OpenGL Shading Language** (abreviado **GLSL** o **GLslang**) es un lenguaje de alto nivel de sombreado con una sintaxis basada en el lenguaje de programación C y shader es el programa generado en este lenguaje.
* **OpenGL**: **Open Graphics Library** es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.
* **WebGL**: es una especificación estándar que define una API implementada en JavaScript para la renderización de gráficos en 3D dentro de cualquier navegador web.
* **Visual Studio** es un editor de código fuente desarrollado por Microsoft para Windows.
* **Kanban:** es una de las llamadas metodología ágiles, aquellas que buscan gestionar de manera generalizada cómo se van [completando las tareas](http://www.billage.es/es/gestion-de-proyectos).
* **Gantt:**
* **SCRUM:**
* **Sprint:**

# Bibliografía

**Metodología SCRUM**: https://blog.ida.cl/estrategia-digital/metodologia-scrum-en-proyectos-digitales/