# Desarrollo del proyecto

En este apartado se detallarán las propiedades del sistema, como también detalles sobre el desarrollo de éste. El principal objetivo es la creación de una herramienta software capaz de ayudar al alumno a crear sus shaders y entender si lo hizo bien.

## Primera Iteración

Lo primero que se hizo fue un storyboard de la interfaz que debería tener la aplicación, de esta forma podríamos pensar en cómo debería funcionar para que fuera lo más intuitivo posible.

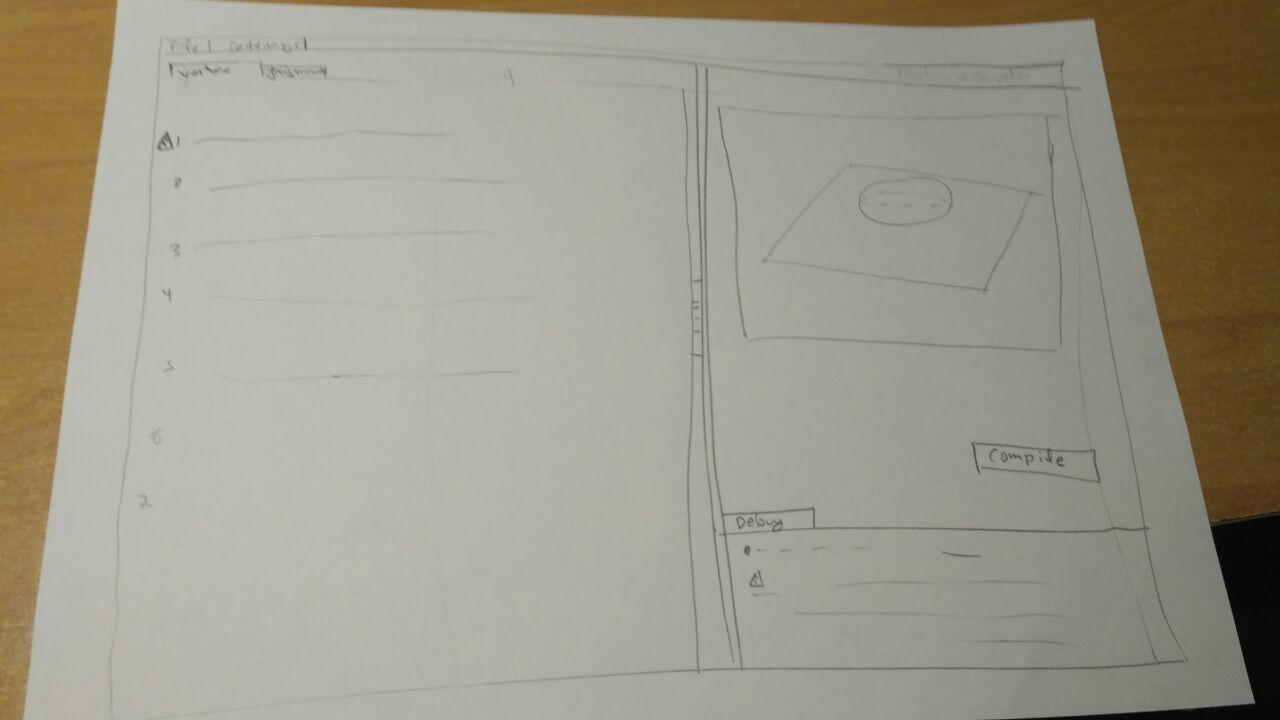


Ilustración 1 Boceto Primera Interfaz

Pero antes de poder empezar a crear la aplicación, se tenía que superar un obstáculo, el de familiarizarse con el entorno de desarrollo de QT y las particularidades de este entorno de programación.

Lo más complicado fue elegir la forma de integrar OpenGL con los widgets existentes en QT, ya que dependiendo de lo que se eligiera podría marcar una gran diferencia con respecto a la versión y formas de trabajar con los shaders.

Después de un tiempo de investigación para lograr encontrar una forma sencilla y potente de conseguir el máximo potencial de OpenGL, la primera aplicación que logró unir Qt y OpenGL, fue la que acabó mostrando una imagen en 2d:

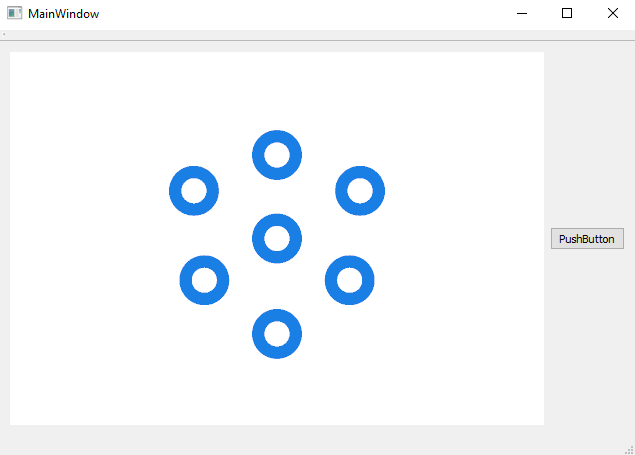


Ilustración 2 Primera aplicación en QT

Partiendo de esta primera aplicación, ya se podía saber que se usaría QT sin problemas por lo tanto se empezó a trabajar en la estructura del código para mostrar una versión parecida al primer boceto de la interfaz.

## Segunda Iteración

Tras un tiempo de trabajo y unos baches por el camino, se consigue la integración de la parte de PAG con QT, y se puede mostrar de una manera rudimentaria el objeto de un peón de ajedrez en modo de líneas:

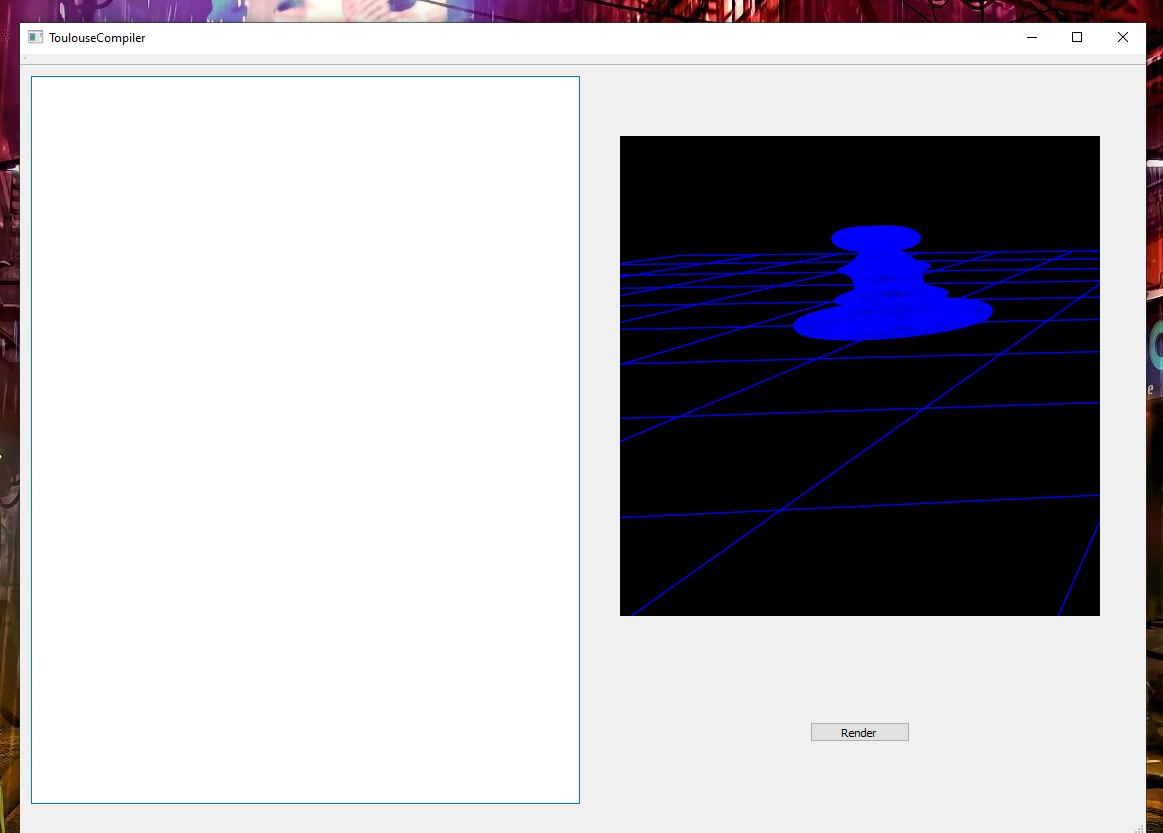


Ilustración 3 Primera versión con shaders

El principal problema reside en el que se tiene que pasar el contexto de “*openglFunctions”* desde la clase que se encarga de renderizar todo el mundo 3D (“*openglWidget”*) al resto de clases que necesitaran usar cualquier parámetro de OpenGL. Tras haber detectado esto, la aplicación fue avanzando hasta lograr lo visto en la ilustración anterior, una versión funcional sin funcionalidad con las luces, pero con los shaders básicos que se enseñaban en PAG.

Tras este logro, se llega a la conclusión de que hace falta un nuevo diseño de la interfaz, ya que hay que permitir una manera sencilla de añadir Uniforms personalizados por el usuario.

Viendo que el storyboard inicial no cumple con lo descrito anteriormente se diseña el siguiente storyboard para la interfaz:

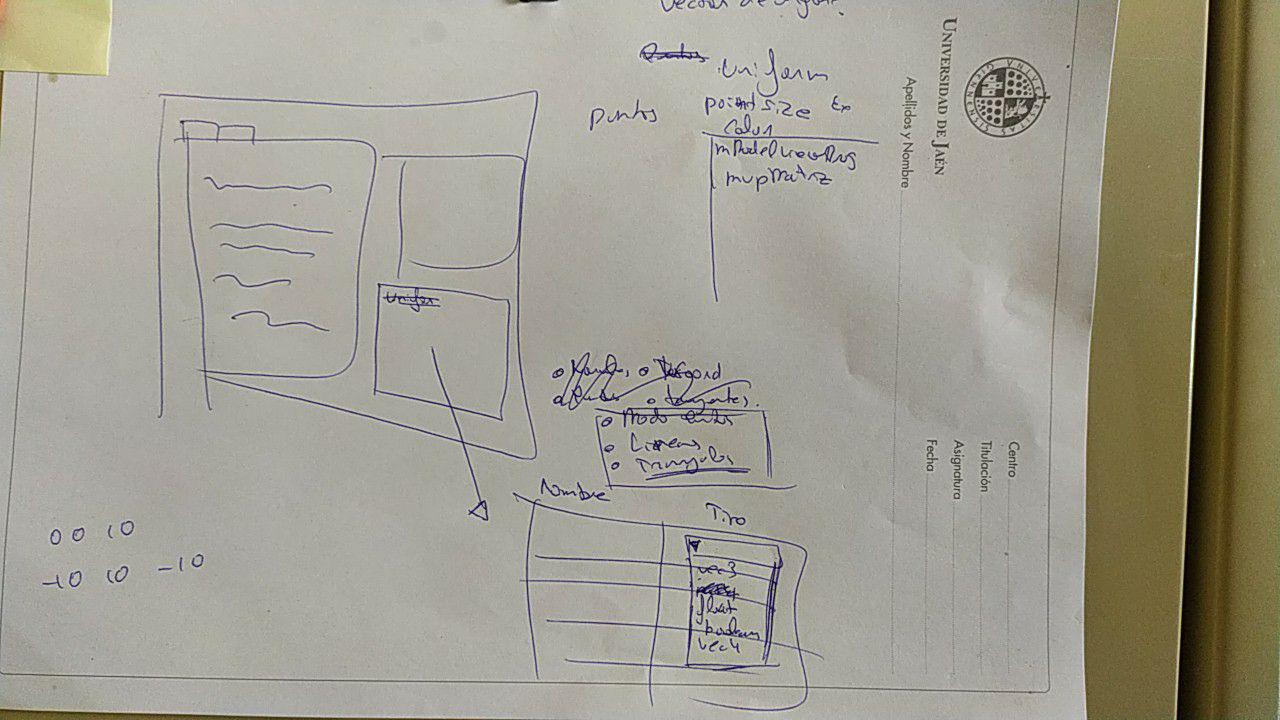


Ilustración 4 Boceto de la segunda Interfaz

Como se puede ver en el boceto, se especifica que la zona de escritura de los shaders será como la típica vista de pestañas de otros editores de código y que además debería haber una tabla para poder añadir aquellos Uniforms que el usuario considere oportunos.

Del mismo modo se hace clara la necesidad de que la interfaz debe tener un método para cambiar el tipo de IBO, dependiendo del shader que quiera usar el usuario y de que hay que especificar al usuario que Uniforms rellena el programa por defecto.

## Tercera Iteración

Lo siguiente es llevar a cabo lo que se creó en el boceto de la anterior iteración, el problema es trabajar con los canvas de QT Designer, ya que hay que tener sumo cuidado con las restricciones de tamaño, espaciado, posición... que se usan ya que mover un widget de un sitio a otro genera como que a la hora de enlazarlo con código haya que acceder entre jerarquías como que las restricciones puestas ya no sean válidas.

Un aspecto importante en cuanto a la creación de la interfaz es que fuera adaptativa dependiendo de las distintas resoluciones posibles, así que teniendo en cuenta este enfoque se crea una primera versión más avanzada del proyecto:

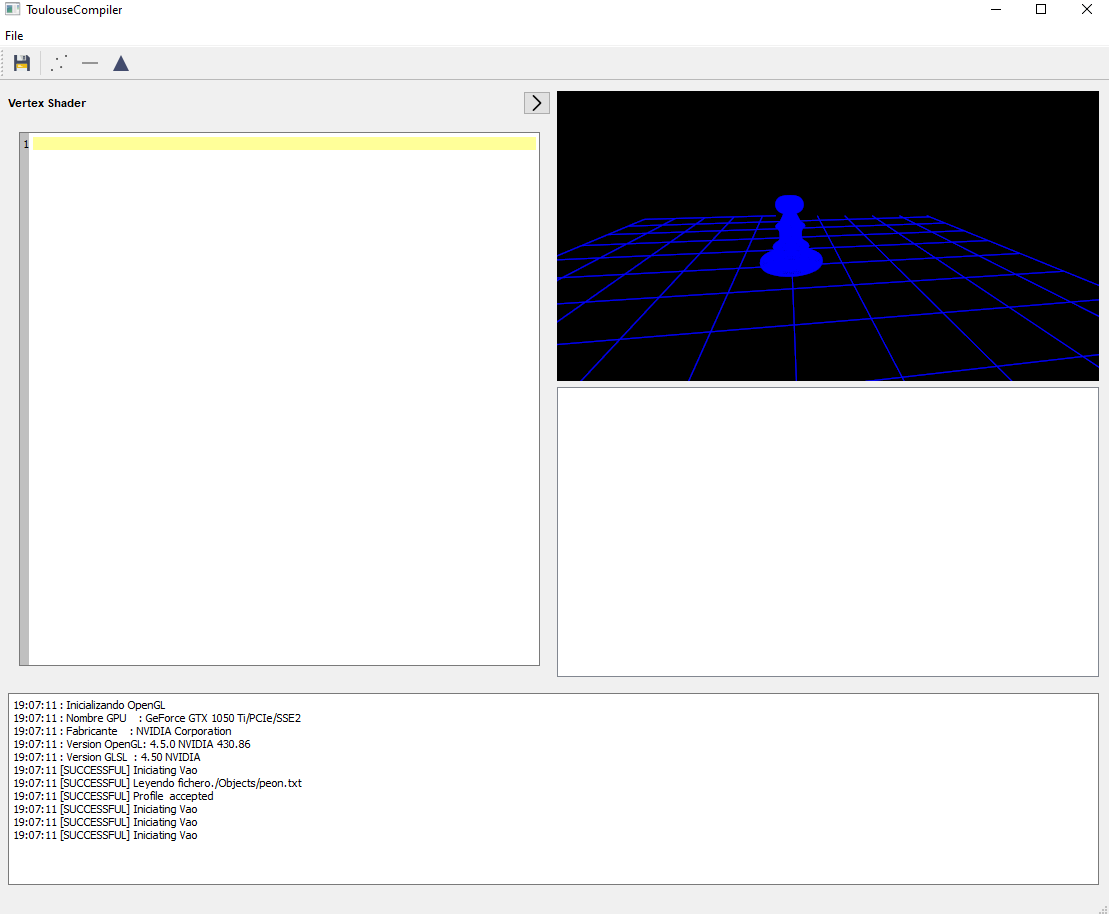


Ilustración 5 Aplicación con varios widgets

En la ilustración X se muestra una interfaz en la cual ya se puede apreciar los botones para cambiar el tipo de IBO a usar, como una tabla para añadir los uniforms necesarios.

Hay que especificar que la aplicación funciona sin tener que escribir nada ya que se esta probando las partes técnicas de visualización y edicion de uniforms y elección del tipo de IBO.

Tras ver que la aplicación responde como lo esperado se empieza a crear una experiencia mas atractiva a la hora de usar la aplicación, como puede ser la visualización de los errores, una ventana emergente para añadir los uniforms, botones con efecto de encendido al pulsarlos y botones para cambiar de tipo de shader.

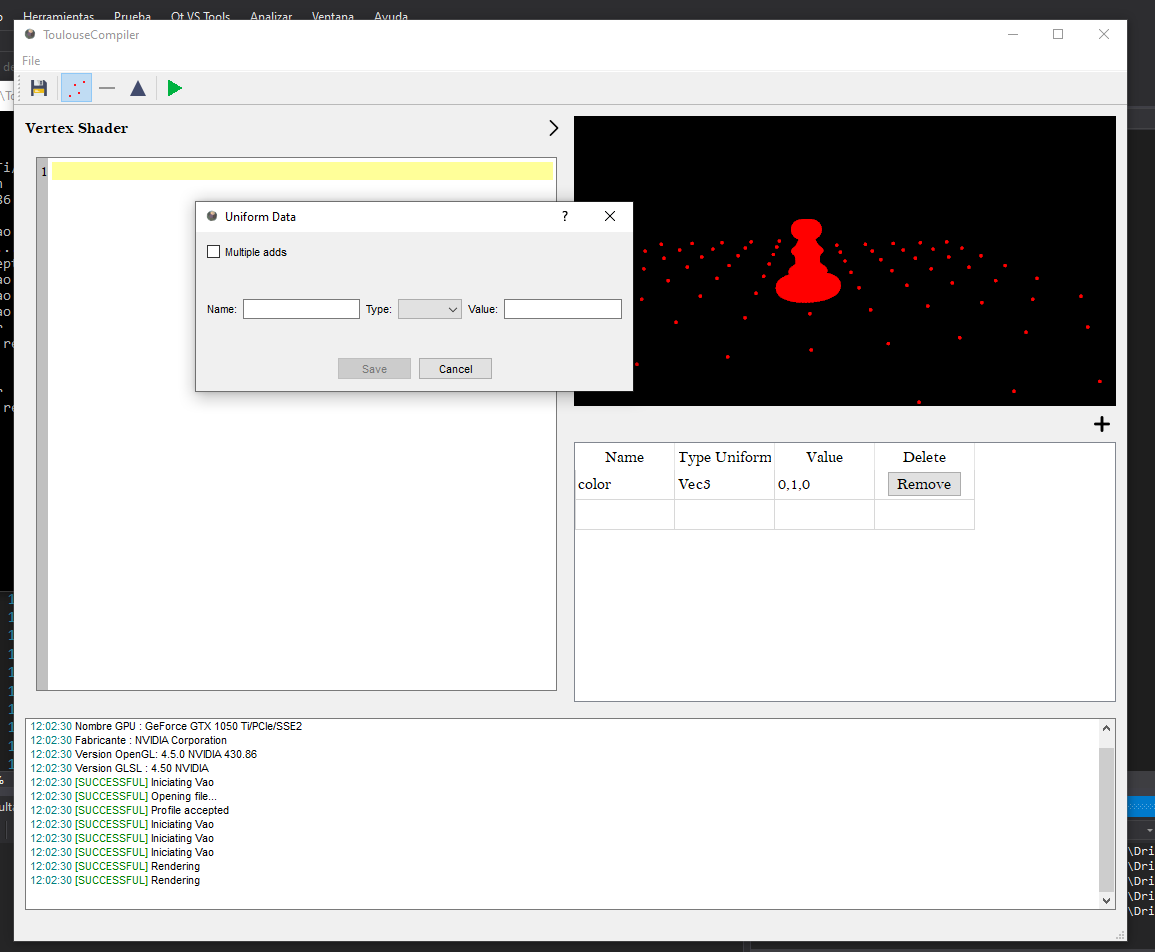


Ilustración 6 Aplicación funcionando con uniforms

## Cuarta Iteración

En esta iteración he conseguido cambiar significativamente la visualización del apartado de escritura de los shaders ya que he dejado las vistas como pestañas que quedan más intuitivas a la hora de programar con shaders.

En cuanto a la tabla de los Uniforms, he puesto una edición y eliminación sencilla para el usuario como más adelante se verá en la ilustración.

He implementado una cámara 3D más acorde con la finalidad del programa, por lo que ahora se puede apreciar los cambios producidos por el shader de una manera más clara. Esta cámara puede rotarse alrededor del objeto y hacer zoom sobre él, con el movimiento y las teclas del ratón.

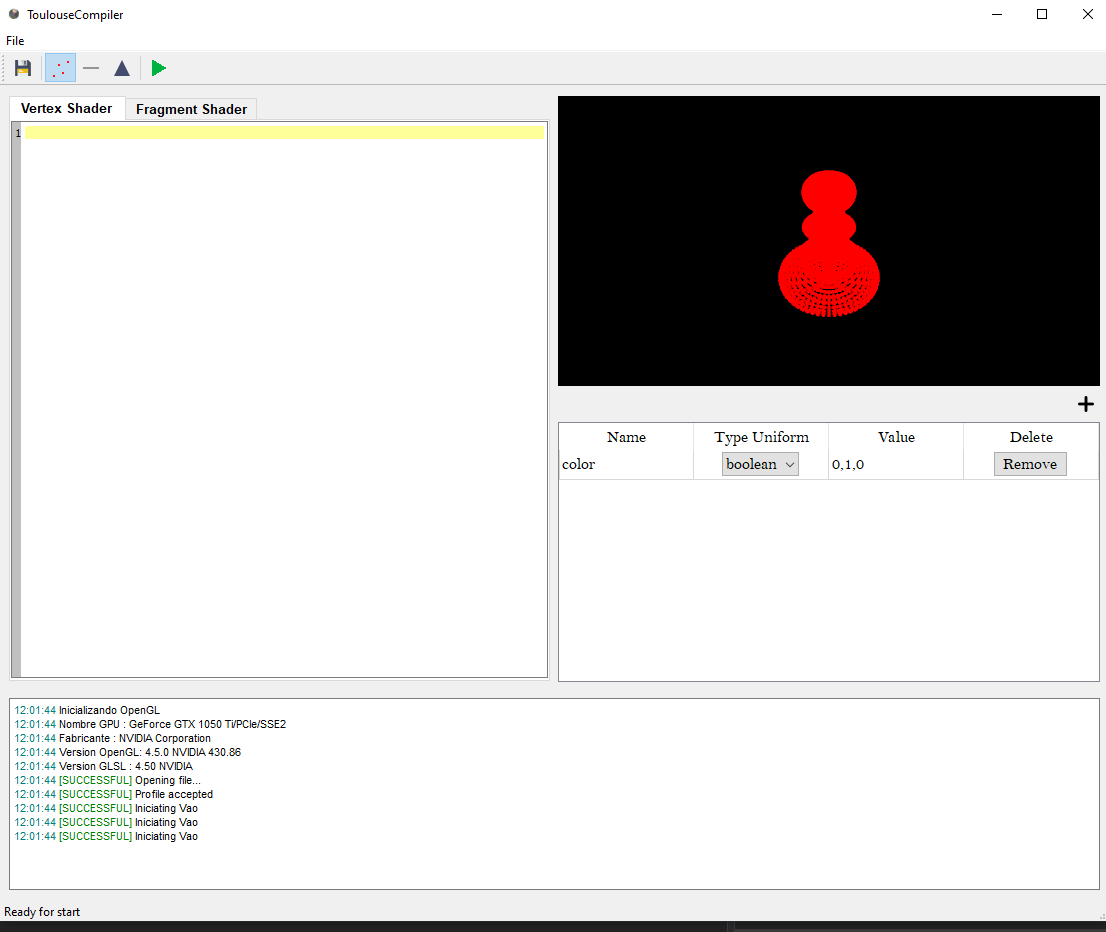


Ilustración 7 Aplicación con cámara modificada

## Quinta Iteración

Ya he conseguido aprender cómo funciona QT en cuanto la maquetación de los widgets, así que aumento el espacio para visualización del objeto 3D y muevo la parte de los uniforms a otra ventana, para así conseguir ese espacio, dentro de una pestaña de herramientas.

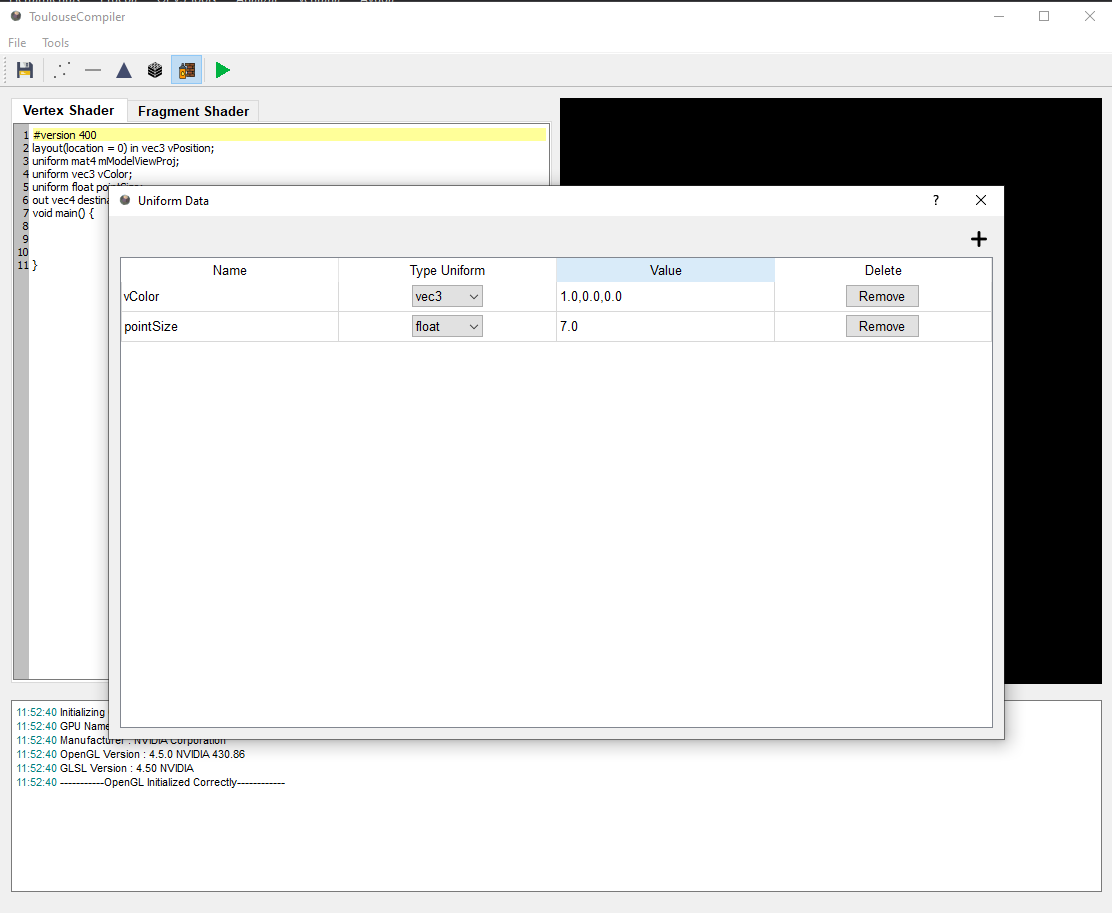


Ilustración 8 Ejemplo de la nueva tabla de uniforms

Implemento la parte de las luces, materiales y texturas, de esta forma ya se pueden probar los shaders pertinentes.

También de implementa el funcionamiento de escribir los shaders y ser recogidos por el sistema, por lo tanto, se deja de lado la parte de pruebas y ya se pasa a abrir los shaders desde el explorador de archivos.

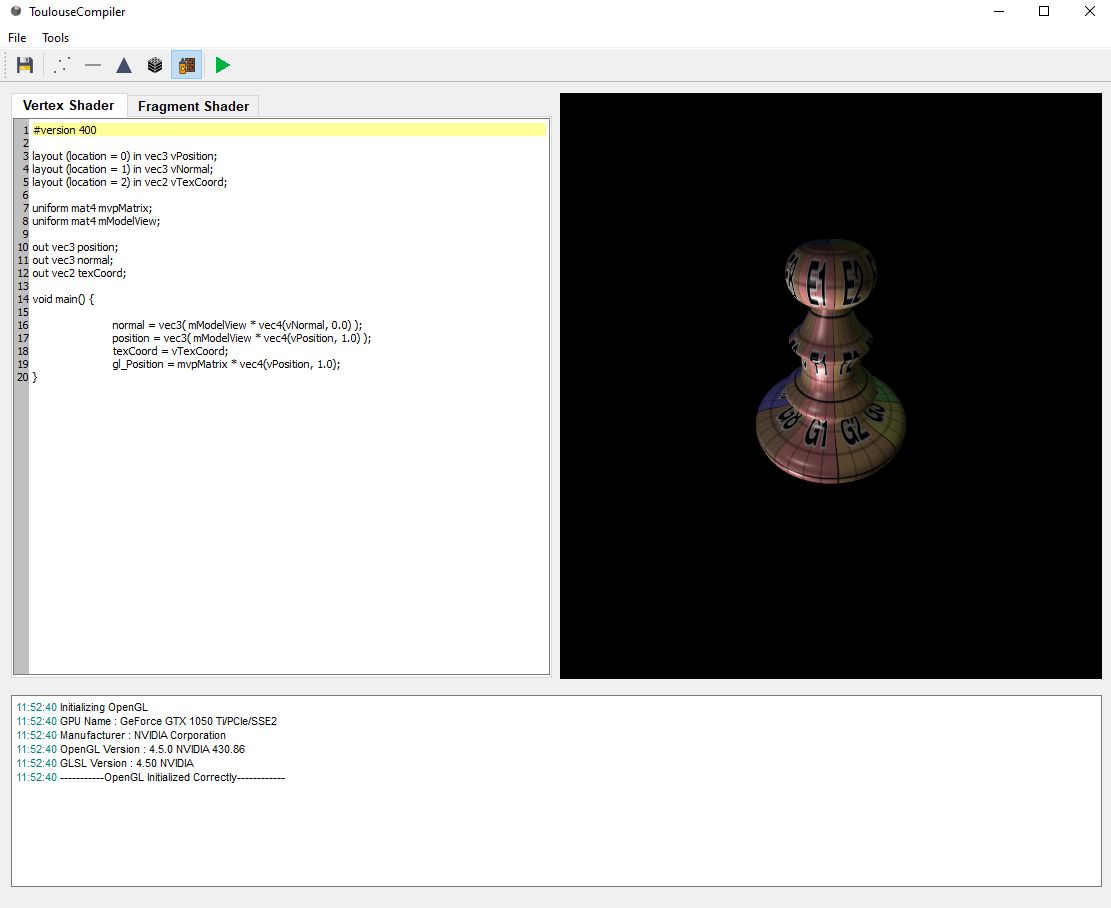


Ilustración 9 Aplicación con más espacios

Se puede ir cambiando de tipo de shader de forma sencilla, pero se detecta un error visual al mover la cámara 3D.

## Sexta Iteración

Aquí se pasa más a la parte visual de la aplicación, por lo que se añaden 2 objetos más para probar el efecto de los shaders. Además, consigo arreglar el error visual producido por un orden incorrecto en la inicialización del programa.

Se empieza a preparar el programa para la elección de idioma, pero prefiero dejarlo para otra iteración. Se diseñan nuevos iconos para el IBO de puntos, líneas y triángulos:



Ilustración 10 Iconos para tipo de IBO

Se crean mensajes de error para posibles inserciones de uniforms incorrectos.

Por otra parte, se añaden una nueva interfaz en la parte de la tabla de uniforms.

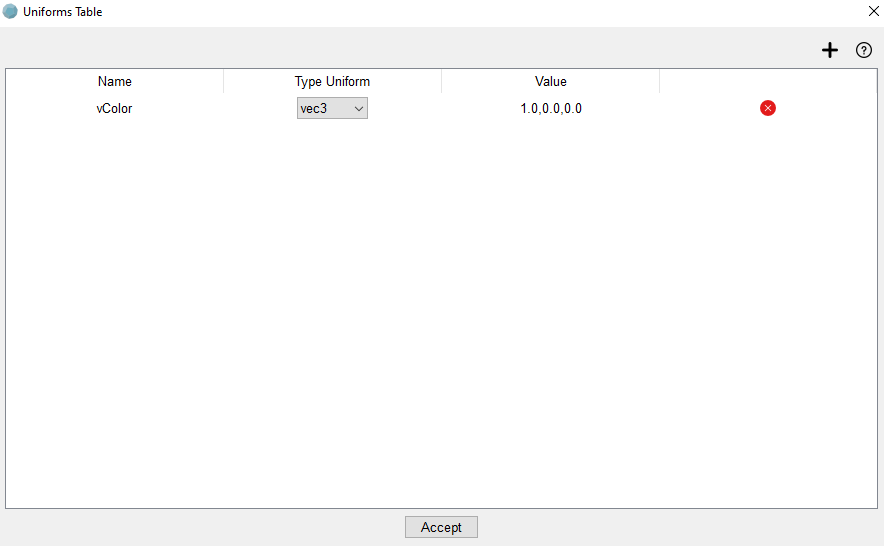


Ilustración 11 Tabla de uniforms

Resumiendo, la aplicación queda así:

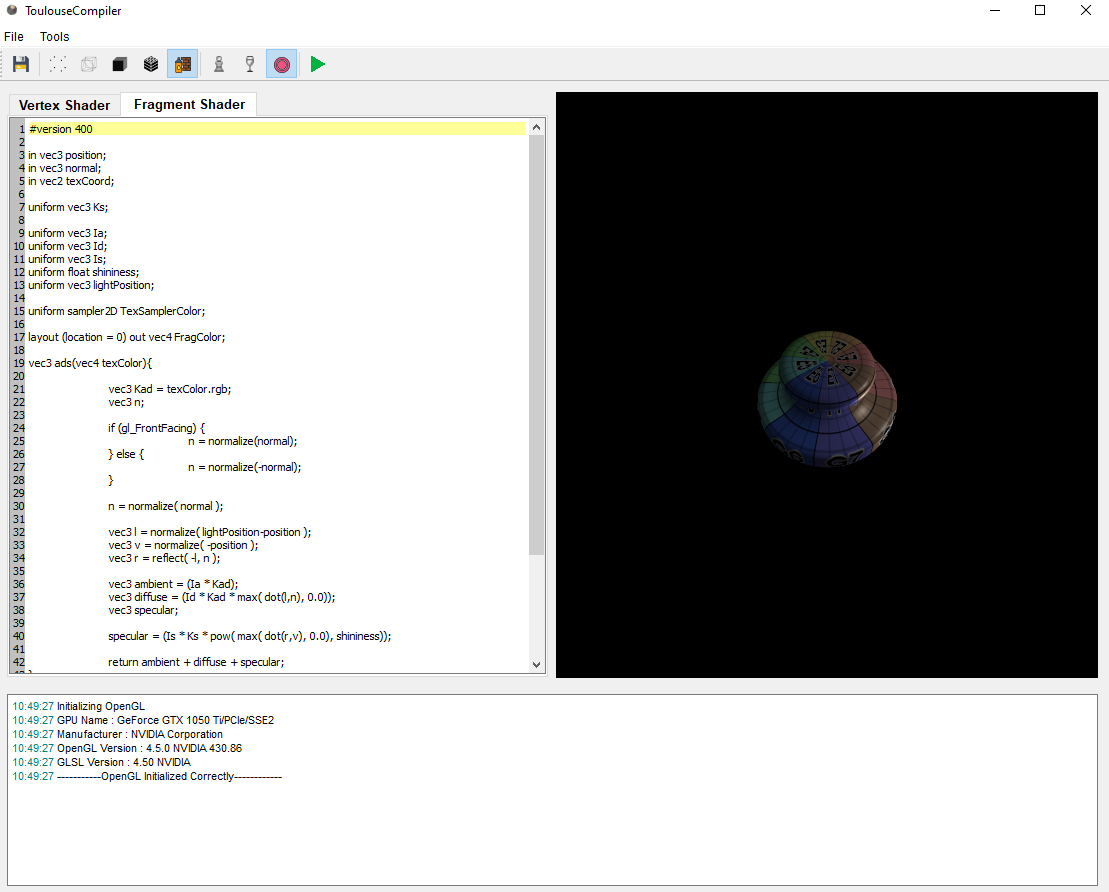


Ilustración 12 Aplicación casi final

## Última Iteración

Se incluye el resaltado de las palabras reservadas a la hora de escribir los shaders.

También se lleva a cabo a el menú para seleccionar idioma, así como ciertos retoques en la visualización de errores de los shaders.