Escuela Politécnica Superior de Jaén
Grado en Ingeniería Informática

Universidad de Jaén
Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

Telescopio Virtual

Alumno

Juan José Cruz Lorite

Tutores

Lidia Ortega Alvarado
(Departamento de Informática)

José Luis Garrido Pestaña
(Departamento de Física)

Septiembre, 2021

When I heard the learn'd astronomer,

When the proofs, the figures, were ranged in columns before me,

When I was shown the charts and diagrams, to add, divide, and measure them,

When I sitting heard the astronomer where he lectured with much applause in the lecture-room,

How soon unaccountable I became tired and sick,

Till rising and gliding out I wander'd off by myself,

In the mystical moist night-air, and from time to time,

Look'd up in perfect silence at the stars.

Walt Whitman

Índice

1. Especificación del trabajo **6**
1.1. Introducción . . . 6
1.2. Objetivos del trabajo . . . 6
1.3. Antecedentes y estado del arte . . . 7
1.3.1. Antecedentes: Catálogos de estrellas . . . 7
1.3.2. Estado del arte: Software astronómico . . . 7
1.4. Entorno de partida . . . 11
1.5. Requisitos iniciales . . . 12
1.5.1. Representación espacial del dataset . . . 12
1.5.2. Movimiento de las estrellas . . . 12
1.5.3. Filtros . . . 12
1.5.4. Resaltar distintos parámetros . . . 13
1.5.5. Otras representaciones del dataset . . . 13
1.6. Hipótesis y restricciones . . . 13
1.7. Descripción de la solución propuesta . . . 13
1.8. Tecnologías utilizadas . . . 14
1.9. Metodología de desarrollo de software . . . 15
1.10. Estimación de tamaño y esfuerzo . . . 16
1.11. Planificación temporal . . . 17
1.12. Presupuesto . . . 18

2. Diseño inicial **20**
2.1. Especificaciones del sistema . . . 20
2.1.1. 1ª Iteración . . . 20
2.1.2. 2ª Iteración . . . 20
2.1.3. 3ª Iteración . . . 21
2.1.4. 4ª Iteración . . . 22
2.1.5. 5ª Iteración . . . 22
2.2. Análisis y diseño del sistema . . . 22
2.2.1. Diseño inicial de clases . . . 22

3. Desarrollo **23**
3.1. 1ª Iteración . . . 23
3.1.1. Análisis y diseño . . . 23
3.1.2. Carga de datos . . . 25
3.1.3. Análisis del sistema de coordenadas . . . 25
3.1.4. Análisis del brillo . . . 27
3.1.5. Análisis del color . . . 30
3.1.6. Cámara libre . . . 32
3.1.7. Modo de cámara centrada en el origen . . . 33
3.1.8. Pruebas de verificación y validación . . . 34
3.2. 2ª Iteración . . . 34
3.2.1. Análisis y diseño . . . 34
3.2.2. Movimiento de las estrellas . . . 36
3.2.3. Implementación del movimiento . . . 39
3.2.4. Planos fundamentales . . . 40

3.2.5. Pruebas de verificación y validación . . . 45
3.3. 3ª Iteración . . . 46
3.3.1. Análisis y diseño . . . 46
3.3.2. Filtros . . . 47
3.3.3. Exportar selección . . . 49
3.3.4. Visualización 2D . . . 50
3.3.5. Diagrama Hertzsprung–Russell . . . 51
3.3.6. Pruebas de verificación y validación . . . 53
3.4. 4ª Iteración . . . 53
3.4.1. Análisis y diseño . . . 53
3.4.2. Colorización por parámetros . . . 54
3.4.3. Información estadística . . . 56
3.4.4. Selección de estrellas . . . 57
3.4.5. Modo de cámara orbit . . . 59
3.4.6. Pruebas de verificación y validación . . . 59
3.5. 5ª Iteración . . . 60
3.5.1. Filtro de distancia automático . . . 60
3.5.2. Nombres de estrellas . . . 62
3.5.3. Formato FITS . . . 63

4. Conclusiones y trabajos futuros **64**
4.1. Didáctica . . . 64
4.2. Investigación . . . 64
4.3. Trabajos futuros . . . 65

5. Manual de usuario **66**
5.1. Instalación y puesta en marcha . . . 66
5.2. Descargar conjuntos de datos . . . 66
5.3. Cargar conjuntos de datos . . . 68
5.4. Ajustando la visualización . . . 69

6. Definiciones y abrevieturas **69**
Definiciones . . . 69
Abreviaturas . . . 71

7. Bibliografía **72**

Índice de figuras

1. Paisaje de un planeta generado con Space Engine [18] 8
2. Imagen de Plutón en Celestia [6] 9
3. La Tierra en Open Space [15] 9
4. Visualización del Octree generado por Gaia Sky [16] 10
5. Práctica de Algoritmos Geométricos, 2019 11
6. Diagrama UML preliminar 23
7. Estructura interna del VBO en la 1ª iteración 24
8. Casos de uso en la 1ª iteración 24
9. Sistema de coordenadas ecuatoriales 26
10. Relación del paralaje con la distancia 26
11. Representación de distintos valores de magnitud aparente 28
12. Funciones gaussianas 29
13. Sliders para configurar la visualización 30
14. Relación entre temperatura y color sobre el diagrama CIE 31
15. Valores de temperatura y color para incrementos del *Índice de color B-V* [7] 32
16. Visualización del catálogo de pruebas de GEDR3 en cámara libre . . . 33
17. Visualización de HYG en el modo de cámara centrada en el origen . . . 34
18. Diseño UML en la 2ª Iteración 35
19. Casos de uso en la 2ª iteración 36
20. Estrella de Barnard, estrella con mayor movimiento propio 36
21. Componentes del movimiento propio 37
22. Parámetros relativos al movimiento de las estrellas 38
23. Vectores de velocidad 40
24. Coordenadas horizontales [4] 42
25. Planos ecuatorial y eclíptico 43
26. Coordenadas galácticas 44
27. Imagen saturada por los vectores 45
28. Visualización con límite de distancia y transparencia 46
29. Diseño UML en la 3ª Iteración 46
30. Casos de uso en la 3ª iteración 47
31. Interfaz para modificar los filtros 48
32. Visualización aplicando filtro de color y distancia 49
33. Diálogo estándar para cargar/guardar/cerrar ficheros 49
34. Visualización 2D de un catálogo en modo ecuatorial y galáctico 51
35. Diagrama HR con estrellas de Hipparcos y Gliese [10] 52
36. Diagrama HR con distintas configuraciones de atenuación de transparencia 52
37. Casos de uso en la 4ª iteración 54
38. Estrellas coloreadas por velocidad 55
39. Interfaz de colorización por filtros 55
40. Representación del brillo sobre una proyección 2D del dataset 56
41. Representación de la distancia sobre el diagrama HR 56
42. Diálogo de información estadística 57
43. Rendering visual y rendering interno al hacer click 58
44. Estrella seleccionada y sus parámetros 58

45. Tasas de FPS para distintas configuraciones 61
46. Visualización del nombre de algunas estrellas 62
47. Diálogo de selección de formato 64
48. Interfaz gráfica de Gaia Archive 67
49. Interfaz ADQL de Gaia Archive 68
50. Diálogo para tratar campos nulos 69

Índice de tablas

1. Principales catálogos de estrellas 7
2. Historia de usuario: Visualización espacial 16
3. Historia de usuario: Movimiento de las estrellas 16
4. Historia de usuario: Filtros 16
5. Historia de usuario: Distintas representaciones 17
6. Historia de usuario: Resaltar distintos parámetros 17
7. Diagrama Gantt 18
8. Presupuesto estimado 19

1. Especificación del trabajo

En este capítulo se presenta la especificación del presente trabajo de fin de grado, 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 cuya descripción aparecen en la *Sección 6 Definiciones y abrevieturas*.

1.1. Introducción

El presente trabajo consiste en el desarrollo de un software gráfico, que denominaremos Telescopio Virtual (TV), orientado a la visualización de datos sobre las estrellas del firmamento y a medir distintas magnitudes físicas relativas a las estrellas y a otros muchos objetos interesantes, mayoritariamente en nuestra galaxia, la Vía Láctea, pero también en otras galaxias.

La Agencia Espacial Europea (ESA) lleva bastante tiempo trabajando en varias misiones para cartografiar y medir distintos parámetros relativos a las estrellas, como el brillo, el color, la velocidad o la distancia a la que se encuentran.

Cada cierto tiempo, la ESA publica la última actualización del catálogo. Actualmente, es la misión GAIA la que se encarga de actualizar el catálogo en base a los últimos datos recibidos por el satélite destinado a ello (también llamado Gaia). La última actualización del catálogo es la Gaia Early Data Release 3 (GEDR3), que contiene en torno a unas ~ 1.800 millones de estrellas. La ESA permite descargar este catálogo o subconjuntos del mismo mediante el servicio Gaia Archive [1], una sencilla aplicación web que nos facilita los datos para ser descargados en un fichero que será la entrada de datos de nuestro software.

1.2. Objetivos del trabajo

El principal objetivo de este trabajo es el de facilitar una herramienta orientada a un uso científico y didáctico, que permita obtener diferentes representaciones visuales de datasets de la última publicación de GAIA. Se debe dar una representación espacial del dataset con una navegación en 3D para moverse libremente por el espacio, así

como diferentes representaciones de los principales parámetros de las estrellas.

El software debe ofrecer una interfaz de usuario intuitiva y fácil de utilizar, pensando siempre en ofrecer la mejor experiencia de usuario posible; pero al mismo tiempo ha de ser robusta y rica en características, lo cual distinguirá su aplicabilidad.

Será una aplicación de escritorio y debe estar disponible para los sistemas operativos Windows y Linux.

1.3. Antecedentes y estado del arte

1.3.1. Antecedentes: Catálogos de estrellas

En 1997 la ESA publicó los resultados de la misión Hipparcos. Un satélite que durante un periodo de tres años y medio estuvo cartografiando la bóveda celeste. Tras un análisis de los datos registrados por el satélite, la ESA publicó dos catálogos. El catálogo Hipparcos con unas ~ 120.000 estrellas cuyos parámetros se registraron con alta precisión. Y el catálogo Tycho, mucho más grande (~ 1 millón de estrellas), pero de menor precisión en la medida de los parámetros. En el año 2000 publicaron una actualización de este segundo catálogo llegando a los ~ 2.5 millones de estrellas, pero igualmente la precisión en los parámetros seguía siendo baja. [2]

El satélite Gaia tomó el relevo y su primera publicación marcó un antes y un después en la astronomía. Hasta entonces solo podíamos hablar de unos cientos de miles de estrellas bien catalogadas, o de un par de millones siendo laxos con la precisión. Desde la primera publicación de Gaia en 2016 contamos las estrellas en miles de millones.

Catálogo	Año de publicación	Número de estrellas
Hipparcos	1997	118 218
Tycho		1 058 332
Tycho 2	2000	2 539 913
GDR1	2016	1 142 679 769
GDR2	2018	1 692 919 135
GEDR3	2020	1 811 709 771

Tabla 1: Principales catálogos de estrellas

1.3.2. Estado del arte: Software astronómico

Ya existen muchos programas de astronomía que trabajan con estos catálogos. Vamos a repasar el estado del arte de los principales programas que existen en este

campo.

- **Space Engine**

  Éste es un software privado, está disponible en Steam para Windows y destaca notablemente por su apartado gráfico. Utiliza efectos gráficos propios de los videojuegos como sistemas de partículas o entornos generados procedimentalmente para crear mundos extraterrestres (ver figura 1). Permite viajar a distintos planetas, estrellas o galaxias y utiliza Hipparcos como catálogo de estrellas. Space Engine es un software más divulgativo que científico.

Figura 1: Paisaje de un planeta generado con Space Engine [18]

- **Celestia**

  Celestia es un software de código abierto multiplataforma. También utiliza el catálogo Hipparcos para representar las estrellas, pero este software está mucho más centrado en el sistema solar. Utiliza texturas de altísima resolución para representar la superficie de los planetas o satélites como se aprecia en la figura 2. Cuenta con una comunidad muy activa que contribuye a crear extensiones fáciles de instalar para añadir nuevos elementos y nuevas funciones. Celestia es un software que no maneja catálogos tan grandes como nuestro TV.

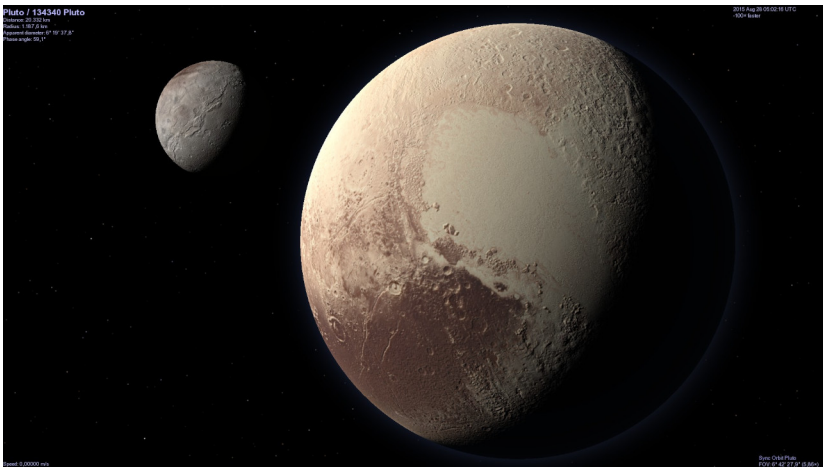

Figura 2: Imagen de Plutón en Celestia [6]

- **Open Space**

  Open Space es uno de los programas más completos en tanto que contempla todos los aspectos del universo que nos podamos imaginar. Hace uso de distintos catálogos para representar distintos tipos de astros, desde asteroides hasta galaxias pasando por satélites artificiales. Utiliza el subconjunto de estrellas para las que se conoce la velocidad radial en GDR2 como catálogo por defecto, unas ~ 7 millones de estrellas. Open Space es un proyecto en versión beta, también es de código abierto y puede ejecutarse en varias plataformas. Open Space, similarmente a Space Engine, tiene muchos detalles más divulgativos que científicos. La siguiente figura muestra una visualización de la Tierra en este software.

Figura 3: La Tierra en Open Space [15]

- **Gaia Sky**

Gaia Sky es junto con Open Space de los programas más completos y relevantes. También es de código abierto y multiplataforma. Permite obtener distintas visualizaciones del universo en todas sus escalas y como su nombre indica, está preparado para trabajar con cualquier catálogo de Gaia. La principal característica de éste software es que está especialmente diseñado para ser capaz de ofrecer representaciones del catálogo incluyendo la mayor cantidad de estrellas posibles en la visualización. Siendo capaz de cargar catálogos increíblemente grandes. Para hacerlo construye un Octree (ver figura 4) para almacenar las estrellas. Los niveles más altos del Octree contienen a las estrellas más brillantes y los niveles más bajos a las menos brillantes. Este programa es capaz de dar resultados visuales espectaculares, pero como los mismos autores explican en su publicación *Gaia Sky: Navigating the Gaia Catalog* [16], obtener estas visualizaciones con catálogos tan grandes solo es posible si disponemos de un muy buen hardware. En el artículo explican como consiguen obtener en un proceso de 6 horas una visualización de un subconjunto de ~ 600 millones de estrellas en un sistema con 2TB de memoria RAM. Es capaz de mantener entre 150 y 200 FPS con catálogos de hasta ~ 5 millones de estrellas.

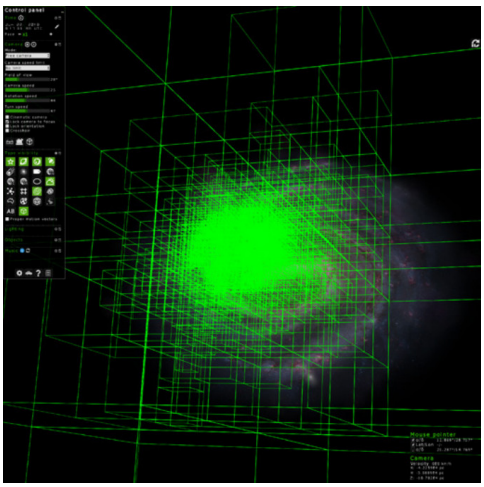

Figura 4: Visualización del Octree generado por Gaia Sky [16]

Nuestro software no aspira a alcanzar los increíbles resultados gráficos que muestran los programas mencionados, pero pretende ser un simulador de todo el universo. Nuestra aplicación no se ceñirá únicamente al estudio de las estrellas, nuestro software estará más orientado a permitir analizar muy distintos grupos de estrellas y otros objetos como por ejemplo quásares y a obtener distintas representaciones de los data-

sets que puedan ser de gran interés científico en los campos, no sólo de la astronomía de posición, sino sobre todo en los de la astrofísica galáctica y extragaláctica.

1.4. Entorno de partida

Para este trabajo se cuenta con cierta experiencia previa por una práctica realizada para la asignatura de Algoritmos Geométricos (ver figura 5) en la que se trabajó con el catálogo de estrellas HYG. Esta práctica fue implementada en Java con Legacy OpenGL. Esta aplicación era muy básica y no tenía muchas funciones, no ofrecía una representación espacial del dataset ni permitía configurar la visualización o estudiar otros parámetros.

El objetivo en aquella práctica era el de obtener una simple visualización de las estrellas desde la perspectiva de la Tierra. El interés de la práctica residía en el uso de estructuras de datos espaciales para realizar búsquedas por rangos de coordenadas y filtrar eficientemente solo las estrellas que pertenecían a la ventana de visión. Las coordenadas de ascensión recta y declinación eran interpretas como puntos sobre un plano bidimensional. Se utilizó un árbol 2-dimensional (KD-Tree) para realizar eficientemente una búsqueda sobre el catálogo solo en el rango de interés. Por lo que ya estamos familiarizados a trabajar con este tipo de información.

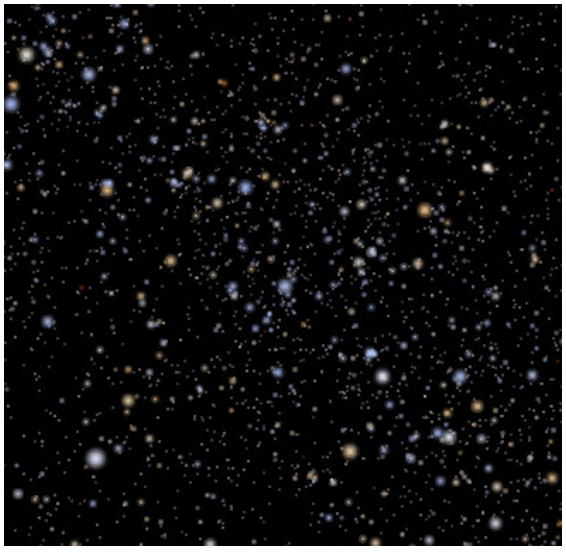

Figura 5: Práctica de Algoritmos Geométricos, 2019

Las misiones de la ESA no se encargan de catalogar las estrellas más brillantes. Los sensores de los satélites son altamente sensibles y están diseñados para detectar

precisamente las estrellas menos brillantes. Si estos satélites intentaran observar alguna estrella muy brillante como las que podemos ver a simple vista en el cielo nocturno, el sensor quedaría dañado e inservible.

Por esta razón, si queremos obtener una visualización de la bóveda celeste tal y como la vemos desde la Tierra tenemos que recurrir a catálogos como HYG [13]. Este catálogo es la unión del catálogo Hipparcos con otros dos catálogos que recogen a las estrellas más brillantes, *Yale Bright Star* y *Gliese Catalog of Nearby Stars*.

1.5. Requisitos iniciales

A continuación se enumeran los requisitos principales a alto nivel que el software debe tener, fijando los requisitos esperados para cada uno de los entregables.

Una especificación más en detalle de estos requisitos se expone en la *Subsección 2.1 Especificaciones del sistema*.

1.5.1. Representación espacial del dataset

El software debe ser capaz de permitir explorar cualquier dataset de GEDR3 de forma gráfica. Mediante una visualización en 3D y una interacción que permita al usuario moverse libremente por el espacio para visualizar en detalle cualquier sección del espacio que resulte de interés. También se debe poder configurar la visualización para realzar el brillo o el tamaño, y se deben mostrar los parámetros de una estrella concreta al hacer click sobre ella.

1.5.2. Movimiento de las estrellas

Para aquellas estrellas donde se conozca, además de la distancia, los parámetros de *Movimiento Propio* y *Velocidad Radial*, el software debe ofrecer una visualización del movimiento de las estrellas a lo largo del tiempo.

1.5.3. Filtros

También se deben poder filtrar las estrellas a visualizar mediante filtros para los principales parámetros. Permitiendo ajustar la visualización al conjunto de estrellas

que cumplan ciertas condiciones, como por ejemplo, visualizar solo las estrellas que estén a una distancia concreta.

1.5.4. Resaltar distintos parámetros

Se debe poder modificar la visualización de las estrellas para resaltar algún parámetro concreto. Por ejemplo, modificar el color de las estrellas en base a su brillo para visualizar las más brillantes de un color y las más tenues de otro.

1.5.5. Otras representaciones del dataset

Además de una visualización espacial, el software también debería ofrecer otras representaciones del dataset como una proyección en 2D del catálogo, o el *Diagrama Hertzsprung–Russell* [11]. Además para el modo en 3D también se debe poder alternar entre los diferentes sistemas de referencia utilizados en astronomía, como son los terrestres horizontal y ecuatorial, el eclíptico solar y el galáctico; lo cual es algo de capital importancia para estudios no sólo galácticos sino también incluso cosmológicos.

1.6. Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

1.7. Descripción de la solución propuesta

Para llevar a cabo este proyecto, el planteamiento será un proyecto en C++ con OpenGL. Utilizaremos CLion como entorno de desarrollo (IDE), este entorno está enfocado en los lenguajes C y C++ . La compilación será gestionada con CMake, esta herramienta nos abstrae el proceso de compilación a un simple fichero `CMakeLists.txt` donde especificaremos los ficheros fuentes del proyecto y algunas opciones de compilación.

Algunas funciones, como por ejemplo, las encargadas de leer y escribir ficheros, deberán ser sobrescritas. Es decir, tendremos que implementar dos versiones diferentes de la misma función para cumplir con el objetivo de compatibilidad en Windows y Linux. Especificaremos los condicionales necesarios tanto en el código como en el fichero `CMakeLists.txt` para que el compilador sepa qué función debe considerar dependiendo del sistema operativo en el que se esté compilando la aplicación. Con la idea de que el mismo proyecto pueda compilarse en ambos sistemas sin tocar una sola línea del código.

1.8. Tecnologías utilizadas

- **Lenguaje de programación**

  Como lenguaje de programación hemos optado por **C++** . La experiencia previa con proyectos OpenGL en distintas asignaturas ha sido principalmente en este lenguaje, por lo que estamos más familiarizados con él. Una alternativa contemplada fue Java porque la aplicación sería multiplataforma de facto, pero realmente los cambios necesarios en el código para adaptar la aplicación a distintos sistemas operativos no son tantos. Eso junto al factor de la experiencia previa terminaron descartando esta opción.

- **Gráficos**

  OpenGL será la API que utilizaremos para renderizar los gráficos. Una alternativa podría haber sido Vulkan que está más orientado al mundo de los videojuegos y ofrece muchas funciones avanzadas como el Ray Tracing. Pero para los objetivos de nuestra aplicación, no buscamos opciones gráficas tan avanzadas y además, por el mismo argumento de la experiencia previa, esta opción queda descartada.

- **Gestor de eventos y ventanas**

  Graphics Library Framework (**GLFW**) es la librería que suele usarse de forma estándar para gestionar los eventos en proyectos OpenGL. Nos facilita la tarea de detectar las pulsaciones del teclado o los eventos del ratón, así como la gestión de las ventanas.

- **Interfaz**

Para la interfaz de usuario nos apoyaremos en **ImGui** [8]. Esta opción destaca frente a otras alternativas como Qt por su sencillez, incluir sliders, botones, menús desplegables u otros elementos es extremadamente fácil. En muy pocas líneas de código se pueden construir interfaces muy completas y con una apariencia visual agradable.

- **Librería de inicio de OpenGL**

  Esta librería es necesaria para iniciar el contexto OpenGL al iniciar la aplicación. Normalmente utilizaríamos OpenGL Extension Wrangler Library (GLEW), pero resulta ser incompatible con ImGui, en su lugar utilizaremos **Glad** [9].

- **Librería matemática**

  OpenGL Mathematics (**GLM**) está concebida para trabajar en contextos OpenGL, facilita muchísimas operaciones matemáticas y funciones muy prácticas para obtener las matrices de visión y proyección necesarias en el proceso de rendering.

- **Carga de datos**

  En la etapa final del desarrollo del proyecto se incluyen las librerías **CCfits** y **CFITSIO** para poder cargar ficheros en formato FITS. Los motivos que justifican el uso de este formato se detallan en la *Subsubsección 3.5.3 Formato FITS*.

1.9. Metodología de desarrollo de software

La descripción de nuestros requisitos no plantea muchas dependencias, más bien son una serie de funcionalidades independientes entre ellas. Por lo que prácticamente podemos implementarlas en cualquier orden, y en cualquier momento podemos tener que añadir o modificar alguna funcionalidad que no estuviera prevista. Esta flexibilidad que tenemos por la naturaleza de nuestros requisitos hace que optemos por una metodología ágil. Repartiremos las funcionalidades a implementar en las distintas iteraciones que consideremos necesarias. Cada iteración tiene un análisis propio sobre las funcionalidades que se vayan a implementar. En lugar de tener un análisis y diseño exhaustivo de todo el proyecto antes de empezar con la implementación, en esta metodología de desarrollo, este análisis y diseño se va completando conforme se van implementando las iteraciones.

1.10. Estimación de tamaño y esfuerzo

En el desarrollo ágil una de las formas de estimar el esfuerzo es mediante las historias de usuario. Éstas describen muy brevemente ciertas características que el software debe tener en un lenguaje llano desde la perspectiva del usuario final. A cada historia se le asignan unos puntos de esfuerzo que representan la dificultad de implementar dicha característica. Cuanta más dificultad suponga la implementación de la característica, más puntos de esfuerzo se le asignan a la historia de usuario. Sirve como base para la estimación temporal del proyecto. Basándonos en estas historias, el esfuerzo total es de unos **39** puntos de esfuerzo. A continuación mostramos las historias de usuario para las distintas iteraciones.

- **1ª Iteración**

Visualización espacial	
Como	Usuario
Quiero	Una visualización 3D de las estrellas
Para	Viajar por el espacio y explorar distintas regiones
Puntos	**12**

Tabla 2: Historia de usuario: Visualización espacial

- **2ª Iteración**

Movimiento de las estrellas	
Como	Usuario
Quiero	Ajustar la velocidad para avanzar o retroceder en el tiempo
Para	Ver dónde estuvieron las estrellas en el pasado y donde estarán en el futuro
Puntos	**7**

Tabla 3: Historia de usuario: Movimiento de las estrellas

- **3ª Iteración**

Filtros	
Como	Usuario
Quiero	Descartar las estrellas cuyos parámetros estén fuera del rango que yo decida
Para	Ajustar la visualización a las estrellas que cumplan un criterio concreto
Puntos	**5**

Tabla 4: Historia de usuario: Filtros

Distintas representaciones	
Como	Usuario
Quiero	Otras representaciones del dataset
Para	Estudiar el dataset desde otras perpectivas
Puntos	**10**

Tabla 5: Historia de usuario: Distintas representaciones

- **4ª Iteración**

Resaltar distintos parámetros	
Como	Usuario
Quiero	Resaltar en la visualización los valores de algún parámetro concreto
Para	Estudiar en detalle la distribución de valores para dicho parámetro
Puntos	**5**

Tabla 6: Historia de usuario: Resaltar distintos parámetros

1.11. Planificación temporal

El desarrollo de este proyecto comienza a mediados de Mayo y debe estar finalizado para finales de Agosto. Este periodo supone un plazo de unos 3 meses y medio o unas 14 semanas. Nuestras principales tareas en la planificación temporal serán 4 iteraciones que se corresponderán con los requisitos básicos expuestos en la *Subsección 1.5 Requisitos iniciales*. Cada iteración conlleva un tiempo de análisis e investigación sobre los conceptos teóricos implicados en la implementación de las funcionalidades demandadas por los distintos requisitos. Al tratarse de un desarrollo iterativo, la tarea de análisis e investigación estará presente en cada iteración, será una tarea paralela a todo el desarrollo del proyecto. Adicionalmente, incorporaremos una 5ª iteración para tratar posibles bugs o posibles cambios finales, terminar de pulir algún detalle o introducir alguna funcionalidad imprevista. La propia documentación de la memoria es también una tarea que estará presente durante todo el desarrollo. Asignaremos a cada iteración una duración proporcional a sus puntos de esfuerzo.

	20/5/2021 - 31/8/2021													
Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Planificación e investigación														
1ª Iteración														
2ª Iteración														
3ª Iteración														
4ª Iteración														
5ª Iteración														
Documentación														

Tabla 7: Diagrama Gantt

1.12. Presupuesto

A continuación vamos a dar una estimación del presupuesto desglosado por partidas.

- **Gasto hardware**

  El único hardware involucrado en la elaboración del proyecto es un ordenador portátil marca Acer modelo Aspire E1-571, procesador Intel i7-3612QM 2.1 GHz, 8GB de RAM, 750GB de disco duro, pantalla de 15,6 pulgadas con un valor de 578,00 €.

- **Gasto software**

  En este apartado debe incluirse el coste de cualquier software involucrado en el desarrollo del proyecto. Principalmente suelen incluirse el coste de las licencias de los diferentes programas que se hayan utilizado. En nuestro caso, todo el software utilizado ha sido gratuito y aquellos programas que requieren de una licencia de pago hemos podido obtenerlos mediante la licencia de estudiante que facilita la universidad, como ha ocurrido con el entorno de desarrollo CLion.

  Por tanto, en nuestro caso esta partida no supone ningún coste.

- **Gasto de personal**

  Para fijar el presupuesto de personal se tomará como referencia la *Resolución de 22 de febrero de 2018, de la Dirección General de Empleo, por la que se registra y publica el XVII Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública* [22] publicada en el Boletín Oficial

del Estado. Este proyecto solo cuenta con un programador para su realización. El salario base establecido en este convenio para el perfil de Analista programador es de 21.555,66 €$/año$ para un trabajo a jornada completa. En nuestro caso aplicaremos el salario equivalente a media jornada para los meses estimados de duración del proyecto.

$$\frac{21.555,66\ €}{2} \cdot \frac{3,5\ meses}{12\ meses} = 3.143,53\ €$$

- **Amortización**

  Tanto el coste hardware como software pueden amortizarse. Esto se hace para calcular el desgaste de los activos debido a su uso en el desarrollo del proyecto.

  Para el hardware y el software la Agencia Tributaria establece los coeficientes en el 20 % y 33 % [19] respectivamente para una amortización lineal.

  En nuestro caso solo podemos amortizar el gasto hardware. Primero calculamos la amortización anual con este coeficiente.

$$20\,\%\ 578,00\ € = 115,6\ €$$

  Y la ajustamos a la duración del proyecto.

$$Amortización = 115,6\ € \cdot \frac{3,5\ meses}{12\ meses} = 33,71\ €$$

- **Tabla resumen**

Concepto	Coste
Hardware	578,00 €
Software	0,00 €
Personal	3.143,53 €
Amortización	33,71 €
Total	**3.755,24 €**

Tabla 8: Presupuesto estimado

2. Diseño inicial

2.1. Especificaciones del sistema

A continuación se profundiza en los detalles de los principales requisitos que el software debe tener para cada uno de los entregables separados por iteraciones.

2.1.1. 1ª Iteración

Para la primera iteración crearemos la estructura básica del proyecto. Cargaremos un catálogo e implementaremos una representación 3D del mismo. Asignaremos a las estrellas un tamaño y un color en función de sus parámetros y también implementaremos los siguientes modos de cámara.

- **Cámara geocéntrica**

  En este modo se visualizará la bóveda celeste desde la perspectiva de la Tierra. Una interacción básica con el ratón debe permitir cambiar la ventana de visión y hacer zoom para poder explorar la bóveda celeste de manera fácil e intuitiva.

- **Cámara libre**

  El usuario puede moverse libremente por el espacio. Con el ratón se cambia la dirección hacia donde se mira. Con el teclado se controla el desplazamiento con las teclas WASD, tal y como suele hacerse de forma estándar para controlar el movimiento en cualquier software 3D y en los videojuegos.

También es interesante que se pueda configurar la visualización de las estrellas a gusto del usuario: modificar el tamaño del punto que representa a la estrella, alternar entre usar un tamaño proporcional al brillo de la estrella o usar un tamaño fijo para todas las estrellas, o aplicar un factor arbitrario de atenuación en función de la distancia, son características que el software debe tener. Esta iteración se centra especialmente en aspectos de visualización.

2.1.2. 2ª Iteración

- **Movimiento de las estrellas**

Los parámetros de movimiento propio y velocidad radial permiten calcular la velocidad (relativa al Sol) a la que se mueve una estrella. Por tanto, el software tendrá la opción de mostrar una animación del movimiento de las estrellas a lo largo del tiempo. La velocidad que obtenemos es una instantánea que se corresponde al periodo durante el que el satélite Gaia realizó la medida. Pero sabemos que la influencia gravitatoria va modificando esta velocidad, en consecuencia, nuestra animación no será físicamente exacta, pero al menos para periodos cortos de tiempo será una aproximación razonable.

- **Sistemas de coordenadas**

  En astronomía, no hay un solo sistema de coordenadas, además del ecuatorial, existen otros muchos como el horizontal, eclíptico, galáctico o galactocéntrico. Cada uno tiene su propio plano fundamental y el software debe permitir cambiar de un sistema a otro.

2.1.3. 3ª Iteración

En esta iteración implementaremos los **filtros** por parámetros para descartar las estrellas fuera del rango de valores que decida el usuario. Los parámetros que vamos a considerar son: distancia al Sol, brillo, color y velocidad.

Aparte de la visualización en 3D, el software debe ser capaz de ofrecer otras representaciones del conjunto de datos con el que el usuario esté trabajando. Incluiremos las siguientes representaciones del catálogo.

- **Proyección 2D**

  Para poder visualizar todo el conjunto de datos a la vez utilizando los parámetros de ascensión recta y declinación como puntos (x, y) sobre el plano, las cuales son las coordenadas polares que se usan mayoritariamente en astronomía.

- **Diagrama Hertzsprung-Russell**

  El diagrama de Hertzsprung-Russell (o diagrama HR) representa las estrellas sobre un plano bidimensional, ordenadas en el eje horizontal por su color, y en el eje vertical por su brillo. Es de gran utilidad en astronomía estelar y no puede faltar en nuestro software.

2.1.4. 4ª Iteración

En esta iteración nos centraremos en el estudio de los parámetros. Cumpliremos con el requisito de ofrecer una visualización gráfica para resaltar los valores de los distintos parámetros con los que estamos trabajando. Mostraremos alguna información estadística básica sobre los mismos.

También implementaremos una solución para que el usuario pueda hacer click sobre una estrella para seleccionarla y mostrar sus parámetros.

2.1.5. 5ª Iteración

Dejaremos una última iteración para solucionar posibles bugs, hacer algún cambio pertinente de última hora, o implementar alguna característica que pueda suponer una mejora en nuestro software.

2.2. Análisis y diseño del sistema

La clase Renderer gestionará toda la lógica de la aplicación. Como en cualquier software interactivo, estamos aplicando el paradigma de programación dirigida por eventos. El método `main()` de nuestra aplicación se limitará a gestionar el bucle básico de ejecución; hacer una llamada al Renderer para que renderice la escena, detectar los eventos e invocar a los distintos Callbacks que a su vez, invocarán a métodos del Renderer.

2.2.1. Diseño inicial de clases

Al trabajar con un modelo de desarrollo incremental basado en iteraciones, el diseño definitivo se irá perfilando poco a poco en cada iteración. La figura 6 muestra el diagrama de clases UML preliminar sin entrar mucho en detalles.

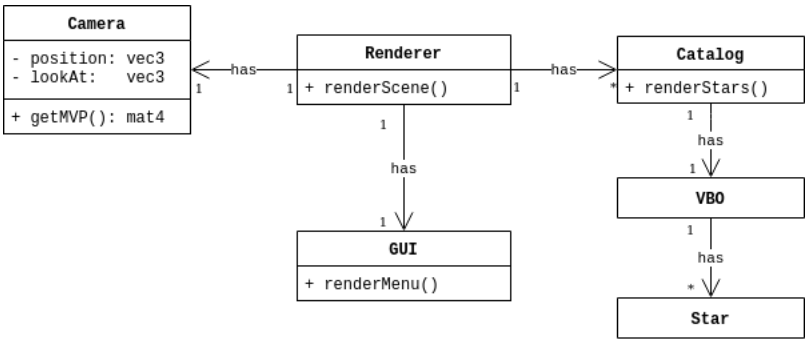

Figura 6: Diagrama UML preliminar

3. Desarrollo

En esta sección entraremos en los detalles específicos de la implementación de todas las funcionalidades que ya hemos especificado.

3.1. 1ª Iteración

En esta primera iteración se creará la primera versión del proyecto. Como punto de partida, el objetivo es crear la estructura básica de un proyecto OpenGL, cargar un catálogo, visualizar las estrellas de dicho catálogo en 3D, e implementar una cámara libre básica para que el usuario pueda explorar el espacio de forma sencilla.

3.1.1. Análisis y diseño

Para visualizar el catálogo, deberemos almacenarlo en la memoria de la tarjeta gráfica (VRAM) utilizando un *Vertex Buffer Object* (VBO), la estructura que *OpenGL* provee para gestionar vértices, normales o cualquier parámetro que influya en la visualización de la escena a renderizar.

Para visualizar el catálogo, basta con renderizar la posición de las estrellas como una nube de puntos. Almacenaremos las coordenadas cartesianas de cada estrella en el VBO. Además de la posición, también necesitaremos asignarle a los puntos diferentes tamaños en función de su brillo gracias al parámetro de *Magnitud Aparente*, sin embargo, utilizaremos la *Magnitud Absoluta* en lugar de la *Magnitud Aparente* por razones detalladas en la *Subsubsección 3.1.4 Análisis del brillo*. La distancia a la que nos encontremos de la estrella también influye en el tamaño que debemos asignar al

punto, por tanto, este tamaño lo calcularemos en función de la posición de la cámara dentro del *Vertex Shader*. También le asignaremos un color (R, G, B) a cada punto gracias al parámetro *Índice de color B-V*.

Al renderizar las estrellas como una nube de puntos, nuestro VBO no necesitará ningún *Index Buffer Object* (IBO) auxiliar.

Considerando estos tres parámetros, posición, brillo y color, la siguiente figura muestra la estructura que tendrá nuestro VBO, y más específicamente, la clase Star.

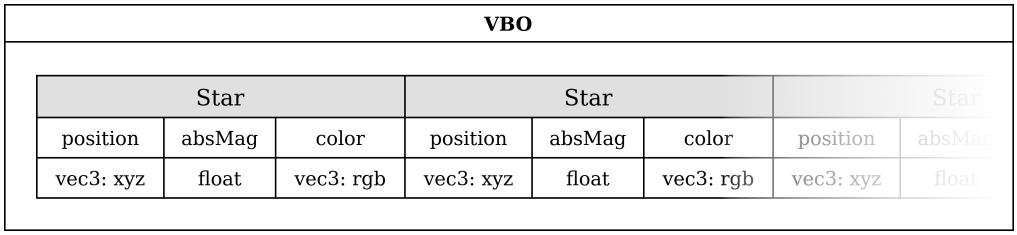

Figura 7: Estructura interna del VBO en la 1ª iteración

Sin embargo, los datasets no ofrecen la posición de la estrella en coordenadas cartesianas, sino en coordenadas esféricas, la *Magnitud Aparente* no puede utilizarse directamente para asignarle un radio al punto, y el *Índice de color B-V* no es un valor (R, G, B).

En las siguientes subsecciones analizamos cómo transformar estos parámetros del dataset para almacenarlos en el VBO.

La siguiente figura muestra los casos de uso que vamos a tratar en esta iteración. En nuestro caso solo tenemos el rol del usuario final.

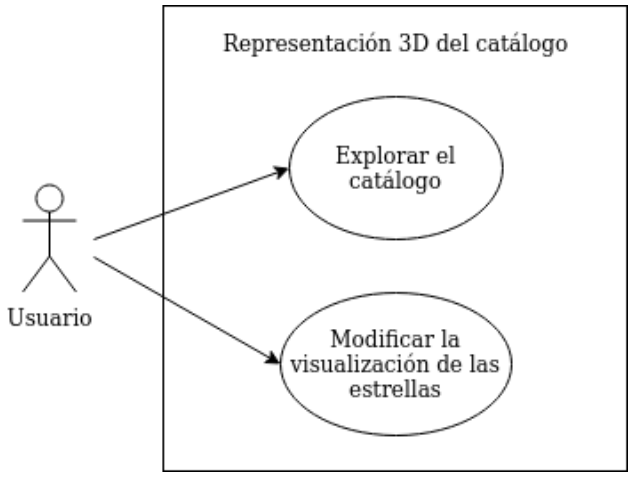

Figura 8: Casos de uso en la 1ª iteración

3.1.2. Carga de datos

Aunque ya sabemos que el objetivo es trabajar con catálogos de GEDR3, a modo de catálogo por defecto utilizaremos HYG. Para ir haciendo pruebas con catálogos de GEDR3 nos descargaremos un conjunto de medio millón de estrellas desde Gaia Archive ordenadas por el parámetro Random Index. La utilidad de este Random Index es que al ordenar las estrellas por este parámetro en la consulta a la base de datos, tenemos garantizado que el conjunto que obtendremos será estadísticamente representativo del conjunto total.

Nos descargaremos este catálogo en un fichero CSV, y para cargar ambos catálogos solo tendremos que parsear estos ficheros. Hay que tener en cuenta que los nombres de las columnas son distintos en HYG y en GEDR3.

3.1.3. Análisis del sistema de coordenadas

El sistema de coordenadas ecuatoriales utiliza la Tierra como referencia. Una proyección del ecuador terrestre sobre la bóveda celeste define el ecuador celeste. El plano que contiene a este ecuador es el plano fundamental del sistema de coordenadas. La ubicación de la estrella viene dada en forma de coordenadas esféricas, ascensión recta (α) y declinación (δ) como ilustra la figura 9.

La ascensión recta representa un ángulo a lo largo del ecuador celeste. Su valor crece hacia el Este, está dentro del rango $[0, 24]$ y suele expresarse en formato de horas, minutos y segundos. Este valor lo pasaremos a radianes en el rango $[0, 2\pi]$. La declinación representa el ángulo perpendicular al ecuador celeste, valores positivos se ubican en el hemisferio norte y los negativos en el hemisferio sur. A diferencia de la ascensión recta, ésta se mide en grados sexagesimales siendo su rango $[-90, 90]$, lo cambiaremos al rango $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

La dirección principal del sistema se define con el equinoccio de primavera. Es decir, la zona del cielo a la que apuntan las coordenadas $(0, 0)$ coincide con la dirección Sol-Tierra justo en el momento del equinoccio.

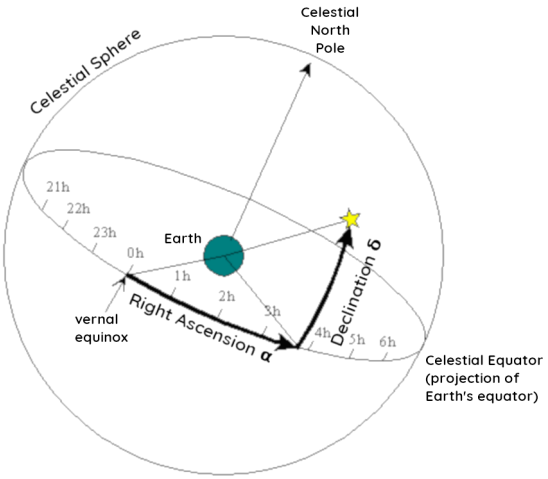

Figura 9: Sistema de coordenadas ecuatoriales

Además de estos dos ángulos, para ubicar a la estrella en el espacio, necesitamos saber a qué distancia se encuentra. En el catálogo HYG, la distancia viene dada directamente como atributo, pero en Gaia, en lugar de eso tenemos el paralaje (ver figura 10). El paralaje es la diferencia entre el ángulo que podemos medir al observar una estrella un día concreto del año, y el ángulo que medimos de esa misma estrella cuando la Tierra esté en el punto diametralmente opuesto de su órbita transcurridos seis meses.

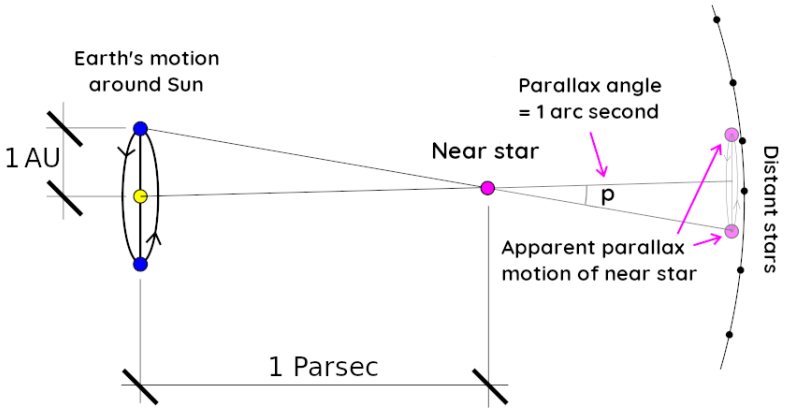

Figura 10: Relación del paralaje con la distancia

Estos ángulos son extremadamente pequeños, del orden de los milisegundos de arco (mas), cuanto más lejos se encuentre una estrella, menor será su ángulo de paralaje. Como la distancia Tierra-Sol es conocida, la trigonometría esférica nos permite calcular la distancia (D). Dado que las distancias en astronomía con increíblemente grandes, se utiliza una nueva unidad de medida, el Parsec. El Parsec como unidad de

medida astronómica se definió como la distancia a la que esté un objeto con 1 arco-segundo de paralaje, de ahí su nombre ***Par**allax of one arc **sec**ond*. De esta forma la distancia es tan solo la inversa del paralaje en arcos de segundo. Como Gaia facilita el paralaje en milisegundos de arco tendremos que multiplicar por 1000.

$$Distance\ (parsecs) = \frac{1000}{Parallax\ (mas)}$$

Para ubicar las estrellas en el espacio necesitamos transformar estos parámetros o coordenadas esféricas en coordenadas cartesianas $(D, \alpha, \delta) \rightarrow (x, y, z)$. Las fórmulas para calcularlas son bastante sencillas aunque han de ser manejadas cuidadosamente.

$$x = D \cdot \cos(-\alpha) \cdot \cos(\delta)$$
$$y = D \cdot \sin(\delta)$$
$$z = D \cdot \sin(-\alpha) \cdot \cos(\delta)$$

3.1.4. Análisis del brillo

Para determinar el brillo de una estrella disponemos del parámetro de la *Magnitud Aparente*, que representa cuán brillante aparenta ser una estrella observada desde la Tierra. Este valor está en una escala inversa y logarítmica (ver figura 11). Esto quiere decir que los valores más pequeños representan a las estrellas más brillantes, los valores más grandes representan a las menos brillantes, y una diferencia de 1.0 en este valor se corresponde con un factor de brillo de $\sqrt[5]{100} \approx 2.512$. Es decir, una estrella de magnitud n es 2.512 veces más brillante que una de $n + 1$, y 100 veces más brillante que una de $n+5$. El Sol tiene una magnitud de -26.7. Sirio, la estrella más brillante del cielo nocturno, tiene una magnitud de -1.46. Las estrellas más tenues que podemos observar a simple vista sin ayuda de telescopios tienen una magnitud de $\sim +6.0$. El satélite Gaia detecta hasta una magnitud de aproximadamente $\sim +21$, espectacularmente débil.

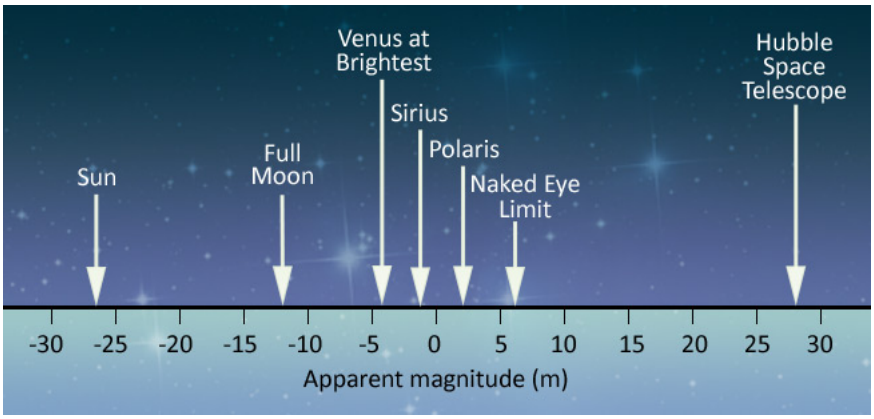

Figura 11: Representación de distintos valores de magnitud aparente [20]

Este parámetro por sí solo no sirve para discernir qué estrella es más brillante. Al ser una medida del brillo aparente que observamos desde nuestra perspectiva en la Tierra, no podemos asegurar si una estrella aparenta ser muy brillante por estar muy cerca de nosotros o porque realmente es muy brillante. Para solventar este problema, tenemos el parámetro de la *Magnitud Absoluta*, que se define como la magnitud que mediríamos de una estrella si estuviera a una distancia de exactamente 10 parsecs.

Si conocemos la *Magnitud Aparente* (m) y la distancia (D) a la que se encuentra una estrella, podemos calcular su *Magnitud Absoluta* (M) mediante la siguiente fórmula [3].

$$m - M = 5 \log \frac{D}{10}$$

Almacenaremos la *Magnitud Absoluta* en el VBO y en el *Vertex Shader* volveremos a utilizar esta fórmula para calcular la nueva *Magnitud Aparente* en función de la posición de la cámara. Una vez tenemos la *Magnitud Aparente*, hay que obtener un radio para especificarle a *OpenGL* el tamaño del punto a renderizar para representar la estrella, y esto puede hacerse de varias maneras.

Thor Olson en su publicación *The Colors of the Stars* [14] aborda ampliamente varios aspectos que inciden en la visualización de las estrellas, y señala el problema de relegar únicamente en el tamaño del punto para representar el brillo, las estrellas más brillantes tienden a ser desproporcionadamente grandes. También señala que en lugar de un punto, las estrellas deben ser representadas con una función de dispersión

de punto. Es decir, en lugar de puntos, deben ser halos más opacos en el centro y más transparentes hacia la circunferencia.

Existen tantas alternativas y fórmulas para renderizar las estrellas como papers se han escrito al respecto, las hay de mayor y menor complejidad y las hay más o menos realistas desde el punto de vista físico. En nuestro caso no buscamos una representación totalmente fidedigna a la realidad, pero sí buscamos un resultado visualmente agradable, y sobre todo riguroso para hacer análisis astrofísicos. Una aproximación sencilla de implementar y que pueda ser fácilmente configurable por el usuario.

Implementaremos una atenuación del radio con la fórmula de la función gaussiana (ver figura 12).

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

Donde a es el máximo de la función, b es el centro de la campana gaussiana, y c determina la anchura de la campana.

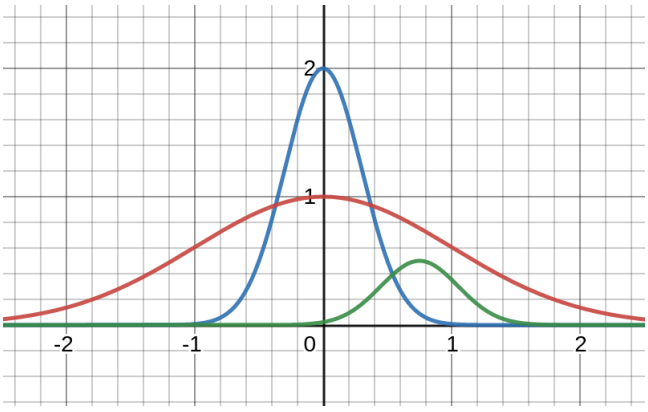

Figura 12: Funciones gaussianas para distintos valores de a, b y c. En rojo a=1 b=0 c=1, en verde a=0.5 b=0.75 c=0.3 y en azul a=2 b=0 c=0.3

La interfaz permitirá al usuario modificar el valor de estas tres variables para configurar el comportamiento de la función (ver figura 13).

Utilizando la *Magnitud Aparente* calculada en el *Vertex Shader* como variable x, obtenemos una atenuación adecuada para el tamaño del punto. Solo nos interesa utilizar esta función en los valores de x positivos, o mejor dicho, cuando $x \geq b$, para atenuar entre el máximo de la campana (a) y 0. Cuando el valor de *Magnitud Aparente* esté en la mitad izquierda de la campana $(x < b)$ asignaremos directamente el máximo de la campana, de esta forma, el parámetro b establece el tamaño máximo que se le puede

asignar a una estrella. La interfaz también permitirá la opción de renderizar la escena asignando un tamaño de punto fijo para todas las estrellas sin considerar su brillo.

Además del tamaño del punto, para representar gráficamente una estrella, también consideraremos asignar una opacidad, un valor de transparencia (canal alfa) en función de la distancia a la que nos encontremos de la estrella, y renderizaremos halos en lugar de puntos. Del mismo modo que con el tamaño, el valor de transparencia y el halo también tendrán sus funciones gaussianas asociadas y la interfaz permitirá configurarlas también.

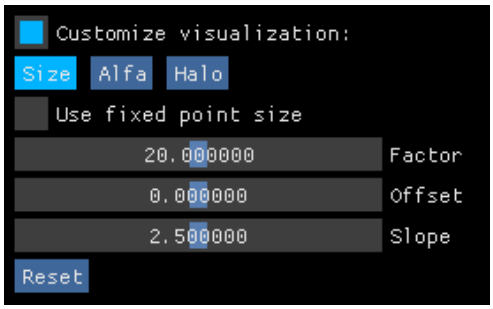

Figura 13: Sliders para configurar la visualización

3.1.5. Análisis del color

Para asignar un color a las estrellas tenemos el parámetro *Índice de color B-V*, un valor numérico que es la diferencia entre la magnitud medida utilizando filtros en distintas longitudes de onda, la azul B (442 nm) y la visible V (540 nm), en ocasiones a la visible también se la conoce como verde G. Este valor sirve para determinar el color o la temperatura de una estrella. Valores más bajos se corresponden con estrellas más azules y calientes, valores más altos indican estrellas más rojas y frías. Este índice también sirve para clasificar a las estrellas según su tipo espectral, una clasificación que asigna a cada estrella una letra en función del valor de este índice. La siguiente fórmula nos permite estimar la temperatura T de la estrella en grados Kelvin a partir de este índice.

$$T = 4600 \left(\frac{1}{0.92(B-V)+1.7} + \frac{1}{0.92(B-V)+0.62} \right)$$

Con la temperatura podemos obtener las coordenadas (x, y) del color que debería tener la estrella sobre el diagrama CIE [12], tal y como ilustra la siguiente figura.

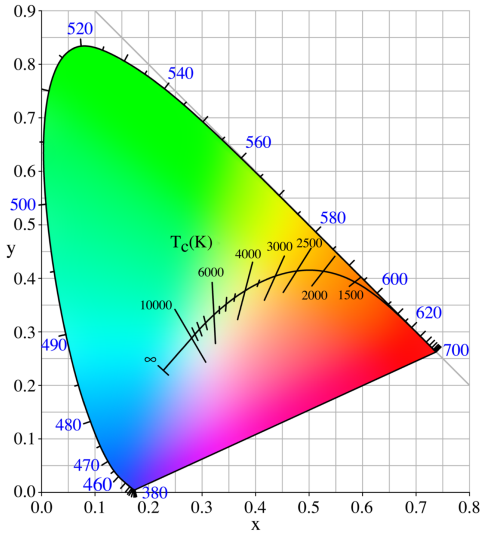

Figura 14: Relación entre temperatura y color sobre el diagrama CIE

$$x = \begin{cases} -0.2661239\frac{10^9}{T^3} - 0.2343589\frac{10^6}{T^2} + 0.8776956\frac{10^3}{T} + 0.179910 & 1667 \leq T \leq 4000 \\ -3.0258469\frac{10^9}{T^3} + 2.1070379\frac{10^6}{T^2} + 0.2226347\frac{10^3}{T} + 0.240390 & 4000 \leq T \leq 25000 \end{cases}$$

$$y = \begin{cases} -1.1063814x^3 - 1.34811010x^2 + 2.18555832x - 0.20219683 & 1667 \leq T \leq 2222 \\ -0.9549476x^3 - 1.37418593x^2 + 2.09137015x - 0.16748867 & 2222 \leq T \leq 4000 \\ +3.0817580x^3 - 5.87338670x^2 + 3.75112997x - 0.37001483 & 4000 \leq T \leq 25000 \end{cases}$$

Estas variables (x, y) junto con otra variable Y forman el espacio de color xyY, donde $Y \in [0, 1]$ representa la luminancia del color. Necesitamos convertir estas variables al espacio de color XYZ, donde la variable Y es la misma que en el espacio xyY y se le suele asignar el valor 1 para no alterar su luminosidad, y las variables X y Z se obtienen con las siguientes fórmulas.

$$X = \frac{Y}{y}x \quad , \quad Z = \frac{Y}{y}(1 - x - y)$$

Una vez tenemos el color en el espacio XYZ podemos hacer la transformación al

espacio RGB multiplicando por una matriz [14].

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 3.275 & -1.557 & -0.511 \\ -0.781 & 1.693 & 0.045 \\ -0.030 & -0.119 & 1.054 \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Pero en lugar de realizar estos cálculos, optaremos por partir de la tabla que muestra la figura 15 con los valores RGB ya calculados para incrementos de 0.05 en el *Índice de color B-V* en el rango $[-0.4, 2]$. Para hayar el color de un índice concreto interpolaremos linealmente entre los valores más cercanos y para los índices fuera del rango asignaremos los valores de los extremos.

-0.40	113017	#9bb2ff	0.25	7483	#eeefff	0.90	5052	#ffe8ce	1.55	3892	#ffd29c
-0.35	56701	#9eb5ff	0.30	7218	#f3f2ff	0.95	4948	#ffe6ca	1.60	3779	#ffd096
-0.30	33605	#a3b9ff	0.35	6967	#f8f6ff	1.00	4849	#ffe5c6	1.65	3640	#ffcc8f
-0.25	22695	#aabfff	0.40	6728	#fef9ff	1.05	4755	#ffe3c3	1.70	3463	#ffc885
-0.20	16954	#b2c5ff	0.45	6500	#fff9fb	1.10	4664	#ffe2bf	1.75	3234	#ffc178
-0.15	13674	#bbccff	0.50	6285	#fff7f5	1.15	4576	#ffe0bb	1.80	2942	#ffb765
-0.10	11677	#c4d2ff	0.55	6082	#fff5ef	1.20	4489	#ffdfb8	1.85	2579	#ffa94b
-0.05	10395	#ccd8ff	0.60	5895	#fff3ea	1.25	4405	#ffddb4	1.90	2150	#ff9523
-0.00	9531	#d3ddff	0.65	5722	#fff1e5	1.30	4322	#ffdbb0	1.95	1675	#ff7b00
0.05	8917	#dae2ff	0.70	5563	#ffefe0	1.35	4241	#ffdaad	2.00	1195	#ff5200
0.10	8455	#dfe5ff	0.75	5418	#ffeddb	1.40	4159	#ffd8a9			
0.15	8084	#e4e9ff	0.80	5286	#ffebd6	1.45	4076	#ffd6a5			
0.20	7767	#e9ecff	0.85	5164	#ffe9d2	1.50	3989	#ffd5a1			

Figura 15: Valores de temperatura y color para incrementos del *Índice de color B-V* [7]

Esta alternativa resulta menos compleja que realizar el cálculo exacto del color en el espacio CIE, y el resultado visual es prácticamente idéntico.

3.1.6. Cámara libre

La implementación de la cámara es trivial, tendremos una clase Camera que encapsule los parámetros de posición y orientación de la cámara. Hay que implementar los métodos de Dolly y Truck para el movimiento y los métodos de Tilt y Pan para cambiar de dirección [21]. En la gestión de eventos de GLFW deberemos invocar adecuadamente a estos métodos en las teclas WASD y en el desplazamiento del ratón cuando se mueva con el click izquierdo pulsado.

Para explorar el espacio cómodamente con esta cámara es imprescindible poder controlar fácilmente la velocidad de desplazamiento. En zonas de menor densidad de

estrellas nos interesará movernos más rápido y viceversa. La cámara tendrá un parámetro de velocidad (10 parsecs/s por defecto) y el evento de mover la rueda del ratón hacia arriba o hacia abajo, incrementará o decrementará este parámetro en un 15 %. Incorporaremos una pequeña ventana en la esquina inferior derecha mostrando la localización de la cámara para ayudar al usuario a ubicarse. Al hacer click sobre esta ventana alternamos entre las coordenadas esféricas o cartesianas. La siguiente figura muestra una representación de un catálogo tras haber implementado todos los aspectos que hemos analizado.

Figura 16: Visualización del catálogo de pruebas de GEDR3 en cámara libre

3.1.7. Modo de cámara centrada en el origen

También es interesante un modo de cámara para visualizar la bóveda celeste tal y como la vemos desde la Tierra, o bien desde el Sol o desde el centro de la Vía Láctea. En este modo la cámara está ubicada en el origen de coordenadas y no puede moverse, solo se puede cambiar la dirección hacia donde mira con el ratón. En este modo, la ventana que implementamos mostrando las coordenadas de la cámara, ahora mostrará las coordenadas hacia donde la cámara está mirando, justo el centro de la pantalla. El movimiento de la rueda del ratón servirá en este caso para hacer zoom modificando el campo de visión de la cámara (FOV), de forma similar a lo que hicimos con la velocidad en la cámara libre. La interfaz dará opción de activar un grid (ver figura

17). Unas líneas que dividen el cielo a modo de cuadrícula y facilitan al usuario localizar unas coordenadas concretas.

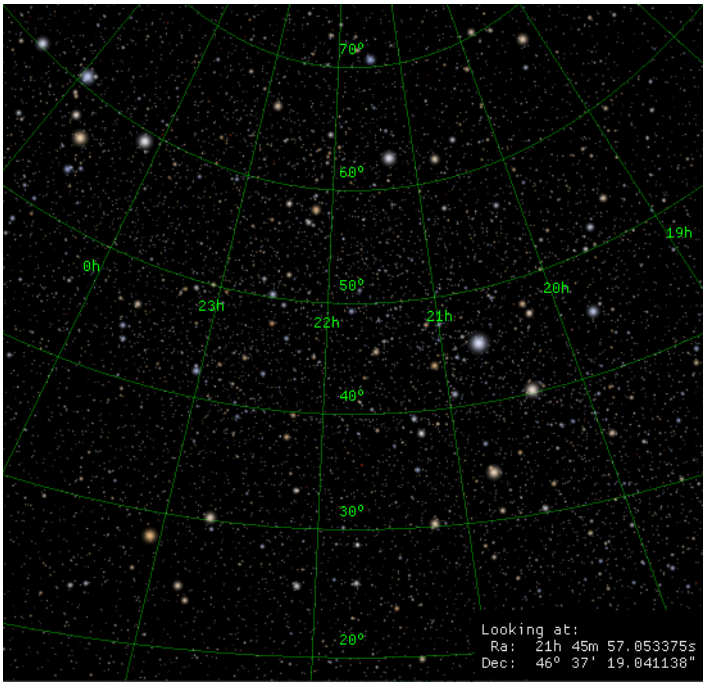

Figura 17: Visualización de HYG en el modo de cámara centrada en el origen

3.1.8. Pruebas de verificación y validación

Un problema que hemos detectado con la cámara libre es que al aumentar la velocidad en un 15 % cada vez que el usuario mueve la rueda del ratón hacia arriba, se puede conseguir muy rápidamente y sin mucho esfuerzo aumentar la velocidad hasta tal punto que desborden las coordenadas (x, y, z) de la cámara, pasando a ser valores nan, generando un bug importante. Simplemente limitaremos la velocidad máxima de la cámara a $100.000\ parsecs/s$, más que suficiente para poder explorar cualquier región del catálogo y demasiado poco como para desbordar las coordenadas de la cámara fácilmente.

3.2. 2ª Iteración

3.2.1. Análisis y diseño

En esta iteración debemos considerar nuevos parámetros relativos a la velocidad. La interfaz de usuario empieza a tener más importancia y tiene que modificar paráme-

tros propios de la clase Renderer. Surge el problema de la inclusión mutua, el Renderer tiene que poder invocar a la GUI para renderizar el menú, y la GUI tiene que invocar al Renderer para modificar sus parámetros. Para evitar esta situación vamos a encapsular todos los parámetros relevantes en una única instancia de la nueva clase Parameters (ver figura 18). El Renderer y la GUI accederán a esta instancia para operar con los parámetros.

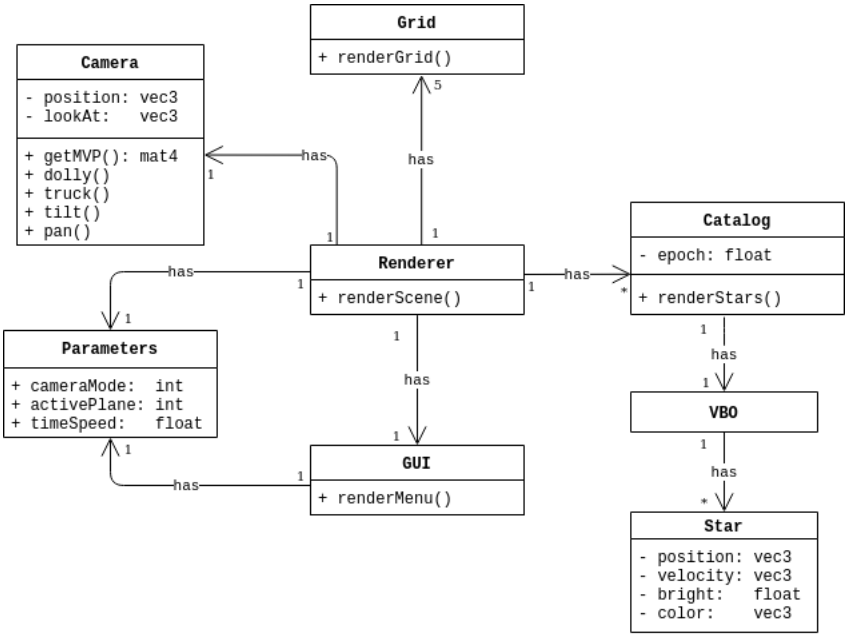

Figura 18: Diseño UML en la 2ª Iteración

En esta figura vemos los casos de uso que quedarán resueltos en esta iteración.

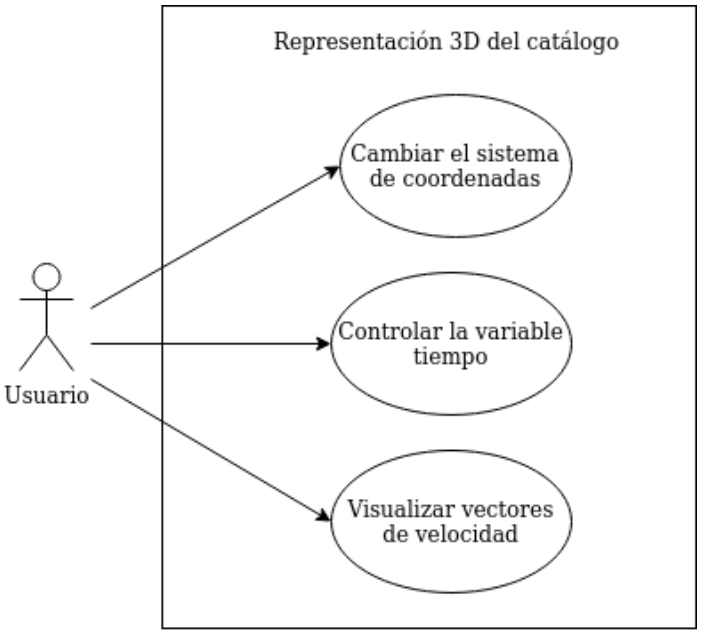

Figura 19: Casos de uso en la 2ª iteración

3.2.2. Movimiento de las estrellas

Todas las estrellas se mueven en alguna dirección a cierta velocidad. Esta velocidad podemos obtenerla a partir de las dos componentes que podemos medir de este movimiento, que son la velocidad angularmente proyectada en un plano tangente a la esfera celeste en cada objeto, y la velocidad en la dirección radial. Describiremos a continuación la primera componente y en el último párrafo de esta Sección la segunda. El movimiento de las estrellas puede observarse en el Movimiento Propio (μ) que apreciamos al observarlas durante largos periodos de tiempo (ver figura 20). Este parámetro se obtiene midiendo un ángulo que indica cuántos grados se ha desplazado la estrella sobre el firmamento durante un periodo de tiempo.

Figura 20: Estrella de Barnard, estrella con mayor movimiento propio

Se suele medir en milisegundos de arco al año y nos sirve para obtener la Velocidad Transversal, una de las dos componentes que necesitamos para obtener la velocidad real.

El Movimiento Propio se expresa con dos parámetros, descompuesto en las dos coordenadas del sistema ecuatorial. Se denominan movimiento propio en ascensión (μ_α), y movimiento propio en declinación (μ_δ). Para hallar el módulo del movimiento propio recurrimos al teorema de Pitágoras.

$$\mu^2 = {\mu_\delta}^2 + {\mu_\alpha}^2 \cdot \cos^2 \delta$$

El factor $\cos^2 \delta$ se debe a que trabajar con coordenadas esféricas es similar a trabajar sobre la superficie de una esfera, no es un espacio euclídeo. Los meridianos se van estrechando al acercarnos a los polos, por lo que el valor del movimiento propio en ascensión debe ser corregido considerando la declinación a la que se encuentre la estrella. Para simplificar esta expresión los astrofísicos definen otra versión del movimiento propio en ascensión.

$$\mu_{\alpha*} = \mu_\alpha \cdot \cos \delta$$

Éste valor ($\mu_{\alpha*}$) ya está corregido considerando la declinación, y es directamente el valor que ofrece Gaia para simplificarnos la fórmula.

$$\mu^2 = {\mu_\delta}^2 + {\mu_{\alpha*}}^2$$

Esta figura ilustra las componentes del movimiento propio.

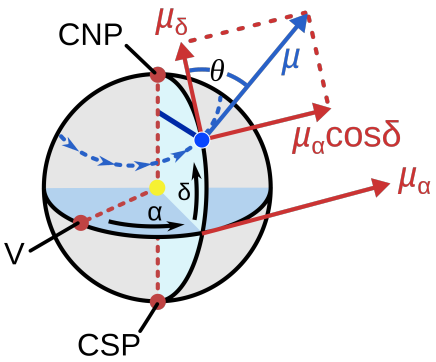

Figura 21: Componentes del movimiento propio

Una vez que tenemos el módulo del movimiento propio tenemos que transformarlo

a un valor de Velocidad Transversal (v_T). El Movimiento Propio es tan solo la velocidad del cambio del ángulo que medimos al observar la estrella. Para obtener una velocidad tenemos que considerar la distancia (D) a la que se encuentra la estrella, según la fórmula de abajo. Dos estrellas con el mismo valor de movimiento propio pueden tener velocidades transversales muy dispares.

$$v_T = 4.74 \cdot \mu \cdot D$$

Donde la distancia D está en parsecs y el movimiento propio μ está en arcos de segundo por año. Gaia nos facilita este parámetro en milisegundos de arco, al usar esta fórmula deberemos dividir este valor entre 1000. El factor 4.74 es para realizar la conversión de unidades, de arcos de segundo al año por parsec, a kilómetros por segundo.

Con esto ya tenemos la primera componente de la velocidad de la estrella. Todavía nos falta otra componente para conocer la velocidad real. La Velocidad Transversal no nos dice nada sobre si la estrella se acerca o se aleja de nosotros, para saber eso necesitamos conocer la Velocidad Radial (v_R). Midiendo la longitud de onda de las líneas espectrales de absorción o emisión de la luz que recibimos de una estrella podemos deducir si se acerca o se aleja de nosotros gracias al efecto Doppler. Si una longitud de onda sufre un cierto corrimiento al rojo quiere decir que la fuente de luz emisora se aleja de nosotros, análogamente si el corrimiento es hacia el azul, la fuente se está acercando hacia nosotros. Este parámetro está disponible en Gaia directamente en km/s. La siguiente figura ilustra los principales parámetros relacionados con la velocidad de las estrellas.

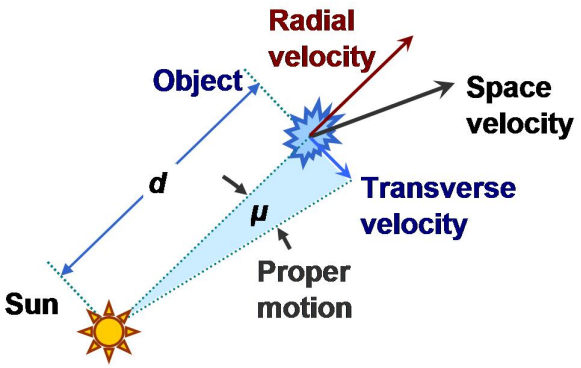

Figura 22: Parámetros relativos al movimiento de las estrellas

Solo tenemos que calcular correctamente las componentes (x, y, z) de los vectores de Velocidad Transversal y Velocidad Radial. Para ello tendremos que tener en cuenta la posición de la propia estrella. Una vez las tengamos, la velocidad real de la estrella es la suma vectorial de estas dos.

$$\vec{v} = \vec{v_T} + \vec{v_R}$$

Hay que considerar que esta velocidad que hemos calculado no es la velocidad absoluta de la estrella sobre la galaxia, sino una velocidad relativa al Sol. Si quisiéramos obtener una velocidad relativa a la galaxia tendríamos que descontar el movimiento de nuestro Sol sobre la galaxia. Para los objetivos de este trabajo nos vale con que la velocidad sea relativa al Sol.

3.2.3. Implementación del movimiento

Para utilizar esta velocidad tendremos que incluirla en nuestro VBO y el vertex shader tendrá que calcular la nueva posición de la estrella en función del tiempo. Esto lo haremos incluyendo en nuestro catálogo la variable *época de referencia*, un valor numérico que representa el instante medio de las observaciones. Se mide en tiempo coordinado baricéntrico y en años Julianos debido al movimiento perpetuo del punto de referencia, el equinoccio de primavera. En GEDR3 todas las estrellas tienen como época J2016.0 TCB. Implementaremos en la interfaz la opción de modificar la velocidad del paso del tiempo hacia el futuro y hacia el pasado para apreciar visualmente el movimiento de las estrellas a lo largo del tiempo. Esto modificará nuestra variable *época de referencia* que pasaremos al Vertex Shader como variable uniform para que el cálculo de la nueva posición de la estrella sea tan simple como

$$p_t = p + t \cdot \vec{v}$$

Con esta fórmula si $t = 0$ obtenemos la posición de la estrella en la época de referencia, en este caso 2016, si $t = 1$ obtenemos su posición en el 2017. Esta velocidad $\vec{v}$ está expresada en Km/s, pero en nuestra representación interna de la posición de las estrellas, la unidad de distancia es el párse c y en la fórmula anterior, el tiempo está expresado en años, por lo tanto, en el VBO almacenaremos la velocidad en párse cs/año.

Esta forma de simular el movimiento asume que las estrellas siguen un movimiento rectilíneo y uniforme, lo cual es completamente falso. Las estrellas están orbitando el centro galáctico y además se atraen entre sí. La velocidad que hemos calculado en base a los valores de Movimiento Propio y Velocidad Radial solo nos dan una instantánea de su trayectoria en un momento concreto, por lo que esta simulación no es creíble para grandes lapsos de tiempo, pero para periodos más cortos es una aproximación razonable.

Introduciremos la opción de activar la visualización de los vectores de velocidad (figura 23), esto es muy útil pues ayuda a detectar fácilmente estrellas binarias o cúmulos abiertos y globulares o corrientes de estrellas, los cuales muestran las interacciones gravitatorias y la dinámica de nuestra Vía Láctea.

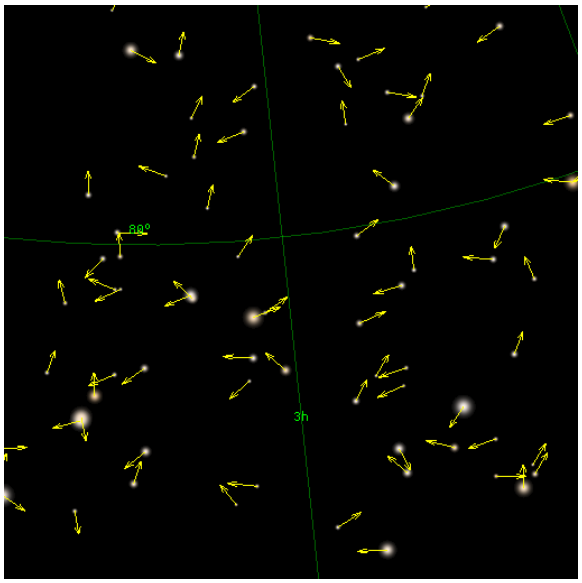

Figura 23: Vectores de velocidad

Para renderizar estos vectores en forma de flecha utilizaremos un Geometry Shader específico para calcular la geometría de las líneas que forman la flecha, y todas las flechas tendrán la misma longitud.

3.2.4. Planos fundamentales

De momento, estamos trabajando únicamente en coordenadas ecuatoriales. Esto quiere decir que el plano fundamental del sistema (el plano horizontal o el plano XZ), coincide con el plano que contiene al ecuador de la Tierra, y el eje de rotación de la Tierra coincide con el eje Y. Pero dependiendo de qué plano fundamental nos interese

tenemos varios sistemas de coordenadas. Para pasar de un sistema a otro hay que multiplicar por una matriz con la transformación de plano adecuada. A continuación analizamos como obtener la matriz correcta para pasar las coordenadas cartesianas de un punto (x, y, z) calculadas a partir de sus coordenadas ecuatoriales a cada uno de los distintos sistemas de coordenadas.

Algunos de estos sistemas de coordenadas pueden considerarse tanto geocéntricos como heliocéntricos, es decir, el origen de coordenadas, el punto $(0, 0, 0)$ puede ser la Tierra o el Sol. En la práctica la diferencia entre un sistema u otro es despreciable pues las distancias interestelares son inconmensurablemente grandes en comparación con la distancia Tierra-Sol.

Los principales parámetros que necesitamos para calcular estas matrices son dos vectores, un vector que indique cuál es la dirección principal del sistema, y otro vector perpendicular a este que apunte hacia el polo norte del sistema de coordenadas en cuestión. La dirección principal del sistema es la dirección hacia las coordenadas esféricas $(0, 0)$, esta dirección va a coincidir con el eje X (la dirección del equinoccio). La dirección hacia las coordenadas del polo norte coincidirá con el eje Y.

La función `lookAt()` de la librería glm que utilizamos para calcular la matriz de visión de la cámara también nos es útil para calcular estas matrices. Esta función utiliza la posición de la cámara, la dirección hacia la que mira (vector lookAt) y su vector up para obtener una matriz que transforme las coordenadas del mundo al espacio de cámara. Especificando el origen de coordenadas $(0, 0, 0)$ como posición de la cámara, y la dirección principal y el polo norte como vectores lookAt y Up, obtenemos justo la matriz de transformación que necesitamos. Solo hay que aplicarle una rotación extra de 90º sobre el eje Y en sentido antihorario para hacer coincidir la dirección principal con el eje X positivo.

- **Coordenadas horizontales**

Este sistema es para el uso divulgativo de nuestro software, algo que no hemos olvidado. Este sistema da una perspectiva del cielo en función de nuestra posición en la Tierra. El sistema es heliocéntrico pero el plano fundamental lo definen nuestras coordenadas de longitud y latitud sobre la superficie de la Tierra. La longitud horizontal se denomina Azimuth (A), y la latitud horizontal se denomina Altitud (a). A diferencia del resto de sistemas que veremos a continuación, éste es el único cuyo valor de longitud

crece en sentido horario, la dirección principal es la que apunta al norte y el valor crece según giramos hacia el Este. La siguiente figura ilustra este sistema de coordenadas.

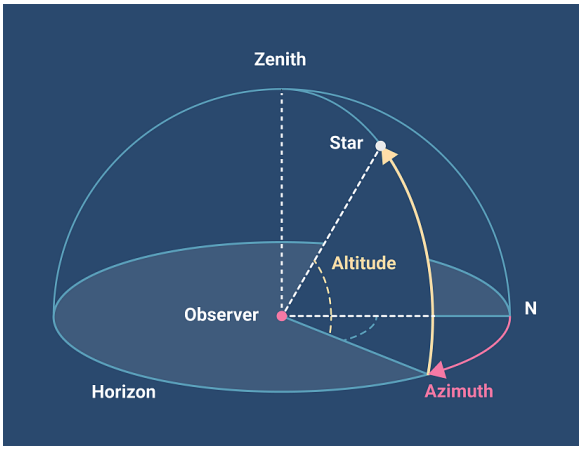

Figura 24: Coordenadas horizontales [4]

Este sistema no sirve para dar coordenadas absolutas de ningún cuerpo celeste. Las coordenadas horizontales de una estrella están cambiando constantemente conforme la Tierra va rotando. Incorporando para este modo la opción de activar una matriz de rotación para simular la rotación de la Tierra conseguimos observar el movimiento aparente de las estrellas sobre el cielo desde distintas latitudes.

- **Coordenadas eclípticas**

En este sistema el plano fundamental es el plano de la órbita de la Tierra o plano eclíptico (ver figura 25). La dirección principal del sistema se define con el equinoccio de primavera al igual que las coordenadas ecuatoriales. Ambos sistemas comparten dirección principal $(0, 0)$ al definirse con el equinoccio, el momento justo en el que la Tierra pasa por la intersección de los planos ecuatorial y eclíptico. En este sistema los parámetros de coordenadas se conocen como longitud eclíptica (λ) y latitud eclíptica (β). Estos parámetros funcionan de forma análoga a la ascensión recta y declinación del sistema ecuatorial, con la diferencia de que la longitud eclíptica toma valores entre 0º y 360º, sigue un formato de grados, minutos y segundos sexagesimales, a diferencia de la ascensión recta ecuatorial que usa grados horarios entre 0h y 24h. En ambos sistemas éste ángulo de longitud crece en sentido antihorario si observamos desde el hemisferio norte.

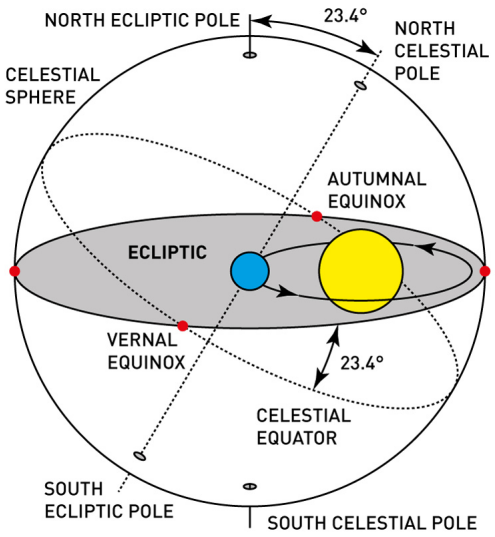

Figura 25: Planos ecuatorial y eclíptico

La coordenadas ecuatoriales del polo norte eclíptico son:

$$\alpha : 18h\ 0m\ 0s$$

$$\delta : 66^\circ\ 33'\ 38.55"$$

Nótese que la declinación de este punto es el resultado de restarle a 90° la inclinación del eje de rotación de la tierra con el plano de la órbita $\sim 23.4^\circ$.

- **Coordenadas galácticas y galactocéntricas**

Este sistema es el más útil para los estudios galácticos y extragalácticos para los que es más útil nuestro software. En este sistema el plano fundamental es el plano de nuestra galaxia, la Vía Láctea. Los parámetros aquí se conocen como longitud galáctica (l) y latitud galáctica (b) (ver figura 26).

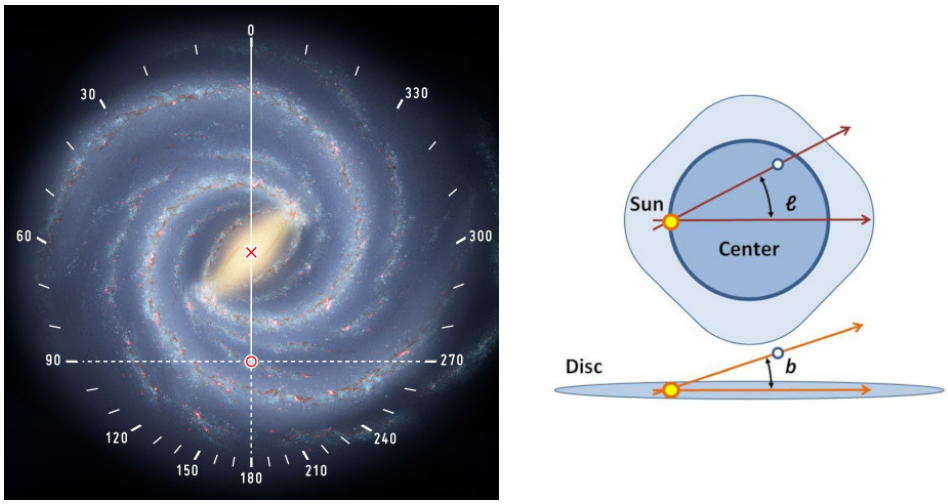

Figura 26: Coordenadas galácticas

La dirección principal del sistema apunta justo hacia el centro de la galaxia, región conocida como Sagitario A*, donde se encuentra un agujero negro supermasivo cuyas coordenadas son:

$$\alpha : 17h\ 45m\ 36s$$

$$\delta : -28^\circ\ 56'\ 24"$$

Y las coordenadas del polo norte galáctico

$$\alpha : 12h\ 51m\ 24s$$

$$\delta : 27^\circ\ 7'\ 48"$$

Estos valores de coordenadas están extraídos de la publicación *The New I.A.U. System of Galactic Coordinates (1958 Revision)* [5].

Hasta ahora todos los sistemas de coordenadas que hemos visto son heliocéntricos, es decir, el origen del sistema de coordenadas siempre ha sido el Sol, lo único que cambia es el plano horizontal.

En el sistema galactocéntrico el origen de coordenadas es el centro de la galaxia, el agujero negro. El plano fundamental es también el plano de la galaxia como en el sistema galáctico, y la dirección principal en este caso es la dirección que va desde el

centro de la galaxia hacia el Sol. Para obtener la matriz, en este caso podemos partir de la matriz del plano galáctico y aplicarle una rotación de 180º para invertir la dirección principal y aplicarle también una traslación sobre la dirección principal con la distancia hasta el centro galáctico ~ 8178 parsecs.

Para todos estos planos implementaremos los correspondientes grids para el modo de cámara centrada en el origen.

3.2.5. Pruebas de verificación y validación

Surge un pequeño problema con la visualización de los vectores de velocidad. Como muestra la figura 27, los vectores saturan en exceso la escena, en especial en las zonas de mayor densidad de estrellas.

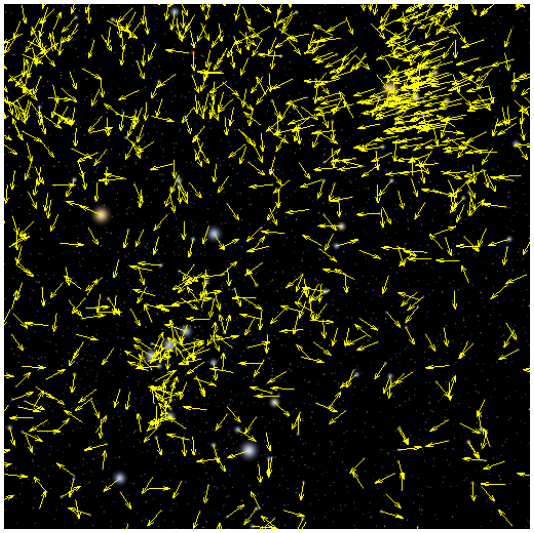

Figura 27: Imagen saturada por los vectores

Para solventarlo vamos a incluir en la interfaz un slider para permitir al usuario controlar la distancia máxima a la que renderizar estos vectores. De este modo el usuario puede adaptar este límite en función de la densidad de estrellas de la zona que esté observando. Además vamos a aplicarle a los vectores una transparencia en función de su distancia hasta este límite. De este modo se da más importancia a los vectores de las estrellas más cercanas y se ignoran las que están demasiado lejos como se aprecia en la siguiente figura.

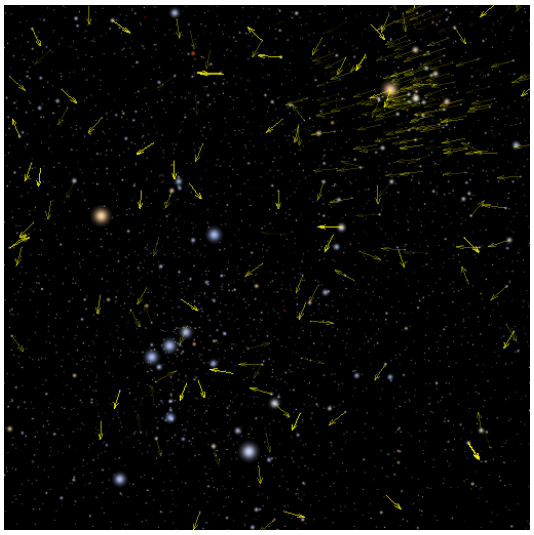

Figura 28: Visualización con límite de distancia y transparencia

3.3. 3ª Iteración

3.3.1. Análisis y diseño

En esta iteración vamos a implementar los filtros. Para representar un filtro basta con dos números que representen las cotas inferior y superior. Implementaremos la opción de guardar catálogos y otras representaciones distintas de la visualización espacial en 3D. En la figura 29 tenemos la actualización del diagrama UML.

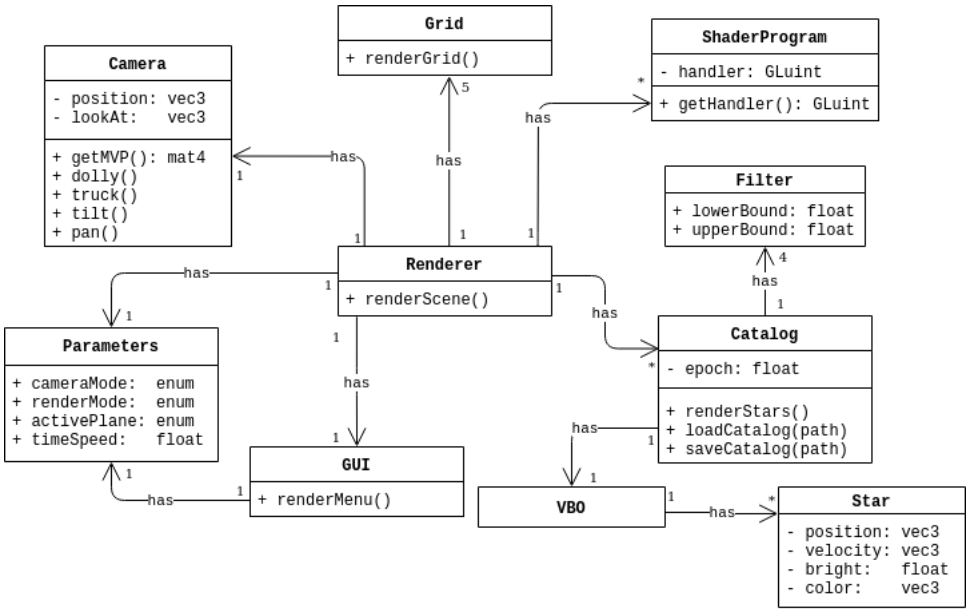

Figura 29: Diseño UML en la 3ª Iteración

Para esta iteración, la siguiente figura muestra los casos de uso que abordaremos.

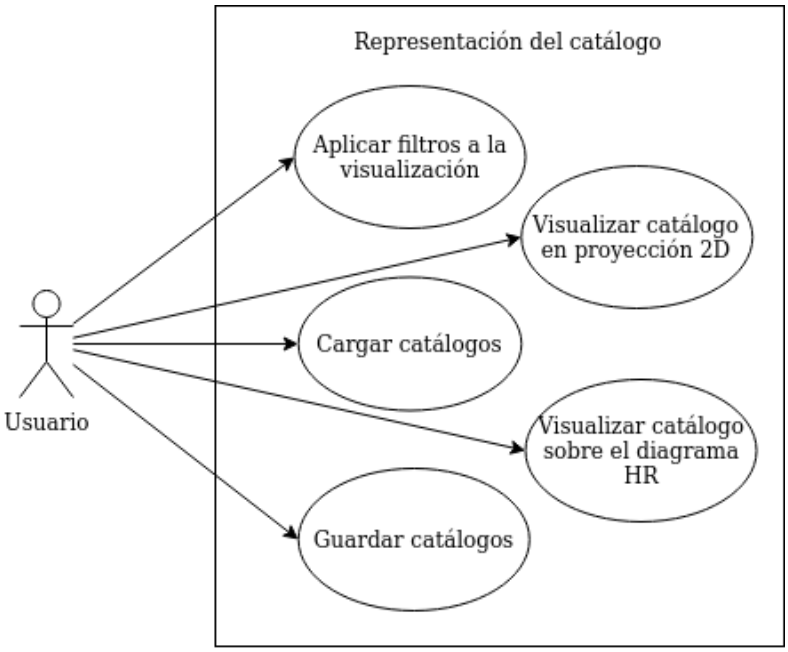

Figura 30: Casos de uso en la 3ª iteración

3.3.2. Filtros

Una utilidad básica y de interés científico es poder aplicar filtros para visualizar una selección concreta del catálogo e ignorar el resto de estrellas. Los principales filtros que vamos a considerar son: filtro de distancia, para visualizar una selección de estrellas en función de su distancia al Sol (ver figura 31); filtro por *Magnitud Absoluta*, para seleccionar solo las estrellas dentro de un rango de brillo; filtro por *Índice de color B-V*, para establecer un rango de temperatura o tipo espectral; y filtro por el módulo de la velocidad, para discernir entre las más rápidas y las más lentas.

Todos estos filtros se definen con una cota inferior y superior para cada uno de los parámetros mencionados. Estas cotas se podrán modificar mediante sliders en la interfaz de usuario y pasarán al Vertex Shader que realizará el test para saber si una estrella pasa todos los filtros y por tanto debe ser renderizada, o en caso contrario descartar la estrella y no renderizarla. Para descartar la estrella no disponemos de una función discard como sí ocurre en el Fragment Shader. Cuando una estrella no pase los filtros, le asignaremos unas coordenadas fuera del espacio de cámara para que sea descartada en la fase de clipping.

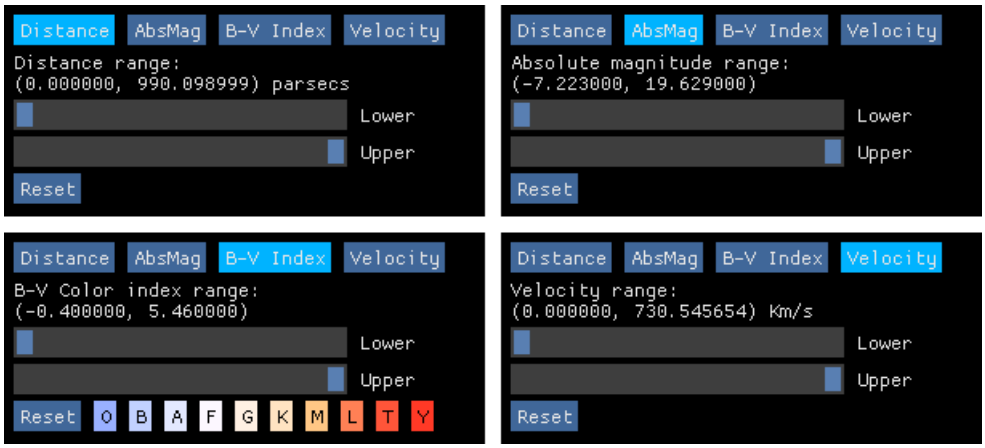

Figura 31: Interfaz para modificar los filtros

En el filtro del *Índice de color B-V* hemos incluido una representación de la selección de las clases espectrales. En todos los filtros el botón de reset establece la cota inferior y superior a los valores mínimo y máximo del parámetro en el catálogo activo.

Para que el Vertex Shader pueda hacer estas comparaciones necesitamos incluir algunos parámetros en el VBO. La Magnitud Absoluta ya la tenemos. La distancia y la velocidad podría calcularlas el Vertex Shader haciendo la raiz de la suma de las componentes al cuadrado de la posición y la velocidad que ya tenemos almacenadas en el VBO. Sin embargo, vamos a intentar reducir tanto como sea posible los cálculos dentro del Vertex Shader, aunque sea redundante, incorporaremos explícitamente los parámetros de distancia al Sol y módulo de la velocidad. Con el Índice de color B-V ocurre algo parecido. Aunque ya tenemos el color RGB en el VBO, no nos sirve para comparar con los límites del filtro, por lo que a excepción de la Magnitud Absoluta, el resto de parámetros tendremos que incorporarlos expresamente. La siguiente figura muestra una visualización aplicando filtros.

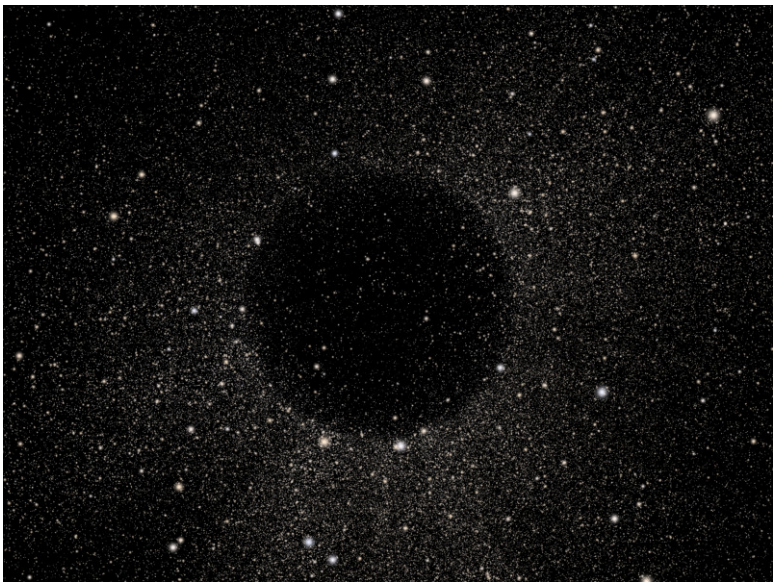

Figura 32: Visualización aplicando filtro de color y distancia

3.3.3. Exportar selección

Ya que hemos implementado estos filtros, tiene sentido incorporar la función de guardar en un fichero las estrellas que pasen estos filtros y aislar en un fichero las estrellas seleccionadas.

Hasta ahora el programa al iniciar la ejecución carga automáticamente los catálogos con los que estamos trabajando, pero el usuario no tiene opción de poder cargar otro catálogo. Incorporaremos en el menú las opciones para cargar o guardar un catálogo, se podrán cargar varios catálogos y cerrar algún catálogo cargado previamente. Para hacerlo, hay que implementar una llamada al explorador de archivos para que el usuario pueda seleccionar la ruta del fichero a cargar o guardar. Esta llamada se implementa de diferentes formas según el sistema operativo, por lo que tendremos que implementar las funciones para Windows y para Linux especificando los condicionales de compilación adecuados para que el compilador sepa cuál debe usar. La siguiente figura muestra estas funciones básicas en la interfaz.

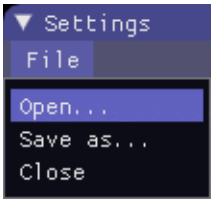

Figura 33: Diálogo estándar para cargar/guardar/cerrar ficheros

3.3.4. Visualización 2D

Otro modo de visualización necesario es una proyección 2D del catálogo, para visualizarlo en una sola imagen como si fuera un mapa. Para ello necesitamos un nuevo shader que interprete los valores de longitud y latitud como valores x e y respectivamente, y estos valores dependerán del plano fundamental activo. Podemos hacer que el shader haga la conversión de la posición 3D que tenemos en el VBO a las coordenadas de longitud y latitud, pero estaríamos realizando en cada frame un cálculo un poco costoso. En lugar de ello y para ahorrarle esta tarea al shader, vamos a utilizar más memoria en nuestro VBO para almacenar explícitamente estas variables x e y. Cada vez que se active este modo de visualización o cada vez que, estando en este modo se cambie de plano fundamental, accederemos al VBO para realizar el cálculo de estos valores de longitud y latitud en función del plano fundamental y actualizar estas posiciones x e y. Esta operación se realiza en O(n), pero solo se hará cuando el usuario realice alguna de estas acciones y no estaremos constantemente calculando la misma operación en cada frame. En este modo la rueda del ratón servirá para hacer zoom y al hacer click y mover el ratón podemos cambiar la posición de la proyección. En la figura 34 se aprecia la visualización de esta proyección.

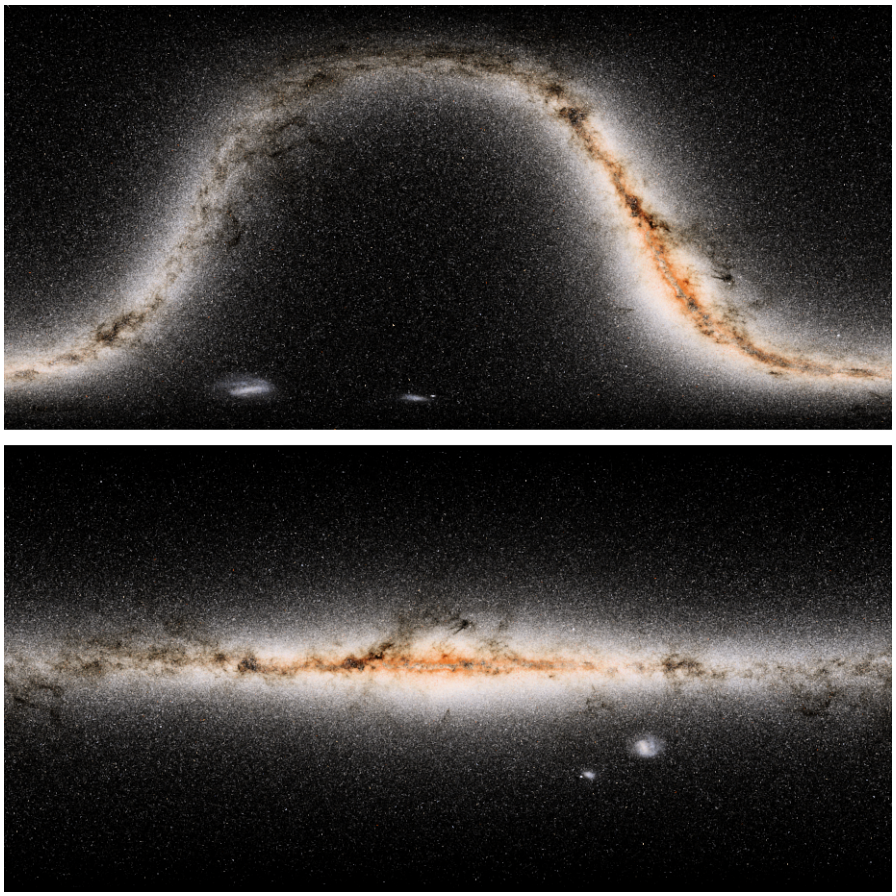

Figura 34: Visualización 2D de un catálogo en modo ecuatorial y galáctico

3.3.5. Diagrama Hertzsprung–Russell

La visualización del diagrama Hertzsprung–Russell (o diagrama HR) [11] es un requisito básico que no puede faltar en nuestro software (figura 35). Este gráfico, que es básico en astronomía estelar, ubica a las estrellas en un plano 2D ordenadas por su temperatura en el eje X y por su brillo en el Y. Dependiendo de en qué región del diagrama se encuentre una estrella, se la puede categorizar como una estrella de la secuencia principal, una enana blanca, o una supergigante por ejemplo. Este diagrama fue fundamental en el entendimiento de la evolución estelar. Podemos observar este diagrama en la siguiente figura.

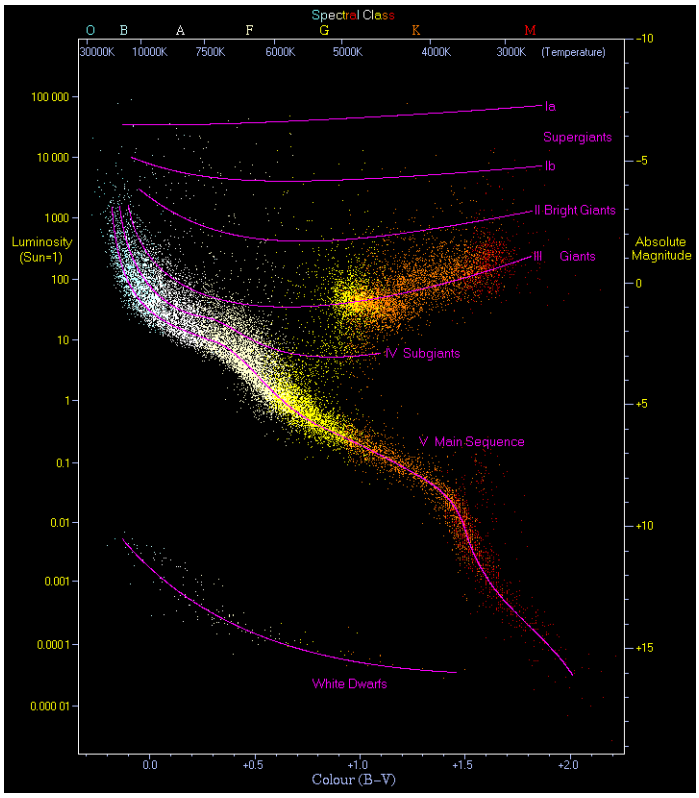

Figura 35: Diagrama HR con estrellas de Hipparcos y Gliese [10]

Implementar esta visualización es tan sencillo como hacer otro shader que interprete el Índice de color B-V y la Magnitud Absoluta que ya están presentes en el VBO como valores x e y. El control con el ratón será el mismo que en el modo 2D. La figura 36 muestra como jugando con el slider de atenuación de transparencia en función de la distancia, podemos notar las diferentes regiones de mayor densidad de estrellas sobre este gráfico.

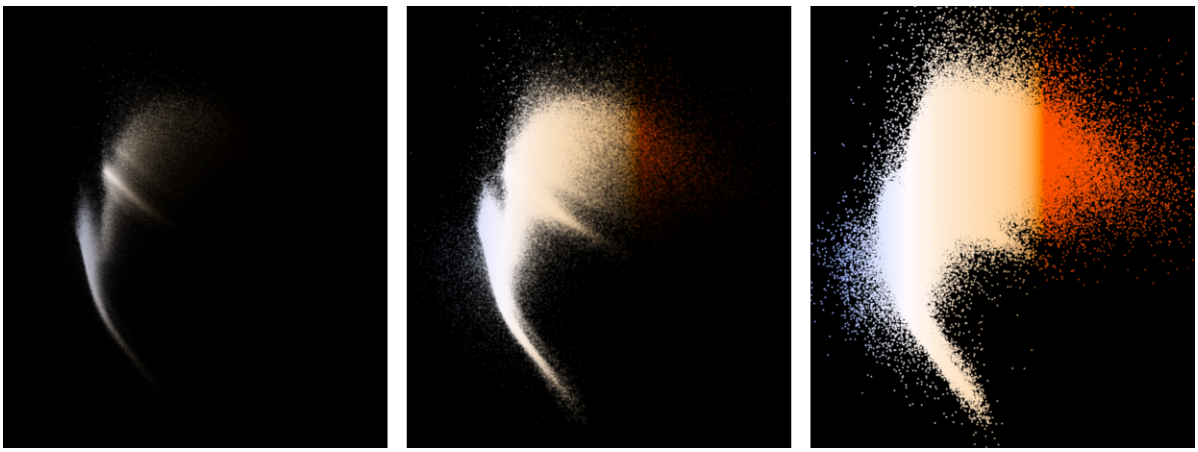

Figura 36: Diagrama HR con distintas configuraciones de atenuación de transparencia

3.3.6. Pruebas de verificación y validación

Probando los distintos filtros encontramos algunos problemas de usabilidad. Los sliders no nos permiten modificar las cotas inferior y superior de forma precisa. El problema está en que al slider le especificamos los valores mínimo y máximo de dicho parámetro y utiliza la anchura de la barra del slider para interpolar entre ambos. El problema es que dependiendo de la distribución de valores del dataset, pueden concentrarse muchas más estrellas en una parte del slider que en otra. La consecuencia es que a veces, moviendo aunque sea muy poco un slider estamos descartando muchas más estrellas de las que queríamos descartar.

Para solventarlo vamos a cambiar el comportamiento del slider. En lugar de especificarle los valores mínimo y máximo del parámetro para el dataset cargado, vamos a especificarle como valores mínimo y máximo, el valor de la cota menos un porcentaje como valor mínimo, y el valor de la cota más el mismo porcentaje como valor máximo. De esta forma la anchura de la barra se utiliza para interpolar entre valores cercanos al valor actual de la cota. Esto cambia el comportamiento del slider. Ahora en lugar de desplazar el marcador del slider, simplemente haciendo click sobre la barra en la parte derecha o izquierda del marcador la cota empieza a aumentar o disminuir de forma suave dando un control mucho mas preciso.

3.4. 4ª Iteración

3.4.1. Análisis y diseño

Finalmente en esta última iteración vamos a centrarnos en el estudio de los distintos parámetros. La siguiente figura muestra los casos de uso que vamos a implementar.

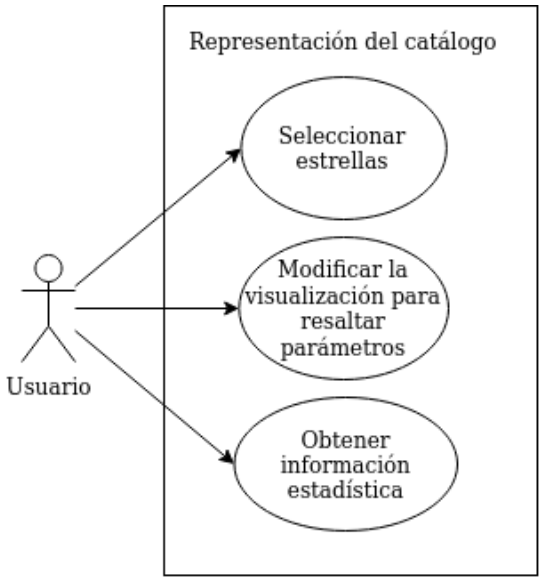

Figura 37: Casos de uso en la 4ª iteración

3.4.2. Colorización por parámetros

Para resaltar el valor de los distintos parámetros y obtener una representación visual, vamos a cambiar el color de las estrellas. Utilizaremos las cotas inferior y superior de los diferentes filtros como puntos entre los que interpolar dos colores distintos para poder visualizar, por ejemplo, las estrellas más lentas en rojo, y las más rápidas en azul (figura 38). La interfaz permitirá al usuario elegir ambos colores y aplicar la colorización en función del filtro activo. Accederemos al VBO para sobrescribir los valores RGB por el nuevo color resultado de interpolar el parámetro afectado entre las cotas del filtro activo.

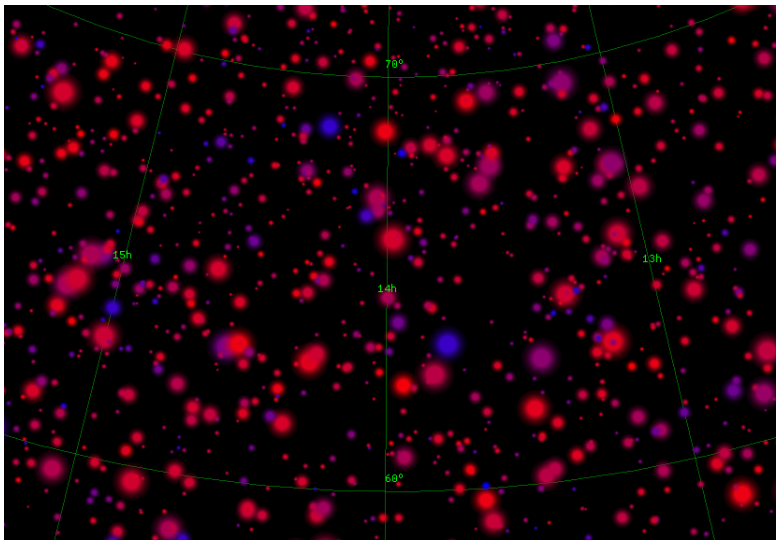

Figura 38: Estrellas coloreadas por velocidad

La interfaz tendrá un botón para restablecer el color al valor que le corresponda por su temperatura con el Índice de color B-V. Para añadir más contraste a esta visualización, vamos a incluir un tercer color para asignar justo al valor medio entre las dos cotas del filtro. Incluiremos también un slider para controlar la importancia que se le da a este tercer color en la interpolación. La siguiente figura muestra todos estos elementos en la interfaz.

Figura 39: Interfaz de colorización por filtros

Con esta nueva función combinada con las distintas representaciones del catálogo y configuraciones de visualización, obtenemos un gran abanico de posibilidades para crear distintos gráficos del dataset. Las siguientes figuras son solo unos ejemplos.

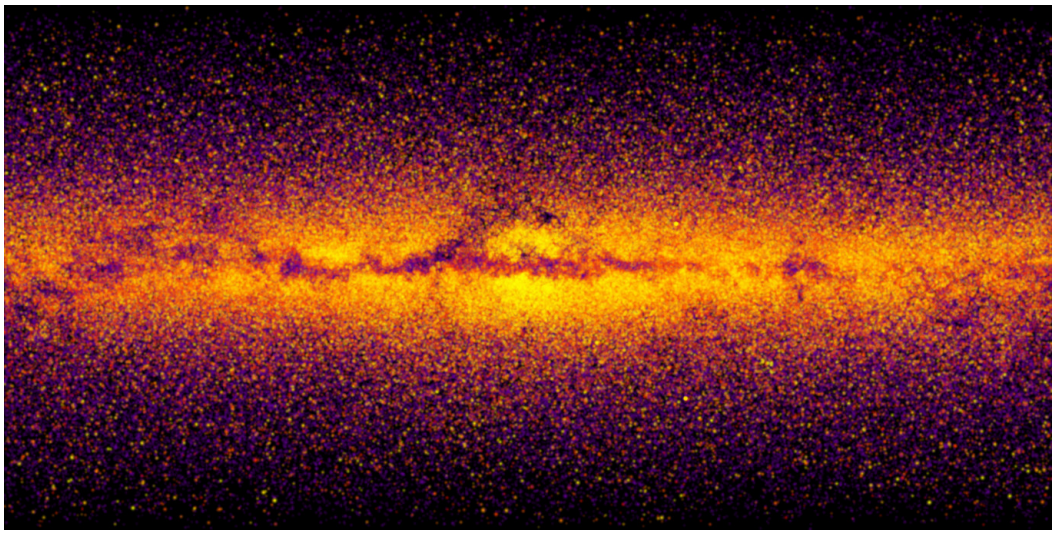

Figura 40: Representación del brillo sobre una proyección 2D del dataset

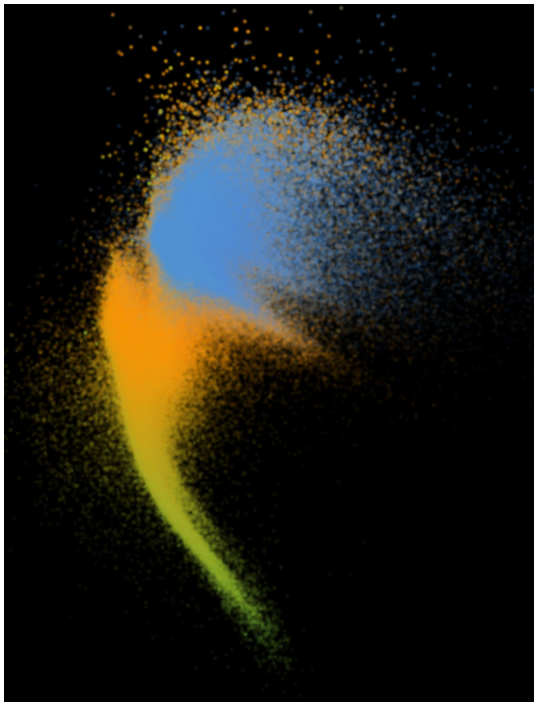

Figura 41: Representación de la distancia sobre el diagrama HR

3.4.3. Información estadística

Una utilidad de gran interés científico es mostrar alguna información estadística sobre el conjunto que se esté visualizando. Mostrar el número total de estrellas que están interviniendo en la visualización, y los valores medios para los distintos parámetros con los que estamos trabajando, así como medidas de dispersión básicas.

Un simple botón en la interfaz abrirá una ventana mostrando esta información para

el conjunto de estrellas que pasen los filtros. En la siguiente figura vemos cómo aparece esta información.

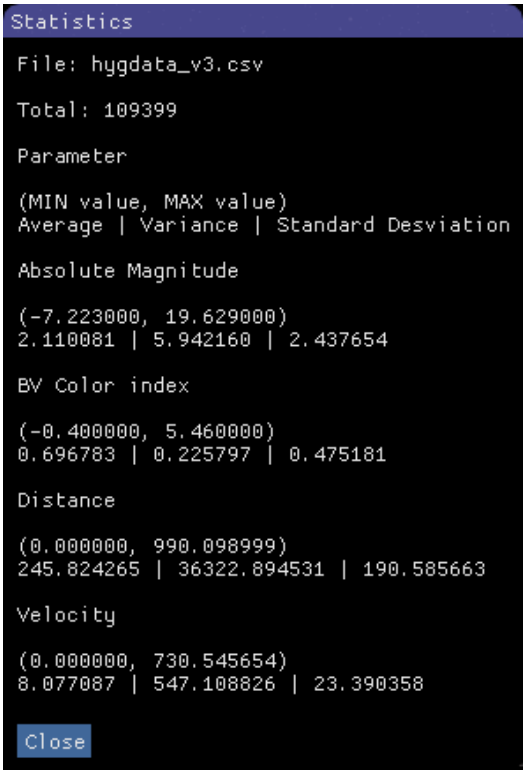

Figura 42: Diálogo de información estadística

3.4.4. Selección de estrellas

Una función básica a implementar es la selección de estrellas mediante el ratón. Al hacer click sobre una estrella debe aparecer una ventana mostrando información básica sobre esa estrella. Al hacer click con el ratón, la gestión de eventos de GLFW nos facilita las coordenadas (x, y) del pixel sobre el que se ha hecho click, pero deducir de ahí cual es la estrella seleccionada no es trivial. En general para obtener el objeto sobre el que se ha hecho click en un entorno 3D, hay que realizar el trazado de un rayo desde la cámara y usar alguna función de intersección con primitivas como el triángulo o la esfera para discernir el objeto seleccionado. Sin embargo en nuestra aplicación las estrellas no tienen un modelo 3D asociado con el que comparar, solo son puntos sin volumen. Podríamos asignarle a cada estrella una esfera de radio proporcional a su brillo, pero existe una alternativa más simple y eficiente utilizando el propio pipeline de rendering.

Esta técnica consiste en que al hacer click, vamos a renderizar un frame de la

escena pintando cada estrella con un color distinto. Asignaremos a cada estrella un ID único de 32 bits que un shader específico interpretará como valor RGB usado por el Fragment Shader. Tras renderizarse el frame, una función de OpenGL nos permite obtener el color RGB del pixel (x, y) donde el usuario hizo click. Reinterpretando este color como un valor entero de 32 bits, obtenemos el ID de la estrella seleccionada. Con este ID podemos acceder directamente a la posición de la estrella seleccionada consultando en el VBO. La siguiente figura muestra los colores generados por los ID.

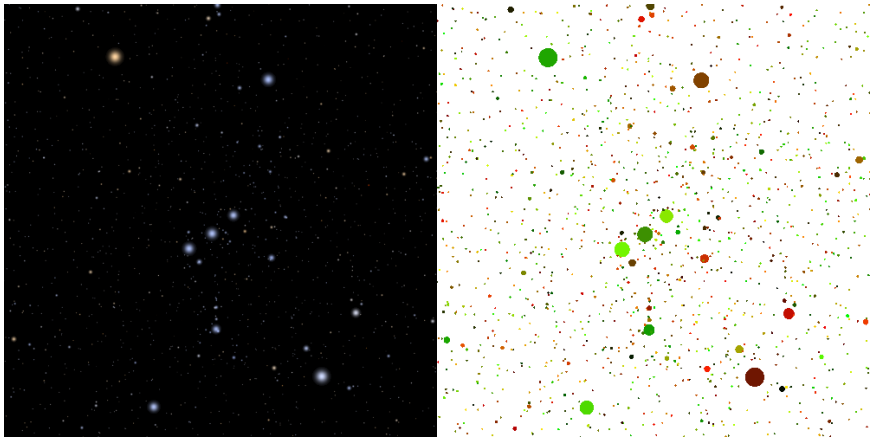

Figura 43: Rendering visual y rendering interno al hacer click

Una vez sabemos sobre qué estrella se ha hecho click, renderizaremos un rectángulo sobre la estrella seleccionada para transmitir feedback al usuario. La interfaz mostrará en una ventana los parámetros de dicha estrella como refleja la siguiente figura.

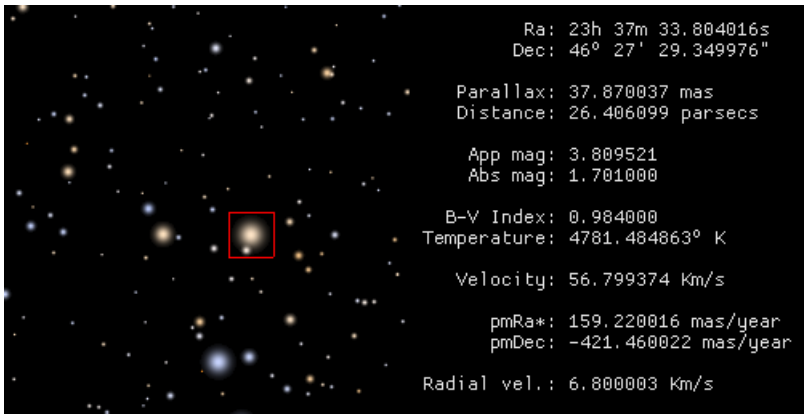

Figura 44: Estrella seleccionada y sus parámetros

Para que la selección de estrellas con el ratón siga funcionando en los modos de visualización 2D y diagrama HR, tenemos que crear un duplicado de estos shaders

que también rendericen las estrellas con el color del ID.

3.4.5. Modo de cámara orbit

Esta selección de estrellas que hemos implementado ha inspirado la idea de poder enfocar la atención de la cámara a la estrella seleccionada, aunque no estaba considerado en el diseño inicial, vamos a incluir un nuevo modo de cámara.

Para la visualización en 3D incluiremos un modo de cámara adicional en el que la cámara siempre mira hacia un punto concreto y solo puede moverse orbitando sobre dicho punto. Con la rueda del ratón en este caso modificamos la distancia a la que nos encontramos del punto orbitado, especificando un aumento o reducción del 10 % en la distancia cada vez que se mueve la rueda del ratón, se produce una experiencia de usuario satisfactoria. Al hacer doble click sobre una estrella, las coordenadas de dicha estrella pasan a ser el nuevo punto a orbitar. Esto permite enfocar la atención de la cámara hacia otra zona del catálogo, y combinado con el movimiento de rueda del ratón, se consigue una nueva forma de explorar el catálogo incluso más efectiva que utilizar la cámara libre.

3.4.6. Pruebas de verificación y validación

Al testear estas implementaciones hemos detectado algunos problemas. En ocasiones al hacer click sobre las estrellas, especialmente cuando visualizamos muchas desde lejos, aparece un bug donde la estrella sobre la que se hace click no es la que se selecciona, es otra distinta, y a veces al hacer click la aplicación se cierra por violación de segmento. Analizando y depurando el código encontramos que a veces, al hacer click, el ID que obtenemos del color RGB excede sobremanera el tamaño total de estrellas del catálogo cargado, esto explica la violación de segmento. Estamos intentando acceder a memoria fuera de nuestro VBO. Pero la razón por la que estamos obteniendo estos valores es porque el antialiasing, al suavizar el borde de los puntos, está generando valores RGB que no teníamos previstos. Al iniciar GLFW activamos el multisampling con 4 muestras, lo desactivaremos especificando 0 muestras. Con esto no solo desaparece el bug, sino que el rendimiento de la aplicación mejora notablemente y el resultado visual es idéntico.

3.5. 5ª Iteración

Ya hemos implementado todos los requisitos básicos que el software debe tener. En esta iteración vamos a implementar algunos aspectos que se nos han ido ocurriendo durante el desarrollo y que suponen una mejora para nuestro software.

3.5.1. Filtro de distancia automático

El rendimiento de la aplicación está íntimamente ligado al número de estrellas que tenga el catálogo que hayamos cargado, en particular, el número de estrellas que se estén visualizando. Por cada estrella estamos realizando una llamada al Vertex Shader, pero especialmente, por cada estrella dentro del campo de visión de la cámara estamos realizando múltiples llamadas al Fragment Shader. En catálogos grandes la experiencia de usuario se ve seriamente comprometida con bajas tasas de FPS.

Cuando cargamos catálogos grandes, en el modo de cámara centrada en el origen, podemos obtener una tasa de FPS aceptable, pero al cambiar al modo Orbit o cámara libre notamos una caída en el rendimiento. En el modo de cámara centrada en el origen, la cámara nunca puede obtener una perspectiva global del catálogo. El número máximo de estrellas que puede visualizar depende del FOV. Al estar ubicada en el origen de coordenadas, aunque el FOV estuviera al máximo, la cantidad máxima de estrellas que podría abarcar está acotada al semiespacio hacia el que está mirando la cámara, descartando todas las que están detrás de ella. Es decir, en este modo el rendimiento es algo mejor porque por lo general, en todo momento estamos observando un porcentaje relativamente pequeño del catálogo, pero en el modo Orbit o cámara libre, podemos alejarnos y observar una perspectiva global desde la distancia, con lo que abarcamos muchas más estrellas y las llamadas al Fragment Shader se multiplican. Además la configuración del tamaño del punto también incide en el número de llamadas al Fragment Shader.

Para informar al usuario sobre el rendimiento de la aplicación vamos a mostrar los FPS (figura 45) en la esquina superior izquierda, para reforzar el feedback de este indicador, su color cambiará como un semáforo, rojo, ámbar y verde para tasas bajas, medias y altas de FPS.

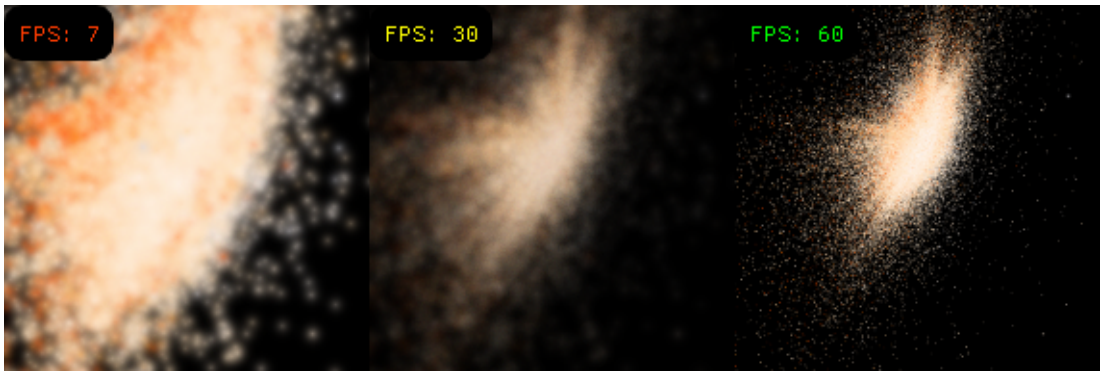

Figura 45: Tasas de FPS para distintas configuraciones

En especial, las caídas más pronunciadas de FPS ocurren cuando miramos desde lejos al origen de coordenadas, la zona de mayor densidad de estrellas. Es más fácil detectar las estrellas cercanas, hay muchas estrellas poco brillantes que por estar muy lejos no llegamos a detectar. En consecuencia, el catálogo Gaia tiene una densidad de estrellas mucho mayor en la región del espacio más cercana a nuestro sistema solar, y esta densidad va bajando conforme nos alejamos de él.

Realmente no podemos garantizar siempre un rendimiento óptimo, dependiendo del hardware sobre el que se ejecute la aplicación tendremos un límite en la cantidad máxima de estrellas. A partir de este límite el rendimiento empieza a verse comprometido. Pero si que podemos recurrir a un truco para paliar la situación. Considerando que el problema se agrava al observar el catálogo desde lejos, y que precisamente cuando estamos lejos, las estrellas más cercanas al origen son menos relevantes en la visualización al quedar ocultas tras las más distantes, podemos utilizar el filtro por distancia para descartar estas estrellas y mejorar el rendimiento. Si el usuario activa la opción de usar este filtro de distancia automático, cuando los FPS bajen de 30, la cota inferior del filtro empezará a subir hasta que los FPS se mantengan como mínimo en 30. Si el usuario se mueve a zonas donde visualice menos estrellas y los FPS suben por encima de 60, el filtro automáticamente bajará para volver a visualizar las estrellas descartadas hasta que los FPS se estabilicen en 60. De este modo la cota inferior del filtro de distancia se ajustará automáticamente para mantener en todo momento la cantidad adecuada de estrellas para que los FPS se mantengan dentro de un rango que garantice una experiencia de usuario agradable.

Para mejorar aún más la eficiencia de este filtro, vamos a ordenar las estrellas en el VBO por distancia al Sol. Estando ordenadas, podemos aplicar el filtro simplemente

especificándole a OpenGL que renderice solo las estrellas del VBO entre dos índices inferior y superior que se corresponderán con las cotas inferior y superior del filtro de distancia. De esta forma no hace falta que el Vertex Shader compruebe si la estrella pasa el filtro de distancia, ahora el filtro se aplica de forma implícita en la llamada a la función de renderizar el VBO. Además, de esta forma, para las estrellas que no pasen el filtro de distancia, no solo nos estamos ahorrando las llamadas al Fragment Shader sino que también nos ahorramos las llamadas al Vertex Shader.

3.5.2. Nombres de estrellas

Para el catálogo por defecto, HYG, en el archivo CSV disponemos de una columna con los nombres propios de las estrellas más cercanas y/o más brillantes, incluiremos la opción de activar la visualización de estos nombres en éste catálogo (ver figura 46). Renderizar texto en OpenGL no es trivial. Para renderizar estos nombres optaremos por la alternativa de apoyarnos en ImGui. Esto implica que para obtener las coordenadas (x, y) del nombre de la estrella sobre la pantalla, tendremos que hacer en CPU el proceso de pipeline de rendering explícitamente para estas estrellas. Evidentemente esta no es la alternativa más eficiente, pero es mucho más fácil de implementar. Al haber muy pocas estrellas con nombre propio, el rendimiento no se ve seriamente perjudicado.

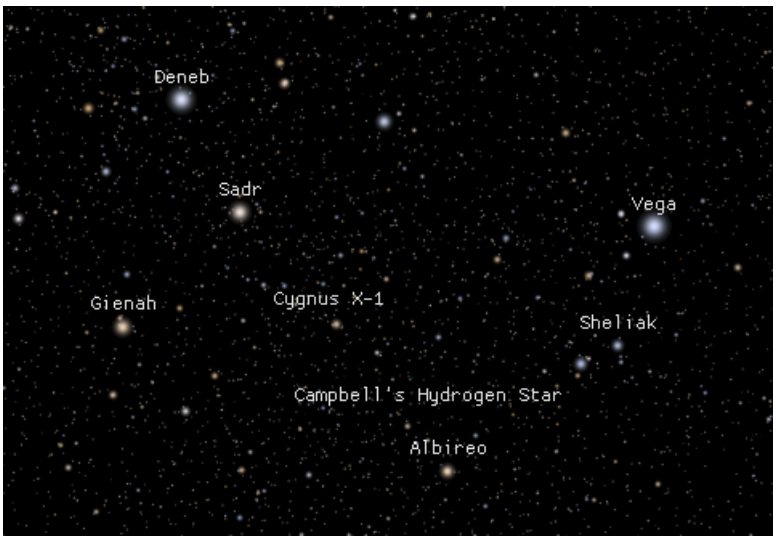

Figura 46: Visualización del nombre de algunas estrellas

3.5.3. Formato FITS

Gaia Archive ofrece varios formatos entre los que elegir para descargar los datasets. Hasta ahora hemos estado trabajando con datasets en formato CSV porque trabajar con este formato es sencillo e intuitivo. El problema de este formato, es que al ser texto plano, cada valor numérico se representa en ASCII donde cada dígito ocupa un byte. Esto hace que los datasets en este formato ocupen mucho tamaño, especialmente cuando los datasets empiezan a tener una cantidad considerable de estrellas.

Por esta razón tenemos que contemplar trabajar en otros formatos. No hace falta que nuestro programa pueda leer todos los formatos que ofrece Gaia, pero sí que es necesario que al menos pueda leer algún formato binario donde los valores numéricos se representan en el estándar IEEE 754 y los archivos son mucho más ligeros.

De todos los formatos que ofrece Gaia, solo dos son binarios: VOTable y FITS. Para manipular estos ficheros necesitaremos de una librería que nos facilite las operaciones de lectura y escritura. Para el formato VOTable, en C++ hay muy pocas librerías disponibles y con pobre documentación. Para el formato FITS, en cambio, existen varias alternativas. La *Administración Nacional de Aeronáutica y el Espacio* del gobierno de los Estados Unidos (NASA) mantiene una librería en C++ para trabajar en este formato; *CCfits* [17]. Esta librería cuenta con un manual donde se explica como instalarla y configurarla en nuestro proyecto, tanto en Windows como en Linux. *CCfits* depende de otra librería llamada *CFITSIO*, que deberemos instalar previamente. Las instrucciones para hacerlo vienen perfectamente detalladas en el manual.

El formato FITS es ampliamente utilizado en astronomía, principalmente suele usarse como un formato de imagen, pero también es utilizado como tabla binaria. *CCfits* facilita una interfaz orientada a objetos para leer, escribir, añadir o quitar las columnas que forman la tabla, por lo que podemos implementar los métodos para leer y escribir ficheros en este formato de forma trivial.

A partir de ahora el programa podrá cargar catálogos en FITS y en CSV, y cuando el usuario vaya a guardar el catálogo activo, la interfaz mostrará la opción de elegir en qué formato guardarlo tal y como muestra la siguiente figura.

Figura 47: Diálogo de selección de formato

4. Conclusiones y trabajos futuros

Aparte de las evidentes aplicaciones didácticas de TV, su diseño e implementación han sido pensados para hacer descubrimientos astronómicos y astrofísicos mediante su inspección visual, de ahí que agradezcamos su rigor y riqueza en características. También inmediata es su generalización a escalas cosmológicas y para encontrar aceleraciones.

4.1. Didáctica

El uso didáctico de TV es natural. Por ejemplo, podemos mostrar nuestra vecindad más cercana. Para ello basta que seleccionemos todos los objetos a menos de 10 parsecs. Estos incluyen muchos tipos de estrellas con historias relevantes. Por ejemplo, alpha Centauri A (visible a simple vista) es igual que nuestro Sol, próxima Centauri C tiene el planeta extrasolar más cercano, la estrella de Barnard es una estrella enana con el mayor movimiento propio de todo el cielo, Luhman 16 AB es un sistema binario de enanas marrones, 61 Cygni es el objeto en el que se midió el primer paralaje. TV es capaz de hacer mapas de estos objetos y por supuesto animaciones para ayudar en esta divulgación.

4.2. Investigación

En cuanto a las aplicaciones en cinemática y dinámica tanto Galáctica como Extragaláctica, sin ser imaginativo, encontramos los siguientes usos del programa:

1. **Física Galáctica**

    - Seleccionando estrellas de alta velocidad, del orden de la velocidad de escape de nuestra Galaxia, podemos observar interacciones de alta energía

con agujeros negros. Esto es extremadamente interesante pues sólo se conocen unas decenas de estas estrellas.

- Seleccionando estrellas de tipo y velocidad similares a altas latitudes galácticas podríamos descubrir grupos móviles de estrellas que son corrientes originadas por interacciones gravitatorias.
- Inspeccionando las posiciones tridimensionales podríamos identificar sobredensidades (densidades mayores que su vecindad) que sean comóviles en el espacio de movimientos propios. Éstas posiblemente podrían confirmarse como nuevos cúmulos abiertos de estrellas; o bien como cúmulos globulares si miramos en el halo estelar de la Vía Láctea.
- Seleccionando diferentes poblaciones de estrellas, por una parte estrellas jóvenes de tipo A y F y por la otra estrellas viejas G y K, se pueden hacer análisis cinemáticos del disco Galáctico y los brazos espirales.

2. **Física Extragaláctica**

   - Podemos mirar galaxias satélite de la Vía Láctea. Seleccionando paralajes y movimientos propios consistentes con la mediana del satélite podríamos calcular sus órbitas. Una excelente aplicación sería nuestra vecina Andrómeda.
   - Podemos identificar una nueva y deseada muestra de cuásares puntuales y núcleos galácticos activos simplemente encontrando la selección de las mayores distancias.

4.3. Trabajos futuros

TV resulta ser una herramienta útil y con potencial. Es una alternativa más humilde pero más útil para un uso científico que por ejemplo, GaiaSky. Nuestros planes son los dos siguientes.

- Además de toda esta utilidad potencial de TV y especialmente dado su diseño planeamos también cruzar los datos de Gaia con los de Hipparcos. Esto nos permitirá encontrar aceleraciones en el plano del cielo, lo cual es absolutamente

novedoso y extremadamente útil. La representación en TV de los vectores velocidad en 3D está especialmente diseñada para esto.

- Finalmente, encontramos interesante y fácil generalizar este software para manejar otras bases de datos de naturaleza cosmológica como el Sloan Digital Sky Survey (SDSS) de galaxias a gran escala en el espacio de corrimientos al rojo.

5. Manual de usuario

Este manual incide en el uso del servicio web Gaia Archive. Se dan unas simples directrices para que el usuario inexperto sepa cómo descargar correctamente los catálogos para poder visualizarlos con TV.

5.1. Instalación y puesta en marcha

TV no requiere de instalación, basta con descomprimirlo en cualquier directorio de nuestra elección. Una vez descomprimido podremos acceder a los ejecutables para Windows y Linux.
En Windows basta con hacer doble click sobre el ejecutable. Si estamos en Linux debemos abrir un terminal en la ubicación del ejecutable y ejecutar `./TV`. Si no tiene los permisos de ejecución podemos dárselos con el comando `chmod +x TV`.

5.2. Descargar conjuntos de datos

Para descargar conjuntos de datos debemos utilizar el servicio web de la Agencia Espacial Europea, Gaia Archive.

`https://gea.esac.esa.int/archive/`

En la sección **Search** (ver figura 48), tenemos una interfaz para seleccionar un subconjunto específico del catálogo.

Debemos seleccionar el catálogo `gaiaedr3.gaia_source`.

En la sección de **Extra conditions** podemos aplicar tantos filtros por parámetros como queramos.

En la sección **Display columns** debemos asegurarnos de marcar los siguientes atributos:

- `ra`
- `dec`
- `parallax`
- `pmra`
- `pmdec`
- `phot_g_mean_mag`
- `bp_g`
- `dr2_radial_velocity`

Podemos marcar mas atributos si queremos, pero como mínimo estos deben estar presentes.

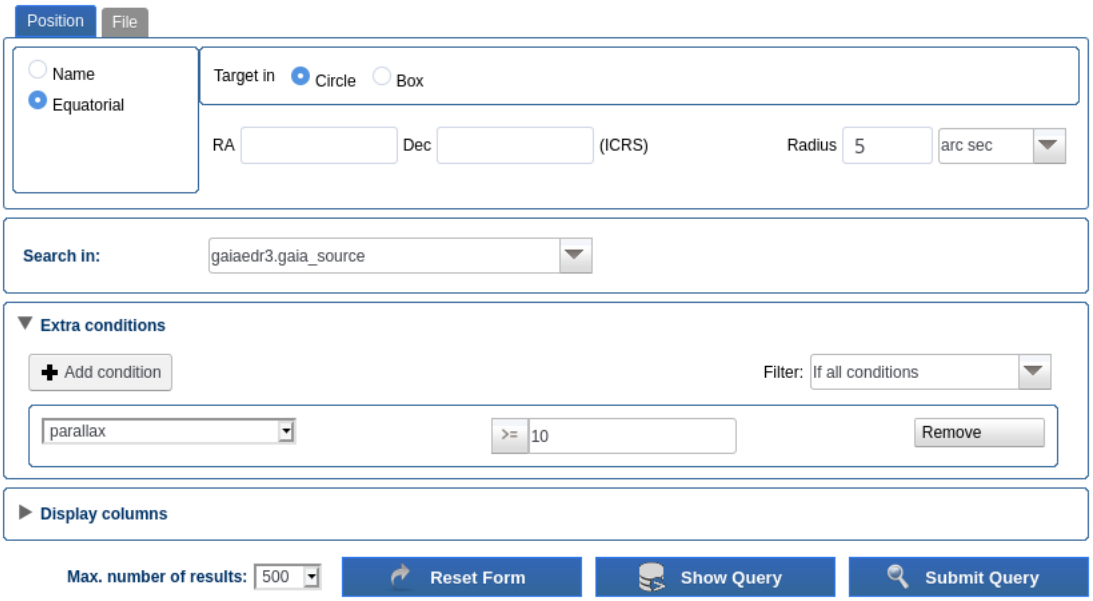

Figura 48: Interfaz gráfica de Gaia Archive

En la sección **Advanced (ADQL)**, nos aparece la consulta que hemos especificado en la interfaz anterior en lenguaje ADQL, que es un lenguaje de consulta muy similar a SQL específicamente diseñado para bases de datos de astronomía. En esta sección podemos especificar condiciones más complejas, como por ejemplo, seleccionar solo aquellas estrellas para las que el error en la medida del paralaje está por debajo de cierto umbral.

Una opción interesante en este modo es ordenar las estrellas por el atributo **Random Index**. Este atributo está pensado para que al ordenar la consulta por este parámetro, el conjunto de salida independientemente de su tamaño, sea estadísticamente representativo del conjunto total.

Una vez especificada la consulta solo queda hacer click en **Submit Query**, tras procesar la consulta nos mostrará el resultado, y si no ha habido ningún error podremos descargar nuestro subconjunto de datos, debemos seleccionar CSV o FITS como formato de descarga. Tal y como se muestra en la siguiente figura.

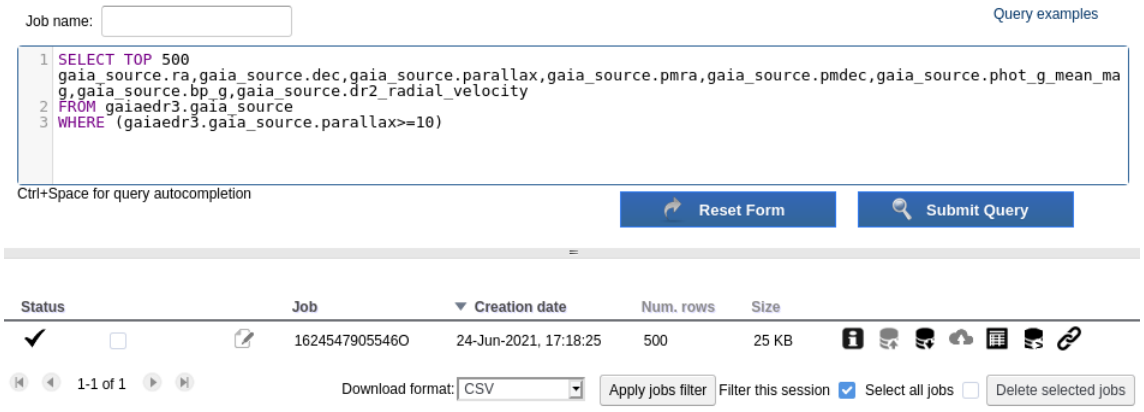

Figura 49: Interfaz ADQL de Gaia Archive

Si utilizamos este servicio sin iniciar sesión, Gaia Archive solo nos permite descargar ficheros de hasta 300.000 estrellas, si queremos catálogos más grandes debemos crearnos una cuenta de usuario y entrar con nuestra cuenta antes de empezar a hacer consultas.

5.3. Cargar conjuntos de datos

Una vez descargado nuestro subconjunto de datos basta con TV, File → Open... Se nos abre un explorador de archivos para seleccionar la ubicación del fichero, lo seleccionamos y hacemos click en Abrir. En la figura 50 se muestra la ventana para seleccionar qué hacer con algunos atributos que podrían ser nulos. Nos da la opción de elegir con qué parámetros queremos que se calcule la velocidad de las estrellas. Hacemos click en **Load** y se cargará nuestro fichero. Por defecto, el programa descartará aquellas entradas con parámetros nulos y solo calculará la velocidad cuando se disponga del Movimiento Propio y la Velocidad Radial.

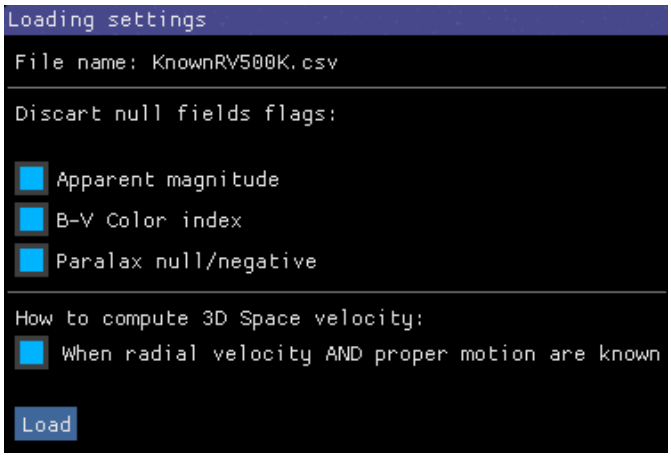

Figura 50: Diálogo para tratar campos nulos

5.4. Ajustando la visualización

Es posible que al cargar nuestro catálogo no veamos ninguna estrella. Esto puede ser porque nuestro catálogo contenga principalmente estrellas muy lejanas o muy poco brillantes y debido a la función de atenuación de tamaño y brillo, estas se hacen invisibles.

Para arreglarlo basta con ir a la sección de **Customize visualization** y jugar principalmente con los sliders en la sección **Size** y en la sección **Alfa**. El slider **Offset** es el que mayor cambio produce en la visualización. También podemos seleccionar usar un tamaño fijo o desactivar la atenuación alfa, esto hará que inmediatamente podamos ver dónde están nuestras estrellas.

6. Definiciones y abrevieturas

Definiciones

C++ Lenguaje de programación de propósito general orientado a objetos.

CLion Entorno de desarrollo enfocado en los lenguajes C y C++ .

CMake Herramienta para facilitar la tarea de compilación en lenguajes C y C++ .

Diagrama Hertzsprung–Russell Diagrama 2D que representa a las estrellas en función de su brillo y color.

Fragment Shader Programa software que se encarga del procesamiento de los fragmentos (píxeles) en el pipeline de rendering.

GAIA Nombre de la misión llevada a cabo por la Agencia Espacial Europea con el objetivo de cartografiar y medir parámetros fotométricos principalmente de las estrellas de nuestra galaxia.

Gaia Archive Aplicación web de la Agencia Espacial Europea que facilita realizar consultas en los distintos catálogos que han publicado.

Geometry Shader Programa software que se encarga de modificar la geometría de las primitivas en el pipeline de rendering.

Gliese Catalog of Nearby Stars Catálogo con todas las estrellas a una distancia de 25 parsecs.

Hipparcos Misión de la Agencia Espacial Europea predecesora de la misión Gaia.

HYG Unión de los catálogos Hipparcos, Yale Bright Star y Gliese Catalog of Nearby Stars.

Java Lenguaje de programación orientado a generar aplicaciones que puedan ejecutarse en cualquier sistema operativo sobre la máquina virtual de Java.

KD-Tree Estructura de datos espacial K-dimensional utilizada para realizar búsquedas eficientes por rango de coordenadas.

Magnitud Absoluta Parámetro que indica el brillo de una estrella observada a 10 parsecs de distancia.

Magnitud Aparente Parámetro que indica el brillo de una estrella observada desde la Tierra.

Movimiento Propio Medida de la velocidad a la que aparentan moverse las estrellas transversalmente.

OpenGL API multiplataforma para renderizar gráficos en 2D y 3D.

Tycho Catálogo publicado en paralelo con Hipparcos.

Velocidad Radial Medida de la velocidad en la dirección estrella-observador.

Velocidad Transversal Velocidad de las estrellas sobre el plano perpendicular a la Velocidad Radial.

Vertex Shader Programa software que se encarga del procesamiento de los vértices en el pipeline de rendering.

Vulkan API multiplataforma para renderizar gráficos orientada al desarrollo de video-juegos.

Yale Bright Star Catálogo de estrellas con magnitud aparente menor a 6,5.

Índice de color B-V Valor numérico utilizado en astronomía que sirve para determinar el color o la temperatura de una estrella.

Abreviaturas

API Application Programming Interface

CIE Commission Internationale de l'Éclairage

CSV Comma Separated Values

ESA European Space Agency

FITS Flexible Image Transport System

FOV Field Of View

FPS Frames Per Second

GEDR3 Gaia Early Data Release 3

GLFW Graphics Library Framework

GUI Graphical User Interface

IBO Index Buffer Object

IDE Integrated Development Environment

NASA National Aeronautics and Space Administration

TV Telescopio Virtual

VBO Vertex Buffer Object

VRAM Video Random Access Memory

7. Bibliografía

[1] European Space Agency. *Gaia Archive*. URL: https://gea.esac.esa.int/archive/ (visitado 10-09-2021).

[2] European Space Agency. *The Hipparcos Space Astrometry Mission*. URL: https://www.cosmos.esa.int/web/hipparcos/home (visitado 10-09-2021).

[3] COSMOS The SAO Encyclopedia of Astronomy. *Absolute Magnitude*. URL: https://astronomy.swin.edu.au/cosmos/a/Absolute+Magnitude (visitado 10-09-2021).

[4] Konstantin Bikos. *Altitude & Azimuth: The Horizontal Coordinate System*. URL: https://www.timeanddate.com/astronomy/horizontal-coordinate-system.html (visitado 10-09-2021).

[5] A. Blaauw y col. «The New I.A.U. System of Galactic Coordinates (1958 Revision)». En: *Monthly Notices of the Royal Astronomical Society* 121.2 (ago. de 1960), págs. 123-131. ISSN: 0035-8711. DOI: 10.1093/mnras/121.2.123. eprint: https://academic.oup.com/mnras/article-pdf/121/2/123/8078181/mnras121-0123.pdf. URL: https://doi.org/10.1093/mnras/121.2.123 (visitado 10-09-2021).

[6] *Celestia image of Pluto*. URL: https://celestia.space/images/gallery/pluto-charon.jpg (visitado 10-09-2021).

[7] Mitchell Charity. *Star color*. URL: http://www.vendian.org/mncharity/dir3/starcolor/details.html (visitado 10-09-2021).

[8] Omar Cornut. *Dear ImGui*. URL: https://github.com/ocornut/imgui (visitado 10-09-2021).

[9] David Herberth. *glad: Multi-Language Vulkan/GL/GLES/EGL/GLX/WGL Loader-Generator*. URL: https://github.com/Dav1dde/glad (visitado 10-09-2021).

[10] *Hertzsprung–Russell diagram Figure*. URL: https://commons.wikimedia.org/wiki/File:HRDiagram.png (visitado 10-09-2021).

[11] Ejnar Hertzsprung. «On the Use of Photographic Effective Wavelengths for the Determination of Color Equivalents». En: *Publications of the Astrophysical Observatory in Potsdam* (1911).

[12] Bongsoon Kang y col. «Design of Advanced Color - Temperature Control System for HDTV Applications». En: *Korean Physical Society* 41.6 (2002), pág. 867. URL: https://web.archive.org/web/20190303161843/http://pdfs.semanticscholar.org/cc7f/c2e67601ccb1a8fec048c9b78a4224c34d26.pdf (visitado 10-09-2021).

[13] David Nash. *The HYG Database*. URL: https://github.com/astronexus/HYG-Database (visitado 10-09-2021).

[14] Thor Olson. *The Colors of the Stars*. 1998. URL: http://www.nightscapes.net/techniques/TechnicalPapers/StarColors.pdf (visitado 10-09-2021).

[15] *Open Space earth screenshot*. URL: https://www.openspaceproject.com/images (visitado 10-09-2021).

[16] Antoni Sagristà y col. «Gaia Sky: Navigating the Gaia Catalog». En: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pág. 1078. DOI: 10.1109/TVCG.2018.2864508.

[17] NASA's HEASARC: Software. *CCfits*. URL: https://heasarc.gsfc.nasa.gov/fitsio/CCfits/ (visitado 10-09-2021).

[18] *Space Engine Landscape Image*. URL: http://spaceengine.org/universe/landscapes (visitado 10-09-2021).

[19] *Tabla de coeficientes de amortización lineal*. URL: https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/

Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml (visitado 10-09-2021).

[20] Tim Trott. *The Astronomical Magnitude Scale*. 2008. URL: https://web.archive.org/web/20210720081837/https://lonewolfonline.net/astronomical-magnitude-scale/ (visitado 10-09-2021).

[21] Joey de Vries. *LearnOpenGL, Camera*. URL: https://learnopengl.com/Getting-started/Camera (visitado 10-09-2021).

[22] *XVII Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública*. URL: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2018-3156 (visitado 10-09-2021).