Term Project Final Report

Stars and Universe
December 19, 2024
21-123 황동우, 23-017 김민겸, 23-050 박서진, 23-119 최원 GROUP 4

Simulation Methodology

The model of the simulation was extracted from the website, Github. The link for this website is: https://github.com/beltoforion/Galaxy-Renderer. This zip of code run the physics engine of the spirial galaxy corporating the factors of the density wave theory. This code approaches simulation by developing simple mathematical model and running it kinematically. This entire project runs in a complex matnner; each of the files are interrelated like a web, each responsible for major functionalities. The following are the explanations for the major C++ header files.

main.cpp

This file is a center of the web. It connects all the header and footer files and forms a huge simulation model. No additional functions were included in this file. Yet, it is an important file that combines all mathematical function into one simulation.

CumulativeDistributionFunction.cpp

The CumulativeDistributionFunction class is responsible for generating a probabilistic distribution of stellar positions based on the intensity profile of the galaxy. The mathematical model accounts for the radial dependence of intensity, ensuring that the simulated stars follow realistic spatial distributions in the galaxy.

The intensity distribution is inspired by empirical models of galaxies, which consist of two primary components: the bulge and the disk.

The bulge intensity follows the well-known **de Vaucouleurs' law**, which describes how the brightness decreases in the central region of galaxies. It is expressed as:

$$I_{\text{bulge}}(R) = I_0 \cdot e^{-k \cdot R^{1/4}}$$

where. k is a decay factor and R is a radial distance from the center.

Besides, for the galactic disk, the brightness distribution follows an *exponential decay* model, known as **Freeman's law**. It is given by:

$$I_{\text{disk}}(R) = I_0 \cdot e^{-\frac{R}{a}}$$

where a is a scale length, which determines the rate of decay.

These processes were done by randomly generating a star and integrating the cumulative distribution function. The cumulative distribution function is built using a numerical integration of the intensity profile. The intensity $I(R)$ at a given radial distance R is combined into a probability distribution function, normalized as:

$$P(r) = \frac{\int_{R_{\text{min}}}^{r} I(x)\,dx}{\int_{R_{\text{min}}}^{R_{\text{max}}} I(x)\,dx}$$

These processes were easily done by applying a Simpson's rule, which efficiently approximates the integration.

Galaxy.cpp

The code is written after a theoretical analysis of how stars would move at certain physical conditions. Instead of inserting a differential equation of the motion, kinematic formulas were utilized to predict the motion of the stars. By introducing a dark matter, this model was able to successfully explain the conventional method of illustrating the spirial arms.

The eccentricity $e(r)$ of star orbits varies with the radial distance r, transitioning smoothly across different regions of the galaxy:

$$e(r) = \begin{cases} 1 + \frac{r}{r_{\text{core}}} \cdot (e_{\text{inner}} - 1), & r < r_{\text{core}}, \\ e_{\text{inner}} + \frac{r - r_{\text{core}}}{r_{\text{galaxy}} - r_{\text{core}}} \cdot (e_{\text{outer}} - e_{\text{inner}}), & r_{\text{core}} \leq r \leq r_{\text{galaxy}}, \\ e_{\text{outer}} + \frac{r - r_{\text{galaxy}}}{r_{\text{farfield}} - r_{\text{galaxy}}} \cdot (1 - e_{\text{outer}}), & r_{\text{galaxy}} < r < r_{\text{farfield}}, \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

Here, e_{inner} and e_{outer} are the eccentricities at the core edge and disk edge, respectively, and r_{core}, r_{galaxy}, and r_{farfield} represent the boundaries of the core, disk, and outer region.
The class calculates the orbital velocity of stars using Newtonian mechanics, incorporating the contribution of dark matter. The velocity $v_{\text{dark}}(r)$ at a distance r is given by:

$$v_{\text{dark}}(r) = \sqrt{\frac{GM(r)}{r} + \frac{GM_{\text{halo}}(r)}{r}}, \tag{2}$$

where G is the gravitational constant, $M(r)$ is the baryonic (stars and gas) mass enclosed within r, and $M_{\text{halo}}(r)$ is the dark matter halo mass. If dark matter is excluded, the velocity reduces to:

$$v_{\text{nodark}}(r) = \sqrt{\frac{GM(r)}{r}}. \tag{3}$$

GalaxyWnd.cpp

To simulate spiral density waves, sinusoidal perturbations are applied to the stellar positions. For a radial distance r and an angular offset θ, the perturbed coordinates are:

$$x' = x + \frac{r}{A}\sin(2\theta N), \tag{4}$$

$$y' = y + \frac{r}{A}\cos(2\theta N), \tag{5}$$

where A is the perturbation amplitude damping factor, N is the number of perturbations per ellipse, and θ is the azimuthal angle. These perturbations modify the ideal elliptical orbits to produce a realistic spiral pattern.

The temperature gradient of stars influences their color, which is calculated using the empirical color-temperature conversion functions:

$$R(T) = \begin{cases} 255, & T \leq 6600, \\ 329.69873 \cdot \left(\frac{T}{100} - 60\right)^{-0.13320}, & T > 6600, \end{cases} \tag{6}$$

$$G(T) = \begin{cases} 99.47080 \cdot \ln\left(\frac{T}{100}\right) - 161.11957, & T \leq 6600, \\ 288.12217 \cdot \left(\frac{T}{100} - 60\right)^{-0.07551}, & T > 6600, \end{cases} \tag{7}$$

$$B(T) = \begin{cases} 0, & T \leq 2000, \\ 138.51773 \cdot \ln\left(\frac{T}{100} - 10\right) - 305.04479, & 2000 < T \leq 6600, \\ 255, & T > 6600. \end{cases} \tag{8}$$

Here, T is the stellar temperature in Kelvin, and $R(T)$, $G(T)$, and $B(T)$ represent the red, green, and blue color channels, respectively. This equation was written based on the Planck's law of radiation and a calibration of a discrepency that occurs when the computer montior displays the color.
The galaxy also includes dust clouds and HII regions, which are initialized probabilistically. Dust clouds exhibit a temperature gradient, causing the inner regions to appear yellow and the outer regions blue, consistent with star formation zones. HII regions are modeled as bright red cores surrounded by diffuse particles.

Helper.cpp

This file simply defines the constants that are often used in the calculations. It is different from the role of the header files in that header files only mentions the fact that they will define a certain constant, function, or an event for the memory management. However, in this file, the value of the constant are defined to utilize it in the actual calculations. The table below is the table of the constant defined, and their values.

Table 1. Constants defined in Helper.hpp

Constant Name	Value	Meaning
PC_TO_KM	$3.08567758 \times 10^{13}$ km	Converts 1 pc to km.
SEC_PER_YEAR	31,557,600.0 seconds	Number of seconds in one year.
DEG_TO_RAD	0.017453 radians/degree	Converts degrees to radians.
RAD_TO_DEG	57.2958 degrees/radian	Converts radians to degrees.
CONTANT_OF_GRAVITY	$6.674 \times 10^{-11}\,\mathrm{m^3 kg^{-1} s^{-2}}$	Gravitational constant G.
PI	3.14159265359	Mathematical constant π.

SDLWnd.cpp

This file is responsible for initializing and managing the rendering environment using the SDL library. This file serves as the backbone for rendering shaders, textures, and world geometry, acting as a bridge between the graphical operations of OpenGL and the SDL framework.

- **Initialization of the Rendering Context:** The file initializes the SDL window, OpenGL context, and related attributes such as double buffering, accelerated visuals, and color buffer sizes.
- **Camera Management:** The file implements a virtual camera system by setting up a view matrix using the `glm::lookAt` function and a projection matrix using `glm::ortho`. This allows transformations like scaling, positioning, and orientation changes of the rendered objects.
- **Axis Scaling and Projection Control:** Users can dynamically adjust the field of view (FOV) and scale axes, which impacts how the galaxy is visualized in the rendered scene.

The SDLWnd.cpp file provides a robust interface between the galaxy simulation and the graphics rendering pipeline. By managing OpenGL states, camera matrices, and event-driven updates, it ensures smooth rendering and real-time interaction with the simulated galaxy. This file is essential for visualizing the galaxy as it allows the program to draw and update complex structures, including stars, density waves, and axes, in an efficient and interactive manner.

TextBuffer.cpp

The **TextBuffer.cpp** file handles the rendering of on-screen text using SDL2_TTF (TrueType Font) and OpenGL. This file is crucial for displaying simulation parameters, labels, and other dynamic text-based information directly within the rendering window.

- **Font Initialization:** The file loads font files (e.g., `arial.ttf`, `consola.ttf`) at different sizes for rendering regular text, captions, and small annotations.
- **Shader Program Setup:** A custom vertex and fragment shader program is used to render the text as textures mapped to 2D quadrilaterals. The vertex shader applies the orthographic projection to position text correctly on the screen.
- **Texture Generation:** Text is rendered onto SDL surfaces using `TTF_RenderText_Blended`, then converted into OpenGL textures. These textures are uploaded to the GPU for rendering.

The TextBuffer.cpp file is critical for providing real-time feedback and information to the user during the simulation. By displaying labels (e.g., axis markers), simulation parameters (e.g., galaxy radius, eccentricity), and help screens, it enhances the usability and interactivity of the program. The use of OpenGL for rendering ensures that text overlays seamlessly integrate with the graphical output without significant performance overhead.