Vanier College

# Celestial Simulator

**Project plan**

**East Meets West**

Ethan Tran 2270000

Dmitrii Cazacu 2264570

Edmon Shi 2249487


Integrative Project in Computer Science and Mathematics

420-204-RE

Yi Wang

March 3, 2024

| Title | Description | Requirements | Dependencies | Priority + Sub Tasks | Due Date+ Assignee |
|---|---|---|---|---|---|
| 1) Create the project environment | Create the project files/java class files, and configure the template and import/configure the libraries. Make sure the project works with the project management software (GitHub). | IntelliJ<br>GitHub<br><br>Libraries:<br>JavaFX<br>Lombok<br>JUnit | Independent | Priority 1<br><br>Subtasks:<br>1A<br>1B<br>1C<br>1D | Due Date:<br>March 1<br><br>Dmitrii<br>Edmon |
| 1A) Creating the Java project in IntelliJ | Opening the JavaFX application template and changing basic settings. | JavaFX template | First thing to be done.<br>Depends on the project plan. | ↑ | Due Date:<br>March 1<br>Ethan |
| 1B) Creating all of the necessary packages and classes from the class diagram | Creating packages: *tests* containing a Driver class, *controller*, *ui*, and *models*.<br>Classes will be created as needed. | IntelliJ - creating packages more easily | Depends on 1A. | ↑ | Due Date:<br>March 1<br>Ethan |
| 1C) Importing all of the necessary libraries | Import local and online libraries to the Java project. | Libraries:<br>JavaFX<br>Lombok<br>JUnit | Depends on 1A, 1B. | ↑ | Due Date:<br>March 1<br>Edmon |
| 1D) Creating a project repository on Github and pushing the | Creating the repository on GitHub and sharing it with the other team members. Putting | GitHub | Depends on 1A,1B,1C. | ↑ | Due Date:<br>March 2<br>Ethan |

| | | | | | |
|---|---|---|---|---|---|
| project into Github. | the configured project onto the GitHub platform. | | | | |
| 2) Implementing a basic version of the simulation | Before anything else can be done, first, the simulation must function in a basic form. | Java/JavaFX | Depends on 1. | Priority 1<br><br>Subtasks:<br>2A<br>2B<br>2C | Due Date:<br><br>March 12 |
| 2A) Implementing a camera | Implementing a camera that can pan and zoom to navigate the simulation | Camera class in JavaFX | | Priority 1 | Due Date:<br>March 12<br>Ethan |
| 2B) Implementing a grid and a stage | Create a basic non-interactive grid in a Pane of JavaFX that can be adjusted by the size of the stage. | JavaFX<br>FXyz3D | | Priority 1 | Due Date:<br>March 12<br>Ethan |
| 2C) Implementing the Body and Vector3D class | Implement all the attributes and most basic methods of the Body and the Vector3D class for them to be ready for tests. | JavaFX | | Priority 1 | Due Date:<br>March 14<br>Dmitrii |
| 2D) Implement an *AnimationTimer* to compute gravity forces for each body every frame | Implement a direct-sum algorithm $O(n^2)$ for calculating gravitational forces for bodies | JavaFX | | Priority 1 | Due Date:<br>March 15<br>Ethan |
| 2E) Implement logic for collisions between bodies | Write algorithm that computes collision vectors based on distance (only collide if distance between two bodies is less | JavaFX | | Priority 1 | Due Date:<br>March 15 |

| | | | | | |
|---|---|---|---|---|---|
| | than sum of radius) | | | | |
| 3) Implement the Particle class | Implement the Particle class and decide what rules it will follow in the simulation and how it will follow them | JavaFX | Depends on 1 and 2 | Priority 1 | Due Date: March 16 Edmon |
| 4) Create a basic version of the GUI in Scene Builder and bind it to the project | Create scenes in Scene Builder as it is much faster to create and to make changes compared to writing the UI manually. | Scene Builder/JavaFX | Depends on 1, 2, 3 | Priority 1<br><br>Subtasks:<br>4A<br>4B<br>4C | Due Date: March 16<br><br>Edmon |
| 4A) Creating the main scene in Scene Builder | Creating a scene that contains the most important buttons/features of the simulation: Sliders for *speed of the Simulation,* buttons (in a file menu) like Start, Stop, Restart, Add Body and Remove Body. | Scene Builder | Depends on 1, 2, 3 | Priority 1 | Due Date: March 16 Edmon |
| 4B) Building the Scene and controller in IntelliJ | Bind the fxml file to the JavaFX project. Create a controller class to contain methods and linking | JavaFX | Depends on 4A | Priority 1 | Due Date: March 17 Edmon |
| 4C) Binding FXML to controller | Bind UI controls (buttons, sliders, etc.) to variables inside controller class | JavaFX | Depends on 4A, 4B | Priority 1 | Due Date: March 18 Ethan |

| | | | | | |
|---|---|---|---|---|---|
| 5) Implementing body selector | Implementing the selector feature which allows to select a body and see its properties in a HBox on the right of the simulation which will also contain a toggle to anchor the body in time and space | JavaFX | Depends on 1, 2, 4 | Priority 2 | Due Date: March 24 Dmitrii |

| | | | | | |
|---|---|---|---|---|---|
| 6) Polish the GUI for UX | Add the remaining UI elements that we need. Strategically place the most used UI elements for a better user experience. | SceneBuilder, JavaFX | Depends on 1, 2, 4 | | Due Date: March 28 Edmon |
| 7) Optimize simulation using Barnes-Hut algorithm | Instead of direct-sum, recursively subdivide space into smaller regions (TreeNode) and store them in an OctTree. Calculate the center of mass of all bodies within each node. For each body, travel the tree and estimate distant cells under the Barnes-Hut Criterion | | Depends on 1, 2, 4 | | Due Date: March 31 Ethan |

| | | | | | |
|---|---|---|---|---|---|
| | (threshold) to be a single body. | | | | |
| 8) Add different preset bodies under the add objects UI control. | Add a submenu for when the user selects the Add Object option. This will provide a selection of preset celestial bodies for the user to add to the simulation. | | Depends on 1, 2, 4 | | Due Date: March 31 Dmitrii |
| 9) Saving and loading the settings | Implementing the *Save Settings* and *Load Settings* buttons and bind them to methods in controller | Serializable | Depends 1, 2, 4 | Priority 2 | Due Date: April 16 |
| 9A) Create class | Create the Level class containing a list of bodies and settings and parameters for the simulation (such as gravity, speed of simulation, camera position and angle, etc.) to be serialized | Serializable | Depends on 6 | Priority 2 | Due Date: April 24 |

| | | | | | |
|---|---|---|---|---|---|
| 9B) Logic for saving and loading | Implement the logic for saving all the parameters, settings, bodies (and their respective data) into a Level class and serializing it. Users should be able to load a file containing the byte stream of the Level class | Serializable | Depends on 6A | Priority 2 | Due Date: April 24 |
| 10) Implement Help Dialog | Create the modal help dialog. It will contain frequently asked questions (FAQs) and help text to help the user navigate through the application. | Scene Builder, FXML | Depends on 1, 4 | Priority 3<br><br>Subtask 10A) 10B) | Due Date: April 29 |
| 10A) Create a popup using fxml | Create the fxml scene and the controller class for this simple popup in scene builder. Link it to the code with a controller class. | Scene Builder, FXML | Depends on 1, 4 | Priority 3 | Due Date: April 29 |
| 10B) Add dialog to code | Implement integration of the FXML file from 10A into the Java Project | FXML | Depends on 1, 4 | Priority 3 | Due Date: April 29 |