Group 90 | MS2 Report

Vision:

Our vision has evolved significantly since MS1. Initially, we considered a Minesweeper implementation with procedural generation, but after discussions with our PM, we pivoted to an N-body gravitational physics simulation. Following feedback from our project guide about the need for greater user interactivity, we expanded our scope beyond a simple 3-body problem demonstration. Our current vision is to create an interactive physics sandbox where users can observe and manipulate gravitational interactions between celestial bodies. Users will be able to control simulation parameters such as time step speed, and we envision features like a "creative mode" where users can place bodies with custom masses, positions, and velocities, experiment with different orbital configurations, and visualize the resulting gravitational dynamics in real-time.

Summary Of Progress:

Between MS1 and MS2, we implemented the core physics engine and established a working graphical interface. We built a complete N-body gravitational simulation from scratch, including a 3D vector mathematics library, gravitational force calculations using Newton's law, and numerical integration using the Euler method. On the frontend, we created a real-time visualization using OCaml's Graphics library that displays a stable 2-body orbital system with interactive speed controls (pause/unpause, speed up/slow down). The system successfully demonstrates accurate orbital mechanics with bodies orbiting around their common center of mass, providing the foundation for expanding to N-body scenarios and additional user interactions in the next milestone. Our next step is a 3D GUI.

Activity breakdown:

Felix Grimm (fjg45)

- **Responsibilities:** Backend physics engine development
- **Activities Participated In:**
    - Core engine architecture design
    - Physics algorithm implementation
    - Module interface design and documentation
- **Features Delivered:**
    - 3D vector mathematics library (vec3.ml/mli) with vector arithmetic, dot product, and normalization
    - Celestial body representation module (body.ml/mli) with immutable update functions
    - Physics engine (engine.ml/mli) including:
        - Gravitational force computation using Newton's law of universal gravitation
        - Net force accumulation across multiple interacting bodies
        - Numerical integration using the Euler method for time-stepping
        - Immutable, functional world state updates

- **Hours Spent:** 5.5 hrs

Nathnael Tesfaw (nbt26)

- **Responsibilities:** Frontend GUI development and visualization
- **Activities Participated In:**
    - Graphics system implementation
    - Physics-to-GUI integration
    - User interaction design
- **Features Delivered:**
    - Main GUI application (main.ml) using OCaml Graphics library
    - Unit-scaling system adapting physics engine to pixel coordinates
    - Two-body system initialization with center-of-mass calculations
    - Real-time rendering pipeline with double-buffered graphics
    - User controls: speed adjustment (Z/X keys), pause/unpause (P key), exit button
    - 60 FPS frame timing and simulation loop
- **Hours Spent:** 6-7 hrs

Anthony Paredes-Bautista (ap2357)

- **Responsibilities:** Documentation and project coordination
- **Activities Participated In:**
    - MS2 submission preparation
    - Documentation writing
    - Project organization
- **Features Delivered:**
    - RepoURL.txt
    - gallery.yaml with project demo
    - MS2Report.pdf (this document)
- **Hours Spent:** 3.5 hrs

Maliq Barnes (db927)

Responsibilities: Testing and quality assurance

Activities Participated In:

- Test suite development
- Test case design for physics modules
- Code validation and debugging

Features Delivered:

- Test cases for Vec3 module (vector operations, normalization, edge cases)
- Test cases for Body module (construction, accessors, immutable updates)

- Test cases for Engine module (gravitational force calculations, net force accumulation, time-stepping)
- OUnit2 test framework setup (test/dune, test/test_group90.ml)

Hours Spent: 2-3 hrs

Productivity analysis:
As a team, we were reasonably productive and accomplished the core objectives outlined in our sprints. We successfully delivered both the physics engine and GUI as planned, meeting our MS2 goals of a working simulation with user controls. However, our time estimates were somewhat inaccurate; we underestimated the complexity of the GUI implementation relative to the physics engine. The backend physics modules came together more quickly than anticipated, as the mathematical concepts translated smoothly into code. In contrast, integrating the Graphics library, implementing the unit-scaling system to bridge physics and pixel coordinates, and creating smooth user interactions took more time than we initially allocated. Despite this imbalance in effort distribution, we completed our sprint commitments and delivered a functional product. Moving forward to MS3, we now have a better understanding of the relative complexity of frontend versus backend work, which should help us create more accurate estimates for features like creative mode, additional user controls, and N-body expansion.