Final Project Proposal
Benjamin Collins (brcolli2), Sanchay Javeria

1. **Abstract**
    i. **Project Purpose**
    The purpose of this project is to design a planetary simulation in Unity, allowing the user to place their own planetary body, star, and other interstellar objects. The idea behind this is to help learn simple planetary physics with an easy-to-use interface.
    ii. **Background/Motivation**
    I have always been interested in Astronomy, hence why my major is Astronomy + Computer Science. Therefore, I think it would be interesting to apply the knowledge and techniques I have learned in my Astronomy courses with those that I have learned with my Computer Science courses. Doing this in Unity allows me to learn more about a new language (that is, C#), and work with a useful tool.
2. **Technical Specifications**
    i. **Platform**
    Windows application.
    ii. **Programming Languages**
    C#.
    iii. **Stylistic Conventions**
    Description of each class, method, and large bodies of code where the function isn't immediately clear. Separate types of methods and functions within each class, AKA split class methods, constructors, getters/setters. Private class attributes are camelCase with a leading '_', methods are CapitalCase, local variables are camelCase.
    iv. **SDK**
    Not applicable.
    v. **IDE**
    Visual Studio.
    vi. **Tools/Interfaces**
    Unity and Blender.
    vii. **Target Audience**
    Astronomy enthusiasts.
3. **Functional Specifications**
    i. **Features**
        - Place planetary and stellar objects.
        - Control flow of time.
        - Change physical and (limited) visual properties.
        - Simulate real life solar system bodies.
        - Introduce outside or spurious forces.
        - Get simple, realistic data (i.e. forces applied in N-body simulations).
        - Have fun.
    ii. **Scope of project**

- Large systems (N > 10) may slow down calculations. When this happens, possibly change to more simplistic calculations or do not allow the user to place a large amount of objects.
- Large time intervals make calculating future placements difficult.
- Very massive objects ($M > 8M_{\odot}$) paired with very small objects ($M \sim M_E$).

4. **Timeline**
   i. **Week 1**
      - Add UI elements for body controls, such as placement, movement, statistical display, and property management.
      - Add basic physics operations and interaction.
      - Integrate collision detection system.
      - Allow change in y position during placement.
      - Hide UI upon placement.
   ii. **Week 2**
      - Implement raycast system for object placement.
      - Introduce known planet and star values.
      - Allow for setups of known systems (such as solar system).
      - Add saves.
   iii. **Week 3**
      - Allow object removal.
      - Control amount of objects/add restraints on properties.
      - Allow time controls, unit changes.
      - Allow time reversal.
   iv. **Week 4**
      - Add start screen, credits screen, help screen.
      - Add textures to objects.
      - Add background.
      - Allow for importing of images.
      - Add animations (collisions).
5. **Future Enhancements**

   Add other structures, such as galaxies, clusters, nebulae, black holes, neutron stars. Possibly include gas physics. Add dark matter patches if feeling brave. I would like to work on it in the future, but potentially only get around to half of the features mentioned.