Gravitatonal N-Body Simulations

By: Ankit & Ryan

Brief description of the system

We created a simulation which describes the motion of planets or stars according to Newton's Law of Gravity.
Our simulation is 2D, handle systems of arbitrary size however computation time needed for each step increases dramatically with the addition of additional bodies.

Goals / Objectives

Primary Objectives:
- Solve the system using multiple computational methods
- Compare the computational cost of each method with the error induced by the method
- Compare the Brute force and reduced calculation techniques for each method
- Methods under question: Euler, Runge-Kutta 2, Runge-Kutta4

Secondary Objectives:
- Witness effects of collision handling
- Learn about Stable versus Chaotic systems

Graphical Model of the System

Model accuracies / inaccuracies

- Modelling collisions to be perfectly inelastic conserves momentum but does not account for realistic astronomical collisions
- Newtons Law of Gravity exactly represents instantaneous force on objects
- Our System considers no other forces which could possibly exist in a real astronomical system

Simulation Implementation

- This simulation was done in vpython it has a full graphical view of orbiting bodies
- Calculated the position changes from Force using:
  - Euler
  - Runge-Kutta 0(2)
  - Runge-Kutta 0(4)

Result / Analysis

Floating Point Operations per update

Euler: n^2 + 2n
Euler Reduced: n^2

RK2: 2n^2 + 5n
RK2Reduced: 2n^2 + 4n

RK4: 5n^2 + 12n
RK4Reduced: 2n^2 + 11n

Questions?

Demo

photo credit Nasa / Goddard Space Flight Center / Reto Stöckli

Gravitatonal N-Body Simulations

By: Ankit & Ryan

Brief description of the system

We created a simulation which describes the motion of planets or stars according to Newton's Law of Gravity.
Our simulation will handle systems of arbitrary size however computation time needed for each step increases dramatically with the addition of additional bodies.

Goals / Objectives

Primary Objectives:
- Solve the system using multiple computational methods
- Compare the computational cost of each method with the error introduced by the method
- Compare the Brute force and reduced calculation techniques for each method
- Methods under question: Euler, Runge-Kutta 2, Runge-Kutta4

Secondary Objectives:
- Witness effects of collision handling
- Learn about Stable versus Chaotic systems

Graphical Model of the System

Model accuracies / inaccuracies

- Modeling collisions to be perfectly inelastic conserves momentum but does not account for realistic astronomical collisions
- Newtons Law of Gravity exactly represents instantaneous force on objects
- Our System considers no other forces which could possibly exist in a real astronomical system

Simulation Implementation

- This simulation was done in xpython it has a full graphical view of orbiting bodies
- Calculated the position changes from Force using:
  - Euler
  - Runge-Kutta O(2)
  - Runge-Kutta O(4)

Questions?

Result / Analysis

Floating Point Operations per update

Euler: n^2 + 2n
Euler Reduced: n^2

RK2: 2n^2 + 5n
RK2Reduced: 2n^2 + 4n

RK4: 5n^2 + 12n
RK4Reduced: 2n^2 + 11n

Demo

photo credit Nasa / Goddard Space Flight Center / Reto Stöc

Brief description of the system

We created a simulation which describes the motion of planets or stars according to Newton's Law of Gravity.
Our simulation will handle systems of arbitrary size however computation time needed for each step increases dramatically with the addition of additional bodies.

Goals / Objectives

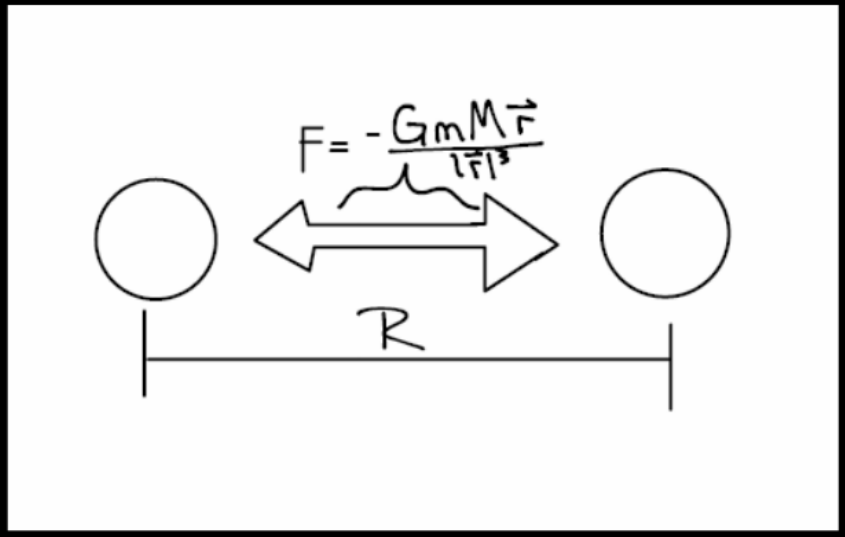
Primary Objectives:

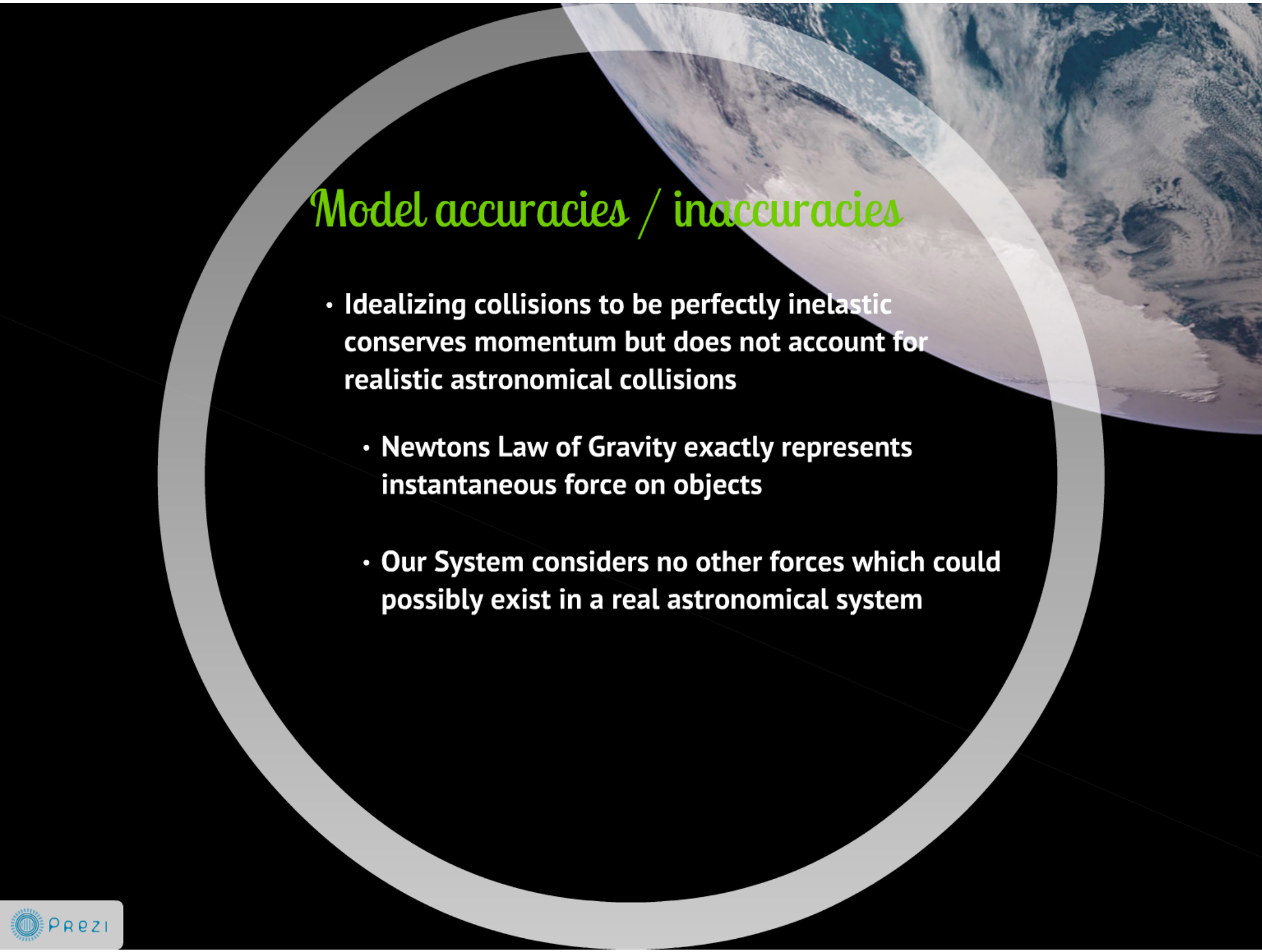
- Solve the system using multiple computational methods
- Compare the computational cost of each method with the error induced by the method
- Compare the Brute force and reduced calculation techniques for each method
- Methods under question: Euler, Runge-Kutta 2, Runge-Kutta4

Secondary Objectives:

- Witness effects of collision handling
- Learn about Stable versus Chaotic systems

Graphical Model of the System





Model accuracies / inaccuracies

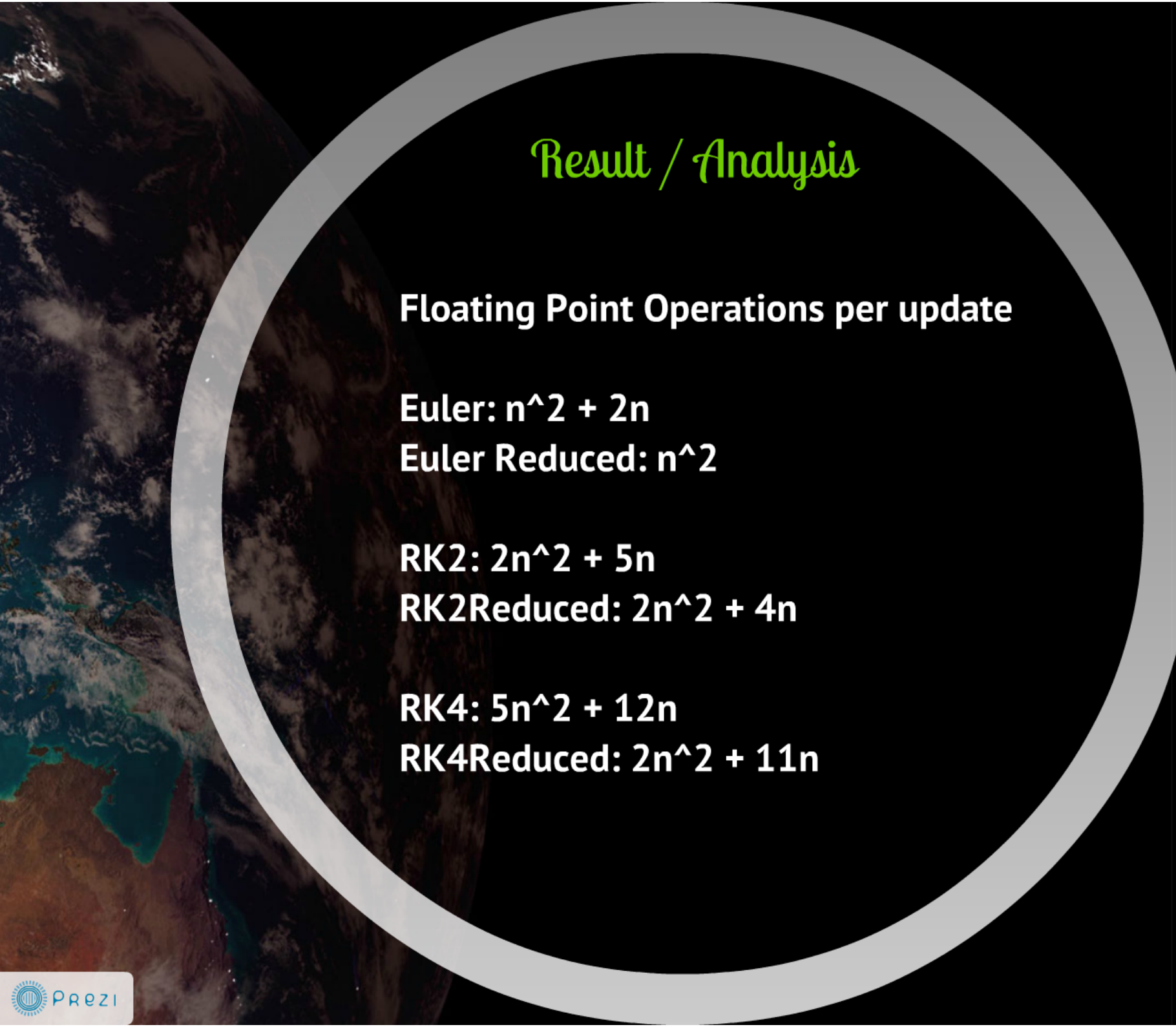
- Idealizing collisions to be perfectly inelastic conserves momentum but does not account for realistic astronomical collisions
  - Newtons Law of Gravity exactly represents instantaneous force on objects
  - Our System considers no other forces which could possibly exist in a real astronomical system

Simulation Implementation

- This simulation was done in vpython it has a full graphical view of orbiting bodies

- Calculated the postion changes from Force using:
    - Euler
    - Runge-Kutta O(2)
    - Runge-Kutta O(4)


Demo


Result / Analysis
Floating Point Operations per update
Euler: n^2 + 2n
Euler Reduced: n^2
RK2: 2n^2 + 5n
RK2Reduced: 2n^2 + 4n
RK4: 5n^2 + 12n
RK4Reduced: 2n^2 + 11n


Questions?