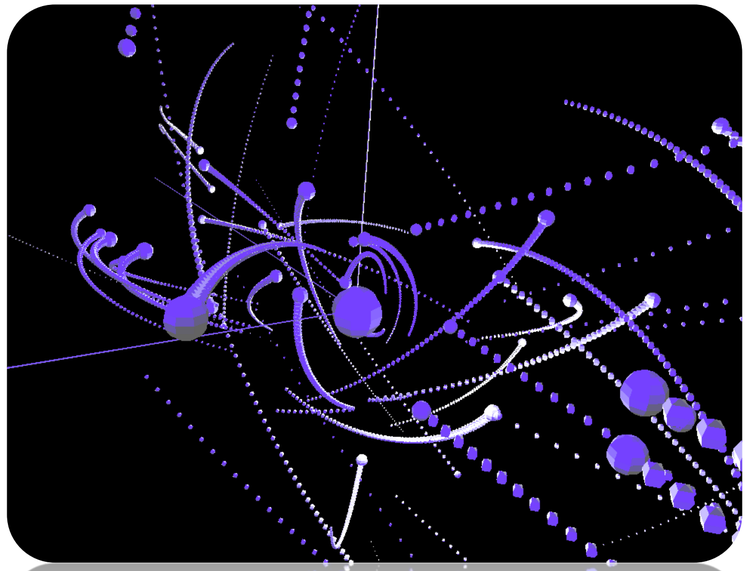
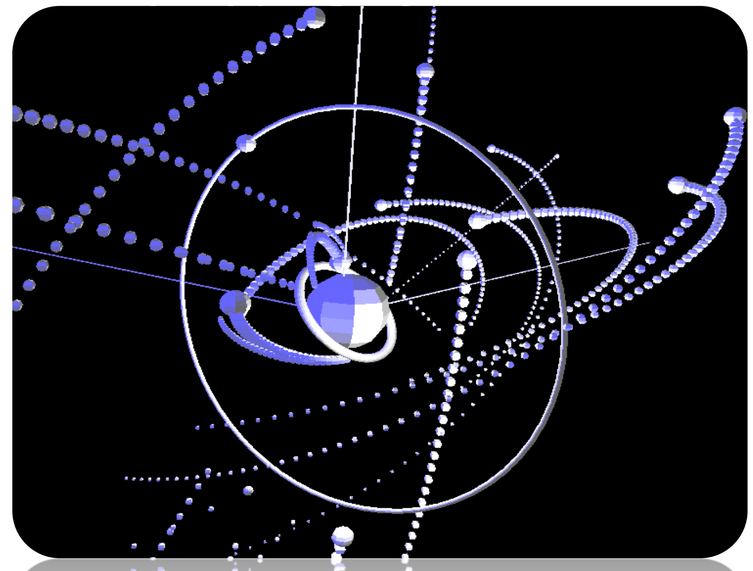


GSim

An Interactive Gravity Simulator

Peter Cottle
CS 184 – Fall 2011
UC Berkeley



3-2-1 Euler Angles

GSim uses an innovative navigation and orientation method that uses a set of 3-2-1 Euler angles and local angular velocities. When the user wants to rotate the current orientation, each arrow key induces a local angular velocity. These local angular rates are transformed into rates of change of each Euler angle with the following relation:

$$\begin{bmatrix}\dot{\psi}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta)\\ 0 & cos(\phi) & -sin(\phi)\\ 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta)\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}$$

GSim then uses these global Euler angles to calculate the orientation at each frame draw. This results in an orientation method that runs in constant time, requires constant memory, retains the orthogonality of the rotation matrix, **avoids gimbal lockup**, retains information on the global yaw amount for user interaction, and allows drawing from identity on each frame.

Integration Method

The integration method for gravity simulators must be chosen carefully, but common explicit integration schemes like the Euler method or Runge-Kutta do not preserve the energy of the dynamic system. This is because they assume a constant acceleration over a timestep, when acceleration is actually a function of position (and thus time). The Verlet method of integration overcomes this challenge by using the following implicit formulation:

$$\vec{x}(t+\Delta t) = \vec{x}(t)+\vec{v}(t)\Delta t+\frac{1}{2}\vec{a}(t)\Delta t^2$$

$$\vec{v}(t+\Delta t) = \vec{v}(t)+\frac{\vec{a}(t)+\vec{a}(t+\Delta t)}{2}\Delta t$$

When acceleration is simply a function of position (and not velocity), the second equation changes from an implicit formulation to an explicit one and implementation is fairly simple.

GSim allows integration in both directions in time, and one can integrate forward for several minutes, reverse direction, and eventually arrive at the **same original configuration**.

User Interaction in 3D

The most difficult user interaction issue was particle insertion in 3D space. Because a particle must have a specified position, velocity, and mass, this translates to 7 different variables that must be mapped through mouse clicks. Furthermore, clicking on a 3D image is mapped to a 2D screen coordinate, which inherently looses the sense of “how far” the user wants to click.

To overcome this obstacle, a Google Sketchup® inspired intersection plane is assumed for all clicks. This plane changes depending on the yaw angle, allowing for fully 3D insertion that is dependent on local configuration. An intuitive graphical interface is overlayed in space during particle insertion, making the sequence of three click-drags responsive and intuitive.

When tested on newcomers to the program, new users were able to insert desired particles on the first try after one brief demonstration. This process combined with file import / export has led to a number of interesting initial configuration setups that demonstrate the beauty and complexity of an evolving gravitational system over time.