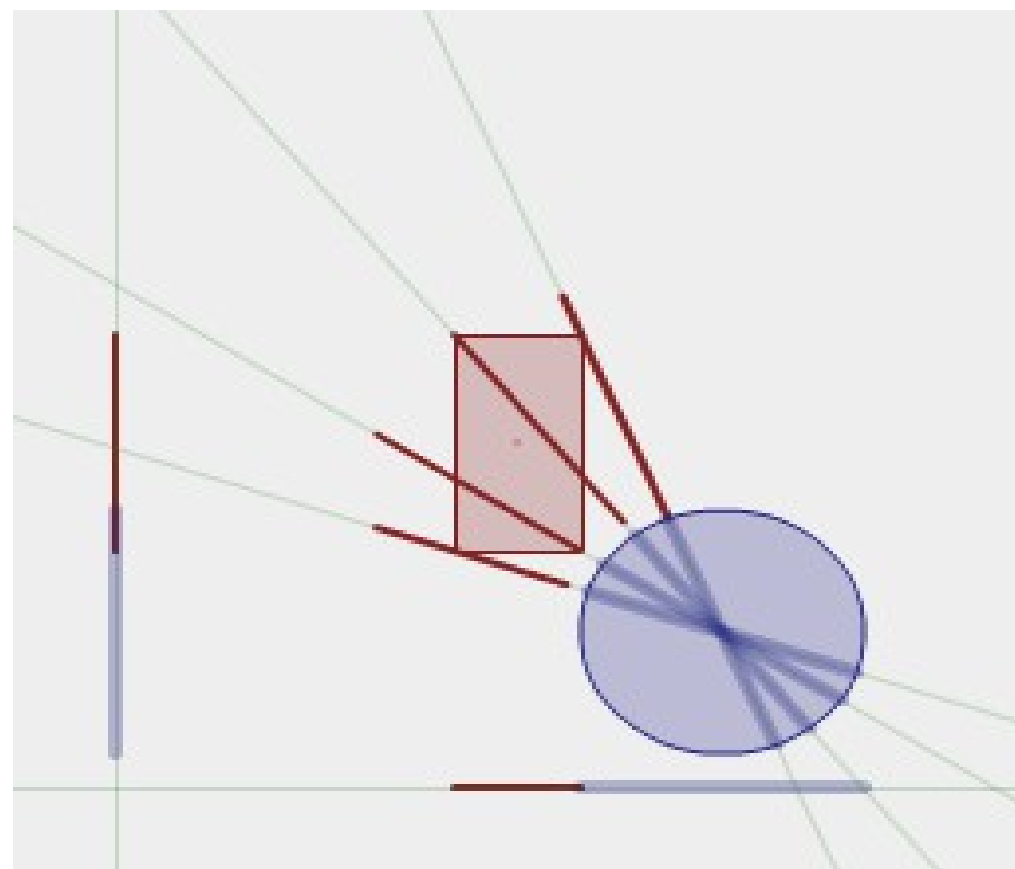


Developing a Rigid-Body Dynamics Physics Engine

Neal Milstein
Computer Systems Lab

Abstract

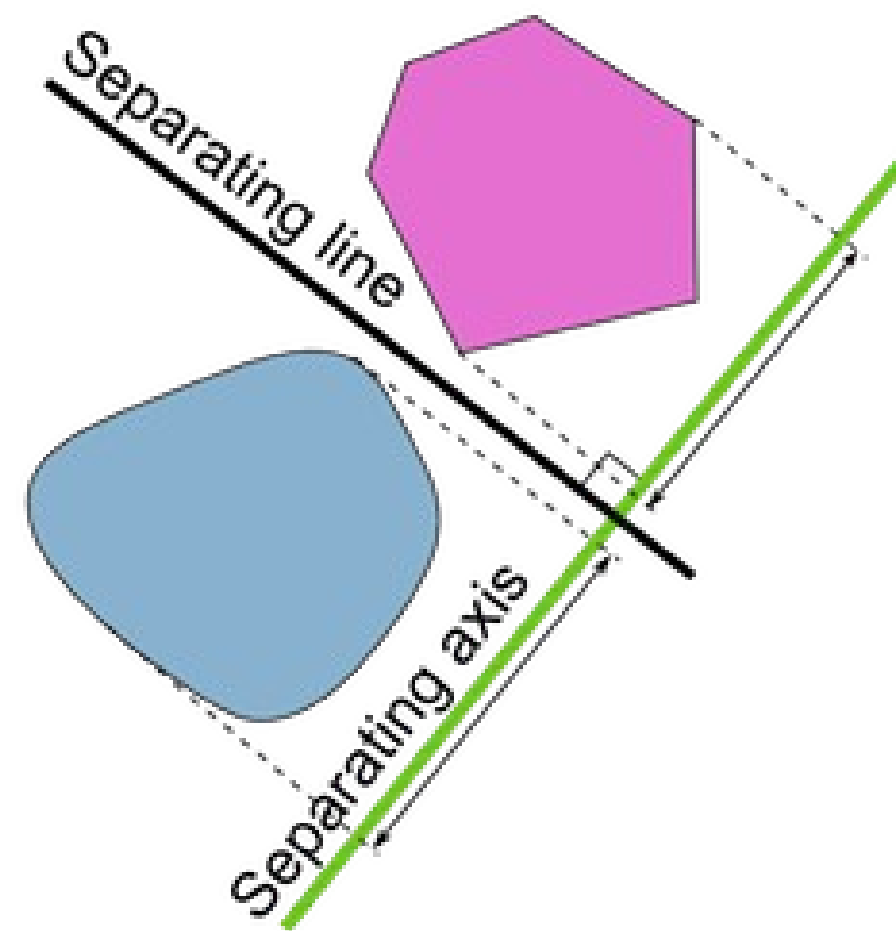
The goal of this project is to create a rigid-body dynamics physics engine in order to allow physics students to visualize and solve physics models. Rigid-body dynamics is the study of the motion of non-deformable, free-moving bodies through space. Such setups involving rigid bodies are prevalent throughout physics courses, leading to the value of such a simulation. The engine will improve upon many current models by focussing on distinctly accurate mathematic techniques. The two modern techniques that will be developed and investigated for optimizations throughout this project are Runge-Kutta 4 integration -- iterative methods developed by German mathematicians to approximate differential equations -- and the Separating Axis Theorem, a mathematical theorem used to detect collisions of convex objects lying on a 2-dimensional plane. The engine interface will allow for the speedy input of a variable number of rigid body and interaction mechanisms, specifically optimized for an educational environment.



The separate Axis Theorem used for Circular Objects. My project uses Runge-Kutta 4 integration do derive results for the responses to these collisions.

Background

Physics engines present the unique problem of combining small-scale rigid-body oriented interactions with large-scale physics engine scalability. Research into rigid body dynamics engines can be broken down into the microcosmic lower-level interactions between rigid bodies, and the macrocosmic large-scale design structures of physics engines. On the microcosmic level, the simulation uses the Runge-Kutta 4 methods of integration and the Separating Axis Theorem to calculate impulse forces and collisions. On the macrocosmic level, the model-view-controller design is used by the engine to separate collision detection algorithms and other interactions from the rest of the engine.



An illustration of the Separating Axis theorem. If all of the separating axes of an two objects are colliding with each other, then the objects are in a state of collision.

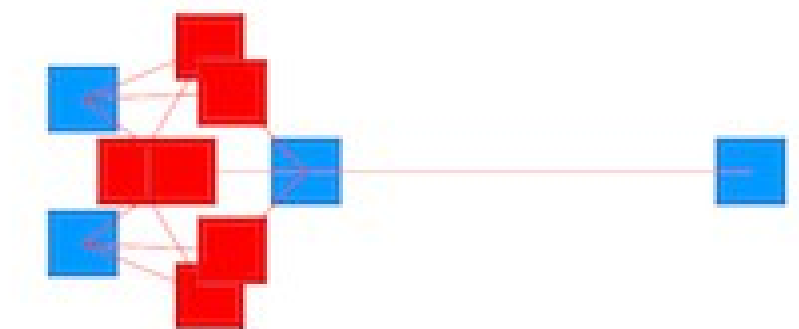
Developments

The engine uses the Model-View-Controller (MVC) programming paradigm to develop my project. The architectural program is used by a number of software systems, such as Microsoft's .NET framework, the open source Ruby on Rails framework, and a number of common software platforms, such as the Facebook application platform and twitter. The MVC design differs from the typical two-part design of physics simulators, where the interface is separated from the simulator, by adding a third module: the model, used to represent the underlying data of the engine.

Discussion of Results

The primary output that will be extracted from the physics engine are numerical results pertaining to the physical states of a simulation after a time step. A number of online physical models, in addition to textbook setups, are available to test the numerical results of my simulation. Furthermore, the graphics displayed by my physics engine can be evaluated for their resemblance to realistic physical phenomena.

One of the most important aspects of my simulation will be its effectiveness in an educational setting. To test the straightforwardness and academic flexibility of my program, I will employ the use of physics students to draw their physical models into the application, after which the models can be evaluated for their similarity to the models portrayed in the textbook. The graphical user interface will be successful if it is fully functional, expandable, straightforward, and easy to use. The time it takes to draw physical models into the application will be measured. Any additional tools required to solve physics setups will be added, and the measuring capabilities required to extract proper physical information will be added. If the physics engine is fully functional and accurate, in addition to providing an educational and straightforward interface, then my project will be a success.



A simulation of a grid of bodies connected by springs using my engine.