Project 1

Simulation Physics, Fall 2012

September 18, 2012

1 Introduction

At the end of this project, you and your partner will have created a fully functional implementation of the classic MS-DOS game Gorillas. There are a few goals:

- Study numerical integration for Newton's laws.
- Practice using a clock.
- Practice using polymorphism in C++.
- Learn about force generators.
- Learn basic collision detection/response.

2 Requirements

1. **Find a partner.** You *must* work on this project with one other person. Pairs should post their names to Piazza *immediately*. You will lose points if you don't do it by 11:59pm on Tuesday, September 18, 2012. There are currently an odd number of students in the class. There will be *one* group of three. The group of three will be the first group that posts on Piazza. If, at the first due date, there are two groups of one, *both groups will receive zeros.*
2. **Velocity Verlet.** Right now the engine uses Euler integration. In class we saw how this can be very bad. For integrating Newton's law we can use a better method called *Verlet Integration*. Read about Velocity Verlet (say, on Wikipedia) and replace the Euler integration in the engine with Velocity Verlet.
3. **Numerical Comparison.** In class we looked at an example comparing Euler Integration and RK4 integration for the simple case of acceleration due to gravity. Write a program that does the same thing, but compares the error due to using Euler integration, Velocity Verlet, and RK4. Write a paragraph explaining the result.
4. **Time Scaling Flags.** In Quiz 2 you implemented a clock that was capable of speeding up, slowing down, and pausing time. A similar clock is provided in this project. Implement a command line flag that allows the user to set the speed of the simulation on startup.
5. **Gravity Force Generator.** Implement a force generator for gravity. Use reasonable values for gravity.
6. **Wind Force Generator.** Implement a force generator that simulates the effect of wind.
7. **Force Registry.** Implement a registry that pairs force generators with particles that the forces apply to. Use the registry in the physics engine in place of the ad hoc method that's there now.
8. **Input Conversion.** User input for the game will come in the form of (θ, v), where θ is an angle and v is a velocity. Write a routine that converts the input to a muzzle velocity vector. More on this after 5 October.
9. **Collisions.** Implement collisions with buildings. More on this after 5 October.

3 Due Dates

There are several due dates for this project:

1. **Due Immediately:** Finding a partner. This is critical. You shouldn't do anything until this is resolved.
2. **Due 28 September 2012:** Implementing Velocity Verlet, performing numerical comparison, and implementing the time scaling flags.
3. **Due 5 October 2012:** Implementing the two force generators and the force registry.
4. **Due 19 October 2012:** Implementing input conversion and collisions.