Institut für Informatik — TU München
Scientific **C**omputing in **C**omputer **S**cience
O. Meister, R. Wittmann

SS 2015

Bachelor Lab Scientific Computing (Game Physics) Worksheet 1: Introduction

The idea of this worksheet is to get you familiar with certain prerequisites that are important when working with the physics engine. There is only one short programming assignment in this first worksheet. However, it is highly recommended that you also elaborately study the other assignments. These assignments are meant as a starting point to programming in C++ and understanding the main parts of the engine.

We recommend to use Eclipse with the CDT plugin and Linux as the development platform since we only give support for this development platform!

Assignment 1: Getting familiar with C++

Many concepts of C++ are similar to those of Java. However, there are several differences. For example, the C++ language does not have a garbage collection implemented. The main advantage of C++ is, that programs, if developed appropriately, are executed faster without a Java virtual machine running in background. Also, several features like inlining and the possibility of hardware oriented programming (cache-efficiency) can be used to gain performance.

Inform yourself on the following topics and answer the respective questions:

- Write your own "Hello World" program. Compile it with the GNU C++ compiler / Visual Studio and run it.
- How are arrays implemented in C++?
- Inform yourself on references and pointers. What is the difference between them?
- Get familiar with the implementation and concept of classes in C++.
- Are constructors/ destructors similar to those in Java?
- A very special feature in C++ are templates. What are the advantages/ disadvantages of templates?
- What is the idea of creating header files? What is the drawback compared to Java development?

For the familiarization with C++, we recommend to read the information provided on the following websites:

- `http://www.cplusplus.com/doc/tutorial/`
  PDF: `http://www.cplusplus.com/files/tutorial.pdf`
- `http://www.cprogramming.com/`

Assignment 2: Have a look at the basic engine code

Get familiar with the general tree structure of the code to answer the following possible questions:

- Compile and run the engine code! You should be able to see different *scenes* by pressing the keys 0 to 9 and as F1 to F10. F11 and F12 are intended for scenes created by yourself in upcoming assignments.
- Consider the separation of .hpp/.h files (header files) and .cpp files (implementation). What might this structure be useful for?
- The physics engine handles different types and quantities of objects. Find out which objects will be supported according to the header definitions. Which properties are needed to describe these objects? Also refer to the physical property section discussed in the lecture.
- In order to advance our simulation in time, we need to perform a discrete time integration. Find the respective integrator()-method within the headers of the physics engine.
- What is the default time step size used in the engine?

Assignment 3: Explicit Euler time-stepping

Your task is to implement the time-stepping to advance both the position $\vec{x}$ and the velocity $\vec{v}$ of an object over time. The explicit Euler method provides the following update rule:

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \Delta t \cdot \vec{v}(t) \tag{1}$$

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \Delta t \cdot \vec{a}(t) = \vec{v}(t) + \Delta t \cdot \frac{1}{m_{object}} \vec{F}(t) \tag{2}$$

where Δt denotes the timestep, $F(t)$ the force at time t acting on the object and m_{object} the mass of the respective object. The acceleration $\vec{a}(t) := \frac{1}{m_{object}} \vec{F}(t)$ is equivalent to the force formulation and will be used within the engine.

Here, the only force we consider is gravity, acting on the center of mass of each object. Thus, the total force is given by

$$\vec{F}(t) := m_{object}\, \vec{g} = m_{object} \begin{pmatrix} 0 \\ -g \\ 0 \end{pmatrix} \tag{3}$$

where $g := 9.81 \frac{m}{s^2}$ is the standard gravity.

In order to insert acceleration due to gravity in the code, the acceleration term from Eq. (3) is implemented inside (*updateConstantAcceleration()*) for all rigid body objects.

- Insert an appropriate call to this method in *cPhysicsEngine_Private::simulationTimestep()*

The integrator()-method (see Assignment 2) updates the positions and velocities of all rigid body objects in our simulation. For additional information have a look at the comments in the source code.

- Implement the Euler update from Eq. (1) in the integrator() method.
- Add assertions to verify that position and velocity are set correctly and that the total energy (= the sum of kinetic and potential energy) does not increase.
- When you run the code in Debug mode you will observe that the last assertion fails and that the system generates energy. A more stable update rule is required apparently – change your implementation accordingly and verify that all assertions pass now.

Compile and run your code. The outcome of your simulation should be falling objects in almost all scenes.

FAQ

If you have questions, first take a look at our FAQ Website:
`http://www5.in.tum.de/wiki/index.php/PSE_Game_Physics_-_Summer_14_-_FAQs`!

Good luck,

Roland & Oliver