

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 you 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 for 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 timestep size used in the engine?

Assignment 3: Explicit Euler time-stepping for falling sphere

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. We therefore restrict ourselves to the explicit Euler method:

$$\begin{aligned}\vec{x}(t + \Delta t) &= \vec{x}(t) + \Delta t \cdot \vec{v}(t) \\ \vec{v}(t + \Delta t) &= \vec{v}(t) + \Delta t \cdot a(t) = \vec{v}(t) + \Delta t \cdot \frac{1}{m} \vec{F}(t)\end{aligned}\tag{1}$$

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

Here, we only want to consider a sphere that is falling to the ground due to gravity. Thus, the only force applied to our system is given by

$$\vec{F}(t) := m \cdot g \cdot (0, -1, 0)^T\tag{2}$$

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

In order to insert the acceleration effects due to gravity, set the acceleration term from Eq. (2) inside (`updateConstantAcceleration()`) for all rigid body objects. Insert an appropriate call to this method in `cPhysicsEngine_Private::simulationTimestep()`

The implementation of the Euler update from Eq. (1) needs to be accomplished within the `integrator()`-method (see Assignment 2) which 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.

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

Assignment 4: 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_11_-_FAQs!](http://www5.in.tum.de/wiki/index.php/PSE_Game_Physics_-_Summer_11_-_FAQs)

Good luck,

Atanas, Martin & Philipp