Course name: Physics
Course code: GT1 (FGA2.GT1-03)
Academic year: 2013-2014
Lecturer: Giuseppe Maggiore
Number of EC's: 4

1 Introduction

In this document we describe the *Physics* course.

The course aims at teaching how basic rigid body physics can be simulated in real-time on a modern computer.

The objective is for students to be able to: *(i)* intuitively understand the basic equations of rigid body physics; *(ii)* know how to translate those equations into a working physics engine; and *(iii)* know the trade-offs that a physics engine must chose between.

At the end of the course the students will be able to program a simplified, robust, and general purpose physics engine.

The course will require *multiple assignments*, roughly one after each lecture or two. The various assignments build towards a working physics engine. Each assignment requires handing in two deliverables: *(i)* a working program, its source, its compiled executable, and a video of the program running; *(ii)* a written discussion and description of the implementation. The program is built [1] in groups of two to three students, while the discussion is handed in (and written) by each student individually.

1.1 Relation to other courses

The course is related the preceding **Programming 1** through **4** courses, in that it requires students to be fluent with structured, higher level programming language such as C++. The course is also related with the **Mathematics 1** through **4** courses, in that it requires students to work with mathematical functions, derivatives, integrals, differential equations, numerical methods, trigonometry, and algebra.

1.2 Relation to the industry

In modern games, physics is not just an interesting addition; rather, it is a necessary component that is expected to work flawlessly.

In order to be able to delivery correct-looking physics simulations in games, a practitioner needs to understand the underlying theoretical framework of rigid body (Newtonian) physics. Moreover, a physics engine developer must be aware of the mainstream techniques for performing (and optimizing) collision detection and also collision resolution through impulse computation in the presence of (multiple, simultaneous) constraints.

[1] In either C++ (recommended) or C# (acceptable)

Even in those cases when an existing, off-the-shelf physics engine is used, knowledge of the above topics is a requirement in order to be able to make an informed choice.

Moreover, the computation of external forces to the bodies of a physics engine (whether custom or existing) are an extremely important aspect that needs to be modelled separately. Depending on the game setting, different forces will be at play: friction, Earth-surface-gravity, general gravity, Magnusson, tires on asphalt, wings on air, etc. Studying such forces is a requisite for being able to use a physics engine to build a domain specific physical simulation.

1.3 Competences

The course relates to competences **P3.Game Engine**, and **P4.Development with Resource Constraints**.

1.4 Course learning objectives

The learning objectives of this course are the following, marked with the corresponding voice of Bloom's Taxonomy:

- **Build** a basic kinematic simulator with RK2 or RK4. The simulator tracks position, velocity, rotation, angular rotation, mass, intertia tensor of arbitrary convex polytopes. **Understand** the difference in precision, stability, and performance of different integration methods. **Understand** numerical drift for rotation matrices and quaternions, and how to compensate through normalization.
- **Build** a system for SAT computation of arbitrary convex polytopes. **Understand** the various kinds of contact manifold determination, and build at least one.
- **Understand** how to reduce the number of expensive collision tests with broad-phase methods such as bounding spheres, bins, and axis algined bounding boxes (AABBs). **Build** AABBs collision detection with the method of overlapping intervals. **Understand** how to reduce the number of SAT tests by exploiting the symmetry of a convex polytope.
- **Understand** the Projected Gauss-Seidel method. **Build** an impulse-based collision response system for all pairs of objects simultaneously.
- **Understand** forces from various domains of physics: gravity, friction, springs, bullets, and cars. **Build** at least one of those into the simulation.

2 Course structure

2.1 Number of hours

2 hours per week, contact time, and approximately 92 hours total study time.

2.2 Attendance policy

Attendance is not mandatory but students may miss valuable information if they do not attend the classes.

2.3 Teaching method

Traditional frontal lectures.

3 Assessment & deadlines

The course has a series of assessments: *(i)* a series of pieces that end up becoming a physics engine, each divided in a *group assignment of coding* and an *individual assignment of writing a report.*

All assessments must qualify for a full pass (6+) in order for the course to pass.

3.1 Assignment: building a physics engine

This assignment requires groups **of up to four** students to write a C++ or C# physics engine. This assignment is not graded directly, but only indirectly thorugh of assignment 2.

The grading criteria will be:

- performance 30%
- general purpose structure 30%
- credibility/correctness of the physical simulation 30%
- quality of the description document 10%

The partial assignments are:

- Build a basic kinematic simulator with RK2 or RK4 (20%)
- SAT/contact manifold computation (at least for OBBs, better for arbitrary meshes) (20%)
- Collision culling with bounding spheres, AABBs, and bins (20%)
- Collision response (20%)
- Forces for domain-specific scenarios (20%)

The partial assignments are due, alternatively:

- the end of the week after presentation in class (bonus $\times 1.1$), printed, with CD, in the lecturer's pigeon-hole
- at the end of the course, all together (no bonus); the deadline in this case is *Friday of the exam week, at midnight*

4 Materials

4.1 Literature

The course will be based on:

- The book *Game Physics - Second Edition*, by David Eberly
- The book *Physics for game programmers*, by Grant Palmer
- The paper *Iterative Dynamics with Temporal Coherence*, by Erin Catto
- The tutorial *Car physics for games*, by Marco Monster
- The Siggraph '97 course notes *An Introduction to Physically Based Modeling: Rigid Body Simulation I - Unconstrained Rigid Body Dynamics* by David Baraff

4.2 Software packages

The course will make use of Visual Studio 2010 or newer, with any graphics library associated such as DirectX, OpenGL, XNA, MonoGame, etc.

5 Lecture topics

The lectures will cover the following topics, approximately one per lecture:

- **Topic 1 - basic concepts from physics**: translational and rotational Newtonian physics, numerical integration, equations of motion for a system of bodies
- **Topic 2 - narrow phase of collision detection**: separating axis, collision response
- **Topic 3 - broad phase of collision detection**: axis aligned bounding boxes, bounding spheres, etc.
- **Topic 4 - simultaneous resolution of multiple constraints**: constraints as a system of equations, the Gauss-Seidel method
- **Topic 5 - force computation**: ballistic forces (Magnusson, friction, gravity), car forces, plane forces, etc.
- **Optional topic 1 - preprocessing of models for collision detection** BSP for faster collision detection
- **Optional topic 2 - preprocessing of generic models** calculating the *inertia tensor* of arbitrary polytopes

Topic 5, and the optional topics may not be entirely included in the lectures. Precedence will be given to correct assimilation of the previous topics, and the final schedule will also be dependent on the students response to the topics.

6 Conclusions

In this document we described the *Physics* course. The course focuses on understanding basic physics, and learning how to build a basic physics engine for games.

At the end of the course the students will be able to understand the structure of a physics engine, and will know how to build their own.