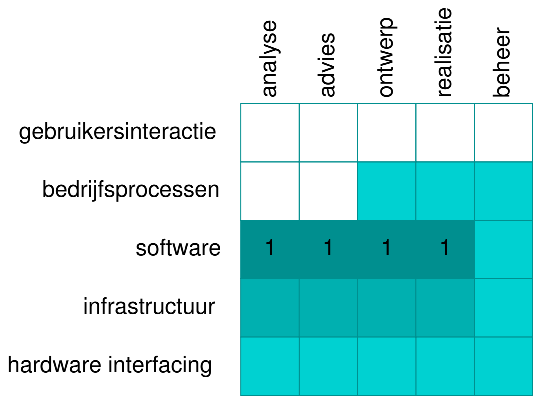
HOGESCHOOL ROTTERDAM / CMI

Numerical approximation for physics simulations

TINWIS01-8

ECTS:	3 ects
Module responsible:	Gerard van Kruining, Giuseppe Maggiore

Module description

Module name:	Numerical approximation for physics simulations
Module code:	TINWIS01-8
Number of ECTS and number of individual study hours:	This module gives 3 ectsECTS, which corresponds to 84 hours. • 8 × 120 minutes frontal lecture • 8 × 120 minutes practicum • 12 × 120 minutes individual study
Examination:	Practical assignments
Course structure:	Lectures and practicums
Required knowledge:	All programming course, linear algebra.
Learning tools:	• Book: Game Physics, author: David Eberly • Book: Friendly F# (Fun with game physics), authors: Giuseppe Maggiore, Marijn Tamis, Giulia Costantini • Text editors: Emacs, Notepad++, Visual Studio, Xamarin Studio, etc.
Connects to competencies :	

	analyse	advies	ontwerp	realisatie	beheer
gebruikersinteractie					
bedrijfsprocessen					
software	1	1	1	1	
infrastructuur					
hardware interfacing					

Overall learning goal:	The student is able to describe, define, and then implement numerical approximation techniques to physics simulations.
Detailed learning goals:	• The student is able to distinguish the aspect of kinematics: linear and rotational motion, and associated forces (learning goal *analysis*); • The student is able to give advice over the design and realisation of a kinematics simulation (learning goal *advice*); • The student is able to design the structure and architecture of a kinematics simulation (learning goal *design*); • The student is able to realise a working kinematics simulation (learning goal *realisation*); • The student is able to communicate in correct Dutch or English, using the correct jargon, about physics simulations and kinematics, etc. (learning goal *communication*).
Module responsible:	Gerard van Kruining, Giuseppe Maggiore
Date:	29 april 2015

1 Algemene omschrijving

The overall goal of the course is to provide a detailed answer to the questions:

- what are the kinematics equations?
- how does a physical simulation work?

In this section we discuss further the full breadth of what the course covers, plus the desired level of skills achieved by the students.

1.1 Introduction

Simulation of the physical world is one of the most powerful fields of application of modern programming disciplines. Thanks to such simulations it is possible to set up virtual experiments, build virtual reality worlds, and many more modern complex applications.

Since analytical solutions are very rarely available, building such simulations requires the use of approximation techniques, which themselves are also applied to many other fields such as computer vision, artificial intelligence, and robotics.

The goal of the course is to provide a definition of the physical laws of kinematics, and of numerical approximations of complex dynamics. Moreover, we shall learn how to translate this knowledge into a working physical simulator.

1.2 Relationship with other teaching units

This module builds over all modules of programming, and is also strongly connected with previous knowledge about mathematics, and linear algebra.

1.3 Learning tools

Obligatory:

- Presentations and sources presented during lectures (found on N@tschool);
- Assignments to work on during practicums (found on N@tschool);
- Text editors: Emacs, Notepad++, Visual Studio, Xamarin Studio, etc.

Facultative:

- Book: Game Physics, author: David Eberly
- Book: Friendly F# (Fun with game physics), authors: Giuseppe Maggiore, Marijn Tamis, Giulia Costantini

2 Content

Structure of lectures The lectures are an adaptation of traditional frontal lectures. In order to improve attention and retention, the following interactive elements are also used: *i*) questions and quizzes to the class, followed by discussion; *ii*) short group assignments, followed by discussion.

List of topics The following is a comprehensive and detailed list of the program of the course:

1. Linear motion
2. Rotational motion
3. Linear acceleration
4. Rotational acceleration
5. Inertia tensor
6. Euler integration
7. Runge-Kutta integration
8. Quaternions
9. Time-derivative of matrices
10. Time-derivative of quaternions

3 Testing and evaluation

In this section we discuss the testing procedure of this course, and the grading criteria.

3.1 Overall description

This module is tested with a series of practical assignments. These assignments can be found on N@tschool.

Foreword and notes:

- The practical assignments determine the final grade.
- The practical assignments are made up of elements of an interpreter or a compiler which is either incomplete or wrongly built. The students task is that of extending or fixing such elements.
- The practical assignments can be made in pairs.
- The practical assignments must contain extensive, individually written documentation.

This manner of examination is chosen for the following reasons:

- By reading existing sources students must read and reason about code (learning goals *analysis* and *advice*).
- By correcting or extending the sources students must write code (learning goals *design* and *realisation*).
- By writing documentation students must communicate about their code (learning goal *communication*).

The grade of each practicum assignment is determined by:

- The correctness of the underlying physical and approximation rules (60%).
- Completeness and clarity of the documentation (40%).

3.2 Assignments

3.2.0.1 Assignment 1 - linear motion Students must define an Euler-integrated simulation over position, velocity (or linear momentum) and force.

3.2.0.2 Assignment 2 - rotational motion Students must define an Euler-integrated simulation over rotation, angular velocity (or angular momentum) and torque.

3.2.0.3 Assignment 2B - Gram-Schmidt ortho-normalization Students must define a Gram-Schmidt ortho-normalization.

3.2.0.4 Assignment 2C - quaternions Students must replace rotation matrices with quaternions.

3.2.0.5 Assignment 2D - Runge-Kutta integration Students must replace the Euler integration with a Runge-Kutta integration of order 4.

3.3 Grades

Assignments 1 and 2 are obligatory. With these assignments the maximum grade possible is a seven. The other assignments are all optional and give each one additional point.

Assignments:	Value in grades
1, 2	7
2B	1
2C	1
2D	1

3.4 Deadlines

The assignments must be handed in, printed on paper, before the end of the last lecture on week 8 of the period. *Herkansing* can be done by handing in the assignments before the end of week 1 of the following period (that would be right after the summer holiday).

3.5 Feedback

It is possible to discuss the assignments (and their evaluation) during the practicum lectures, or during week 10 of the period (the same holds for the *herkansing*, but in reference to the following period).

Appendix 1: Examination matrix

Learning goal	Dublin descriptors	Assignments
The student is able to distinguish the aspect of kinematics: linear and rotational motion, and associated forces	1, 5	1, 2
The student is able to give advice over the design and realisation of a kinematics simulation	1, 3, 5	Documentation of all
The student is able to design the structure and architecture of a kinematics simulation	1, 3, 4	1, 2
The student is able to realise a working kinematics simulation	2	All
The student is able to communicate in correct Dutch or English, using the correct jargon, about physics simulations and kinematics, etc.	2	Documentation of all

Dublin-descriptoren:

1. Knowledge and insight
2. Application of knowledge and insight
3. Making judgments
4. Communication
5. Learning skills