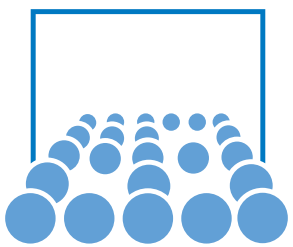
PSE Game Physics

Introductory session

Atanas Atanasov, Philipp Neumann, Martin Schreiber

6. Mai 2011



Outline

Mathematics and Physics

- Physics and Mathematical Modeling
- Mathematical Description: Ordinary Differential Equations
- Time Discretisation: Finite Difference Schemes
- Properties of Finite Difference Schemes: Consistency, Stability and Convergence

C++

- Live demonstration in eclipse
- Templates

Outline

Mathematics and Physics
- Physics and Mathematical Modeling
- Mathematical Description: Ordinary Differential Equations
- Time Discretisation: Finite Difference Schemes
- Properties of Finite Difference Schemes: Consistency, Stability and Convergence

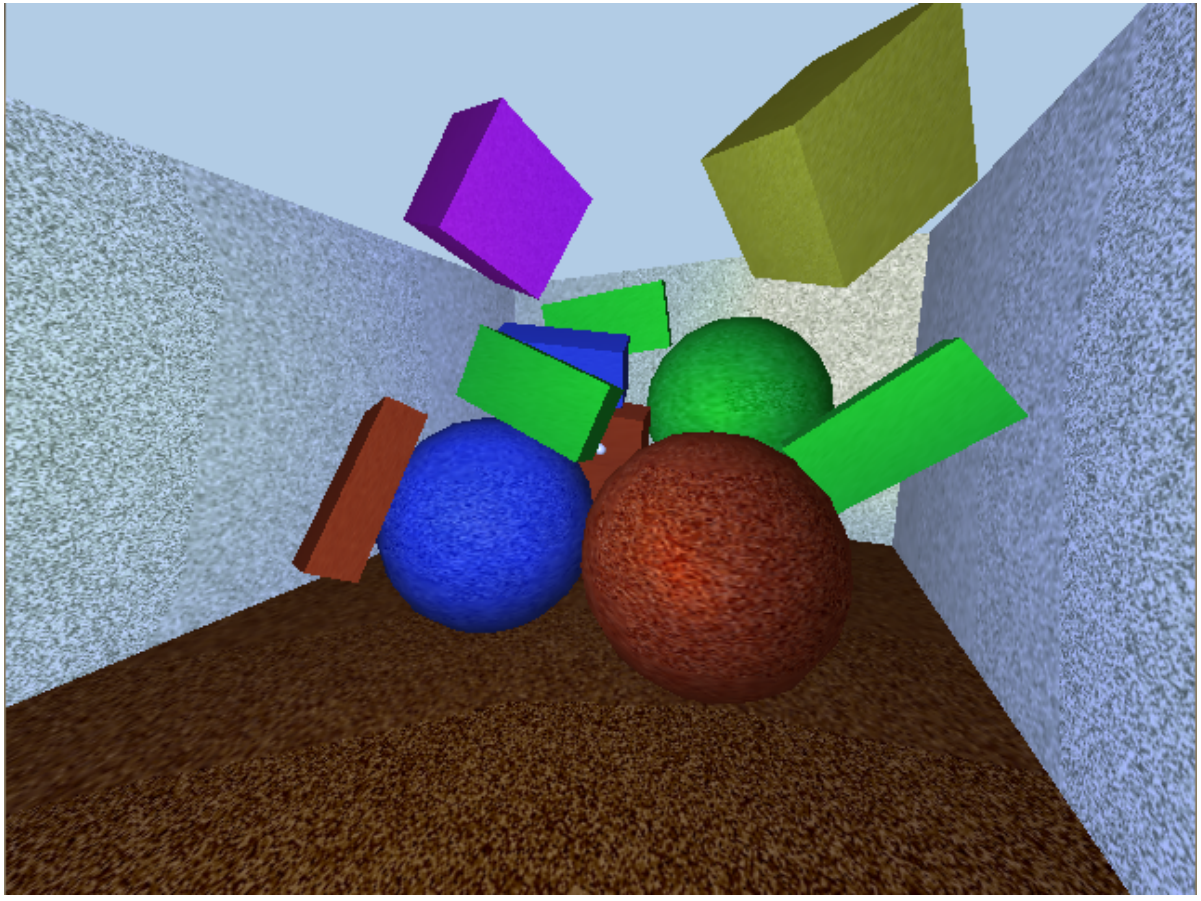
C++
- Live demonstration in eclipse
- Templates

Physical systems - Our model

Physics engine - What do we talk about here?

Physical systems - Our model

Physics engine - What do we talk about here?



Physical systems - Our model

Physics engine - What do we talk about here?

- Starting point: Physical system with certain properties
- Determine laws which hold for these properties

Our physical system:

- Set of rigid bodies n, $n \in \{1, ..., N\}$, $N \in \mathbb{N}$
- Each body is described by
  - its position $\vec{x} \in \mathbb{R}^D$
  - its velocity $\vec{v} \in \mathbb{R}^D$
  - its mass $m \in \mathbb{R}$
  - its geometrical form (we first only consider spheres!)

Property relations (1)

What do we know about the relations of these properties?

- Position $\vec{x}(t)$ changes according to the body's velocity $\vec{v}(t)$ over time t
- Velocity of a rigid body changes due to
  - collisions with other rigid bodies (more information in the next lab)
  - external forces acting on the bodies, e.g. gravity

Property relations (2)

How do we put our knowledge about the properties into a mathematical description?

Velocity $\Leftrightarrow$ **change of position over time** $\Leftrightarrow$ $\frac{d\vec{x}}{dt}$

Forces $\Leftrightarrow$ **change in velocity over time** $\Leftrightarrow$ $\frac{d\vec{v}}{dt}$

Conservation laws - The equations of motion

$$\begin{aligned} \frac{d\vec{x}}{dt} &= \vec{v} \\ \frac{d\vec{v}}{dt} &= \frac{1}{m}\left(\vec{F}_{coll} + \vec{F}_{ext}\right) \end{aligned} \tag{1}$$

with external forces $\vec{F}_{ext}$, collisional forces $\vec{F}_{coll}$ and **initial conditions** $\vec{x}(t=0) := \vec{x}_0$, $\vec{v}(t=0) := v_0$.

Remark: Within the lab, we use the equations of motion as **mathematical model** that we try to solve. The difference between the “real-world solution” and the mathematical solution of our model is called **model error**.

Mathematical description - Basics

How do we solve Eq.(1)?

$\rightarrow$ A little bit of formalism...

- *Definition:*
  $\frac{df}{dt} := \lim_{dt \to 0} \frac{f(t+dt)-f(t)}{dt}$ is called **(time) derivative of a function** $f : \mathbb{R} \rightarrow \mathbb{R}$.
  Analogously, the derivative for a vector-type function can be defined as the derivatives of each vector component.
- *Definition:*
  An equation of the type $F(t, \frac{df}{dt}, \frac{d^2 f}{dt^2}, ..., \frac{d^l f}{dt^l}) = 0$ is called **ordinary differential equation** of order l.
- *Definition:*
  A system of equations of the form

$$F_k(t, \frac{df_1}{dt}, ..., \frac{df_N}{dt}, ..., \frac{d^l f_1}{dt^l}, ..., \frac{d^l f_N}{dt^l}) = 0, \quad k = 1, ..., K$$

  with functions $f_1(t), ..., f_N(t)$ is called a **system of ordinary differential equations**.

Let's consider our mathematical model...

$$\begin{aligned} \frac{d\vec{x}}{dt} &= \vec{v} \\ \frac{d\vec{v}}{dt} &= \frac{1}{m}\left(\vec{F}_{coll} + \vec{F}_{ext}\right) \end{aligned} \tag{1}$$

How can we describe our Eq.(1) in terms of our definitions?

Let's consider our mathematical model...

Answer: We need to deal with a system of ordinary differential equations of order 1, consisting of $2 \cdot D \cdot N$ equations.

Which general approaches are there to solve the system?

- **Analytically** $\rightarrow$ Not always possible (many rigid bodies, complex geometries, complex forces); available for simple problems
- **Numerically** $\rightarrow$ Can handle many rigid bodies, complex geometries, complex forces; in most cases: **cannot** solve the problem exactly, but up to a certain accuracy

Towards the numerical solution: The three steps of discretisation

- Step 1:
  Instead of solving the problem on a certain time interval $[0, t_{end}]$, we try to solve it only at **discrete** points in time
  $t_0 := 0 < t_1 < t_2 < ... < t_T := t_{end}$.
  In our case, $t_i := t_0 + i \cdot \Delta t$, $i = 0, ..., T$, $\Delta t := \frac{t_T - t_0}{T}$ **timestep**
- Step 2:
  We try to find **discrete** approximate solutions $\vec{x}^h(t), \vec{v}^h(t)$ that are "close" to the exact solutions $\vec{x}(t), \vec{v}(t)$.
- Step 3:
  We try to find a **discrete** form for our differential operators from Eq.(1).

Remark: The error between the exact solution of our mathematical model and the solution of our discretised problem is called **discretisation error**.

Towards the numerical solution: The three steps of discretisation

How can we approximate the differential expressions from Eq.(1)?

Let's consider the position vector $\vec{x}(t)$ and define $x^d(t)$:= d-th component of vector $\vec{x}(t)$.

Assume that our position evolution is sufficiently "smooth", i.e. as often differentiable as we need it to be :-)

Towards the numerical solution: Finite differences (step 3)

Then, use Taylor expansion for $x^d(t + \Delta t)$:

$$\begin{aligned} x^d(t+\Delta t) &= x^d(t) + \frac{dx^d}{dt}\Delta t + O(\Delta t^2) \\ &\Leftrightarrow \\ \frac{dx^d}{dt} &= \frac{x^d(t+\Delta t) - x^d(t)}{\Delta t} + O(\Delta t) \end{aligned} \tag{2}$$

So, we can approximate $x^d(t + \Delta t)$ from $x^d(t)$ as follows:

$$\boxed{\begin{array}{c} x^d(t+\Delta t) = x^d(t) + \Delta t \cdot v^d(t) \\ \\ \text{Explicit Euler approximation} \end{array}} \tag{3}$$

Remark: Expressions of the form $\frac{x^d(t+\Delta t) - x^d(t)}{\Delta t}$ are called **finite differences**.

Towards the numerical solution: Finite difference schemes

Approximation	Scheme	Error
$\frac{x^d(t+\Delta t)-x^d(t)}{\Delta t} = v^d(t)$	Explicit Euler	$O(\Delta t)$
$\frac{x^d(t+\Delta t)-x^d(t)}{\Delta t} = v^d(t+\Delta t)$	Implicit Euler	$O(\Delta t)$
$\frac{x^d(t+\Delta t)-x^d(t-\Delta t)}{2\Delta t} = v^d(t)$	Central difference	$O(\Delta t^2)$

Can you think of advantages/ disadvantages of these schemes?

Questions revisited

How "close" are our approximate solutions to the continuous solution of the mathematical model?

How big should we choose the timestep?

Does our discrete solution approach the continuous solution when we decrease the timestep?

What happens to **numerical errors***, that is errors introduced by e.g. the limited computational precision of our computers?*

Consistency

Definition:
A numerical method (for a differential equation) is called **consistent**, if the solution $x_{\Delta t}(t)$ according to our solving algorithm approaches the exact solution $x(t)$ of the mathematical model for $\Delta t \to 0$ in a single timestep, i.e.

$$\lim_{\Delta t \to 0} x_{\Delta t}(t) = x(t) \tag{4}$$

Consistency: Error estimate

The consistency of our Euler scheme is...

Consistency: Error estimate

The consistency of our Euler scheme is...

wait for it...

Consistency: Error estimate

The consistency of our Euler scheme is...

wait for it...

trivial :-), as long as we stick to $x(t) \in C^2$:

$$\|x_{\Delta t}(t + \Delta t) - x(t + \Delta t)\| \leq \frac{C}{2} \max_t \left\| \frac{d^2 x(t)}{dt^2} \right\| dt^2$$

Stability

Definition:
A numerical method is called **stable**, if numerical errors do not increase in subsequent timesteps.

The stability of our scheme is...

Stability

Definition:
A numerical method is called **stable**, if numerical errors do not increase in subsequent timesteps.

The stability of our scheme is...

wait for it...

Stability

Definition:
A numerical method is called **stable**, if numerical errors do not increase in subsequent timesteps.

The stability of our scheme is...

wait for it...

...here it comes...

Stability

Definition:
A numerical method is called **stable**, if numerical errors do not increase in subsequent timesteps.

The stability of our scheme is...

wait for it...

...here it comes...
not that trivial at all :-(

Stability: Remarks

- Purely implicit discretisation schemes typically are unconditionally stable

- Explicit schemes are only stable in a certain discretisation range. For time discretisation schemes, the timestep Δt often needs to be limited:

$$\Delta t < C(\text{Problem})$$

Convergence

Definition:
A numerical method is called **convergent**, if our numerical method yields an approximate (global) solution of the continuous problem.

- Consistency, Stability and Convergence are closely related
- For linear initial value problems, it holds:

$$\boxed{\text{Consistency} + \text{Stability} \Leftrightarrow \text{Convergence}}$$

Questions re-revisited

How "close" are our approximate solutions to the continuous solution of the mathematical model?

How big should we choose the timestep?

Does our discrete solution approach the continuous solution when we decrease the timestep?

What happens to **numerical errors***, that is errors introduced by e.g. the limited computational precision of our computers?*

Talking about errors: Which errors do you remember?

Outline

Mathematics and Physics
- Physics and Mathematical Modeling
- Mathematical Description: Ordinary Differential Equations
- Time Discretisation: Finite Difference Schemes
- Properties of Finite Difference Schemes: Consistency, Stability and Convergence

C++
- Live demonstration in eclipse
- Templates

C++ introduction

Live demonstration in eclipse

Templates

Generics in Java

- Specialized during runtime
- ...

Templates in C++

- Specialized during compilation time
- Advantage: Compiler can run optimizations
- Use inlining as much as possible
- ...