Visual Interactive Simulation D, 5p.
TDBD22, Spring 2005

Physics Engines

Kenneth Bodin
VRlab, HPC2N, Computing Science, Physics ☺
Umeå University, Sweden

holmlund@hpc2n.umu.se

"For every complex problem there is
a solution that is simple, neat and wrong!"

H.L. Mencken (1880-1956)

KISS ("keep it simple stupid")

Ockham's razor:

"Plurality is not to be assumed without necessity."

"What can be done with fewer [assumptions]
is done in vain with more."

William of Ockham (1285-1349)

Why a physics engine?

The same advantages as always with modular and structured software.

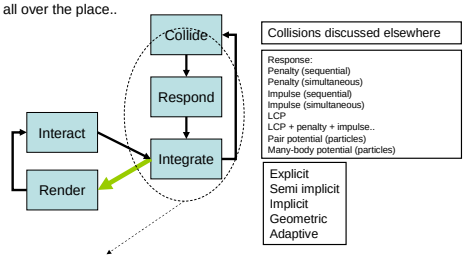
- Reuse
- Evolving
- Generality
- Scalability
- Portability
- Productivity
- Mature API's, file formats
- Documentation
- Quality
- Stability
- Precision
- Optimization (memory, speed)

Most programmers, designers and developers are not very skilled in computational physics or numerical methods and redo the same mistakes over and over...!

Numerical physics is more difficult than the average state machine type of programming...

Architecture of a Physics Engine

API:s all over the place..



What you focus on in this course!

Example: CMLabs, Vortex architecture

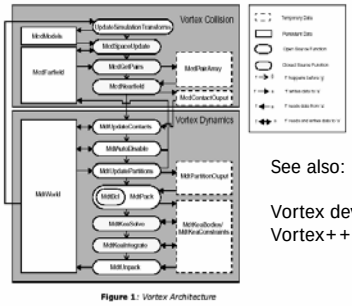


Figure 1: *Vortex Architecture*

See also:

Vortex developers Guide
Vortex++ Vx Guide

Typical design strategy

- Set up the world (gravity and framework to create objects).
- Set up a ground plane (or ground mesh).
- Set up a rigid body system (m, pos/orient, vel/ang vel, inertia).
- Set up collision bodies (primitive, mesh, ...).
- Set up a simulation bodies by combining rigid bodies and collision bodies
- Set up graphical bodies (may be identical to the collision body).
- Make sure the graphical body fetches it's transformation matrix from the simulation body, whenever it needs it.
- Set up joints and constraints.
- Run simulation.

Other physics engines

- www.cm-labs.com
- www.havok.com
- www.ageia.com (also Novodex)
- www.physicstools.org (Novodex etc.)
- ode.org (open source)
- www.meqon.com (Swedish)
- Tocamac
- Newton
- Renderware Physics (aka Mathengine)
- Etc.

Simulation management

Simulation can have different types of loop structure:

Serial loop:

```
while(true) {
  simulation_step(dt)
  render()
  assure real-time sync
}
```

Asynchronous loop (thread/fork):

```
while(true){
  read_simulation()
  render()
}
```

↔

```
while(true){
  simulation_step(dt)
  assure real-time sync
}
```

Ultimately a load balancer that manages real-time sync vs frame-rate etc.

Things to think of in simulation loop

- Design so that you can choose either fixed time step or time step adaptive to frame rate.
- The former requires either many frames per timestep, or many timesteps per frame; or an asynchronous loop.
- If you use an asynchronous loop: make it right! Avoid race conditions and wait states.
- (another possibility is to use signals and time slicing, e.g. the way it is done in Renderware, but this forces you to use algorithms that can handle this!)
- It is useful to have functionality to compare simulation time with clock time (i.e. visualization of time bars). Use microclock or even real-time features of OS if available. Don't get too confused by spatial and temporal scales. A 10x10x10 m3 box moves in a different way than a 10x10x10 cm3 box, when gravity and friction is present!

Data structures

- Rigid body: Position and orientation (R, Q)
- Use of world coordinates recommened in course...
- Memory references
- Memory copies
- Shared memory
- Sockets (client/server, ...)
- Hierarchies in a scenegraph not always so useful in physics since systems are cyclic and change alot! Keep things rather flat and atomistic...
- Saving and loading states
- Debugging

- One object or many?
- E.g. for many particles not efficient to toss many objects around.

- Design and plan data structures and modularity of program/libraries for the entire course!
- Consider using a file format or interactive parameters so that you can prototype, test and vary.

Graphics

- We recommend using www.openscenegraph.org
- Or - your favourite 3D graphics software, including your own stuff
- Don't spend too much time on 3D graphics programming in this course
- But, we do appreciate good looking models and scenes!
- Use 3D graphics as a *library* and don't hand in this code in lab projects!