

vrsode {

...connect physics to vrs objects...



table of contents

- features
 - about ode
 - usage of vrsode
 - basic components
 - physics manager
 - collision body, physics body
 - surface, collision shape, joint
 - a word of warning
 - live coding
 - final words, link list
-
-

features

- encapsulate VRS::SceneThing with physics
 - wraps ode's C api with C++ classes
 - physics world step (main loop) in a separated thread
 - support for several collision geometries, eg.
 - cylinder
 - sphere (automatic generation)
 - box (automatic generation)
 - trimesh (mathematically complex)
 - collision callbacks (into colliding objects)
 - mechanism for visualization of debug geometries
 - includes a simple sample car
-
-

about ode

- vrsode was tested with version 0.7 and 0.8
 - build in release mode (else: mass problem!)
 - enable double precision (way better stability!)
 - trimesh may drive you crazy ;-)
 - the stability of your simulation heavily depends on the resolution of your trimesh data
 - badly chosen trimesh resolutions may even cause the application to crash (in dSolveLCP())
 - take a look at the ode wiki (user guide is outdated)
-
-

usage of vrsode

- has it's own namespace called vrsode
 - depends on
 - vrs, ode, random_utils, sdl
 - build via makefiles
 - tested on linux (gentoo, debian) and os x
 - windows build should not be a problem ;-)
 - doxygen api documentation online available
 - check out <http://rtti.de> for download and api docs
-
-

basic components

- PhysicsManager
 - singleton, access via get()
 - central unit stepping world
 - synchronizes vrs and ode
 - manages lists of objects
 - provides the threaded main loop
 - use stepSize() and stepTime() for speed tweaks
 - set global world parameters
-
-

basic components

- **CollisionBody**
 - object encapsulating `VRS::SceneThing`
 - enables collision detection for this object
 - used for static objects
 - configurable via `surface` and `collisionshape`
 - possible to set shape to “no shape”
 - **PhysicsBody**
 - inherits `CollisionBody`
 - adds dynamic facilities to the object
 - used for movable objects
 - can be interconnected via joints
-
-

basic components

- Surface
 - describes bouncieness, friction, ... for collision bodies
 - CollisionShape
 - describes shape for collision bodies
 - eg. box, sphere, trimesh, cylinder
 - Joint
 - connects two physics bodies, or
 - connects one physics body with static world
 - configurable motors, stops
 - eg. ball, hinge, hinge2, slider
-
-

a word of warning

- for now, only used with RandomRacer
- no unit tests
- one big physics lock
 - think about what you are doing if changing physics parameter while simulation, if you should lock or not
- does not eliminate the need of understanding ode
- 3700+ lines of code
- for now, no further development planned
- may not bug free

“Gegen eine stabile API programmieren kann jeder...” - Prof. R. Hirschfeld - 2007

live coding

- lets see what we can do with some lines of code
 - random racer 1.1.0 svn snapshot as base
 - minimal vrs sample as base
 - create vrs objects
 - create vrsode objects and encapsulate vrs objects
 - collision body: a ground plane
 - physics body: a box
 - apply forces to ball on key press
 - toggle debug geometries on key press
 - attach a ball joint to the box
-
-

finally

- MacOS X patch for VRS
- RandomRacer source may be interesting
- links
 - <http://rtti.de/index.php?id=28>
 - http://files.rtti.de/random_racer_doxygen/namespacevrsode.html
 - <http://files.rtti.de/random-racer-1.1.0-r326.tar.bz2>
 - <http://ode.org>
 - <http://vrs3d.org>
 - <http://libsdl.org>
 - http://opende.sourceforge.net/wiki/index.php/Main_Page

questions anyone?

} /* vrsode */

