MAT 594G / CS 280: Advanced Computer Graphics	Nov. 8, 2012
Fall Quarter, 2012	Assignment 3

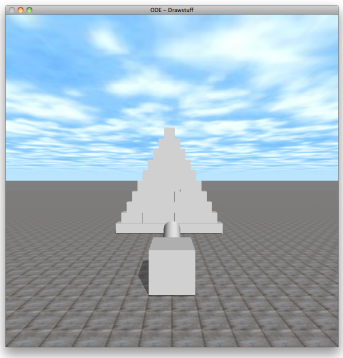
Due by **midnight, the evening of November 15, 2012** via GauchoSpace.

1 Angry Birds 3D

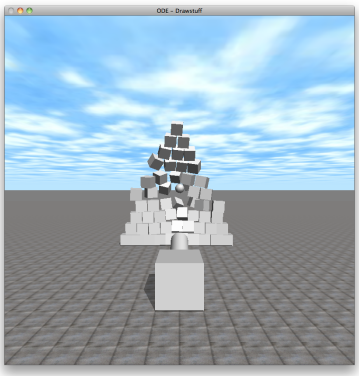
In this assignment, you will gain experience using *Open Dynamics Engine* (ODE), a freely available rigid body dynamics package. The ODE Manual and the demo source code that accompanies the starter code will be your primary references for this assignment.

1.1 Basic Box Stacking

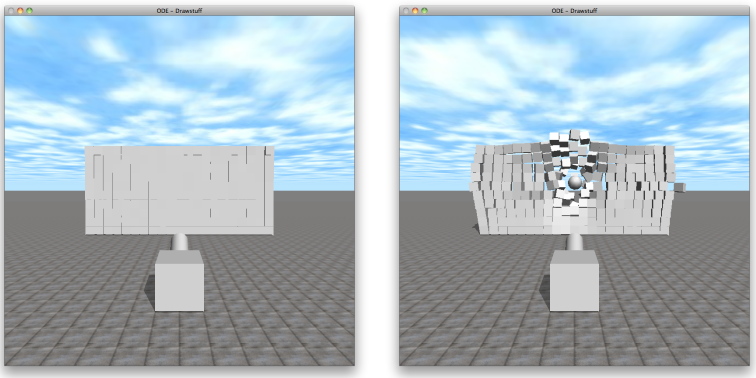
After compiling the starter code (Section 3), you will see a pyramid of blocks and a cannon:



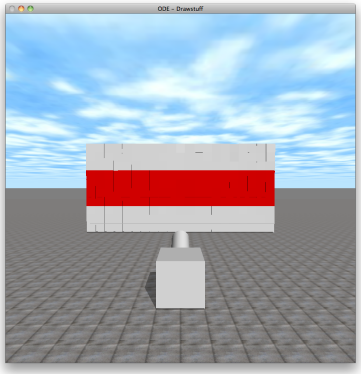
You can reposition the cannon barrel by using `j`, `k`, `l` and `i` as arrow keys, and by hitting the space bar, you can fire a cannon ball to knock over the blocks:



To begin familiarizing yourself with ODE, modify the code to instead draw and simulate a wall of boxes:



Tag some of the boxes as "burstable," and color them red:



ODE comes with a thin `drawstuff` wrapper around OpenGL. When setting an object's color, call `dsSetColor` instead of `glColor4f`.

1.2 Compute Accelerations

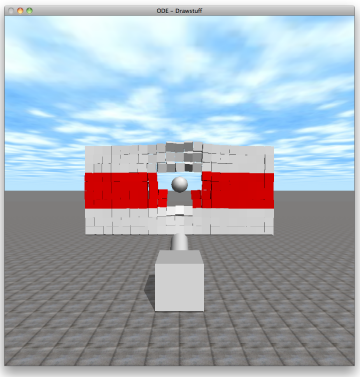
Record a box as 'burst' when it undergoes a large acceleration. ODE only gives you direct access to the position and velocity of its rigid bodies, so you will need to compute the accelerations yourself. Each rigid body i at time t has a velocity v_t^i. We want to know its acceleration, $a_t^i = \frac{\partial v_t^i}{\partial t}$.

In order to estimate this derivative, use *finite differences*. Store each rigid body's velocity from the previous timestep, v_{t-1}^i, and estimate the derivative using a difference equation,

$$a_t^i = \frac{\partial v_t^i}{\partial t} \approx \frac{v_t^i - v_{t-1}^i}{\Delta t}$$

where Δt is the simulation timestep size in ODE. This is the `dt` value passed to `dWorldQuickStep`.

If the acceleration of the translational velocity exceeds a fixed threshold (I found the value 50 worked well), register the box as 'burst' and have it disappear from the scene:

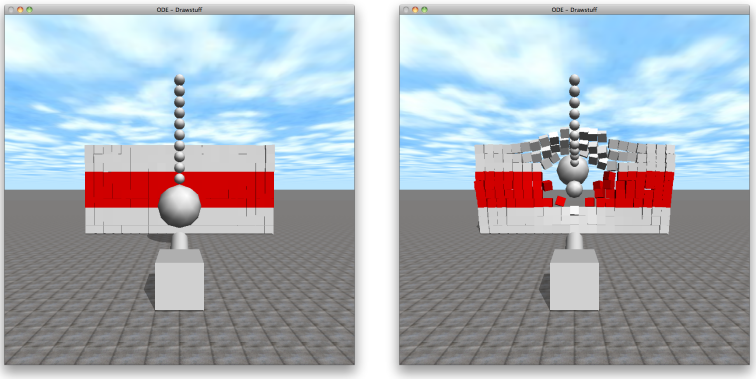


Note in the above image that the center boxes that the ball directly hit have not been knocked over, but instead have disappeared. You can optionally also take into account angular acceleration when bursting your boxes

The functions `dBodyDestroy` or `dBodyDisable` will *not* give you the desired disappearing behavior. You will have to decide how best to achieve this behavior within the constraints of the ODE API.

1.3 Build a Wrecking Ball

Using the articulated body functions of ODE, build a wrecking ball:

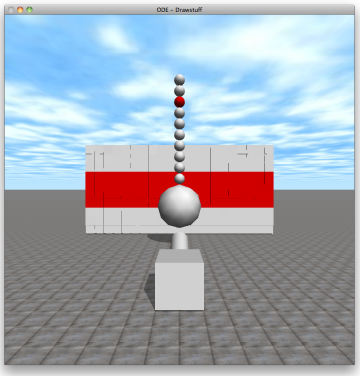


By shooting the ball, you should be able to knock down the wall more effectively. You will need to first build a stationary anchor point at the top of the chain. Use the `dBodySetKinematic` function to achieve this effect. For the remainder of the chain, use

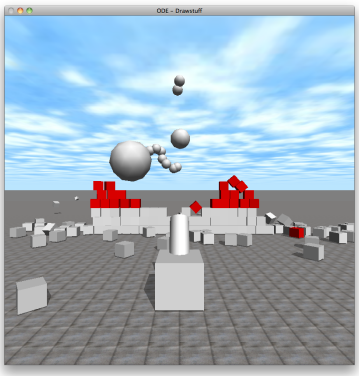
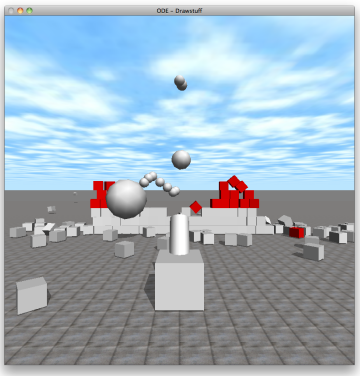
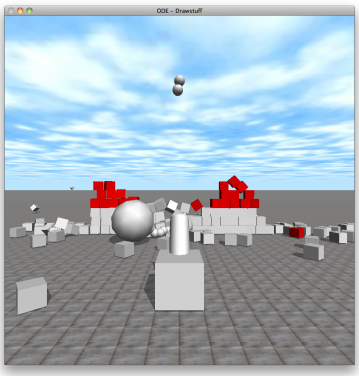
the various `dJoint*` functions available in ODE. You may find the `demo_chain1.c` and `demo_chain2.cpp` examples provided in the starter code to be helpful.

1.4 Shoot the Chain

Finally, allow the wrecking ball chain to break at a single weak link. Color the weak link red (the third sphere down):



If the cannonball comes into contact with the weak link, delete a joint so that the chain breaks:

In this case, do not detect if an acceleration threshold has been exceeded. Instead, explicitly detect if the cannonball and the weak are at all in contact. This can be accomplished within the narrow phase collision detection callback, `nearCallback`.

1.5 Make a Video

Create a video file that shows your implementation in action, and post it to a site such as YouTube or Vimeo. There are many screen capture software packages available, such as

Camtasia, Fraps and glc. Get one of these programs up and running, and record a video that shows all of the required features working.

Some of these programs, such as Fraps, insert a watermark if you do not buy the full version. It is **okay** if your final video contains a watermark. **Do not** buy any software for the sole purpose of this assignment.

2 Points Breakdown

Turn in your files via GauchoSpace in a ZIP file named `HW3-<UCSB Net ID>.zip`. Unless you made some very aggressive modifications, you should only need to turn in the **HW3.cpp** file. Include a `README.txt` that describes what platform you developed on, the link to you YouTube video, and any special instructions for building and running your code you think may be necessary.

1. Get a wall of boxes working. **(10 points)**
2. Tag boxes as burstable and color them red. **(10 points)**
3. Implement acceleration-based box bursting. **(20 points)**
4. Build a wrecking ball. **(20 points)**
5. Implement contact-based breaking of the wrecking ball chain. **(20 points)**
6. Turn in a link to a YouTube or Vimeo video showing everything working **(20 points)**

3 Compiling the code

Only Mac and Linux are currently supported for this assignment. You should be able to build the code by going to the top level directory, `ode-0.12` and executing:

```
./configure
make
```

If you then go to the directory `ode-0.12/ode/demo`, you can execute the starter demo by executing:

```
./HW3
```

The code that corresponds to that executable is `HW3.cpp`, and is in the same directory. You should only need to modify this file for the current assignment.

3.1 Mac Instructions

If you run into trouble when compiling on the Mac, try the following:

- Install MacPorts
- Execute `sudo port install libtool`
- Execute `sudo port install pkgconfig`
- From `ode-0.12`, execute `sh autogen.sh`
- Try `./configure` and `make` again.

If you are using Lion, we have found that calling a non-default `g++` compiler can cause problems. If you perform a `which g++` and it does not point to `/usr/bin/g++`, modify your path so that the `/usr/bin` version gets precedence.

3.2 Linux Instructions

For compiling on Linux, you should modify the Makefile in the following directory: `ode-0.12/ode/demo`. In that Makefile, you should add `-lglut` to the following two lines:

```
LDADD = $(top_builddir)/drawstuff/src/libdrawstuff.la \
        $(top_builddir)/ode/src/libode.la -lglut -lGLU -lGL
```

and

```
GL_LIBS = -lglut -lGLU -lGL
```

3.3 Common Issues

On both Mac and Linux, we have found that if any of the parent directory named contain a space, this can also cause problems. In this case, you should move the `ode-0.12` folder to a different location, or get rid of the spaces in the directories' names.

If you get permission errors during your compilation process, try using `sudo make` instead of `make`.

Thanks are in order for Sahar for figuring out all these compilation issues.