# CS 463 Take-Home Midterm (Dr. Lawlor)

YOUR NAME: ___Answer Key, Jr.___________________

Your finished electronic version of this exam document is due back by midnight 2015-03-06 as a PDF file. Please write clearly and professionally, using correct spelling and complete sentences, but keep things to the point. You are encouraged to use hyperlinks and embedded image files where appropriate. Everything you write should be your own work, in your own words.

This is an exam, but it is open web, open books, and open notes. In fact, you may use any non-living reference material—the spirit world is OK; but classmates, friends, or faculty are not OK, even if reached electronically.

—

**1.)** The attached corporation simulator tracks current net Income, outstanding Loans, operating Capital, and the emotion of Fear.

**1.a.)** The current extremely aggressive investment strategy (briefly, "make hay while the sun shines") produces record short-term profits via a highly leveraged investment bubble at t=3 years, which when interest rates rise, results in a crash and complete bankruptcy at t=3.7 years. Incorporate fear (corp.F) into the investment decisionmaking process and adjust the fear response parameters, or otherwise moderate the overinvestment bubble, allowing the corporation to stay profitable indefinitely.

There are many possible solutions here: don't take on debt (my personal preference, although it increases the chance of a hostile takeover), take on debt more slowly, etc.

**1.b.)** What differential equation is represented by "corp.C+=0.5*corp.I*lib.dt;"?

The time derivative of Capital is proportional to Income.

dC/dt = 0.5*I

**1.c.)** Why would a statement like "corp.L-=lib.dt;" violate the real physics of corporate finance?

Because debt is just decreasing, without any corresponding income or cost.

**1.d.)** Can you state some financial law analogous to Newton's Second Law?

"Money ain't come from nowhere!" (Or a slightly less flippant statement of the conservation of money.)

Point value: 30 points

—

**2.)** We've covered several methods of representing rotation. Recommend one rotation method for each of these cases, and give a representative few lines of example code.

**2.a.)** An ellipsoidal planet rotating around a vector 23.4 degrees from vertical, with a terrestrial XYZ orthogonal coordinate system (where (0,0,0) is the planet's center) used to place objects on the planet's surface.

If you need the vectors for a coordinate frame, you can represent rotation by explicitly adjusting those vectors.

```
planet.X+=planet.Y*dt*rotation_rate_radians_per_second;
planet.Y-=planet.X*dt*rotation_rate_radians_per_second;
// planet.Z never changes.  Don't forget to orthonormalize!
```

**2.b.)** One planet rotating at a fixed rate around the global Z axis

Euler angles.

```
rot.Z+=dt*rotation_rate_radians_per_second;
```

**2.c.)** Tracking the orientations of each asteroid in a belt with 10,000 total objects, on the GPU

One quaternion per asteroid stores orientation, and another stores the rotation rate (updated at collisions). Update them on the GPU using the quaternion derivative formula or the axis-angle formula.

Point value: 25 points

—

**3.)** An existing in-house modeling package uses a strange curved element surface representation called a “curtrangle” (supposedly short for “curved triangle”, but the author's name is also “Curt”). Each curtrangle has six nodes: three vertices at the corners like a normal triangle, and three extra vertices at the midpoint of each edge, which allows the edges to each be curved instead of straight.

**3.a.)** To show the model on a normal GPU, you need triangles. What are some ways you could generate triangles from curtrangles?

The obvious: ignore the “extra” three nodes, and flatten the curtrangles to triangles. Curt would not be proud.

Cut the curtrangle into 4 fixed triangles. This is probably what I would do.

Dynamically subdivide the curtrangle into triangles of fixed curvature. The hard part here is matching the tesselation between the edges.

**3.b.)** How would you recommend calculating a volume mesh for a curtangle model? (When you asked your boss if you could spend the next 6 months investigating the mathematics of curtetrahedra, she said “no”.)

1.) Convert to triangles via any scheme above.

2.) TetMesh

3.) Profit!

Point value: 20 points

—

**4.)** A dirt simulator should support millions of grains of sand, but the code performs terribly:

```
std::vector<sand_grain> sand=read_sand("dirt.xyz");
for (double simtime=0;simtime<sim_end;simtime+=dt) {
      for (int i=0;i<sand.size();i++)
            sand[i].net_force=vec3(0,0,0);
      for (int i=0;i<sand.size();i++)
      for (int j=i+1;j<sand.size();i++)
            if (sand[i].contacts(sand[j]))
                  add_contact_forces(sand[i],sand[j]);
      for (int i=0;i<sand.size();i++)
            sand[i].step(dt);
}
```

**4.a.)** What is the performance problem?

There are O(n^2) grain-to-grain contact calculations.

**4.b.)** How might you get better performance?

Use a collision acceleration structure, like voxels or a tree, to only calculate contact for nearby grains.

**4.c.)** Preparing models by manually entering XYZ center coordinates for each grain of sand has already driven several students insane. How would you recommend preparing a model for a simulated pile of dirt, prior to running a simulation to measure the dirt's compaction strength?

Randomly choose locations to place grains of sand, but do not grains outside the pile, or in contact with an existing grain. Be sure to let the pile settle (using this simulator or a dedicated setup simulator) before running the real simulation.

Point value: 25 points