Homework 1 : The Jello Cube (Continuous Simulation Assignment)

Dr. Joseph T. Kider

IDS6938: Simulation Techniques - *Modeling And Simulation*

Institute for Simulation and Training

University of Central Florida

I. Introduction and Background

A Continuous System Simulation describes systematically and methodically how mathematical models of dynamic systems, usually described by sets of either ordinary or partial differential equations possibly coupled with algebraic equations, can be simulated on a digital computer. The purpose of this assignment is to get to know 3D mass-spring particle systems, numeric integration, error analysis, and collision detection and response. **The Assignment is due on Monday, February 27, 2017 at 11:59 P.M.**

Major parts for the Assignment:
You can think of the assignment being broken up into four parts:

- Numerical Analysis of Integration Functions
- Programming a Jello Cube
- Written questions
- Composing a final report

Project Report + (Video)
The goal of this assignment is to become familiar with the concepts in the first third of the class. You will be expected to compose a *final report* which demonstrates your understanding on the material in each section of the assignment. Be visual! - Pictures say a thousand words so you do not have to. Show off your different configurations and really explore the assignment.

Good Luck!

II. Assignment

Part 0: Getting Started. Read the assignment. Sync your fork with the **main IDS6938 repository**. Use CMake to create project files for the Homework 1 assignment (*Hint: and Lecture 6 and Lecture 6 Solution*). Set your *startup project* to the correct project. Test building and executing the homework 1 project. Look over and understand the framework and find the functions you need to edit for the assignment.

Part 1: Numerical Analysis of Integration Functions (25 pts).

We looked at numerical solutions to **Initial Value Problems**. IVPs consist of an *ordinary differential equation* with an initial condition. We learned this **mathematical model** helps us **simulate** a system's evolution over time. We also learned that numerical solutions are only approximations to symbolic (analytical) solutions. To compare the error we solved for the exact solution and compared the results to the numerical solution results (*See provided example on Webcourses - you should understand those modules and build-execute the supplemental source code for the lectures*).

Let's look at another IVP:

$$\frac{dy}{dx} = y - \frac{1}{2} * e^{\frac{x}{2}} * sin(5x) + 5e^{\frac{x}{2}} * cos(5x)$$

where the initial condition for this ODE is defined by $y(0) = 0$.

(a) - 2pts: Solve for the exact symbolic (analytical) solution. (*Hint: take the integral.*)

(b)- 5pts: After step (a) you have the values for the `df` and `exact` functions. Translate the mathematical formulas you now have into the `df` and `exact` functions.

(c)- 5pts: Run three numerical integration solutions: RK1, RK2, and RK4 to generate the numerical and exact values. To start use $h = 0.1$, for $x =< 0.0, 10.0 >$. (*Hint: this produces 100 values for each solution.*) Graph the results of RK1, RK2, and RK4 and the exact solution. (Use whatever approach you like to graph this data: R, Python, Matlab, Excel - you should produce a graph with 4 curves.)

(d)- 5pts: Plot the error percentages for RK1, RK2, and RK4 in another graph. Remeber the error is defined as : $\%error = \frac{|exact - approx|}{exact}$.

(e)- 5pts: Vary the step size $h = n$ where you define three n values for $x =< 0.0, 10.0 >$. Plot the results just for RK4 with 3 different n values (with the exact solution). Plot the error rate.

(f)- 3pts: Analyze you results: (1) Describe how varying the integration method changes the accuracy. (2) What happens as you increase the x value to the accuracy. (3) How does varying the step size effect the accuracy. (4) Which method is the most accurate and why (in particular explain what is taken to account in the solution).

Part 2: Programming a Jello Cube (60 pts).

The code includes an incomplete jello animation system. The starter code has been designed to compile and run on a Windows system running Visual Studio. All assignment code should be added to the **JelloMesh** class which contains function stubs marked with "TODO" to indicate where you should implement your assignment. Although much of the bookkeeping code has been written, the following parts are missing and you are required to write:

- **(10 points) Forward Euler, midpoint integration, and RK4.** All three integration methods should be available and working. In particular, you will need to:
  -Implement **JelloMesh::EulerIntegration()**
  -Implement **JelloMesh::MidpointIntegration()**
  -Implement **JelloMesh::RK4Integration()**

- **(10 points) Particle forces other than gravity.** We've implemented gravity for you, but you should implement spring forces. You can also try implementing other forces if you like, such as those input by a user (see the extra credit for more information). Let us know what you try! In particular, you will need to
  -Implement spring forces in **JelloMesh::ComputeForces(ParticleGrid& grid)**
  -Find good spring constants

- **(10 points) Collision and penetration detection.** The particles should be tested for collisions and contacts with the ground and with the cylindrical objects in the scene. In particular, you will need to Look at the function **JelloMesh::CheckForCollisions(ParticleGrid& grid)**, which collects contacts and collisions into lists
  -Implement **JelloMesh::CylinderCollision()**
  -Implement **JelloMesh::FloorCollision()**

- **(10 points) Collision and penetration response.** The system intends for you to consider "collision" to be the case where a particle is very very close to, but not really inside, an object. In a collision, particles which are moving towards, rather than away from, the object should have a gentle impulse (change in momentum, manifested as a change in velocity) applied to them. The exact details of this are up to you: find something which produces good-looking results. "Penetration", on the other hand, is the situation where the particle is actually inside a cylinder. If there is penetration, you should apply a stiff spring force on the particle out from the surface of the object. In particular, you will need to
  Implement **JelloMesh::ResolveContacts(ParticleGrid& grid)**
  Implement **JelloMesh::ResolveCollisions(ParticleGrid& grid)**

- **(10 points) Extra springs.** A simple network of structural springs connect the particles, but there are no shear or bending springs. The decisions you make on spring placement will have marked effects on the movement of the jello. You should be creative in your decisions, but in any case you should have springs to simulate shear and bending resistance. In particular, you will need to Modify **JelloMesh::InitJelloMesh()** to augment the structural springs with bend and shear springs.

- **(10 points) Implementing 2 features on the extra features list.** Pick any feature on the "extra features" list below to customize your assignment to fit your interests. Please document this in your writeup.

Part 3: Written Questions (10 pts).

Here are five questions for you to think about and (briefly) answer. (Each question is worth 2pts.)

- What is the effect of the K_s and K_d parameters on the jello? .
- What are the benefits and the drawbacks of the collision system used here? What are some different ways in which it could be improved? .
- What are some example systems you could model with Mass-spring simulations? Explain how you would you construct the model.
- Does the jello behave realistically? What integration method did you choose to make the Jello stable?
- How would you model and simulate water (in terms of a continuous simulation)?

Part 4: Final Report (5 pts).

Write up the results to the previous sections in the main **Readme.md** in your forked repository. Turn in the URL for your fork in webcourses. Be visual. The report should contain the graphs and analysis from part 1, discussing the results for your jello cube: how and what did you pick for K_s and K_d on the jello, which integration method produced better results. Screenshots of your test environments - be creative. The responses for part 3. I have high expectations for the documentation here and you should allot the proper time to compose the writeup.

Extra Features (Extra Credit - 25 pts).

You have to implement two features from this list for Part 2. You may choose any two features you wish from this list. (Please explicitly note them in your **Readme.md**)

If you feel like going beyond the scope of the assignment, you should consider implementing more of the following extra features to implement. **Get the assignment working without them first.** You can get a maximum of 25 points in extra credit. Simply implementing these things doesn't guarantee you a 25; you really need to go above and beyond to get the full amount. (*The instructor reserves the right to hand out extra credit as his he sees fit.*)

- (5 points) Support collisions with other shapes, such as an inclined plane, cubes, and spheres (Implement JelloMesh::CubeCollision() , Implement JelloMesh::SphereCollision()). Make a new, interesting scene.
- (5 points) Let the user pick and drage particles on the cube surface
- (5 points) Support user generated forces. When the user drags the mouse, generate a corresponding forces to push the jello.
- (5 points) Support a procedurally generated force field and visualize it.
- (15 points) Support a different jello shape. For example, try tetrahedralizing a mesh using a 3rdparty API such as tetgen or your own OBJ loader (http://tetgen.berlios.de/)
- (25 points) Multiple Jello cubes interacting. Support multiple jello cubes bouncing off each other.

- (5 each) Different and more stable integration methods. Verlet, Velocity Verlet, Leapfrog, or Symplectic. (5 pts each).
- (5 points) Create a movie of your jello cube environment, upload the video to youtube and link it in your assignment writeup.
- (5 points) Create a movie of your jello cube that clearly explores the differences between integration types and spring stiffnesses, upload the video to youtube and link it in your assignment writeup.
- (25 points) Support Advanced Rendering. Render the Jello cube with a Physicall-based renderer of choice (Arnold, Renderman, V-Ray, PBRT, Mitsuba, Mental Ray, ...)
- You are welcome to make suggestions for a feature of your own choosing, but they must be approved by instructor before coding