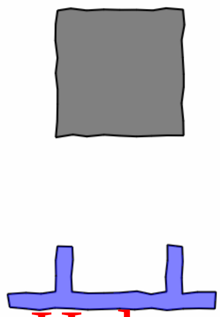
COMP 559 - Winter 2020 - Assignment 3
Finite Elements and Fracture

Available Friday February 28

Due 11:30 pm Friday March 20

Getting Started



The starter code provides you with pieces of working simulation, but it will not do anyhting interesting until you implement the force computation.

Download the provided code from MyCourses, and dump it into a new java project. Do not change the package names as this will interfere with marking. You can create any new classes you like and modify the provided ones ... ts in a few areas (see the TODO comments) ... provided code and javadoc to familiar... through the main display method in the deb... erent classes provided:

- **FractureApp** - Main entry point for the a... ation creates a swing interface for adjusting viewing and simul... plore these parameters). The keyboard interface provides the following:
    - space - toggles running of the simulation
    - s - steps the simulation
    - r - reset
    - c - clear
    - enter - toggles recording of canvas to stills directory
    - esc - quit

  Many of these controls and a variety of other controls are available in the swing interface window. Extend these as you see fit.
- **ConjugateGradientMTJ** - A conjugate gradient solver implmenetation, for implementing implicit integration.
- **Filter** - Velocity filter for pinned DOF constraints in the conjugate gradient solver (see the implementation in FEMSystem).
- **MatrixMult** - An interface used by the conjugate gradients solver for matrix multiplication (implemented by the FEMSystem).
- **FEMTriangle** - An elastic element. **You will need to finish this class.** It needs to have force computation for the element, and in addition, the method to compute force differentials is needed for implicit integration, and for Raleigh damping.
- **Particle** - The particle class keeps track of its position, velocity, and mass. It also includes a force accumulator, and position and force differentials for implicit integration and Raleigh beta damping. It also has an adjacent triangle list (needed

for fracture), and has a **method for computing its separation tensor that you will need to finish.**

- **FEMSystem** - The system keeps a list of particles (DOFs) and force elements, and has a method for advancing time. You will want to **add code to process fracture events**.
- **Edge** - A line segement used to mark boundaries of the model in collision detection.
- **Matrix2d** - A 2D matrix class based on vecmath with useful methods for this assignment, for instance, eigenvalue decomposition, frobenius norm, inverse, and rank1 updates.
- **MouseSpring** - A simple implicit spring that allows you to add forces to a triangle based on the mouse position.
- **Collision** - Computes overlap between triangles and has methods for computing collision response with implicit damping as described in the paper by Parker and O'Brien (this code is a bit buggy, and we may provide you with a patch part way through the assignment).
- **ImageBlocker** - A helper class for loading triangle meshes from images much like those in the rigid body assignment. Each pixel creates 4 triangles. Blue pixels will cause degrees of freedom to be pinned. The mass of triangles is computed from the colour of the triangles (darker is heavier).
- **Triangle** - A helper class for loading triangle meshes created with Triangle. The windows executable is provided with the sample code should you want to try generating new meshes. If you make new meshes, you'll want to read the docs to choose good command line parameters to generate the mesh (see the command line parameters suggested in this Java file).

The provided cod the dataFracture folder. If you make any ne is folder when you subit your assignment.

The code makes use JOGL2, vecmath, mtj, a can reuse the provided jars from the last assignment.

Steps and Objectives (15/15)

Many (perhaps not nall) of the important parts of the program that you will need to modify or write to complete this assignment are labeled with TODO comments (look for them in Eclipse's Tasks pane).

1. ***FEMTriangle forces (4 marks)***

   Implement STVK elastic forces. Use the Matrix2D class to compute the Green strain tensor and then 1st Piola-Kirchoff stress tensor to compute forces as the DOFs following the FEMDefo course notes. For visualizaiton, and fracture computation you must also compute the stress tensor in Equation 7 of O'Brien and Hodgins [1999], i.e., same as the 1st Piola-Kirchoff stress tensor without multiplying on the left by the deformation gradient F.

   Use simple forward Euler explicit integration to test your forces, but note that you will need small time steps for stability.

   I suggest the following parameters for testing this objective:

   - Default step size of 1e-2

    - subsets set to 100 (tiny steps between displayed frames to ensure stability!)
    - implicit integration unchecked (i.e., explicit)
    - Young's modulus of 200
    - Turn off the Raleigh alpha and beta (you probably will not have written them yet anyway)
    - Collision viscous coefficient of 10

2. ***Implicit integration with force differentials (4 marks)***

   Implement implicit integration using the provided conjugate gradients solver. Implement Rayleigh damping too.

3. ***Separtion tensor (3 marks)***

   Following Section 4.1 and 4.2 of the O'Brien and Hodgins paper, split each element's stress tensor into compressive and tensile components, and compute their contributions to the mesh vertices.

4. ***Fracture processing (3 marks)***

   Process fracture events at vertices that have an eigenvalue that exceeds the material toughness. Only process one fracture event per time step (the one that most needs to break). Note that each fracture event will create a new particle and new boundary edges. Careful bookkeeping will be necessary to make sure that the simulation continues correctly. Note that you should not process fracture events that involve pinn n of "hinge" vertices (see the paragrap [2009]).

5. ***Demonstration movies, Novel Scene ie (1 mark)***

   Create a novel test system, and record a ing interesting or amusing. This could be something within ality of the specification and provided code, or some additional feature you add to your code. Be creative! If you make something beautiful, consider recording the images at 720p and deactivating the text overlay.

Other extensions

Here are some improvements or extensions that you might want to think about in the context of course a project.

- The provided collision detection code does an n squared test on triangles that are adjacent to boundary edges. While better than testing all triangles, consdider using spatial hashing to speed up the code by reducing the number of triangle overlap tests.
- The provided collision detection code has some small bugs. Normal computation can fail in some cases. Likewise, repulsion forces are computed but do not work for all types of scenes.
- Consider remeshing to allow fracture along directions other than the existing edges. See the O'Brien 1999 paper for ideas.
- All the concepts apply to 3D just as they do in 2D. Consider extending this assignment to use tetrahedral meshes in 3D. See TetGen software for generating

tetrahedral meshes (tetwild in libigl, and gmsh are other alternatives), or consider using a regular tetrahedral mesh like in Muller's interactive virtual materials paper.

References

E. Sifakis and J. Barbic, FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction, SIGGRAPH Courses, 2012.

E. Parker and J. O'Brien, Real-time deformation and fracture in a game environment, Symposium on Computer Animation, 2009.

J. O'Brien and J. Hodgins, Graphical modeling and animation of brittle fracture, SIGGRAPH, 1999.

M. Muller and M. Gross, Interactive Virtual Materials, Graphics Interface, 2004.

Finished?

Great! Submit your source code and videos as a zip file via MyCourses. ***Do not use anything other than zip to pack your assignment!*** Include the appropriate package structure for your source and data files, and do not include unnecessary files (e.g., class files, your bin directory, or your stills folder). The export filesystem option in eclipse is a good mechanism for preparing for final submission. Include a readme.txt or readme.pdf file with any specific question answers or comments. ***Do not submit a readme without either a pdf or tx*** for your submission video such that it does n **your uploaded submission** as y t files! *Note that you are encouraged to discuss assignments with your* *to the point of sharing code and answers. All code and written answ*