Designing an Interactive OpenGL Slide-Based Presentation of the Softbody Simulation System for Teaching and Learning of Computer Graphics Techniques

Miao Song
Computer Science and Software Engineering
Concordia University
Montréal, Québec, Canada
m_song@cse.concordia.ca

Serguei A. Mokhov
Computer Science and Software Engineering
Concordia University
Montréal, Québec, Canada
mokhov@cse.concordia.ca

Peter Grogono
Computer Science and Software Engineering
Concordia University
Montréal, Québec, Canada
grogono@cse.concordia.ca

ABSTRACT

3D graphics power-point-like slides in OpenGL is a way to present a demo or a teaching item. We focus on how the softbody objects are modeled and rendered with the intent to make a base for teaching, learning, and education of/for computer graphics and integration with the softbody framework. This is work-in-progress on the implementation and publication of this integration and demonstration work.

Categories and Subject Descriptors

I.3.7 [**Three-Dimensional Graphics and Realism**]: Animation

General Terms

Design, Human Factors

Keywords

softbody, real-time, frameworks, OpenGL, physical-based modeling, education, presentation

1. INTRODUCTION

It is useful for the teaching and learning to be able to present some computer graphics (CG) techniques, such as advanced rendering and physical-based real-time animation of softbody objects with and without GPU support [17, 3, 16] and release the resources at the same time. Teaching may be less effective if the examples are not visuallized in class for the students. At the same time it is a burden for the instructor presenting the concepts and switching between the presentation power-point-like slides and the demo program. We argue that it would be more effective and efficient to combine the OpenGL CG programs with OpenGL presentation slides in one, where the traditional power-points and various techniques can be exemplified at run time and the source code can be released to the students later to follow the examples through all angles. For this purpose we are in the process of integration of the physical-based softbody simulation system [11, 12, 13] with the the OpenGL slide presentation system [6, 7].

2. RELATED WORK

The bulk of the related work in this paper is concerned with the two separate frameworks and systems being integrated together to eventually form one CG teaching module, with the physical-based modeling and presentation integrated together.

2.1 Softbody Simulation Framework and System

In our real physical world there exist not only rigid bodies but also soft bodies, such as human and animal's soft parts and tissue, and other non-living soft objects, such as cloth, gel, liquid, and gas. Softbody simulation is a vast research topic and has a long history in computer graphics.

The softbody system has gone through a number of iterations in its design and development. Initially it had limited user interface [11, 12].

Then the fine-grained to high-level level-of-detail (LOD) GLUI-based [9] user interface has been added [13], GPU shading support was added [14] using the OpenGL Shading Language [10], a curve-based animation was integrated, and software engineering re-design has happened.

The example of the common visual design of the LOD interactivity interface is summarized in Figure 1. The LOD components are on the right-hand-side (in their initial state, not expanded), and the main simulation window is on the left (interactivity with that window constitutes for now just the mouse drag and functional keys). Following the top-down approach configuration parameters, that assume some defaults, were reflected in the GUI.

2.1.1 Softbody Framework's Design and Implementation Overview

The framework's design has centered around common dimensionality (1D, 2D, and 3D) of graphical objects for simulation purposes, physic's based integrators, and the user interactive component. At present, the Integrator API of the framework as of this writing is implemented by the well-known Explicit Euler, Midpoint, Feynman, and Runge-Kutta 4 (RK4)-based integrators for their mutual comparison of the run-time and accuracy. The system is implemented using OpenGL [8, 19] and the C++ programming language with object oriented programming paradigm.

This elastic object simulation system has been designed and implemented according to the well known architectural pattern, the Model-View-Controller [18]. This pattern is

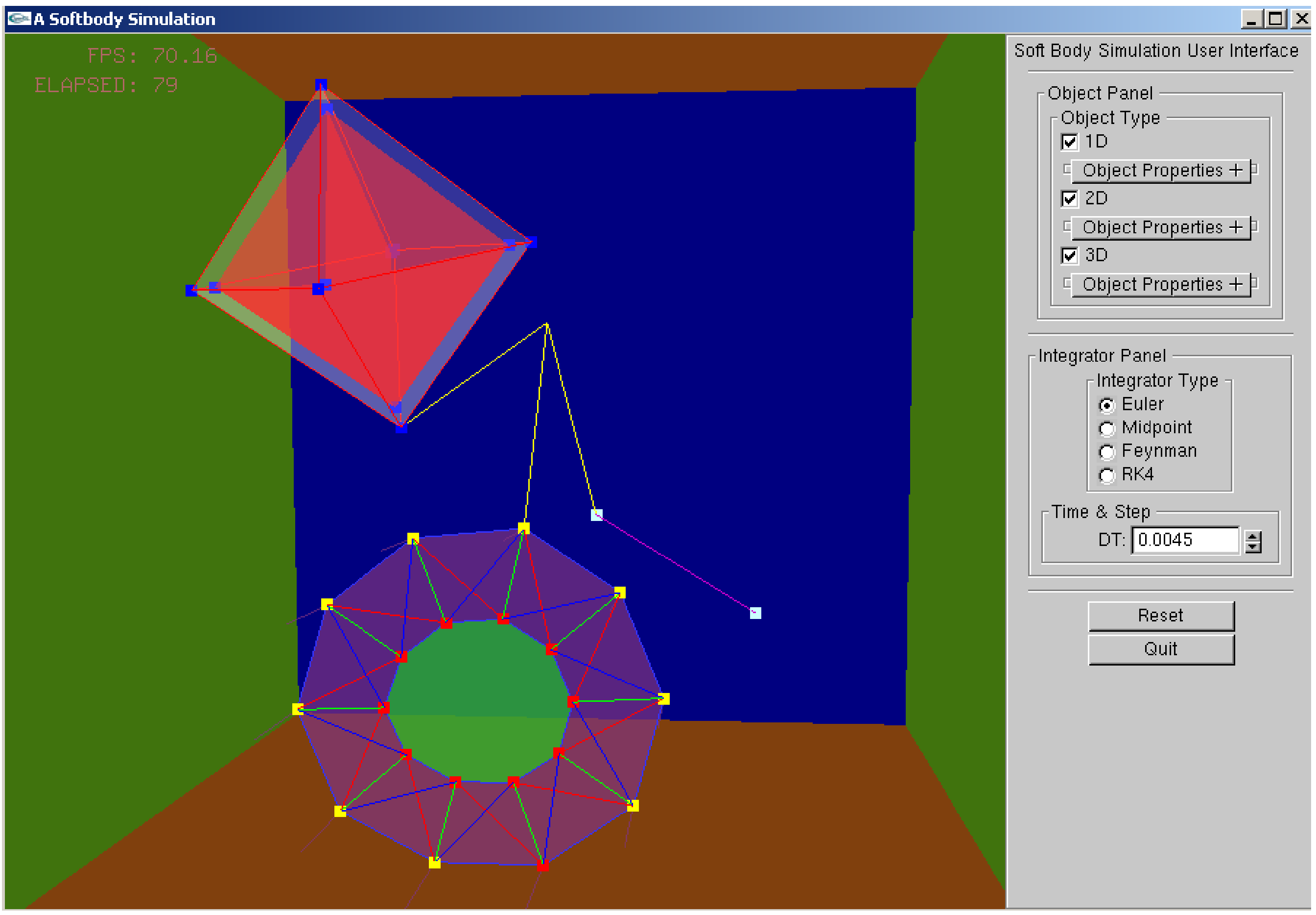


Figure 1: Three Types of Softbody Objects Dragged by Mouse

ideal for real time simulation because it simplifies the dynamic tasks handling by separating data (Model) from user interface (View). Thus, the user's interaction with the software does not impact the data handling; the data can be reorganized without changing the user interface. The communication between the Model and the View is done through another component: Controller. This also closely correlates to the OpenGL state machine, that we use as a core library for our implementation. In our current simulation system, the application has been split into these three separated components [12].

2.1.2 *Example Softbody Animation*

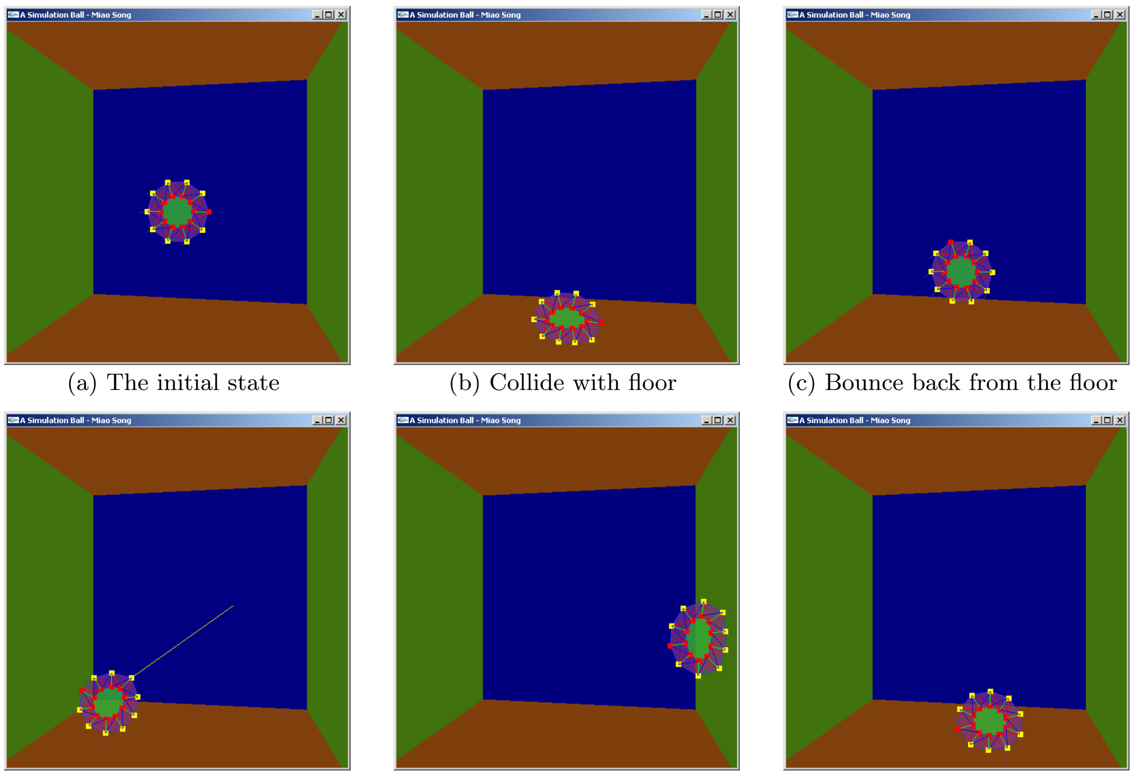
In this section, the two-dimensional and three-dimensional objects are illustrated at different animation sequences, with different simulation parameters, and by simulation with different numerical integration methods. The screenshots present the animation sequence of the two-dimensional, and three-dimensional objects when they are at the initial state, colliding with floor, bouncing back from the floor, responding to user's external dragging, and at the resting state.

2D. Figures 2(a) through 2(f) show how a two-dimensional object moves in a 3D environment. This two-layer object consists of 10 particles and 10 structural springs on both inner and outer circles. Moreover, it contains 10 radius springs, 10 shear left springs, and 10 shear right springs between the inner and outer layers. If a 2D object with only one layer, or the object has no pressure force within, the spring's stiffness has to be a larger value than without, then the object will not collapse. However, as shown in Figure 2(b), if the spring stiffness is small enough, the object does not collapse, neither overlap with the layers because of the stability of the two-layer structure [12].

3D. The simulation as shown in Figures 3(a) through 3(f) is how a 3D uniform facet object moves in a three-dimensional environment. This two-layer object, which is generated by subdividing an octahedron once, consists of 12 particles, 36 structural springs, and 32 faces, on both inner and outer spheres. Moreover, the object also contains 36 radius springs, 36 shear left springs, and 36 shear right springs between the inner and outer layers. Just like in two dimensions, the two-layer structure gives the 3D sphere more stability [12].

```
void Idle() {
  object1D.Update(DT, mousedown!=0, xMouse, yMouse);
  object2D.Update(DT, mousedown!=0, xMouse, yMouse);
  object3D.Update(DT, mousedown!=0, xMouse, yMouse);
  glutPostRedisplay();
}
```

Figure 4: `Idle()` Model Updates



(a) The initial state (b) Collide with floor (c) Bounce back from the floor

(d) Drag the object (e) Response to compact (f) The resting state

Figure 2: Animation Sequence of the 2D Elastic Object

```
void Object::Update(float deltaT, bool drag,
  float xDrag, float yDrag) {
  if(integrator == NULL) {
   switch(integratorType) {
    case EULER:
      integrator = new EulerIntegrator(*this);
      break;
    case FEYNMAN:
      integrator = new FeynmanIntegrator(*this);
      break;
    case MIDPOINT:
      integrator = new MidpointIntegrator(*this);
      break;
    case RK4:
      integrator = new RungeKutta4Integrator(*this);
      break;
    default:
      assert(false);
      return;
    }
    integrator->setDimension(dim);
  }
  integrator->integrate(deltaT, drag, xDrag, yDrag);
}
```

Figure 5: General `Update()` Function

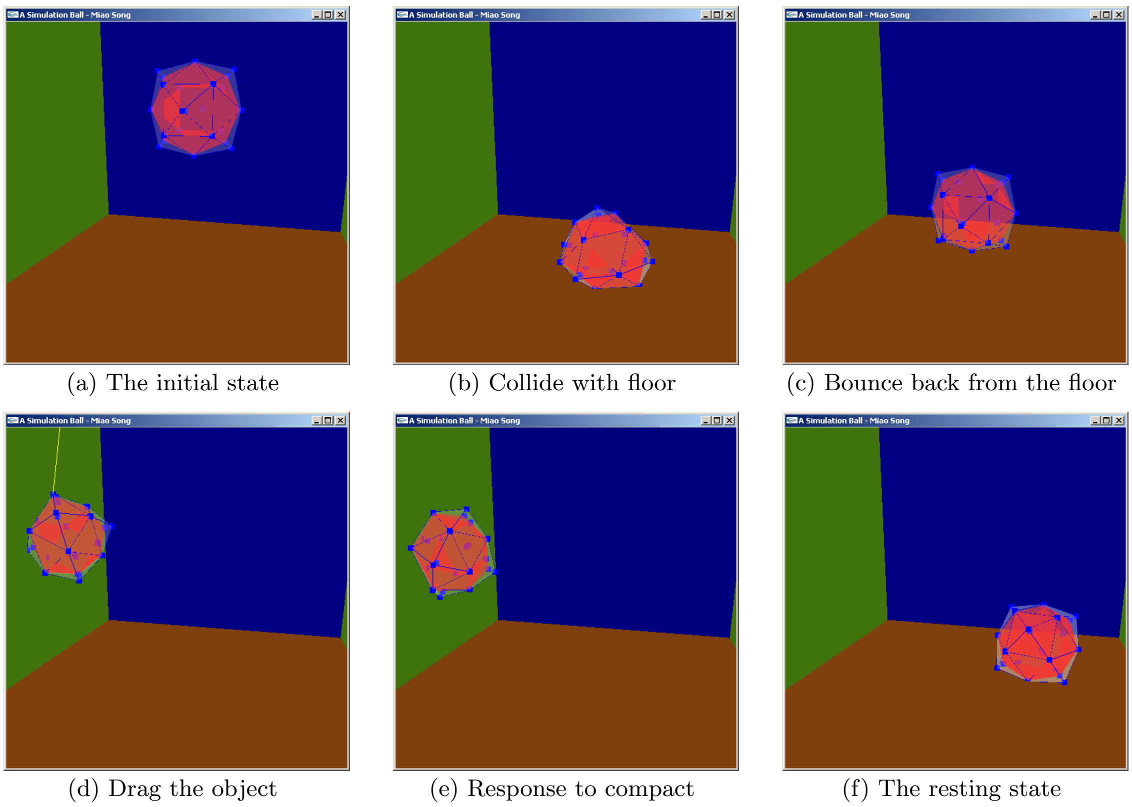
In the main simulation, the `Idle()` function shown in Figure 4, elastic objects update at every time step `DT` to tell the the system how the objects behave and the change for their velocity and position. There are four parameters for `Update()` as shown in Figure 5, the time step `deltaT`, if there exists user interaction `drag=0` by default, the mouse position on `x` and `y` axes (for dragging upon mouse release) is at 0 by default. The general algorithm of the `Update()` presented, illustrates that the most of the actual modifications are based on the dynamically selected integrator and the dimensionality of the simulation object being integrated.

2.1.3 *Summary of the Adjustable Simulation Parameters*

The parameters in the simulation such as mass, spring stiffness, and friction (damping) can be changed. One can drag the object mass with a mouse to change its position. Effects of different simulation parameters are discussed [12].

The parameters that influence the behavior of the simulated environment are summarized below, with their default values. Most initial and default values were based on the 2D case from [4]; otherwise, the values are empirical and are partially dependent on the hardware the simulation is executing on. The values can be changed at real-time with the GUI interface [13].

- ◇ `KS` = 800 where `KS` is structural spring stiffness constant. The larger this value is, the less elastic the object is and it is more resistant to the inner pressure and deformation. The lesser this value is the more object is deformable and a subject to break up if the inner pressure force is high.
- ◇ `KD` = 15 where KD is structural spring damping constant, opposite to the spring retraction force. It denotes how fast the object is to resist its motion.
- ◇ `RKS` = 700 where RKS is radius and shear spring stiffness constant, similar to KS, but for radius and shear springs as opposed to the structural springs.
- ◇ `RKD` = 50 where RKD is radius and shear spring damping constant, similar to KD, but for radius and shear springs.



(a) The initial state (b) Collide with floor (c) Bounce back from the floor

(d) Drag the object (e) Response to compact (f) The resting state

Figure 3: Animation Sequence of the 3D Elastic Object

◇ `MKS` = 150 where MKS is the spring stiffness constant of the spring connected with the mouse and the approximate nearest particle on the object. This constitutes the elasticity of the "drag" spring connected to the mouse: the lesser the value is, the more elastic it is, and the harder it is to drag the object as a result.

◇ `MKD` = 25 where MKD is the damping constant of the spring connect with the mouse and the approximate nearest point on the object.

◇ `PRESSURE` = 20 where PRESSURE is gas constant used in the ideal gas equation mentioned earlier to determine the pressure force inside the enclosed object. If this constant is too high, and the combined spring stiffness for all the spring types is low enough, the object can "blow up".

◇ `MASS` = 1 where MASS is the mass for each particle. The object can be made heavier or lighter if this value is larger or smaller respectively, in order to experiment with the gravity effects. Naturally, the heavier objects will be more difficult to drag upwards in the simulation environment. Conversely, the smaller-mass object can be dragged around with less effort given the rest of the parameters remain constant.

2.2 OpenGL Slides Framework

OGLSF presents a way of making slides, transiting between them using controls, allow for common bulleted textual widgets – the *tidgets*, and allowing overriding the control handling from the main idle loop down to each individual slide in OpenGL. All slides together build up a `Presentation`, which is a collection of slides that uses the Builder pattern to sequence the slides. Each slide is a derivative of the `Slide` class and represents essentially a scene outfitted with the default keyboard controls for the tidgets and navigation.

Each scene on the slide is modeled using traditional procedural modeling techniques and is set as a developer or artist desires. It can include models and rendering of any primitives, complex scenes, texturing, lighting, GPU-based shading, and others, as needed and is fit by the presenter [6, 7].

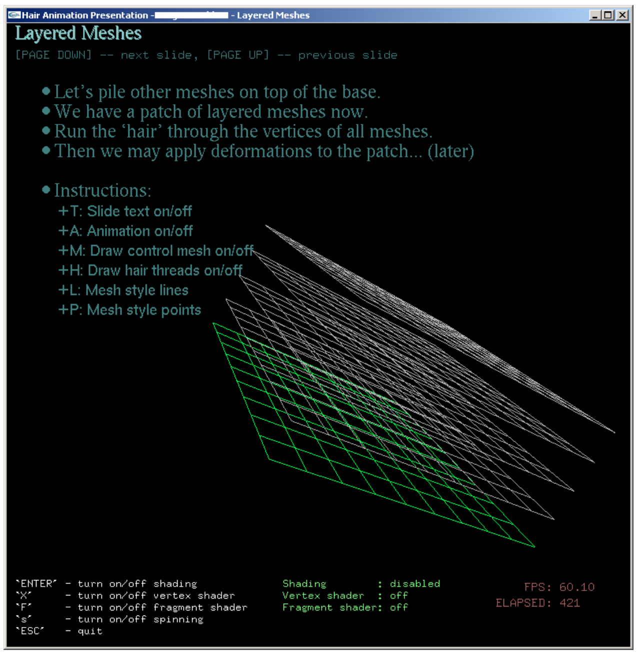
The main program must delegate its handling of the callback controls for keyboard, mouse, and idle down to the presentation that does it down to each current slide in addition to whatever handling it itself requires.

An example of slides made for a project [5] using the OpenGL-based slides is illusrated in Figures 6, 7, 8, and 9. It is imperative that the tigets can be enabled and disabled to allow the main animation to run unobstructed.

3. METHODOLOGY

The methodology consists of the adjustments required for the integration and then following by making the actual presentation with slides. The source code of the presentation is a part of the learning material along the actual content of the material presented.

Initially, both frameworks and implementing systems define the `main()` function, which cannot be included into any of the libraries (both can be compiled into the library files to be linked into other projects) as there will be linker errors when the object code from the two or more system is combined into a single executable. We have to start a new application with a new `main()`, the `SoftBodyPresentation.cpp`.



Hair Animation Presentation - [redacted] - Layered Meshes

Layered Meshes

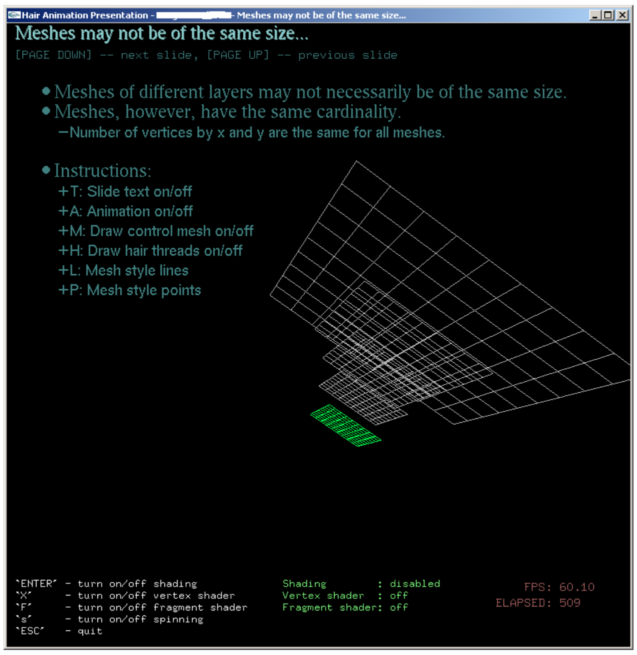
[PAGE DOWN] -- next slide, [PAGE UP] -- previous slide

- Let's pile other meshes on top of the base.
- We have a patch of layered meshes now.
- Run the 'hair' through the vertices of all meshes.
- Then we may apply deformations to the patch... (later)

- Instructions:
  - +T: Slide text on/off
  - +A: Animation on/off
  - +M: Draw control mesh on/off
  - +H: Draw hair threads on/off
  - +L: Mesh style lines
  - +P: Mesh style points

```
`ENTER' - turn on/off shading              Shading        : disabled       FPS: 60.10
`X'     - turn on/off vertex shader        Vertex shader  : off        ELAPSED: 421
`F'     - turn on/off fragment shader      Fragment shader: off
`s'     - turn on/off spinning
`ESC'   - quit
```

Figure 6: OpenGL Slide Example 1



Hair Animation Presentation - [redacted] - Meshes may not be of the same size...

Meshes may not be of the same size...

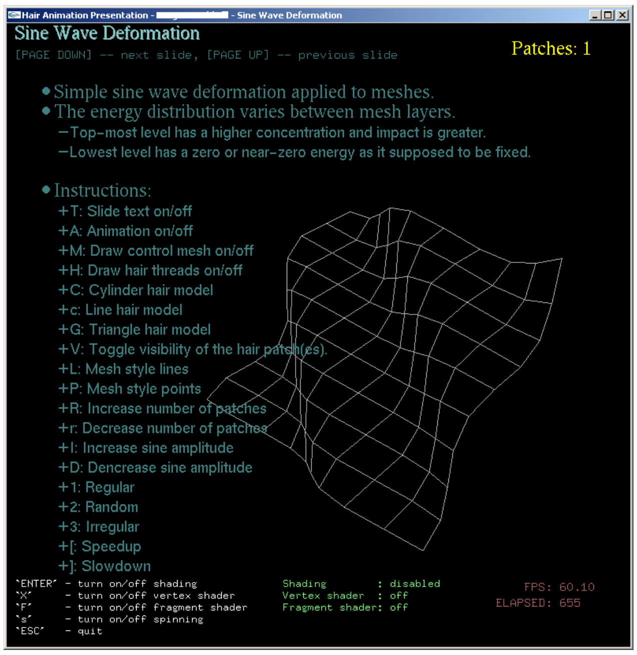
[PAGE DOWN] -- next slide, [PAGE UP] -- previous slide

- Meshes of different layers may not necessarily be of the same size.
- Meshes, however, have the same cardinality.
  - Number of vertices by x and y are the same for all meshes.

- Instructions:
  - +T: Slide text on/off
  - +A: Animation on/off
  - +M: Draw control mesh on/off
  - +H: Draw hair threads on/off
  - +L: Mesh style lines
  - +P: Mesh style points

```
`ENTER' - turn on/off shading              Shading        : disabled       FPS: 60.10
`X'     - turn on/off vertex shader        Vertex shader  : off        ELAPSED: 509
`F'     - turn on/off fragment shader      Fragment shader: off
`s'     - turn on/off spinning
`ESC'   - quit
```

Figure 7: OpenGL Slide Example 2



Hair Animation Presentation - [redacted] - Sine Wave Deformation

Sine Wave Deformation

[PAGE DOWN] -- next slide, [PAGE UP] -- previous slide

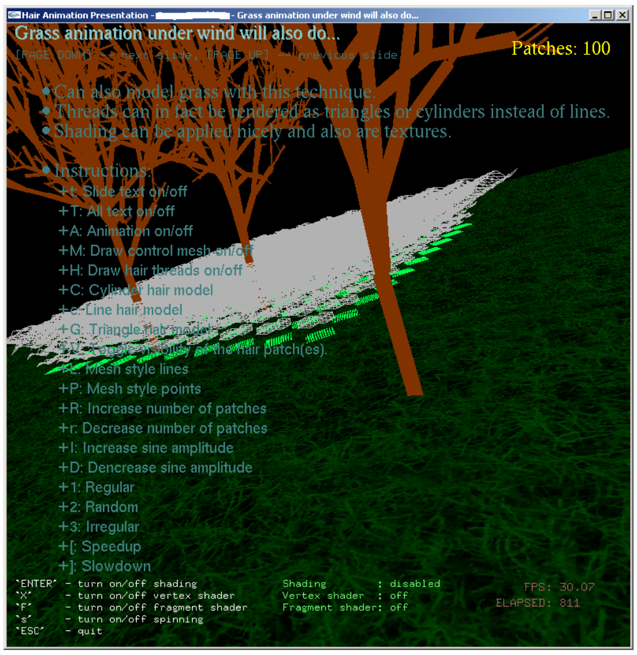
Patches: 1

- Simple sine wave deformation applied to meshes.
- The energy distribution varies between mesh layers.
  - Top-most level has a higher concentration and impact is greater.
  - Lowest level has a zero or near-zero energy as it supposed to be fixed.

- Instructions:
  - +T: Slide text on/off
  - +A: Animation on/off
  - +M: Draw control mesh on/off
  - +H: Draw hair threads on/off
  - +C: Cylinder hair model
  - +c: Line hair model
  - +G: Triangle hair model
  - +V: Toggle visibility of the hair patch(es).
  - +L: Mesh style lines
  - +P: Mesh style points
  - +R: Increase number of patches
  - +r: Decrease number of patches
  - +I: Increase sine amplitude
  - +D: Dencrease sine amplitude
  - +1: Regular
  - +2: Random
  - +3: Irregular
  - +[: Speedup
  - +]: Slowdown

```
`ENTER' - turn on/off shading              Shading        : disabled       FPS: 60.10
`X'     - turn on/off vertex shader        Vertex shader  : off        ELAPSED: 655
`F'     - turn on/off fragment shader      Fragment shader: off
`s'     - turn on/off spinning
`ESC'   - quit
```

Figure 8: OpenGL Slide Example 3



Hair Animation Presentation - [redacted] - Grass animation under wind will also do...

Grass animation under wind will also do...

[PAGE DOWN] -- next slide, [PAGE UP] -- previous slide

Patches: 100

- Can also model grass with this technique.
- Threads can in fact be rendered as triangles or cylinders instead of lines.
- Shading can be applied nicely and also are textures.

- Instructions:
  - +t: Slide text on/off
  - +T: All text on/off
  - +A: Animation on/off
  - +M: Draw control mesh on/off
  - +H: Draw hair threads on/off
  - +C: Cylinder hair model
  - +c: Line hair model
  - +G: Triangle hair model
  - +V: Toggle visibility of the hair patch(es).
  - +L: Mesh style lines
  - +P: Mesh style points
  - +R: Increase number of patches
  - +r: Decrease number of patches
  - +I: Increase sine amplitude
  - +D: Dencrease sine amplitude
  - +1: Regular
  - +2: Random
  - +3: Irregular
  - +[: Speedup
  - +]: Slowdown

```
`ENTER' - turn on/off shading              Shading        : disabled       FPS: 30.07
`X'     - turn on/off vertex shader        Vertex shader  : off        ELAPSED: 811
`F'     - turn on/off fragment shader      Fragment shader: off
`s'     - turn on/off spinning
`ESC'   - quit
```

Figure 9: OpenGL Slide Example 4

Additionally, both frameworks have to declare their own namespaces, which both have not done in the past, similarly like CUGL [2] does because there are some common names of variables, classes, or functions that clash. This will be an overall improvement not only for this work, but also for any similar type of integration with other projects.

Most of the main code from `SoftbodySimulation.cpp` application becomes encapsulated into a generic `SoftbodySimulationSlide` class that includes the default configuration of the parameters presented in the earlier section.

Furthermore, concrete slides that inherit from `SoftbodySimulationSlide` are broken down into some preset distinct configuration defaults and accompanying tidgets. They override the `idle()` function as well as the state LOD parameters per an example slide. These currently include:

◇ `TitleSlide` – the common title slide with the lecture title and the presenter information.

◇ `TOCSlide` – table of contents of the presentation.

◇ `IntroductionSlide` – introduction of the material.

◇ `SoftbodySimulationSlide1D` – a slide featuring a 1D spring object by default encased in the `ViewBox`.

◇ `SoftbodySimulationSlide2D` – a slide featuring a 2D softbody object by default.

◇ `SoftbodySimulationSlide3D` – a slide featuring a 3D softbody object by default.

◇ `SoftbodySimulationSlideAllD` – a slide featuring all types of softbody objects by default, similarly to the Figure 1, except encased into a slide environment.

◇ `SoftbodySimulationSlideAllEuler` – all three objects configured by default to animate under the Explicit Euler integrator.

◇ `SoftbodySimulationSlideAllMidpoint` – all three objects configured by default to animate under the Midpoint integrator.

◇ `SoftbodySimulationSlideAllFeynman` – all three objects configured by default to animate under the Feynman integrator.

◇ `SoftbodySimulationSlideAllRK4` – all three objects configured by default to animate under the RK4 integrator.

◇ `ConclusionSlide` – preliminaty conclusions slide.

◇ `ReferencesSlide` – the list of references.

It is reasonable to expect that the above slides to be presented at the poster session as an example.

4. CONCLUSIONS AND FUTURE WORK

By designing and integrating a softbody simulation system and the OpenGL slides framework together we are coming up with a good CG teaching item that combines the demonstration of the techniques and presentation into one unit, that can be released to the students for learnign purposes in the form of the source code.

4.1 Future Work

There are a number of immediate items of the future work:

◇ Complete the implementation of the teaching item slides.

◇ Make it source code portable to Linux and MacOS X. Currently it only compiles properly under Windows.

◇ We plan on releasing our code as open-source implementation either a part of the Concordia University Graphics Library [2] and/or as part of a Maya [1] plug-in and as a CGEMS [3] teaching module.

◇ Allow advanced controls of the scenes and slides, e.g. by using haptics devices [15].

5. REFERENCES

[1] Autodesk. Maya. [digital], 2008. `autodesk.com`.

[2] P. Grogono. Concordia University Graphics Library (CUGL). [online], Dec. 2005. `http://users.encs.concordia.ca/~grogono/Graphics/cugl.html`.

[3] J. Jorge, F. Hanisch, F. Figueiredo, and R. Schauer. CG Educational Materials Source (CGEMS). [online], 2008. `http://cgems.inesc.pt/`.

[4] M. Matyja. A pressure model for soft body simulation. In *Svenska Föreningen för Grafisk Databehandling (SIGRAD2003)*, November 2003.

[5] S. A. Mokhov. Real-time animation of hair and thread-like objects via deformation of layered meshes. Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2004. Project and report.

[6] S. A. Mokhov and M. Song. An OpenGL-based interface to 3D PowerPoint-like presentations of OpenGL projects. In *Proceedings of CISSE'08*, University of Bridgeport, CT, USA, Dec. 2008. Springer. To appear.

[7] S. A. Mokhov and M. Song. OpenGL project presentation slides interface and a case study. In *Proceedings of GRAPP'09*, Lisboa, Portugal, Feb. 2009. INSTICC. To appear, `grapp.org`. Poster position paper.

[8] OpenGL Architecture Review Board. OpenGL. [online], 1998–2008. `http://www.opengl.org`.

[9] P. Rademacher. GLUI - A GLUT-based user interface library. SourceForge, June 1999. `http://glui.sourceforge.net/`.

[10] R. J. Rost. *OpenGL Shading Language*. Pearson Education, Inc., Feb. 2004. ISBN: 0-321-19789-5.

[11] M. Song. Dynamic deformation of uniform elastic two-layer objects. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, Aug. 2007.

[12] M. Song and P. Grogono. A framework for dynamic deformation of uniform elastic two-layer 2D and 3D objects in OpenGL. In *Proceedings of C3S2E'08*, pages 145–158, Montreal, Quebec, Canada, May 2008. ACM. ISBN 978-1-60558-101-9.

[13] M. Song and P. Grogono. An LOD control interface for an OpenGL-based softbody simulation framework. In *Proceedings of CISSE'08*, University of Bridgeport, CT, USA, Dec. 2008. Springer. To appear.

[14] M. Song and P. Grogono. Application of advanced rendering and animation techniques for 3D games to softbody modeling and animation. In *Proceedings of C3S2E'09*, Montreal, Quebec, Canada, May 2009. ACM. To appear.

[15] M. Song and P. Grogono. Are haptics-enabled interactive and tangible cinema, documentaries, 3D games, and specialist training applications our future? In *Proceedings of GRAPP'09*, Lisboa, Portugal, Feb. 2009. INSTICC. To appear, `grapp.org`. Short position paper.

[16] J. O. Talton. Teaching graphics with the OpenGL Shading Language. *ACM SIGCSE Bulletin archive*, 39(1), Mar. 2007.

[17] D. Tenneson, A. M. Spalter, J. Kumar, I. Medvedev, and A. van Dam. The Graphics Teaching Tool (GTT). [online], Brown University, 2008. `http://graphics.cs.brown.edu/research/gtt/`.

[18] Wikipedia. Procedural modeling. [online], `http://en.wikipedia.org/wiki/`, 2007.

[19] M. Woo, J. Neider, T. Davis, D. Shreiner, and OpenGL Architecture Review Board. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley, 3 edition, Oct. 1999. ISBN 0201604582.