Demo 1: Simple Dynamic Simulation

Demo 1 Location

- Via SVN: https://svn.ece.lsu.edu/svn/gp/gpup/demo-1-simple.cc

Demo 1 Goals

- Simulate ball bouncing over platform.
- Get a feeling for project and context for material to be covered.
- Will focus on physical simulation and code execution.
- Will skip graphics for now.

Overview

- Demonstration of Capabilities
- Simulation State Representation
- Simulation *Time Stepping*
- Code Execution
Demo 1: Simple Dynamic Simulation

Code Capabilities

- Shows yellow ball bouncing over tiled platform.
- User can move light, viewing position.
- User can move and push ball.
- User can play with gravity.

Parts of Code to Study

- Will look at physical simulation.
- Not expected to understand graphics yet.
Dynamic Simulation: World Representation

*World*

- That which one wants to simulate.
- E.g., bouncing ball, airplane flight, baseball game.

*Physical Model*

- Methods used to represent and simulate world.

*Graphical Model*

- Methods used to display physical model.
Dynamic Simulation: State

**State**

- Description of physical world at a point in time.

**Dynamic State**

- State which changes over time as simulation proceeds.
- E.g., In first demo, ball position.

**Static State**

- State which does not change after simulation initialized.
- E.g., In first demo, platform position.

**Interactive State**

- State changed by user input.
- E.g., In first demo, acceleration of gravity.
DS Terminology Example: Demo 1

World

• A ball bouncing on a tiled platform.

Physical Model

• Ball modeled as point mass in a constant gravity field.
• Platform a perfect rectangle and never moves.
• Contact with platform is frictionless and results in energy loss.

Graphical Model

• Ball is tessellated sphere approximation, with radius of 5.
• Platform checkerboard.
• Further details omitted since material not yet covered.
DS Terminology Example: Demo 1 State

Dynamic State

• Position of ball: Ball::position
• Velocity of ball: Ball::velocity

Static State

• Position of platform. World::platform_xmin...

Interactive State

• Acceleration of gravity.
DS: demo-1-program “Our” Files

*These files written for this class.*

demo-1-simple.cc

- Program startup, physical simulation.

demo-1-simple-graphics.cc

- Graphics and user-interface code.

gp/coord.h

- A basic set of vector classes and operators.

gp/util.h

- Code for starting graphics, simple user interface routines, etc.

shapes.h

- Code to render simple shapes, like a sphere.
Major Program Functions

- **Initialization**
  - Set things up.

- **Accept User Input**
  - Do something with key or button press.
  - Changes interactive state.

- **Simulation**
  - Run time-step code to move state ahead in time.
  - Changes dynamic state.

- **Render Frame**
  - Write graphical representation based on state.
DS: Program Structure

Sequencing of Functions

- Initialization, of course, is first.
- Other functions react to events.

Render Frame

- Once for each display device frame (if things changing).
  - May set up our own timer or use some API function.
- Each time window system requests it. (E.g., after removal of something blocking our window.)

User Input

- Whenever user does something.
DS: Demo 1 Program Structure

Initialization

- Code in World::init() and World::graphics_init().
- Occurs when World object in main routine constructed.

Render Frame

- Code in World::frame_callback().
- Called in reaction to:
  - OS re-paint events (OS asks us to paint our window).
  - GPU driver frame notifications (display is at top of frame).
  - A timer (if frame notifications not available).
  - Details provided later.

Simulation

- Code in World::time_step_v0.
- Called as needed by render routine.
DS: Demo 1 Program Structure

User Input

- Code in World::cb_keyboard in file demo-1-simple-graphics.cc.
- Called in reaction to input events.
Term Project

Goal

- Modification of a dynamic simulation, balls, balloon or something else.
- Good use of GPU programming.
- Good overall internal design.
- Visually interesting, fun, witty, etc.

Project Teams

- Composed of three or four students.

Project Progress

- About four submissions during semester.
Term Project: Team

Team

- Composed of 3-4 Students
- Rely on skills of members.

Important Skills

- Programming, Computer Use
- Physics, modeling.
- Creative problem solving, Packaging.
Team Member Skill: Programming, Computer Use

• Primary Language: C++
• Will cover new C-like languages
  - *OpenGL Shader Language* (GPU language for graphics)
  - *CUDA* (GPU language for not-necessarily-graphical computation)
  - GPU Assembly and Intermediate Languages
• Development Tools
  - Text editors (Emacs), makefiles, debuggers (gdb, cuda-gdb).
  - Code repository (SVN).
Term Project Team Skill: Physics, Modeling

Physics, Modeling

- Equations of motion, angular momentum, springs, etc.
- Find simple physical models for imaginary world.
- Not afraid of math.
Creative problem solving, Packaging

• Find interesting, novel, effective ways of achieving goals.
• Focus on purpose without sacrificing last point.
Project Details

Additional project details will be posted later.