Classroom Demonstration Programs

2s-1

Demonstration Programs Used in Class

Simulate an imaginary world.

Bouncing ball, balloon, etc.

Based on simple physical models.

$\overrightarrow{F} = m\overrightarrow{a}$, and not much more.

Programs make use of:

CPU graphics programming (OpenGL).

GPU graphics programming (OpenGL shader language).

CPU and GPU physics programming (CUDA on GPU).

Bouncing Ball Simulation

Simulates a ball bouncing over a platform.

Purpose is to show overall program structure . . .
. . . and simple physical simulation.

These Notes

First, we'll describe the simulation physics.

Then the overall program structure will be described.

2s-3

Simulation of a Bouncing Ball

Representation of Ball

Position: p, a three-element vector: $\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$.

Velocity: v, a three-element vector: $\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$.

Representation in demo-1-simple.cc:

```
class Ball {
public:
  pCoor position;
  pVect velocity;
};
```


2s-3

2s-4

We should already know that under constant acceleration a:

$$v(t) = v(0) + at$$

$$\begin{aligned} p(t_1) &= p(0) + \int_0^{t_1} v(t)\,dt \\ &= p(0) + \int_0^{t_1} (v(0) + at)\,dt \\ &= p(0) + v(0)t_1 + \frac{1}{2}at_1^2 \end{aligned}$$

In demo-1-simple.cc:

```
ball.position +=
  ball.velocity * delta_t + 0.5 * gravity_accel * delta_t * delta_t;
```

What about the platform?

EE 4702-1 Lecture Transparency. Formatted 11:21, 24 August 2018 from lsli2-simple.
2s-4

2s-5

Platform Collision

Let's keep things simple:

The platform is at $y = 0$.

If there is a collision with the platform ...
... the y component of the velocity will be multiplied by -0.9.

The ball will bounce off more slowly than it hit.

The factor -0.9 is not special, just a typical non-ideal bounce.

$$v(t) = \begin{cases} v(0) + at & \text{if } t \le t_c \\ -0.9\,[v(0) + at_c] + a(t - t_c) & \text{if } t > t_c \end{cases}$$

where t_c is the time of collision.

The equation above only considers the first bounce.

2s-6

Closed-Form Equations for $v(t)$ and $p(t)$?

Should we re-write the equations for $v(t)$ and $p(t)$ for any t?

The discontinuity (platform collision) makes things tedious.

But it is still doable for an undergraduate.

But, what if there were two balls?—or three?

Then, a closed-form expression would be impossible.

2s-7

Discrete Interval Simulation

Idea: Consider short time periods called *time steps*.

The overall simulation will occur over many time steps.

Within a time step separately consider:

Free motion (without collisions).

Collisions.

Consider Pseudocode:

```
for ( time = 0;  time < end_of_time;  time++ )
  {
     simulate_free_motion();
     detect_and_resolve_collisions();
  }
```

2s-8

Simulation of Free Motion

Determine forces on object.

Gravity.

Contact.

From forces and mass determine acceleration.

From acceleration update velocity.

Update position.