Papervision3D DisplayObject3D helper baseclass for realtime updates

by Benny Bottema - Monday, June 15, 2009

http://www.bennybottema.com/2009/06/15/papervision3d-displayobject3d-helper-baseclass-for-realtime-updates/

Here's a small class I made for adding realtime updates to DisplayObject3D objects. This class allows for easy time-based updates instead of the traditional frame-based updates.

- RealtimeDisplayObject3D.as

I use the convention of adding update methods to all (my own) DisplayObject3D objects, which then update themselves according to preconfigured settings (eg. passed in the constructor). I pass in a Camera3D and that's it, let the objects do their own thing. I'm making use of this convention by extracting this update method to a baseclass which then can do time-based updates instead.

It only works for new classes you create that normally would extend DisplayObject3D, such as a composite parent DisplayObject3D. For example, if you were to create an Earth object in PV3D, you could create a new DisplayObject3D container class which adds a planet, clouds and a glow to itself as child objects. This Earth object could then extend our special class, RealtimeDisplayObject3D, for controlling time-based updates.

```
public class Planet extends RealtimeDisplayObject3D {

 private var earth:DisplayObject3D;

 // define: 1 update every 200ms (5 updates per second) and process 1
value per second
 public function Planet() {
  super(200, 1);
  // add planet, clouds and glow etc.
 }

 // rotates Earth 5 times per second, with a stepsize of 0.2 (1 second
 / updates per second)
 protected override function realtimeUpdate(camera:Camera3D, stepsize:
Number):void {
  earth.rotate(stepsize);
 }
}
```

With the planet now extending our RealtimeDisplayObject3D class instead of DisplayObject3D, you can provide realistic rotation speed that is not influenced by the FPS the movie is pulling. Instead, it works by applying a set value-per-second (phase speed) and a required update frequency per second; the phase speed is spread out over the number of updates per second.

There is one requirement though: **the fps the movie is pulling cannot go below the updates per second specified**, or the updates per second is capped by the frames per second. If you specify 200ms for each update, this means 5 updates per second (1000 / 200). This won't be a problem. If however you define 50ms for each update, this means 20 updates per second (1000 / 50). You must ensure that your movie is running on 20 frames per second or more.

Combining time-based updates with frame-based updates

You can still update normally on frame-based timetable by overriding the *update(camera)* method of the baseclass. Or, you can combine both type of updates by overriding *update(camera)*, do a frame-based update, call *super.update(camera)* and override the *realtimeUpdate(camera)* method as we did before. Here's an example of that:

```
public class Planet extends RealtimeDisplayObject3D {

 private var earth:DisplayObject3D;

 // define: 1 update every 200ms (5 updates per second) and process 1
value per second
 public function Planet() {
  super(200, 1);
  // add planet, clouds and glow etc.
 }

 // makes the planet face the camera each and every frame
 protected override function update(camera:Camera3D, stepsize:Number):
void {
  earth.lookAt(camera);
  super.update(camera);
 }

 // rotates Earth 5 times per second, with a stepsize of 0.2 (1 second
 / updates per second)
 protected override function realtimeUpdate(camera:Camera3D, stepsize:
Number):void {
  earth.rotate(stepsize);
 }
}
```

PDF generated by Kalin's PDF Creation Station