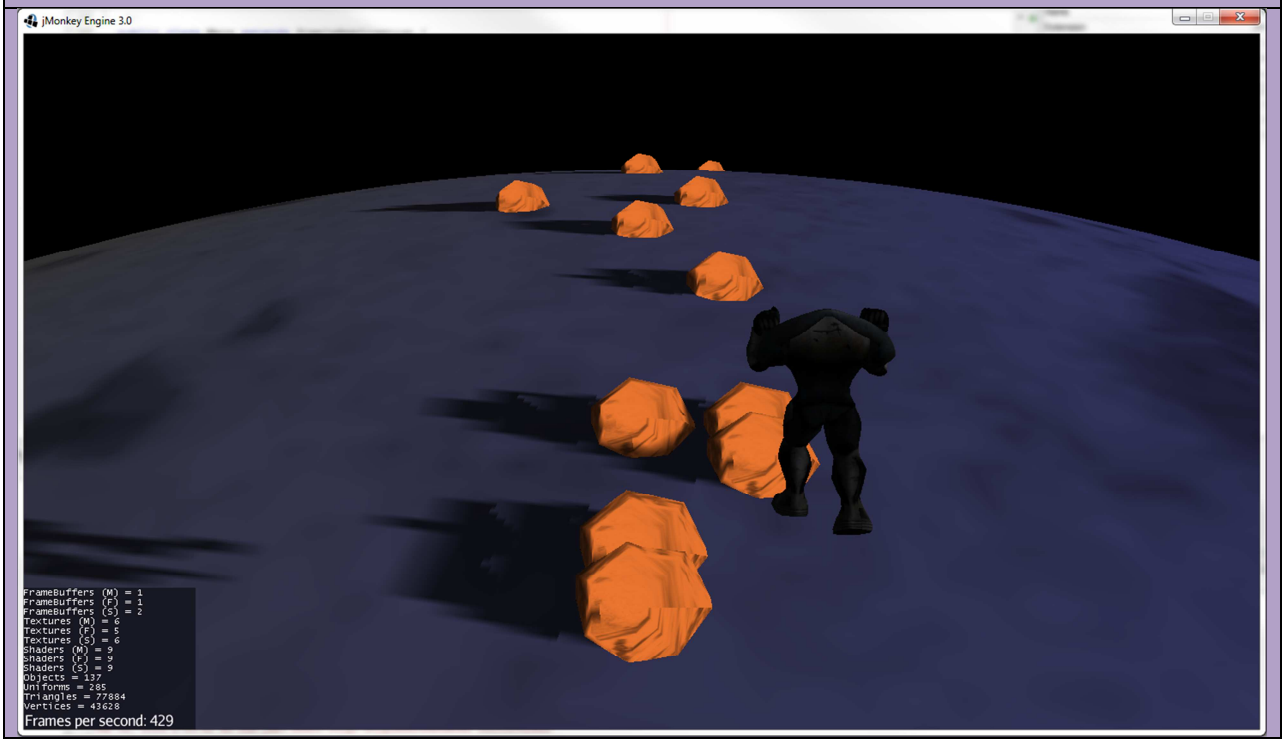
Assignment: Oto's Planet Run: An Animated Running Game on a Sphere



Topic:

Rotations, Animation, Controls, State Machines

Task:

Program a simple running game: Oto the robot is stranded on a fast rotating planet, and decided to spend the rest of his energy on running against the spinning direction, jumping over obstacles. To max the difficulty, he decided to limit himself to an area close to the equator. Interestingly, the obstacles also only appear in that equatorial strip.

Requirements for this assignment:

Everything that has to do with spatial-movement MUST be programmed in special Control-derived classes. The SimpleUpdate() method should be as empty as possible. This means:

You need at least classes for:

- Oto
- Obstacles
- WorldSphere

In these classes you MUST declare local Control-classes.

You are given:

The project “OtoAnimationTest” (see blackboard => content). This project is the perfect starting point. It contains a class Oto, which already has a local class OtoControl. Peruse that source code --- it will clarify a lot. It even contains a small state machine, which you only have to extend to make Oto jump. Please be aware that the animation is a control as well. However, you can set the transformation for the entire robot, using the node “otoNode” (see e.g. the STATE_JUMP handling in the source code).

Hints:

When you create your sphere, attach it to a Node which controls the sphere’s spinning. To the same node, you can attach the obstacles --- they will then just move along with the sphere’s surface.

- The sphere radius is set to 200 units in my example.
- Oto is scaled down by factor 0.5 (see initModel() in source code)
- My camera is initialized using the following parameters:
    - cam.setLocation(new Vector3f(0f, 15f, 15f));
    - cam.lookAt(new Vector3f(0, 5f, 0), Vector3f.UNIT_Y);
- The obstacles in the image are just spheres, initialized with a small number of axial and radial samples, hence they look like rocks:
    - Sphere sphere = new Sphere(5, 7, SPHERERADIUS); // SPHERERADIUS = 2.0f
- If you create you own project and add the Oto source code from my project, you need to add the jme-test-data library (project -> libraries -> add library …)
- Add explosions (blackboard content: SIngleBurstParticleEmitter) if you want to. But who wouldn’t want to.

Important:

We will talk about collision in class. Collision will be based on geometries, hence you don’t need to plan anything ahead --- JME will do the collision testing more or less by itself, with a little programming effort.