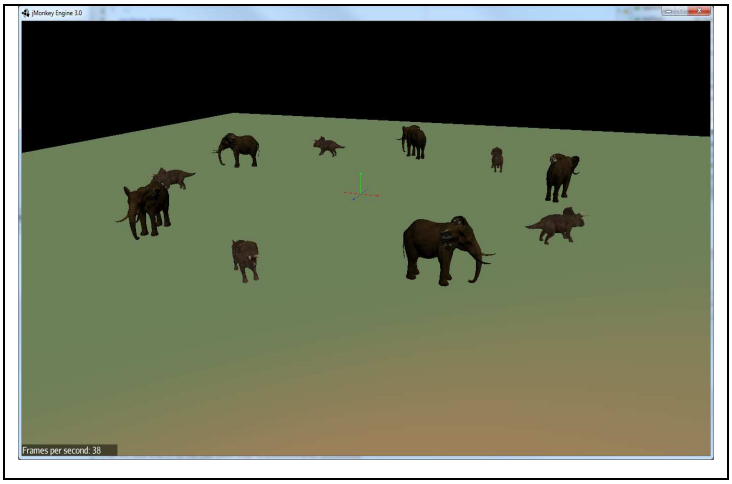
Assignment 2: Darwin's Cannon

Overview

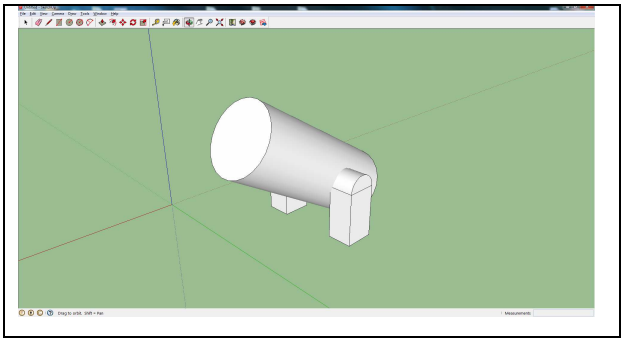
You will create an entire video game in this week. That's quite ambitious! I call this game Darwin's Cannon: as a convinced pacifist, I am happy that with JMonkey, we can generate virtual worlds where cannons perform as they rarely do in reality: in our world they bring progress. The game is simple: you create a (flat) world where dinosaurs roam around. The world also contains a cannon, which you can operate: pitch, yaw, and shoot. The cannonballs however do not bring death, but progress: whenever a dinosaur is hit, it turns into an elephant. When all the dinosaurs are 'progressed' to an elephant, the game ends, the time needed should relate to a score. If you sense a certain weirdness in this game: these are the only two animal models I had at hand. You can import more, if you find anything in the web. However, these two models are pretty:

What you see, is what I prepared for you in a JMonkey project, that's your starting point. There are two classes, Elephant and Dinosaur, extending (in a certain line of inheritance) Node, hence you can just add them to the scene.



What's missing is the cannon. It should resemble something like this:

This is a model just put together with google sketchup. While, eventually, we will use modeling software to create our models, for now I want you to generate the cannon in the game itself. It should consist of a cylinder, and some elements this cylinder is attached to (a stand). The entire cannon must be able to rotate, and to tilt.



This assignment is about multiple coordinate systems. You will need to build scene graph structures that contain the animals, the cannon, and a cannon ball, with all the local and global rotations and translations.

The cannon should react to the keyboard. You can lookup how to read keys in JMonkey in my keyboard example, as well as in the textbook, and online. I will handle it in class shortly (since it's event based programming).

In a later extension, the cannon input will come from accelerometer sensors in smartphones. Please already encapsulate the input accordingly.

Details:

Please break down the game into a sufficient number of useful classes. The cannon, the cannonball, the main game, the animals --- all these are actors in this game, which should be represented by separated classes.

In a real game, the game logic is often represented by so called state-machines. We will handle this in a later class. For now, just try to put it all together your way --- you might discover state machines yourself, or you might find another way.

The animation: the simpleUpdate(float tpf) loop is the starting point. Do NOT put all of the logic into this loop, but place method calls to update methods in your own classes into the simpleUpdate loop.

The game physics: we won't use JMonkey physics for this assignment, you will have to write your own physics. This only means, you have to move the cannon ball. I will talk about it in class.

Collision detection: you need to know when a dinosaur was hit. This is a hard problem. In this version, we make it simple: when the distance between the center of the cannonball and the center of a dinosaur is below a certain threshold, the dinosaur is hit. The Dinosaur (and Elephant) class provides a method 'getCenter()', which returns a point between the shoulders of the animal, in world coordinates.

I leave the game details to you. You can move the animals in a random fashion, or rotate them around the cannon like an old carousel, or whatever you want. Be creative.

Be creative!

You can add anything you want to. For example, if an elephant is hit, it can regress into a dinosaur. Or add another model.

This is still NO groupwork!

Start early.

Get acquainted with the demo programs, and the classes in the project I provided for this assignment.