# Lesson 9: Intro to Panda3D

3D Game Programming With C++

*Digital Media Academy (Summer 2011)*

**Written by:** Andrew Uzilov (andrew.uzilov@gmail.com)
Feel free to contact me with any questions.

This lesson will introduce a lot of concepts to you. **They will probably not make sense at first.** Don't worry about it yet – when you start writing Panda3D code in the next lesson, refer back to these concepts, and they will make more sense.

## Getting your bearings in a 3D world

Before you can start programming in 3D, you must learn some concepts about how 3D models are positioned in a 3D world. There are three things that completely define how a model will appear:

1) The model's X, Y, and Z coordinates. Those specify **where** the center point of the model is **located**. If you change the XYZ coords, you are **translating** (moving) the model.

2) The heading (H), pitch (P), and rotation (R) of the model. Those specify **how** the model is **oriented**. (Where is it facing? Is it tilted? Rotated?) If you change the HPR, the model does not move, but it rotates about one of its axes.

3) The scale. Simply put, what is the **size** of the model, **relative** to its original size?

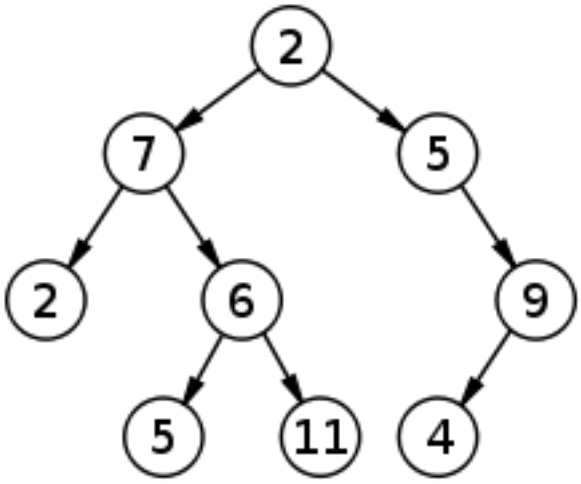
Here is a nice cheat sheet that graphically shows the above concepts (scroll down to **Positioning** section):
http://www.panda3d.org/manual/index.php/Cheat_Sheets

If that doesn't work, try this:
http://www.p3dp.com/doku.php?id=dotherwise:start

## Panda3D Scene Graph and `NodePath` Objects

Each objects in Panda3D's three-dimensional world is described by a **node**. These nodes are connected together to form the **scene graph**.

The word "graph" here means something different than what it usually means – here, "graph" is a bunch of nodes and how they are connected together. For example, here are 9 nodes (each has a number) connected together to form a graph:




*(Image from Wikipedia, http://en.wikipedia.org/wiki/Tree_%28data_structure%29 )*

The arrows connect **parent nodes** to **child nodes**. For example, node #4 is a child of node #9. Therefore, node #9 is the parent of node #4.

Every node has a parent, except for node #2. So, node #2 is the **root**. A node can have multiple (or no) children, but never more than one parent.

Why does this matter in Panda3D? Because **every object** (every 3D model, every camera, etc.) in your 3D game will be represented by a node, and those nodes will be connected to form the scene graph. Most importantly: **an object will NOT be rendered until it is connected to a parent node in the scene graph**. To make an object appear in the game, you have to give it a parent. This is called **reparenting** (parenting "again", because we can change parents anytime we want, many times).

## What does it mean to be a parent?

In Panda3D, a child always follows the parent around. That means:

- If the parent moves, its child node(s) move with it.
- It the parent rotates, its child node(s) rotate with it.
- If the parent is rescaled, so are the child nodes, by the same amount.

Why is this useful? Well, consider these situations:

- A shield that spins around the player because the player picked up a power-up.

- A satellite that rotates around a planet.

- A model of a person riding a skateboard. The person is a model, and the skateboard is a separate model. When the skateboard moves, the person must move with it (until the person falls off!).

In all of these cases, it is **much easier** to keep track of XYZ and HPR of an object (shield, satellite, or person) **relative** to another object (player, planet, or skateboard). So, if we make one object the child of another, they'll move together without any extra effort!

The root node in Panda3D is called **`render`**; we'll show you how to access it and attach nodes to it in the next lesson.

## How do we access Panda3D nodes?

The Panda3D library defines a special class called `NodePath` (don't worry about the fact that it says "path" – the way we'll use it, it will describe a single node). Most of the time, we will not be touching Panda3D nodes directly, but we'll modify them through a `NodePath` object. Think of a `NodePath` object as a "handle" to the actual Panda3D scene graph node – you use a handle to move, rotate, and do other things to a node, but you don't touch the node itself.

So, for each node in the scene graph, we will:

- Create a `NodePath` object that "wraps" the scene graph node.
- Attach that `NodePath` object to the scene graph (reparent it). Usually, we will reparent a `NodePath` object to `render` (the root of the scene graph).
- Use **that** `NodePath` object to move (translate), rotate, or scale the scene graph node.

## Take-home message

We didn't learn any code in this lesson, but we learned some important concepts. When you read and write Panda3D code in the next lesson, **refer back to these concepts so that you can understand the code**. Here is a summary:

- Every object in the game is a node.
- A node must be attached (reparented) to a node in the scene graph to get rendered.
- The root of the scene graph is a special node called `render`.
- Use `NodePath` objects as handles to the actual nodes in the scene graph.