

Writing MoSART Environments for Fun and Profit: Going to the Next Dimension with Direct-3D

by Chen-I Lim
rev. 1.0, April 12th 1998

1.0 Introduction

This document covers writing your own code to implement Direct-3D animation within your MoSART (Modeling, Simulation, Animation, and Real-time Control) Environment. This implementation is specific to the program framework that I have adopted in my own projects, for example in the **interactive MoSART helicopter environment** (we will refer to it as **Heli** here). The examples presented within this guide will refer explicitly to the code in that project.

2.0 Getting Started

This section will get you up to speed in obtaining the **Heli** code, and starting to make sense of it.

2.1 What do I need to know?

This primer assumes that you are familiar with using Windows '95 or NT. Experience with any kind of programming especially in C/C++, or better still, **Visual C++/MFC**, will help a great deal. It is helpful to be a little familiar with 3D animation in general, but not necessary.

2.2 How do I get the Heli Source Code?

The source code (together with the executable program) is available on the web at Dr. Rodriguez's homepage:

<http://www.eas.asu.edu/~aar/research/mosart/projects/projects.html>

or at my own homepage:

<http://www.public.asu.edu/~chenilim/projects>

Note that the **Heli** project is still under intense development, and new versions are continuously being uploaded. If you have obtained an older version from myself or from someone else, you should check the web again for the latest edition.

2.3 OK I got the code, now what do I do with it?

If you aren't familiar with using Visual C++ (version 5.0) you need to play around with it a little first. Load **heli32.dsw**, which is the project workspace file. The code is organized into quite a number of source files (I forget how many exactly). Don't be alarmed at the sheer size of the code! I know one guy has taken the trouble to print it out, and it almost ate through a whole ream of paper. You don't have to do this! There are only a few files that are relevant. The rest of them provide basic and framework functionality such as matrix-algebra, MATLAB interfacing, data arrays, visual indicators, and direct-3D visual objects. Anyway, the first thing you ought to do is to try compiling the code to make sure that it's all there and that it works as promised. If not, check that you've extracted all the files correctly into the proper directories, and that you are using version 5 or higher of Visual C++. If things still don't work, send me some email (cilim@asu.edu).

2.4 I want to know what's going on in the code, where do I start?

The following are the main files in the project:

1. **CHeliDoc.cpp** - Simulation engine
2. **CHeliView.cpp** - User Interface
3. **CVarWnd.cpp** - Real-time variables window
4. **C2DAnimWnd.cpp** - 2D animation window
5. **D3DAnimWnd** - Direct-3D animation window

The above are covered in a separate guide (Writing MoSART Environments for Fun and Profit: A Brief Primer). In this guide, we will focus on the following files:

1. **D3DAnimWnd.cpp** - Direct-3D animation window
2. **CD3DObject.cpp** - Direct 3D Object parent class
3. **CD3DObjHeli.cpp** - Direct 3D Helicopter Object
4. **CD3DObjLand.cpp** - Direct 3D Land (ground) Object

There is another class called **CD3DObjShadowHeli.cpp**, but don't worry about this right now. FYI, this is the object used for overlaying several simulations at once.

3.0 Going Through the Code

First, before you get bogged down in C++ code, run the program and see what it is doing. Unfortunately (at the time of writing this), there is still no online help in the program, nor are there and instructions or tutorials available. The program was (supposed to be) designed to be as intuitive as possible, but this may not always be the case (we only have the programmer to blame for this!). Anyway, start off by clicking on the right-most button on the toolbar to call up a 3D-animation window, and hit the **play** button to run the simulation and see the animation.

Notice that the animation consists of two main objects: the ground, and the helicopter (the animated rotor blade is part of the helicopter). The ground essentially remains fixed in space, while the helicopter is rotated and moved according to the simulation. In the code, each object is represented by a particular class derived from **D3DObject**. **D3DAnimWnd** contains class members for a set of these objects that it initialized and controls.

3.1 CreateScene(): Initializing Direct-3D objects

There is a **CreateScene()** function in both **D3DAnimWnd** and **D3DObject**. This is where the 3D objects are created and placed initially in the virtual world. **CreateScene()** in **D3DAnimWnd** calls the corresponding **CreateScene()** functions in each object. The two parameters passed are **d3drm** and **global**. **d3drm** is a pointer to the global master Direct-3D retained mode (hence 'rm') object which contains all the functionality for Direct-3D. There is only one of this and it's declared in **RMApp**. **global** is a pointer to a Direct-3D scene. A scene is essentially a particular coordinate reference that the subsequent objects attach to. Think of it as an anchor point that everything else is defined relative to. The benefit of having different frames is that all the objects within that frame can be easily translated or rotated simply by translating or rotating the anchor point or the scene. **D3DAnimWnd** has a master scene called **global** that all the other objects are attached to.

To understand **CreateScene()** in the objects, look at **CD3DObjLand** (it's much simpler). It does the following things:

1. Create a ground mesh
2. Create a ground mesh texture
3. Create a ground mesh wrap
4. Create and attach to a ground frame
5. Do some initial translations

These steps are pretty well delineated in the code, and they simply call the appropriate Direct-3D function to do the job. Note that step 2: loading a texture is not really necessary, because a texture file can be explicitly declared within the .X file itself.

CreateScene() in **D3DObjHeli** is pretty much the same thing, except that we have to load the blade mesh as well (it's defined in a separate .X file) and set a callback function to it. A callback function is just a function that gets called by the framework at (not quite) regular intervals, something like a built-in timer or idle function. This callback function (**RotBlade**) just rotates the blade a set number of degrees to produce the necessary animation.

3.2 SetDof(): Moving objects around

This function is defined in **CD3DObject**, the parent class (it means "Set Degrees of Freedom"). What it does is simply to translate and rotate the main frame (a frame is the same thing as a scene discussed above) according to the general 6 degrees of freedom for a 3D object. This is the main way in which animation is achieved by the real-time simulator. What happens is that the simulation code will call **SetDof()** in **D3DAnimWnd**, which simply calls **SetDof()** for **CD3DObjHeli**.

4.0 Making Your Mark

Ok, by now you are probably raring to get your hands dirty and roll some of your own code for your own Direct-3D Animation. Here's what you need to do:

1. Obtain an appropriate Direct3D mesh (.X) file
2. Derive your own **CD3DObject** class for that object
3. Modify the **CreateScene()** function in **D3DAnimWnd**
4. Modify **SetDof()** in **D3DAnimWnd** if necessary.

4.1 Obtain an appropriate Direct3D mesh (.X) file

Mesh files are available through the Internet, or from 3rd party vendors (i.e. free or not free). Richard Metzger (rpm@asu.edu) is the resident expert in Direct-3D and he can create Direct-3D meshes from scratch, so try talking to him first. The Direct-3D SDK comes with a utility to convert 3D-Studio files to the Direct-3D format.

4.2 Derive your own CD3DObject class for that object

The easiest way to do this is to copy **CD3DObjHeli**, change the name, and then modify the **CreateScene()** function to load your new mesh. Next you need to **#include** the header file in **D3DAnimWnd.h** and declare a new member function of your new object.

4.3 Modify the CreateScene() function in D3DAnimWnd

Just change the line that creates a **CD3DObjHeli** object to create your new object. Repeat steps 2-3 in order to add more objects to your project.

4.4 Modify SetDof() in D3DAnimWnd if necessary

Sometimes your system may require more than the basic 6 degrees of freedom. For example, you might be doing a 2-link pendulum that can be graphically modeled as 2 individual pieces. You might want to have two angles, theta and alpha to describe their positions. You might then modify **SetDof()** in **D3DAnimWnd** to accept theta and alpha as inputs, convert them to the necessary (x, y, z, yaw, pitch, roll) positions for each piece and then call **SetDof()** for each object.

5.0 Moving On

In this brief guide we have discussed how to implement your own Direct-3D animation using the framework provided in the helicopter code. Keep in mind that this is not the only way to do Direct-3D! This approach is the one I use in my projects, and thus is compatible with the simulation-engine, user-interface, and other components of the environment. There is no requirement, for example, to split different objects into different classes. However, I believe that this is the easiest and cleanest way to implement the animation module, because you can easily add, modify and change objects around. So while you are not forced to adhere to the structure I have adopted, I suggest that you at least maintain compatibility so that your new modules can be easily incorporated into new environments, new simulation engines, interfaces, etc. when they come about.

Please give me feedback on this document, as it is only in its first draft. I hope I've managed to cover the most important things here so that you can get started on writing your own code as quickly as possible. If anything is not clear or if there are any omissions, please let me know. You can email me at cilim@asu.edu.

Happy Coding!

6.0 References

Direct3D SDK from Microsoft. Check out the help files. This is available online (www.microsoft.com) or from the MoSART lab.

Trujillo, "Cutting Edge Direct3D", Coriolis books.