Multimedia Programming 2004

Assignments No. 6

26-1 2005

All assignments due: 10-2 2005

Goals of the assignments:

- Learn how to do basic debugging
- Learn how to design, implement, and execute a Direct3D program
- Learn basic 3D programming using DirectX: Direct3D Device, Vertices, Matrices, Lights, Textures, and Meshes

Preparations:

1. Download the code (code06a.zip (assignment 1) and code06b.zip (the rest of the assignments)) from the MMP2004 web-site and unzip it to a local directory
2. All further directories mentioned in the assignments can be found in this local directory
3. The code for the assignments is a slightly adapted version of the Direct3D tutorial that you can find in the *DirectX Sample Browser:* select in the <Show> dialog <C++> and <Tutorials> only. Now select the <Direct3D> tab in the browser window. Here you can find the tutorial documentation.

Posting Your Work: See the last page of Assignment Set 1 for the procedure for posting your work. NB Only assignment 8 has to be posted.

Assignment 1: Debugging

1. Use the debugger of MS Visual C++ to find and solve the errors in the projects that you find in the file code06a.zip.

Assignment 2: Direct3D Device

1. Browse to the *Direct3D_01_CreateDevice* directory. Compile and execute the *CreateDevice* code. Press 'r', 'g', and 'b' on the keyboard and see what happens. Read the documentation of Direct3D Tutorial 1.
2. Read the code and the comments carefully. Start as always in the *WinMain()* function. Note that the program uses two global variables *g_pD3D* (used to create the Direct3DDevice) and *g_pd3dDevice* (our rendering device). *WinMain()* is the application's entry point. In *WinMain* the function *InitD3D()* is called, this function initializes Direct3D. *MsgProc()* is the window's message handler, upon receipt of some messages the functions *Render()*, and *Render(int r, int g, int b)* are called. These functions draw the scene. Finally, the function *Cleanup()* releases all previously initialized objects.
3. Change the code such that clear color used in the program changes based on the location of the mouse pointer in the window. You may think of a simple coloring scheme of your own. For example: if the mouse pointer is in the upper left corner then the window is red; if it is in the upper right corner of the window it is green; if it is in the lower right corner

of the window, the window will become blue; whereas the colors change continuously on locations in between.

Assignment 3: Vertices

1. Browse to the *Direct3D_02_Vertices* directory. Compile and execute the *Vertices* code. Read the documentation of Direct3D Tutorial 2.
2. The program is basically the program from Assignment 1. But now, in a new procedure *InitVB(),* a vertex buffer is filled with the 3 vertices of a triangle. These vertices are drawn at the previously indicated position in the *Render()* code (see code of Assignment 1). For the vertex buffer an extra global variable is declared *g_pVB*. Of course this object has to be released in the *CleanUp()* function.
3. Change the code such that a second smaller triangle is drawn inside the existing triangle.

Assignment 4: Matrices

1. Browse to the *Direct3D_03_Matrices_Paused* directory. Compile and execute the *Matrices* code. Resize the application window. You will notice that the triangle rotates.
2. Close the previous Matrices code and browse to the *Direct3D_03_Matrices directory*. Compile and execute the *Matrices* code. You will notice that the triangle rotates by itself. This is because the *Render()* function is now called in the message loop itself, while previously it was only called upon receipt of a WM_PAINT message. It complicates the message loop a little bit, but the code is basically equal to the code of Assignment 2.
3. For the rotation of the triangle (our geometry) several matrices are used in the *SetupMatrices()* function. We have two static matrices: the view matrix in which the view point and orientation of the viewer is specified; and a projection matrix that defines the perspective of the scene: it defines how our geometry looks smaller in the distance and bigger near by. The third matrix is dynamic and defines the rotation of the 3D world in which our geometry is drawn. The matrix changes each time the *SetupMatrices()* function is called. The change is based on the *current time*. It rotates the 3D world every second.
4. Read the documentation of Direct3D Tutorial 3.
5. Change the code such that the triangle rotates two times slower.
6. Change the code such that if you press ‘x’, ‘y’, ‘z’, respectively, the triangle rotates along the x-. y-, z-axis respectively.
7. Optional: play with different parameters for the view point and perspective matrices.

Assignment 5: Lights

1. Browse to the *Direct3D_04_Lights* directory. Compile and execute the *Lights* code.
2. Read the documentation of Direct3D Tutorial 4.
3. Change the code such that you can move the light to the left and right side by pressing the ‘l’ and ‘r’ key on the keyboard, respectively.
4. Change the code such that by pressing other keys the user can influence the color of the light, and the color of the material.

Assignment 6: Textures

1. Browse to the *Direct3D_05_Textures* directory. Compile and execute the *Textures* code.
2. Read the documentation of Direct3D Tutorial 5.

3. Change the code such that the user can switch between two different kind of textures by pressing the ‘t’ key on the keyboard.

Assignment 7: Meshes

1. Browse to the *Direct3D_06_Meshes* directory. Compile and execute the *Meshes* code.
2. Read the documentation of Direct3D Tutorial 6.
3. Double click the *ReadMeTigerMesh.txt* file to get an impression of the way a mesh and the texture mapping is defined. Of course these mesh files are not made by hand but rather generated by a 3D design program.

Assignment 8: Fireworks

1. Browse to the *Fireworks* directory. Compile and execute the *Fireworks* code.
2. The setup of the program is a little bit different than what we saw in the previous assignments. Here there is an application class *CMyD3DApplication* that has different methods for initializing the window, handling the message loop, and initializing the *Direct3D* device, etc. In *WinMain* a simple function/method call *d3dApp.Run()* starts everything.
3. The important drawing/rendering takes place in the methods of the class *CParticleSystem: Render()* and *Update()*. Study these two methods to get some insights in the geometry and physics of the particles that constitute the *fireworks*.
4. This last assignment of Multimedia Programming is an open assignment: change the code such that you get the most impressive fireworks. (Hints: you can change the dynamics of the colors, the particles, etc…)