

ECE 595 / CS 491 / CS 591
**Real-Time Rendering &
Graphics Hardware**

Pradeep Sen
Advanced Graphics Lab

Class 8
February 12, 2007

Announcements

- Intro project due tonight by midnight
- This includes source code and report
- I'll be giving the Sigma Xi talk on "Dual Photography" Thursday Feb 15 at 5pm, UNM Conference Center Room G

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

Last time

- Vertex and Fragment programs

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

Today

- DirectX
- High-level shading languages

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

DirectX

- Really a collection of APIs for different tasks, in the form of COM-compliant objects:
 - DirectInput (to handle IO from input devices such as joysticks, keyboard, etc)
 - DirectPlay (to handle networking between PCs)
 - DirectSound (for playing/recording of sound fx)
 - DirectMusic (for playing audio soundtracks)
 - DirectXMedia (for animation and multimedia streaming apps)
 - DirectDraw (for 2-D drawing to the screen)
 - Direct3D (for 3-D rendering)

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

History of DirectX

- In 1992, a company was founded called RenderMorphics
- They created a 3D API called RealityLab
- In 1995, Microsoft bought RenderMorphics and soon produced the 3D API called DirectX 2.0 for Windows95.
- DirectX 2.0 released June 1996
- DirectX 3.0 released September 1996
- People began to use it...

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

Carmack .plan (December 1996)

"I have been using OpenGL for about six months now, and I have been very impressed by the design of the API, and especially it's ease of use. A month ago, I ported quake to OpenGL. It was an extremely pleasant experience. It didn't take long, the code was clean and simple, and it gave me a great testbed to rapidly try out new research ideas."

"I started porting glquake to Direct-3D IM with the intent of learning the api and doing a fair comparison. Well, I have learned enough about it. I'm not going to finish the port. I have better things to do with my time."

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

Carmack .plan (December 1996)

"Direct-3D IM is a horribly broken API. It inflicts great pain and suffering on the programmers using it, without returning any significant advantages. I don't think there is ANY market segment that D3D is appropriate for. OpenGL seems to work just fine for everything from quake to softimage. There is no good technical reason for the existence of D3D."

"I'm sure D3D will suck less with each forthcoming version, but this is an opportunity to just bypass dragging the entire development community through the messy evolution of an ill-birthed API."

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

Carmack .plan (December 1996)

"The overriding reason why GL is so much better than D3D has to do with ease of use. GL is easy to use and fun to experiment with. D3D is not (ahem). You can make sample GL programs with a single page of code. I think D3D has managed to make the worst possible interface choice at every opportunity."

AGL Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

Carmack .plan (December 1996)

"GL's interface is procedural: You perform operations by calling gl functions to pass vertex data and specify primitives."

```
glBegin(GL_TRIANGLES);
glVertex(0,0,0);
glVertex(1,1,0);
glVertex(2,0,0);
glEnd();
```

D3D's interface is by execute buffers: You build a structure containing vertex data and commands, and pass the entire thing with a single call.

Carmack .plan (December 1996)

"D3D's interface is by execute buffers: You build a structure containing vertex data and commands, and pass the entire thing with a single call. On the surface, this appears to be an efficiency improvement for D3D, because it gets rid of a lot of procedure call overhead. In reality, it is a gigantic pain-in-the-ass."

Carmack .plan (December 1996)

```
v = &buffer.vertices[0];
v->x = 0; v->y = 0; v->z = 0; v++;
v->x = 1; v->y = 1; v->z = 0; v++;
v->x = 2; v->y = 0; v->z = 0;
c = &buffer.commands;
c->operation = DRAW_TRIANGLE;
c->vertices[0] = 0;
c->vertices[1] = 1;
c->vertices[2] = 2;
IssueExecuteBuffer (buffer);
```

"If I included the complete code to actually lock, build, and issue an execute buffer here, you would think I was choosing some pathologically slanted case to make D3D look bad."

History of DirectX

- DirectX 4 was never released (back to the drawing board)
- DirectX 5 was released July 1997
- Latest version is DirectX 10, released Nov 2006 only on Windows Vista

Summary

- DirectX has improved considerably since its modest beginnings
- Many developers swear by it

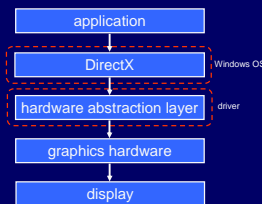
Differences between DirectX and OpenGL

- At a high-level, OpenGL is much easier to get something to draw to the screen.
- DirectX is intended for high-performance applications, so drawing a single triangle is complicated.

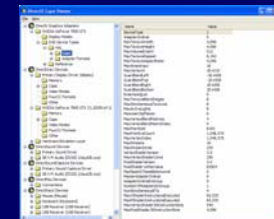
Differences between DirectX and OpenGL

- DirectX is natively left-handed (OpenGL is right-handed)
- Projection matrix maps z-depths to 0 to 1 range (-1 to 1 in OpenGL)
- Row major matrix representation (column major in OpenGL)
- etc.

Overview of DirectX



Viewing device capabilities



Taking a look at some code...

- Acquiring a Direct3D interface handle:


```
IDirect3D9 *md3dObject;
md3dObject = Direct3DCreate9(D3D_SDK_VERSION);
```
- Creating the Direct3D object:


```
IDirect3DDevice9 *gd3dDevice = 0;
md3dObject->CreateDevice(
    D3DADAPTER_DEFAULT, // primary adapter
    D3DDEVTYPE_HAL, // device type
    mhMainWnd, // window
    devBehaviorFlags, // vertex processing
    &md3dpp, // present params
    &gd3dDevice); // return created device
```

Taking a look at some code...

- D3DPRESENT_PARAMETERS:


```
typedef struct D3DPRESENT_PARAMETERS {
    UINT BackBufferWidth;
    UINT BackBufferHeight;
    D3DFORMAT BackBufferFormat;
    UINT BackBufferCount;
    D3DMULTISAMPLE_TYPE MultiSampleType;
    DWORD MultiSampleQuality;
    D3DSWAPEFFECT SwapEffect;
    HWND hDeviceWindow;
    BOOL Windowed;
    BOOL EnableAutoDepthStencil;
    D3DFORMAT AutoDepthStencilFormat;
    DWORD Flags;
    UINT FullScreen_RefreshRateInHz;
    UINT PresentationInterval;
} D3DPRESENT_PARAMETERS;
```

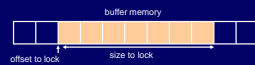
Taking a look at some code...

- Sample settings


```
D3DPRESENT_PARAMETERS d3dpp;
d3dpp.BackBufferWidth = 800;
d3dpp.BackBufferHeight = 600;
d3dpp.BackBufferFormat = D3DFMT_X8R8G8B8;
d3dpp.BackBufferCount = 1;
d3dpp.MultiSampleType = D3DMULTISAMPLE_NONE;
d3dpp.MultiSampleQuality = 0;
d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
d3dpp.hDeviceWindow = hwnd;
d3dpp.Windowed = true;
d3dpp.EnableAutoDepthStencil = true;
d3dpp.AutoDepthStencilFormat = D3DFMT_D24S8;
d3dpp.Flags = 0;
d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
d3dpp.PresentationInterval = D3DPRESENT_INTERVAL_IMMEDIATE;
```

Vertex and index buffers

- Vertex buffer – a chunk of contiguous memory that contains the vertex data (represented by the IDirect3DVertexBuffer9 interface)
- Index buffer – a chunk of contiguous memory that contains the index data (represented by the IDirect3DIndexBuffer9 interface)
- Access this memory through Lock/Unlock methods



Taking a look at some code...

- Creating a vertex buffer


```
gd3dDevice->CreateVertexBuffer(
    8 * sizeof(VertexPos), // num bytes
    D3DUSAGE_WRITEONLY, // dynamic/static
    0, // vertex format
    D3DPOOL_MANAGED, // memory pool
    &nVB, // handle to vb
    0); // n/a
```

Simple example

- Program to draw two shapes


```
// create two vertex buffers, one for each shape
gd3dDevice->CreateVertexBuffer(sizeof(shape),
    D3DUSAGE_WRITEONLY, 0, D3DPOOL_MANAGED, &pShape1VB, NULL);
gd3dDevice->CreateVertexBuffer(sizeof(shape),
    D3DUSAGE_WRITEONLY, 0, D3DPOOL_MANAGED, &pShape2VB, NULL);

// pass down vertex information for both
pShape1VB->Lock(0, sizeof(pData), (void**)&Data, 0); //lock
memcpy(pData, shape1, sizeof(shape)); //copy data to buffer
pShape1VB->Unlock();

pShape2VB->Lock(0, sizeof(pData), (void**)&Data, 0); //lock
memcpy(pData, shape2, sizeof(shape)); //copy data to buffer
pShape2VB->Unlock();
```

Simple example

- Now onto the drawing loop


```
gd3dDevice->Clear(0, NULL, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER,
    D3DCOLOR_XRGB(0,0,0), 1.0f, 0);
gd3dDevice->BeginScene(); // start drawing

// draw 1st shape
gd3dDevice->SetStreamSource(0, pShape1VB, 0, sizeof(D3DVERTEX));
gd3dDevice->DrawPrimitive(D3DPT_TRIANGLELIST, 0, 1);

// draw 2nd shape
gd3dDevice->SetStreamSource(0, pShape2VB, 0, sizeof(D3DVERTEX));
gd3dDevice->DrawPrimitive(D3DPT_TRIANGLELIST, 0, 1);

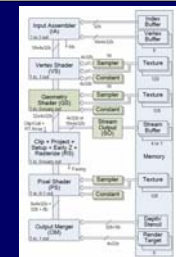
gd3dDevice->EndScene(); // stop drawing
gd3dDevice->Present(NULL, NULL, NULL, NULL); // swap buffers
```

Some things are easier than OpenGL

- Creating and using textures


```
D3DXCreateTextureFromFile(g_App.GetDevice(), "texture.png",
    &pMyTexture);
g_App.GetDevice()->SetTexture(0, pMyTexture);
```

Direct3D 10 rendering pipeline



From Bythe, SIGGRAPH 2006

High-level shading languages

```
ARB_fragment_program
DP3 R0, c[11].xyzk, c[11].xyzk;
RSQ R0, R0.x;
MUL R0, R0.w, c[11].xyzk;
MOV R1, c[3];
MUL R1, R1.x, c[0].xyzk;
DP3 R2, R1.xyzk, R1.xyzk;
RSQ R2, R2.x;
MUL R1, R2.w, R1.xyzk;
ADD R2, R0.xyzk, R1.xyzk;
DP3 R3, R2.xyzk, R2.xyzk;
RSQ R3, R3.x;
MUL R2, R3.w, R2.xyzk;
DP3 R2, R1.xyzk, R2.xyzk;
MAX R2, c[3].z, R2.x;
MOV R2.z, c[3].y;
MOV R2.w, c[3].y;
LIT R2, R2;
...
```

Cg

```
float3 cSpecular = pow(max(0,dot(Nf, H)), phongExp).xxx;
float3 cPlastic = Cd * (cAmbient + cDiffuse) +
                Cs * cSpecular;
```



Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

High-level shading languages

- Real-time Shading Language (Stanford, 2001)
- Cg (NVIDIA, 2003)
- HLSL (Microsoft, 2003)
 - very similar to Cg
- GLSL (3DLabs, 2003)
 - Included in OpenGL2.0 spec



Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007

Reading

- Paper on Direct3D 10 architecture by David Blythe
- Paper on RTSL by Proudfoot et al.
- Paper on Cg by Bill Mark



Real-time Rendering & Graphics Hardware
Pradeep Sen

Class 8 - February 12, 2007