

Re: In window mode, pD3DDevice->SetViewport?

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.graphics/2004-07/010>

From: Hanna-Barbera (*hb_at_nulllink.com*)

Date: 07/05/04

Date: Mon, 5 Jul 2004 13:47:52 -0700

> > *You mean the parameters in D3DPRESENT_PARAMETERS?*
>
> *Yup.*

That doesn't work. For some reason, the back buffer is always stretchblit to the front buffer so I see the entire back buffer, even parts I haven't rendered to.

Resetting D3d with pD3DDevice->Reset(&d3dpp);
gives a D3DERR_INVALIDCALL.

If I get rid of the index buffer and vertex buffers in my code and just not render anything, then Reset works fine.
There is some bug in my code or I have no idea what's the problem.

This is the test code I have (It should compile as is):

```
#include <windows.h> // Header File For Windows
#include <d3dx9.h> // Header File For Direct3D
#include <d3d9.h> // Header File For Direct3D

#define DEGREES_TO_RADIANS 0.017453292519943295769236907684886

#define USEHAL

HDC hDC = NULL; // Private GDI Device Context
HWND hWnd = NULL; // Holds Our Window Handle
HINSTANCE hInstance; // Holds The Instance Of The Application
IDirect3D9 *pD3D = NULL; // Direct3D Version 9
IDirect3DDevice9 *pD3DDevice = NULL; // Direct3D Rendering Device

bool keys[256]; // Array Used For The Keyboard Routine
bool active=true; // Window Active Flag Set To TRUE By Default
```

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
int WindowWidth=300, WindowHeight=300;
```

```
D3DPRESENT_PARAMETERS d3dpp;
```

```
//NEW//////////////////NEW//////////////////NEW//////////////////NEW////////  
////////
```

```
//#define D3DFVF_CUSTOMVERTEX (D3DFVF_XYZ | D3DFVF_DIFFUSE | D3DFVF_TEX2) //  
Flexible vertex format  
#define D3DFVF_CUSTOMVERTEX (D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_DIFFUSE |  
D3DFVF_TEX2) // Flexible vertex format  
#define D3DFVF_CUSTOMVERTEX2 (D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_TEX2) //  
Flexible vertex format
```

```
IDirect3DVertexBuffer9 *pVertexBuffer=NULL; // Our vertex buffer  
IDirect3DIndexBuffer9 *pIndexBuffer2=NULL;  
IDirect3DVertexBuffer9 *pVertexBuffer2=NULL; // Our vertex buffer
```

```
IDirect3DTexture9 *pBaseTexture;
```

```
D3DXMATRIXA16 IdentityMatrix;
```

```
char ErrorMessage[512];
```

```
float ModelTransformMatrix[16];  
D3DXMATRIXA16 D3DModelTransformMatrix;
```

```
struct TVertex // Our vertex struct  
{  
float x, y, z; // Vertex position  
float nx, ny, nz; //Normal  
DWORD color; // Color value in 0xAARRGGBB Format  
float s, t;  
};
```

```
struct TVertex2 // Our vertex struct  
{  
float x, y, z; // Vertex position  
float nx, ny, nz; //Normal  
float s, t;  
};
```

```
//2 triangles, render as strip  
TVertex triangle[4]={ // Vertex array that holds our triangle  
{ 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 1.0, 1.0},  
{ 1.0, -1.0, 1.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 1.0, 0.0},  
{ -1.0, -1.0, 1.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 0.0, 0.0},  
{ -1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 0.0, 1.0}  
};
```

```
unsigned short indexfullscreentriangle[6]={0, 1, 2, 2, 1, 3};
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
TVertex fullscreentriangle[4]={ // Vertex array that holds our triangle
{ 1.0, -1.0, 0.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 1.0, 0.0},
{ 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 1.0, 1.0},
{ -1.0, -1.0, 0.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 0.0, 0.0},
{ -1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0xFFFFFFFF, 0.0, 1.0}
};

D3DLIGHT9 Light;
D3DMATERIAL9 Material;
D3DVIEWPORT9 Viewport;
//NEW//////////NEW//////////NEW//////////NEW////////
////////

void KillD3DScene();
int InitD3D();
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM); // Declaration For
WndProc

//////////
//////////
// THE RESIZE D3D SCENE
//////////
//////////

//About viewport :
//If I set D3DPRESENT_PARAMETERS backbuffer width and height to a large
value,
//then the front buffer seems stretchblit to the window. The right and
bottom have
//junk.
//If I just leave them static and call SetViewport, the viewport flashes and
such, in REF
//it crashes.
//The proper thing to do is pD3DDevice->Reset(&d3dpp); and reuploading the
stuff, but
//I get a D3DERR_INVALIDCALL, and I'm not sure what the source problem is.
//
void ReSizeD3DScene(int width, int height) // Resize And Initialize The
D3D Window
{
HRESULT result;
width=(width<=0) ? 1 : width;
height=(height<=0) ? 1 : height;

if((width!=WindowWidth)||height!=WindowHeight)
{
WindowWidth=width;
WindowHeight=height;
d3dpp.BackBufferWidth=WindowWidth;
d3dpp.BackBufferHeight=WindowHeight;
d3dpp.BackBufferFormat=D3DFMT_X8R8G8B8;//D3DFMT_R5G6B5
}
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
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_D16;

d3dpp.Flags=D3DPRESENTFLAG_DISCARD_DEPTHSTENCIL;//D3DPRESENTFLAG_DISCARD_DEP
THSTENCIL | D3DPRESENTFLAG_DEVICECLIP;
d3dpp.FullScreen_RefreshRateInHz=D3DPRESENT_RATE_DEFAULT;

d3dpp.PresentationInterval=D3DPRESENT_INTERVAL_ONE;//D3DPRESENT_INTERVAL_DEF
AULT;
//Reset D3D device
KillD3DScene();
result=pD3DDevice->Reset(&d3dpp);
if(result==D3DERR_DEVICELOST)
    MessageBox(NULL, "D3DERR_DEVICELOST", "Error", MB_OK);
else if(result==D3DERR_DRIVERINTERNALERROR)
    MessageBox(NULL, "D3DERR_DRIVERINTERNALERROR", "Error", MB_OK);
else if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);
else if(result==D3DERR_OUTOFVIDEOMEMORY)
    MessageBox(NULL, "D3DERR_OUTOFVIDEOMEMORY", "Error", MB_OK);
else if(result==E_OUTOFMEMORY)
    MessageBox(NULL, "E_OUTOFMEMORY", "Error", MB_OK);

////////////////////////////////////
IDirect3DSurface9 *pBackBuffer;
    pD3DDevice->GetBackBuffer(0, 0, D3DBACKBUFFER_TYPE_MONO, &pBackBuffer);
D3DSURFACE_DESC m_d3dsdBackBuffer;
    pBackBuffer->GetDesc(&m_d3dsdBackBuffer);
    pBackBuffer->Release();
////////////////////////////////////

InitD3D();
}

Viewport.X=0;
Viewport.Y=0;
Viewport.Width=width;
Viewport.Height=height;
//Viewport.Width=1024;
//Viewport.Height=1024;
Viewport.MinZ=0.0;
Viewport.MaxZ=1.0;
//result=pD3DDevice->SetViewport(&Viewport);
//if(result!=D3D_OK)
//{
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
// result=result;
//}

//Alignment is accomplished using __declspec(align(16))
//so that new instructions can be used (SSE, 3DNow!)
D3DXMATRIXA16 matProjection;
//D3DXMatrixPerspectiveFovLH(&matProjection, 45.0,
(float)width/(float)height, 0.1, 1000.0);
D3DXMatrixPerspectiveFovLH(&matProjection, 45.0,
(float)width/(float)height, 0.1, 1000.0);

//D3D Transform State
pD3DDevice->SetTransform(D3DTS_PROJECTION, &matProjection);
}

////////////////////////////////////
////////////////////////////////////
// THE DIRECT3D INIT
////////////////////////////////////
////////////////////////////////////
int InitD3D()
{

////////////////////////////////////
////////////////////////////////////
long ReturnVal;

pD3DDevice->SetRenderState(D3DRS_CULLMODE, D3DCULL_NONE);
//pD3DDevice->SetRenderState(D3DRS_CULLMODE, D3DCULL_CW);
pD3DDevice->SetRenderState(D3DRS_LIGHTING, FALSE);
pD3DDevice->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_GOURAUD);
pD3DDevice->SetRenderState(D3DRS_FILLMODE, D3DFILL_SOLID);

pD3DDevice->SetRenderState(D3DRS_ZENABLE, D3DZB_TRUE);
pD3DDevice->SetRenderState(D3DRS_ZWRITEENABLE, TRUE);
pD3DDevice->SetRenderState(D3DRS_ZFUNC, D3DCMP_LESSEQUAL);

pD3DDevice->SetRenderState(D3DRS_ALPHATESTENABLE, FALSE);

pD3DDevice->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_SRCALPHA);
pD3DDevice->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_INVSRCALPHA);
pD3DDevice->SetRenderState(D3DRS_BLENDOP, D3DBLENDOP_ADD);

pD3DDevice->SetRenderState(D3DRS_SPECULARENABLE, TRUE);
pD3DDevice->SetRenderState(D3DRS_COLORVERTEX, FALSE);

D3DXMatrixIdentity(&IdentityMatrix);

memset(&Light, 0, sizeof(D3DLIGHT9));
Light.Type=D3DLIGHT_DIRECTIONAL;//D3DLIGHT_POINT;
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
Light.Ambient.r=0.0;
Light.Ambient.g=0.0;
Light.Ambient.b=0.0;
Light.Ambient.a=0.0;
Light.Diffuse.r=1.0;
Light.Diffuse.g=1.0;
Light.Diffuse.b=1.0;
Light.Diffuse.a=1.0;
Light.Specular.r=1.0;
Light.Specular.g=1.0;
Light.Specular.b=1.0;
Light.Specular.a=1.0;
Light.Position.x=0.0;
Light.Position.y=0.0;
Light.Position.z=0.0;
Light.Direction.x=0.0;
Light.Direction.y=0.0;
Light.Direction.z=1.0;
```

```
Light.Falloff=1.0;
Light.Attenuation0=1.0;
Light.Attenuation1=0.0;
Light.Attenuation2=0.0;
```

```
//Execute D3D functions for light
int LightCounter=0;
pD3DDevice->SetLight(LightCounter, &Light);
pD3DDevice->SetRenderState(D3DRS_LIGHTING, TRUE);
```

```
Material.Ambient.r=1.0;
Material.Ambient.g=1.0;
Material.Ambient.b=1.0;
Material.Ambient.a=1.0;
Material.Diffuse.r=1.0;
Material.Diffuse.g=1.0;
Material.Diffuse.b=1.0;
Material.Diffuse.a=1.0;
Material.Emissive.r=1.0;
Material.Emissive.g=0.0;
Material.Emissive.b=0.0;
Material.Emissive.a=0.0;
Material.Specular.r=1.0;
Material.Specular.g=1.0;
Material.Specular.b=1.0;
Material.Specular.a=1.0;
Material.Power=100.0; //Should be 0.0 to 128.0
pD3DDevice->SetMaterial(&Material);
```

```
//NEW////////////////////////////////NEW////////////////////////////////NEW////////////////////////////////NEW////////
////////
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
HRESULT result;
unsigned char *pVertices;
unsigned char *pindicesPointer;

/*result=pD3DDevice->CreateVertexBuffer(4*sizeof(TVertex),
D3DUSAGE_WRITEONLY, D3DFVF_CUSTOMVERTEX,
        D3DPOOL_DEFAULT, &pVertexBuffer, NULL);
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);
else if(result==D3DERR_OUTOFVIDEOMEMORY)
    MessageBox(NULL, "D3DERR_OUTOFVIDEOMEMORY", "Error", MB_OK);
else if(result==E_OUTOFMEMORY)
    MessageBox(NULL, "E_OUTOFMEMORY", "Error", MB_OK);

result=pVertexBuffer->Lock(0, 0, (void **)&pVertices, 0);
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);

memcpy(pVertices, triangle, sizeof(triangle));
result=pVertexBuffer->Unlock();
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);
////////////////////////////////////*/

result=pD3DDevice->CreateIndexBuffer(4*sizeof(unsigned short),
D3DUSAGE_WRITEONLY, D3DFMT_INDEX16,
        D3DPOOL_DEFAULT, &pIndexBuffer2, NULL);
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);
else if(result==D3DERR_OUTOFVIDEOMEMORY)
    MessageBox(NULL, "D3DERR_OUTOFVIDEOMEMORY", "Error", MB_OK);
else if(result==E_OUTOFMEMORY)
    MessageBox(NULL, "E_OUTOFMEMORY", "Error", MB_OK);

result=pIndexBuffer2->Lock(0, 0, (void **)&pindicesPointer, 0);
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);

memcpy(pindicesPointer, indexfullscreentriangle,
sizeof(indexfullscreentriangle));

result=pIndexBuffer2->Unlock();
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);

/*result=pD3DDevice->CreateVertexBuffer(4*sizeof(TVertex),
D3DUSAGE_WRITEONLY, D3DFVF_CUSTOMVERTEX,
        D3DPOOL_DEFAULT, &pVertexBuffer2, NULL);
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);
else if(result==D3DERR_OUTOFVIDEOMEMORY)
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
    MessageBox(NULL, "D3DERR_OUTOFVIDEOMEMORY", "Error", MB_OK);
else if(result==E_OUTOFMEMORY)
    MessageBox(NULL, "E_OUTOFMEMORY", "Error", MB_OK);

result=pVertexBuffer2->Lock(0, 0, (void **)&pVertices, 0);
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);

memcpy(pVertices, fullscreentriangle, sizeof(fullscreentriangle));
result=pVertexBuffer2->Unlock();
if(result==D3DERR_INVALIDCALL)
    MessageBox(NULL, "D3DERR_INVALIDCALL", "Error", MB_OK);
*/

return TRUE;
}

/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////
// THE DIRECT3D RENDER
/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////
int RenderScene()
{
    HRESULT result=D3D_OK;
    result=pD3DDevice->Clear(0, NULL, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER,
        D3DCOLOR_COLORVALUE(0.0, 0.0, 0.0, 0.0), 1.0, 0);

    Light.Type=D3DLIGHT_POINT;
    Light.Ambient.r=0.0;
    Light.Ambient.g=0.0;
    Light.Ambient.b=0.0;
    Light.Ambient.a=0.0;
    Light.Diffuse.r=1.0;
    Light.Diffuse.g=1.0;
    Light.Diffuse.b=1.0;
    Light.Diffuse.a=1.0;
    Light.Specular.r=1.0;
    Light.Specular.g=1.0;
    Light.Specular.b=1.0;
    Light.Specular.a=1.0;
    Light.Position.x=0.0;
    Light.Position.y=0.0;
    Light.Position.z=10.0;
    Light.Direction.x=0.0;
    Light.Direction.y=0.0;
    Light.Direction.z=-10.0;

    Light.Falloff=1.0;
    Light.Range=1000.0;
    Light.Attenuation0=1.0;
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
Light.Attenuation1=0.0;
Light.Attenuation2=0.0;
result=pD3DDevice->LightEnable(0, TRUE);
result=pD3DDevice->SetRenderState( D3DRS_AMBIENT, 0 );

Material.Ambient.r=0.0;
Material.Ambient.g=0.0;
Material.Ambient.b=0.0;
Material.Ambient.a=0.0;
Material.Diffuse.r=1.0;
Material.Diffuse.g=1.0;
Material.Diffuse.b=1.0;
Material.Diffuse.a=0.5;
Material.Emissive.r=0.0;
Material.Emissive.g=0.0;
Material.Emissive.b=0.0;
Material.Emissive.a=0.0;
Material.Specular.r=1.0;
Material.Specular.g=1.0;
Material.Specular.b=1.0;
Material.Specular.a=0.0;
Material.Power=0.0; //Should be 0.0 to 128.0
result=pD3DDevice->SetMaterial(&Material);

//Execute D3D functions for light
pD3DDevice->SetLight(0, &Light);
//NEW//////////NEW//////////NEW//////////NEW////////
////////
/* D3DXMATRIXA16 ModelviewMatrix;

D3DXMATRIXA16 projectionMatrix;
pD3DDevice->GetTransform(D3DTS_PROJECTION, &projectionMatrix);
D3DXMATRIXA16 currentModelviewMatrix;
pD3DDevice->GetTransform(D3DTS_WORLD, &currentModelviewMatrix);

pD3DDevice->BeginScene();

//Render fullscreen quad
//Set projection and modelview to identity
pD3DDevice->SetTransform(D3DTS_PROJECTION, &IdentityMatrix);
pD3DDevice->SetTransform(D3DTS_WORLD, &IdentityMatrix);
//Turn off depth testing and writing
pD3DDevice->SetRenderState(D3DRS_ZENABLE, FALSE);
pD3DDevice->SetRenderState(D3DRS_ZWRITEENABLE, FALSE);
pD3DDevice->SetFVF(D3DFVF_CUSTOMVERTEX);
pD3DDevice->SetStreamSource(0, pVertexBuffer2, 0, sizeof(TVertex));
//pD3DDevice->SetIndices(pIndexBuffer2);
//result=pD3DDevice->DrawIndexedPrimitive(D3DPT_TRIANGLESTRIP, 0, 0, 4, 0,
2); //2 triangles to render
pD3DDevice->SetRenderState(D3DRS_LIGHTING, FALSE);
result=pD3DDevice->DrawPrimitive(D3DPT_TRIANGLESTRIP, 0, 2);
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
pD3DDevice->SetRenderState(D3DRS_LIGHTING, TRUE);
pD3DDevice->SetRenderState(D3DRS_ZENABLE, TRUE);
pD3DDevice->SetRenderState(D3DRS_ZWRITEENABLE, TRUE);
pD3DDevice->SetTransform(D3DTS_PROJECTION, &projectionMatrix);
pD3DDevice->SetTransform(D3DTS_WORLD, &currentModelviewMatrix);
//////////
```

```
pD3DDevice->SetFVF(D3DFVF_CUSTOMVERTEX);
pD3DDevice->SetStreamSource(0, pVertexBuffer, 0, sizeof(TVertex));
```

```
//Enable blending
pD3DDevice->SetRenderState(D3DRS_ALPHABLENDENABLE, TRUE);
```

```
D3DXMatrixTranslation(&ModelviewMatrix, 0.0, 0.0, 5.0);
pD3DDevice->SetTransform(D3DTS_WORLD, &ModelviewMatrix);
result=pD3DDevice->DrawPrimitive(D3DPT_TRIANGLEFAN , 0, 2);
```

```
D3DXMatrixTranslation(&ModelviewMatrix, 2.0, 0.0, 5.0);
pD3DDevice->SetTransform(D3DTS_WORLD, &ModelviewMatrix);
result=pD3DDevice->DrawPrimitive(D3DPT_TRIANGLEFAN, 0, 2);
```

```
//Disable blending
pD3DDevice->SetRenderState(D3DRS_ALPHABLENDENABLE, FALSE);
```

```
result=pD3DDevice->EndScene();
//NEW//////////NEW//////////NEW//////////NEW//////////
//////////
*/
```

```
RECT rect={ Viewport.X, Viewport.Y, Viewport.Width, Viewport.Height};
result=pD3DDevice->Present(NULL, &rect, NULL, NULL);
```

```
return TRUE;
}
```

```
//////////
//////////
// THE KILL D3D SCENE
//////////
//////////
```

```
void KillD3DScene()
{
```

```
//NEW//////////NEW//////////NEW//////////NEW//////////
//////////
```

```
HRESULT result;
```

```
if(pVertexBuffer)
{
    result=pVertexBuffer->Release(); // Release Vertex Buffer
    pVertexBuffer=NULL;
}
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
}

if(pVertexBuffer2)
{
    result=pVertexBuffer2->Release(); // Release Vertex Buffer
    pVertexBuffer2=NULL;
}

if(pIndexBuffer2)
{
    result=pIndexBuffer2->Release();
    pIndexBuffer2=NULL;
}

//NEW////////////////////////////////NEW////////////////////////////////NEW////////////////////////////////NEW////////
////////

if(pBaseTexture)
{
    pBaseTexture->Release();
    pBaseTexture=NULL;
}

}

////////////////////////////////////
////////////////////////////////////
// THE KILL D3D WINDOW
////////////////////////////////////
////////////////////////////////////
void KillD3DWindow() // Properly Kill The Window
{
    KillD3DScene(); // Release D3D Scene

if(pD3DDevice)
{
    pD3DDevice->Release(); // Release D3D Device
    pD3DDevice=NULL;
}

if(pD3D)
{
    pD3D->Release(); // Release D3D Interface
    pD3D=NULL;
}

if (hDC && !ReleaseDC(hWnd, hDC)) // Are We Able To Release The DC
{
    MessageBox(NULL,"Release Device Context Failed.", "SHUTDOWN ERROR",MB_OK |
    MB_ICONINFORMATION);
    hDC=NULL; // Set DC To NULL
}
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
}

if (hWnd && !DestroyWindow(hWnd)) // Are We Able To Destroy The Window?
{
    MessageBox(NULL,"Could Not Release hWnd.,"SHUTDOWN ERROR",MB_OK |
    MB_ICONINFORMATION);
    hWnd=NULL; // Set hWnd To NULL
}

if (!UnregisterClass("Direct3D",hInstance)) // Are We Able To Unregister
Class
{
    MessageBox(NULL,"Could Not Unregister Class.,"SHUTDOWN ERROR",MB_OK |
    MB_ICONINFORMATION);
    hInstance=NULL; // Set hInstance To NULL
}
}

/* This Code Creates Our Direct3D Window. Parameters Are: *
* title - Title To Appear At The Top Of The Window *
* width - Width Of The D3D Window Or Fullscreen Mode *
* height - Height Of The D3D Window Or Fullscreen Mode *
* fullscreenflag - Use Fullscreen Mode (TRUE) Or Windowed Mode (FALSE) */

////////////////////////////////////
////////////////////////////////////
// THE CREATE D3D WINDOW
////////////////////////////////////
////////////////////////////////////
BOOL CreateD3DWindow(char* title, int width, int height, HINSTANCE
hinstance)
{
    HRESULT result;

    WNDCLASS wc;
    DWORD dwExStyle; // Window Extended Style
    DWORD dwStyle; // Window Style
    RECT WindowRect; // Grabs Rectangle Upper Left / Lower Right Values
    WindowRect.left=(long)0; // Set Left Value To 0
    WindowRect.right=(long)width; // Set Right Value To Requested Width
    WindowRect.top=(long)0; // Set Top Value To 0
    WindowRect.bottom=(long)height; // Set Bottom Value To Requested Height

    hInstance = hinstance; // Grab An Instance For Our Window
    wc.style = CS_HREDRAW | CS_VREDRAW | CS_OWNDC; // Redraw On Size, And Own
    DC For Window.
    wc.lpfnWndProc = (WNDPROC) WndProc; // WndProc Handles Messages
    wc.cbClsExtra = 0; // No Extra Window Data
    wc.cbWndExtra = 0; // No Extra Window Data
    wc.hInstance = hInstance; // Set The Instance
    wc.hIcon = LoadIcon(NULL, IDI_WINLOGO); // Load The Default Icon
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
wc.hCursor = LoadCursor(NULL, IDC_ARROW); // Load The Arrow Pointer
wc.hbrBackground = NULL; // No Background Required For D3D
wc.lpszMenuName = NULL; // We Don't Want A Menu
wc.lpszClassName = "Direct3D"; // Set The Class Name

if (!RegisterClass(&wc)) // Attempt To Register The Window Class
{
    MessageBox(NULL, "Failed To Register The Window
Class.", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE; // Return FALSE
}

{
    dwExStyle=WS_EX_APPWINDOW | WS_EX_WINDOWEDGE; // Window Extended Style
    dwStyle=WS_OVERLAPPEDWINDOW; // Windows Style
}

AdjustWindowRectEx(&WindowRect, dwStyle, FALSE, dwExStyle); // Adjust
Window To True Requested Size

//WindowWidth=WindowRect.right-WindowRect.left;
//WindowHeight=WindowRect.bottom-WindowRect.top;

// Create The Window
if(!(hWnd=CreateWindowEx( dwExStyle, // Extended Style For The Window
    "Direct3D", // Class Name
    title, // Window Title
    dwStyle | // Defined Window Style
    WS_CLIPSIBLINGS | // Required Window Style
    WS_CLIPCHILDREN, // Required Window Style
    0, 0, // Window Position
    WindowRect.right-WindowRect.left, // Calculate Window Width
    WindowRect.bottom-WindowRect.top, // Calculate Window Height
    NULL, // No Parent Window
    NULL, // No Menu
    hInstance, // Instance
    NULL))) // Dont Pass Anything To WM_CREATE
{
    KillD3DWindow(); // Reset The Display
    MessageBox(NULL, "Window Creation
Error.", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE; // Return FALSE
}

if (!(hDC=GetDC(hWnd))) // Did We Get A Device Context?
{
    KillD3DWindow(); // Reset The Display
    MessageBox(NULL, "Can't Create A Device
Context.", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE; // Return FALSE
}
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
pD3D=Direct3DCreate9(D3D_SDK_VERSION); // Check for correct Direct3D
version
if(pD3D==NULL)
{
    KillD3DWindow(); // Reset The Display
    MessageBox(NULL,"Can't find D3D SDK Version
9.", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE; // Return FALSE
}

d3dpp.BackBufferWidth=width;
d3dpp.BackBufferHeight=height;
d3dpp.BackBufferFormat=D3DFMT_X8R8G8B8;//D3DFMT_R5G6B5
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_D16;

d3dpp.Flags=D3DPRESENTFLAG_DISCARD_DEPTHSTENCIL;//D3DPRESENTFLAG_DISCARD_DEP
THSTENCIL | D3DPRESENTFLAG_DEVICECLIP;
d3dpp.FullScreen_RefreshRateInHz=D3DPRESENT_RATE_DEFAULT;

d3dpp.PresentationInterval=D3DPRESENT_INTERVAL_ONE;//D3DPRESENT_INTERVAL_DEF
AULT;

// Check The Wanted Surface Format.
#ifdef USEHAL
    result=pD3D->CheckDeviceFormat(D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL,
        d3dpp.BackBufferFormat, D3DUSAGE_DEPTHSTENCIL,
        D3DRTYPE_SURFACE, d3dpp.AutoDepthStencilFormat);
#else
    result=pD3D->CheckDeviceFormat(D3DADAPTER_DEFAULT, D3DDEVTYPE_REF,
        d3dpp.BackBufferFormat, D3DUSAGE_DEPTHSTENCIL,
        D3DRTYPE_SURFACE, d3dpp.AutoDepthStencilFormat);
#endif

if(result==D3DERR_INVALIDCALL)
{
    KillD3DWindow(); // Reset The Display
    MessageBox(NULL,"Can't Find Surface Format.
(D3DERR_INVALIDCALL)", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE; // Return FALSE
}
else if(result==D3DERR_NOTAVAILABLE)
{
    KillD3DWindow(); // Reset The Display
    MessageBox(NULL,"Can't Find Surface Format.
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
(D3DERR_NOTAVAILABLE)", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE; // Return FALSE
}

// Create The Direct3D Device
#ifdef USEHAL
if(FAILED(pD3D->CreateDevice(D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL, hWnd,
    D3DCREATE_HARDWARE_VERTEXPROCESSING, &d3dpp, &pD3DDevice)))
#else
if(FAILED(pD3D->CreateDevice(D3DADAPTER_DEFAULT, D3DDEVTYPE_REF, hWnd,
    D3DCREATE_HARDWARE_VERTEXPROCESSING, &d3dpp, &pD3DDevice)))
#endif
{
    KillD3DWindow(); // Reset The Display
    MessageBox(NULL, "Can't Create Direct3D
Device.", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE;
}

ShowWindow(hWnd, SW_SHOW); // Show The Window
SetForegroundWindow(hWnd); // Slightly Higher Priority
SetFocus(hWnd); // Sets Keyboard Focus To The Window
ReSizeD3DScene(width, height); // Set Up Our Perspective D3D Screen

if (!InitD3D()) // Initialize Our Newly Created D3D Window
{
    KillD3DWindow(); // Reset The Display
    MessageBox(NULL, "Initialization
Failed.", "ERROR", MB_OK|MB_ICONEXCLAMATION);
    return FALSE; // Return FALSE
}

return TRUE;
}

////////////////////////////////////
////////////////////////////////////
// THE WINDOW PROCEDURE
////////////////////////////////////
////////////////////////////////////
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_ACTIVATE: // Watch For Window Activate Message
    {
        if (!HIWORD(wParam)) // Check Minimization State
        {
            active=TRUE; // Program Is Active
        }
    }
    else
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
{
    active=FALSE; // Program Is No Longer Active
}

return 0; // Return To The Message Loop
}

case WM_SYSCOMMAND: // Intercept System Commands
{
    switch (wParam) // Check System Calls
    {
        case SC_SCREENSAVE: // Screensaver Trying To Start?
        case SC_MONITORPOWER: // Monitor Trying To Enter Powersave?
            return 0; // Prevent From Happening
    }
    break; // Exit
}

case WM_CLOSE: // Did We Receive A Close Message?
{
    PostQuitMessage(0); // Send A Quit Message
    return 0; // Jump Back
}

case WM_KEYDOWN: // Is A Key Being Held Down?
{
    keys[wParam] = TRUE; // If So, Mark It As TRUE
    return 0; // Jump Back
}

case WM_KEYUP: // Has A Key Been Released?
{
    keys[wParam] = FALSE; // If So, Mark It As FALSE
    return 0; // Jump Back
}

case WM_SIZE: // Resize The Direct3D Window
{
    ReSizeD3DScene(LOWORD(lParam), HIWORD(lParam)); // LoWord=Width,
    HiWord=Height
    return 0; // Jump Back
}
}

// Pass All Unhandled Messages To DefWindowProc
return DefWindowProc(hWnd,uMsg,wParam,lParam);
}

////////////////////////////////////
////////////////////////////////////
// THE WINMAIN
```

Re: In window mode, pD3DDevice->SetViewport?

microsoft.public.win32.programmer.directx.graphics: Re: In window mode, pD3DDevice->SetViewport?

```
////////////////////////////////////  
////////////////////////////////////  
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR  
lpCmdLine, int nCmdShow)  
{  
    MSG msg;  
    BOOL done=FALSE; // Bool Variable To Exit Loop  
  
    // Create Our Direct3D Window  
    if(!CreateD3DWindow("APRON TUTORIALS", 800, 800, hInstance))  
    {  
        return 0; // Quit If Window Was Not Created  
    }  
  
    while(!done) // Loop That Runs While done=FALSE  
    {  
        if (PeekMessage(&msg,NULL,0,0,PM_REMOVE)) // Is There A Message Waiting?  
        {  
            if (msg.message==WM_QUIT) // Have We Received A Quit Message?  
            {  
                done=TRUE; // If So done=TRUE  
            }  
            else // If Not, Deal With Window Messages  
            {  
                TranslateMessage(&msg); // Translate The Message  
                DispatchMessage(&msg); // Dispatch The Message  
            }  
        }  
        else // If There Are No Messages  
        {  
            // Draw The Scene. Watch For ESC Key And Quit Messages From  
            DrawD3DScene()  
            if ((active && !RenderScene()) || keys[VK_ESCAPE]) // Active? Was There  
            A Quit Received?  
            {  
                done=TRUE; // ESC or DrawD3DScene Signalled A Quit  
            }  
        }  
    }  
  
    // Shutdown  
    KillD3DWindow();  
    return (msg.wParam); // Exit The Program  
}
```

Re: In window mode, pD3DDevice->SetViewport?