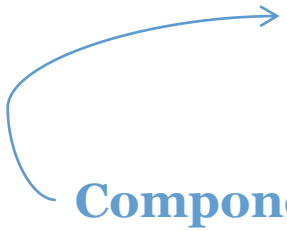
Direct Input

Gail Carmichael

comp3501-fall2012.blogspot.com

COM Design

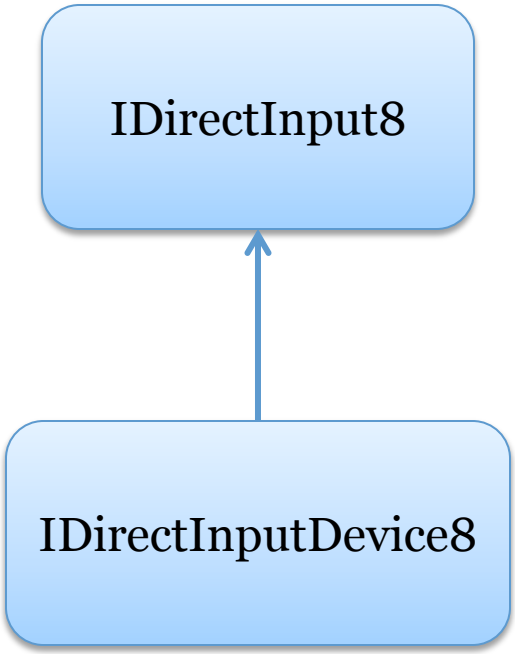


Component Object Model

Declare object pointer

Call function to instantiate objects

Interface Hierarchy



Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

Libraries to link to:

```
dinput8.lib
dxguid.lib
```

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>

LPDIRECTINPUT8 directInputObject;
LPDIRECTINPUTDEVICE8 keyboardDevice;

HRESULT hr = DirectInput8Create(
  appInstance,
  DIRECTINPUT_VERSION,
  IID_IDirectInput8,
  (void**)&directInputObject,
  NULL);

hr = directInputObject->CreateDevice(
   GUID_SysKeyboard,
   &keyboardDevice,
   NULL);
```

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>

LPDIRECT
LPDIRECT                                    e;

HRESULT hr = DirectInput8Create(
  appInstance,
  DIRECTINPUT_VERSION,
  IID_IDirectInput8,
  (void**)&directInputObject,
  NULL);

hr = directInputObject->CreateDevice(
   GUID_SysKeyboard,
   &keyboardDevice,
   NULL);
```

Has to come before header include to avoid warnings

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>

LPDIRECT
LPDIRECT                                    e;

HRESULT hr = DirectInput8Create(
  appInstance,
  DIRECTINPUT_VERSION,
  IID_IDirectInput8,
  (void**)&directInputObject,
  NULL);

hr = directInputObject->CreateDevice(
   GUID_SysKeyboard,
   &keyboardDevice,
   NULL);
```

Passed to application in WinMain

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>

LPDIR                                    t;
LPDIR                                    vice;

HRESULT hr = DirectInput8Create(
  appInstance,
  DIRECTINPUT_VERSION,
  IID_IDirectInput8,
  (void**)&directInputObject,
  NULL);

hr = directInputObject->CreateDevice(
   GUID_SysKeyboard,
   &keyboardDevice,
   NULL);
```

COM identifier for type of object to create

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>

LPDIRECT
LPDIRECT                                    ce;
```

For advanced COM usage
(we don't need it)

```
HRESULT hr = DirectInput8Create(
  appInstance,
  DIRECTINPUT_VERSION,
  IID_IDirectInput8,
  (void**)&directInputObject,
  NULL);

hr = directInputObject->CreateDevice(
   GUID_SysKeyboard,
   &keyboardDevice,
   NULL);
```

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>

LPDIRECTINPUT8 directInputObject;
LPDIRECTINPUTDEVICE8 keyboardDevice;

HRESULT hr = DirectInput8Create(
  appInstance,
  DIRECTINPUT_VERSION,
  IID_I
  (void
  NULL);

hr = directInputObject->CreateDevice(
   GUID_SysKeyboard,
   &keyboardDevice,
   NULL);
```

Indicates type of device we want to use

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
#define DIRECTINPUT_VERSION 0x0800
#include <dinput.h>

LPDIRECTINPUT8 directInputObject;
LPDIRECTINPUTDEVICE8 keyboardDevice;

HRESULT hr = DirectInput8Create(
  appInstance,
  DIRECTINPUT_VERSION,
  IID_IDirectInput8,
  (void**)&directInputObject,
  NULL);

hr = directInputObject->CreateDevice(
   GUID_SysKeyboard,
   &keyboardDevice,
   NULL);
```

For advanced COM usage
(we don't need it)

Keyboard

Create

Configure

Acquire

Get data

Release

```
hr = keyboardDevice->
   SetDataFormat(&c_dfDIKeyboard);

hr = keyboardDevice->
   SetCooperativeLevel(
      hwnd,
      DISCL_FOREGROUND |
      DISCL_NONEXCLUSIVE);
```

Keyboard

Create

Configure

Acquire

Get data

Release

```
hr = keyboardDevice->
   SetDataFormat(&c_dfDIKeyboard);
```

Keyboard device data format – can define your own, but you'll usually use this one

Keyboard

Create

Configure

Acquire

Get data

Release

```
hr = keyboardDevice->
   SetDataFormat(&c_dfDIKeyboard);

hr = keyboardDevice->
   SetCooperativeLevel(
      hwnd,
      DISCL_FOREGROUND |
      DISCL_NONEXCLUSIVE);
```

Foreground access: only when window is active (otherwise, background)

Keyboard

Create

Configure

Acquire

Get data

Release

```
hr = keyboardDevice->
   SetDataFormat(&c_dfDIKeyboard);

hr = keyboardDevice->
   SetCooperativeLevel(
      hwnd,
      DISCL_FOREGROUND |
      DISCL_NONEXCLUSIVE);
```

Non-exclusive: don't prevent other applications from accessing keyboard (otherwise, exclusive)

Keyboard

Create

Configure

Acquire

Get data

Release

```
hr = keyboardDevice->Acquire();
while( hr == DIERR_INPUTLOST )
{
   hr = keyboardDevice->Acquire();
}

if (FAILED(hr))
   return;
```

Keyboard

Create

Configure

Acquire

Get data

Release

```
hr = keyboardDevice->Acquire();
while( hr == DIERR_INPUTLOST )
{
    hr = keyboardDevice->Acquire();
}

if (FA
    ret
```

Happens for non-exclusive foreground windows when user switches away

Keyboard

Create

Configure

Acquire

Get data

Release

```
BYTE keys[256];
ZeroMemory(keys, sizeof(keys));

keyboardDevice->GetDeviceState(
   sizeof(keys), keys);

if (keys[DIK_X] & 0x80)
{
   // x key is pressed
}
```

Keyboard

Create

Configure

Acquire

Get data

Release

```
BYTE keys[256];
ZeroMemory(keys, sizeof(keys));

keyboardDevice->GetDeviceState(
    sizeof(keys), keys);

if (keys[DIK_X] & 0x80)
{
    // x k
}
```

Gets a snapshot of keyboard at that moment in time

Keyboard

- Create
- Configure
- Acquire
- Get data
- Release

```
BYTE key
ZeroMemo

keyboard
   sizeo

if (keys[DIK_X] & 0x80)
{
   // x key is pressed
}
```

8th bit will be set to 1 if the key is pressed

Keyboard

Create

Configure

Acquire

Get data

Release

```
if (keyboardDevice)
{
   keyboardDevice->Unacquire();
   keyboardDevice->Release();
}

if (directInputObject)
{
   directInputObject->Release();
}
```

Mouse

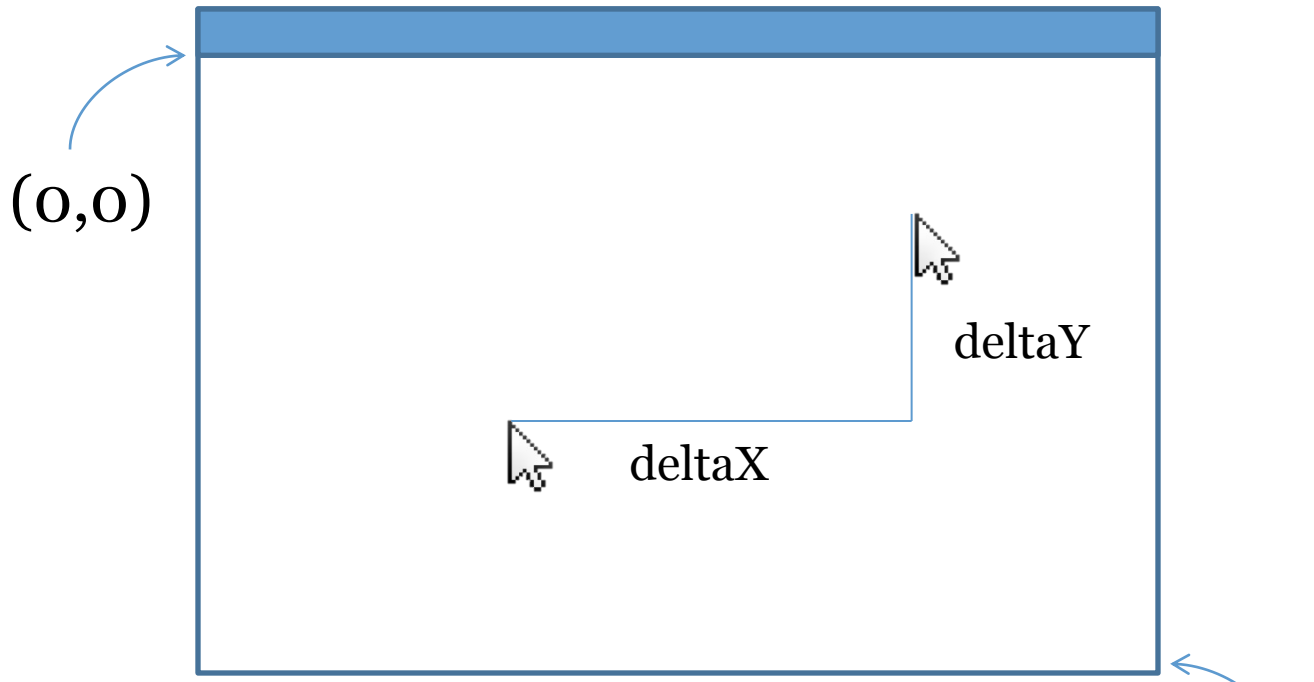
```
… CreateDevice(GUID_SysMouse, …)

… SetDataFormat(&c_dfDIMouse2);

… SetCooperativeLevel(hWnd,
DISCL_FOREGROUND | DISCL_NONEXCLUSIVE);
```

Mouse

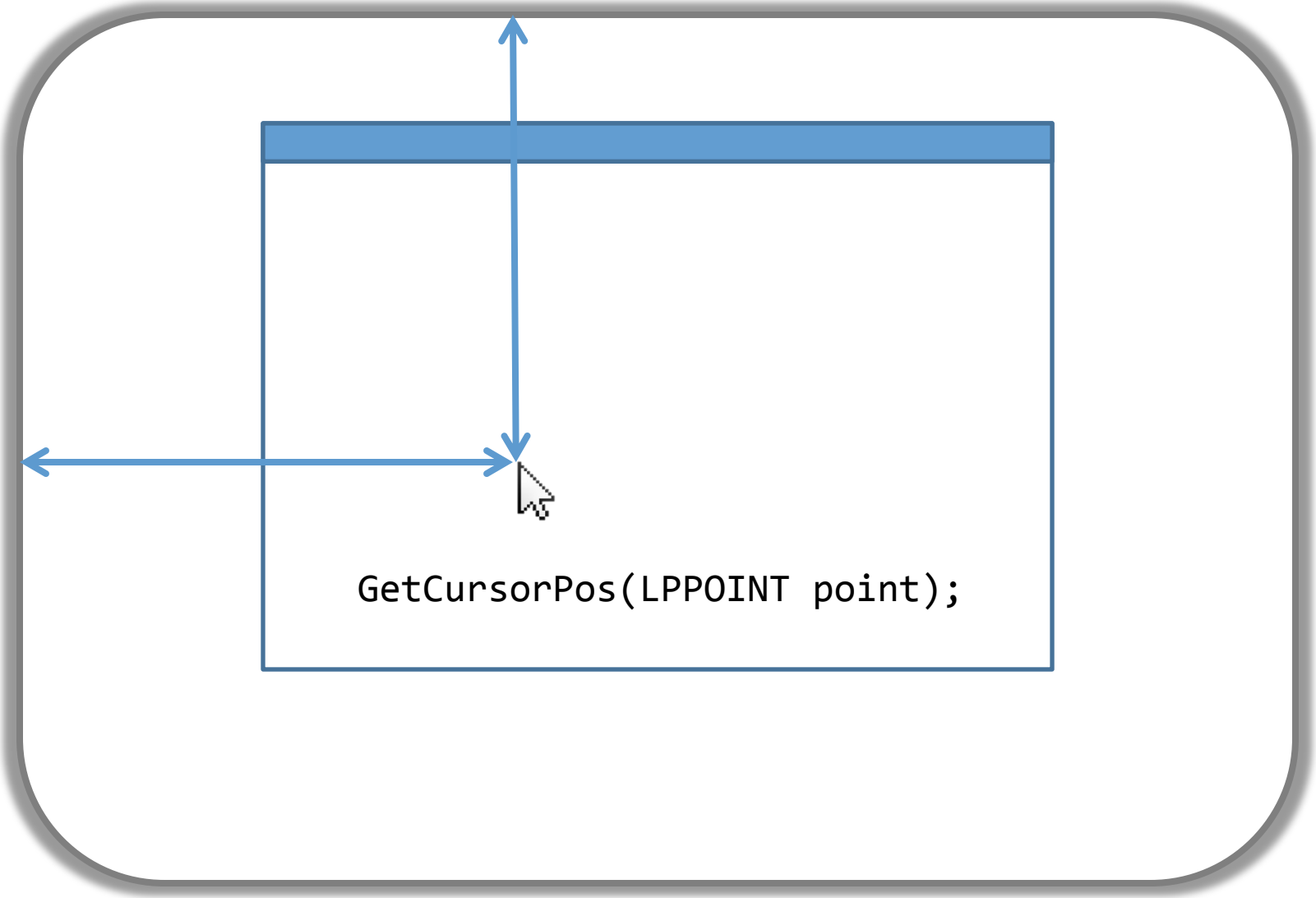
```
DIMOUSESTATE2 mouseState;
ZeroMemory( &mouseState, sizeof(mouseState));

HRESULT hr = mouseDevice->GetDeviceState(
   sizeof(DIMOUSESTATE2), &mouseState);
```

`LONG x; LONG y; LONG z` : x,y,z position of the mouse
`BYTE rgbButtons[8]` : state of each button

Mouse Coordinates



Absolute Coordinates



Displaying Text

Gail Carmichael

comp3501-fall2012.blogspot.com

Displaying Text

Create

Draw

```
ID3DXFont *m_font;

D3DXCreateFont(
   gD3dDevice,
   20,
   0,
   FW_BOLD,
   0,
   FALSE,
   DEFAULT_CHARSET,
   OUT_DEFAULT_PRECIS,
   DEFAULT_QUALITY,
   DEFAULT_PITCH | FF_DONTCARE,
   TEXT("Arial"),
   &m_font );
```

Displaying Text

Create

Draw

```
ID3DXFont *m_font;

D3DXCreateFont(
    d3dDevice,
    20,
    0,
    FW_BOLD,
    0,
    FALSE,
    DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_DONTCARE,
    TEXT("Arial"),
    &m_font );
```

Height of font

Displaying Text

Create

Draw

```
ID3DXFont *m_font;

D3DXCreateFont(
    d3dDevice,
    20,
    0,
    FW_BOLD,
    0,
    FALSE,
    DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_DONTCARE,
    TEXT("Arial"),
    &m_font );
```

Font Weight

Displaying Text

Create

Draw

```
ID3DXFont *m_font;

D3DXCreateFont(
    d3dDevice,
    20,
    0,
    FW_BOLD,
    0,
    FALSE,
    DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_DONTCARE,
    TEXT("Arial"),
    &m_font );
```

Is font italic?

Displaying Text

Create

Draw

```
ID3DXFont *m_font;

D3DXCreateFont(
    d3dDevice,
    20,
    0,
    FW_BOLD,
    0,
    FALSE,
    DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_DONTCARE,
    TEXT("Arial"),
    &m_font );
```

Name of font

Displaying Text

Create

Draw

```
D3DCOLOR fontColor =
   D3DCOLOR_ARGB(255,0,0,255);

RECT rct;
rct.left=2;
rct.right=780;
rct.top=10;
rct.bottom=rct.top+20;

m_font->DrawText(
   NULL,
   "Hello World",
   -1,
   &rct,
   0,
   fontColor );
```

Displaying Text

Create

Draw

```
D3DCOLOR fontColor =
   D3DCOLOR_ARGB(255,0,0,255);
```

```
RECT rct;
rct.left=2;
rct.right=780;
rct.top=10;
rct.bottom=rct.top+20;

m_font->DrawText(
   NULL,
   "Hello World",
   -1,
   &rct,
   0,
   fontColor );
```

Color to draw the font with

Displaying Text

Create

Draw

```
D3DCOLOR fontColor =
   D3DCOLOR_ARGB(255,0,0,255);

RECT rct;
rct.left=2;
rct.right=780;
rct.top=10;
rct.bottom=rct.top+20;

m_font->DrawText(
   NULL,
   "Hello World",
   -1,
   &rct,
   0,
   fontColor );
```

Where to draw the text

Displaying Text

Create

Draw

```
D3DCOLOR fontColor =
   D3DCOLOR_ARGB(255,0,0,255);

RECT rct;
rct.left=2;
rct.right=780;
rct.top=10;
rct.bottom=rct.top+20;

m_font->DrawText(
   NULL,
   "Hello World",
   -1,
   &rct,
   0,
   fontColor );
```

Sprite object – if not given, D3D will use its own internal sprite object to render with

Displaying Text

Create

Draw

```
D3DCOLOR fontColor =
   D3DCOLOR_ARGB(255,0,0,255);

RECT rct;
rct.left=2;
rct.right=780;
rct.top=10;
rct.bottom=rct.top+20;

m_font->DrawText(
   NULL,
   "Hello World",
   -1,
   &rct,
   0,
   fontColor );
```

Text to render

Displaying Text

Create

Draw

```
D3DCOLOR fontColor =
   D3DCOLOR_ARGB(255,0,0,255);

RECT rct;
rct.left=2;
rct.right=780;
rct.top=10;
rct.bottom=

m_font->Dra
   NULL,
   "Hello World",
   -1,
   &rct,
   0,
   fontColor );
```

Number of characters in the string, or -1 to make D3D do the counting

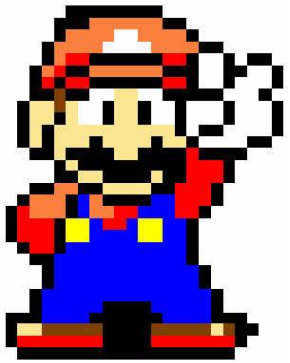
Displaying Text

Create

Draw

```
D3DCOLOR fontColor =
   D3DCOLOR_ARGB(255,0,0,255);

RECT rct;
rct.left=2;
rct.right=780;
rct.top=10;
rct.bottom=rct.top+20;

m_font->DrawText(
   NULL,
   "Hello World"
   -1,
   &rct,
   0,
   fontColor );
```

Format flags (justification, line spacing, etc)



Sprites

Gail Carmichael

comp3501-fall2012.blogspot.com

What is a Sprite?

A 2D texture that can be positioned and rotated on the screen.

Why Use Sprites?

2D Images for 2D Games

Simplify rendering

Increase realism

Drawing Sprites

Create

Render

Release

```
LPD3DXSPRITE sprite=NULL;
LPDIRECT3DTEXTURE9 texture=NULL;

if (SUCCEEDED(
        D3DXCreateSprite(
           device, &sprite))
{
     // created OK
}

D3DXCreateTextureFromFile(
   device, “fileName.bmp”, texture);
```

Drawing Sprites

Create

Render

Release

```
D3DXVECTOR3 pos;

pos.x=10.0f;
pos.y=20.0f;
pos.z=0.0f;

sprite->Begin(D3DXSPRITE_ALPHABLEND);

sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

Drawing Sprites

Create

Render

Release

```
D3DXVECTOR3 pos;

pos.
pos.
pos.

spri                                    ND);

sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

Note:
Must do this in the render loop between scene ‘begin’ and ‘end’ calls

Drawing Sprites

Create

Render

Release

```
D3DXVECTOR3 pos;

pos.x=10.0f;
pos.y=20.0f;
pos.z=0.0f;
```

Where to render the sprite

```
sprite->Begin(D3DXSPRITE_ALPHABLEND);

sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

Drawing Sprites

Create

Render

Release

Some flags specify depth parameters:
D3DXSPRITE_SORT_TEXTURE
D3DXSPRITE_SORT_DEPTH_BACKTOFRONT

```
pos.z=0.0f;

sprite->Begin(D3DXSPRITE_ALPHABLEND);

sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

Drawing Sprites

Create

Render

Release

```
D3DXVECTOR3 pos;

pos.x=10.0f;
pos.y=20.0f;
pos.z=0.0f;

sprite->Begin(D3DX

sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

A rect to say how much of the texture to render, NULL to render everything

Drawing Sprites

Create

Render

Release

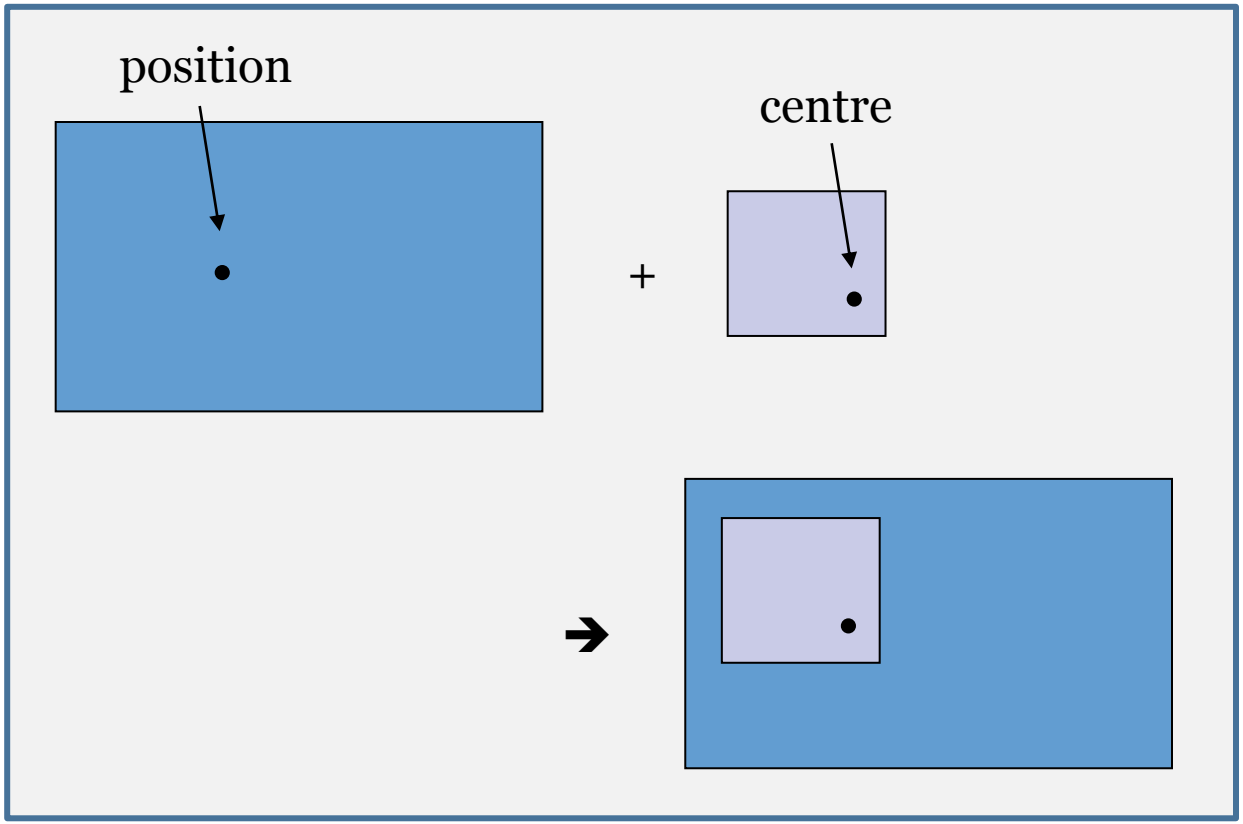
```
D3DXVECTOR3 pos;

pos.x=10.0f;
pos.y=20.0f;
pos.z=0.0f;

sprite->Begin(D3DX

sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

Rotation/positioning point, or top-left if NULL

Create

Render

Release




```
sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

Drawing Sprites

Create

Render

Release

```
D3DXVECTOR3 pos;

pos.x=10.0f;
pos.y=20.0f;
pos.z=0.0f;

sprite->Begin(D3

sprite->Draw(
   texture,NULL,NULL,&pos,0xFFFFFFFF);
```

Can be used to modulate color of texture (this value does nothing)

Drawing Sprites

Create

Render

Release

```
sprite.End();
```

Animation