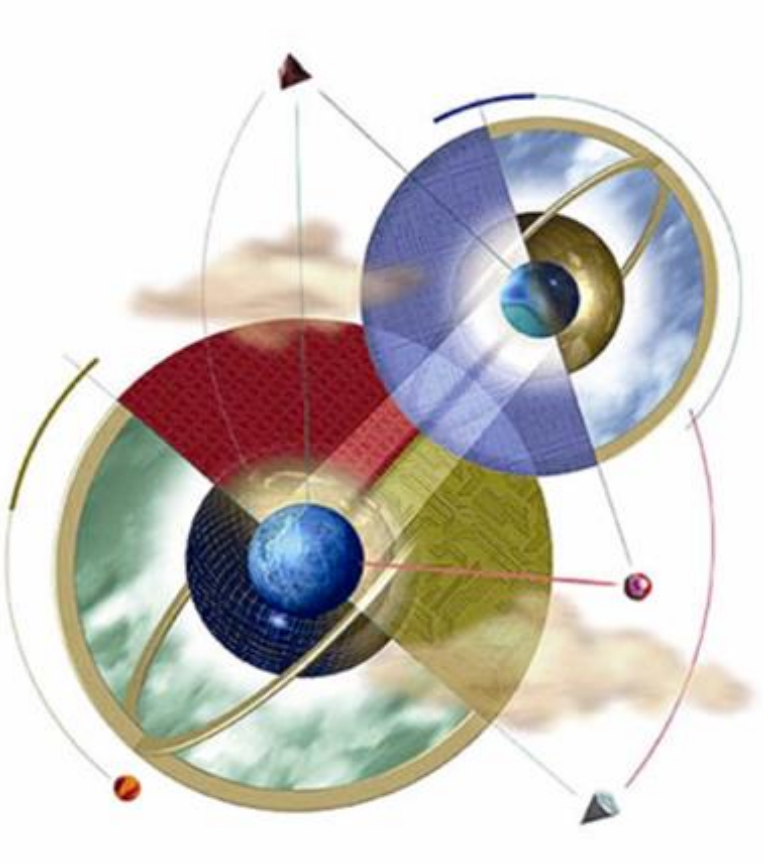


Direct3D Fundamentals

Outline

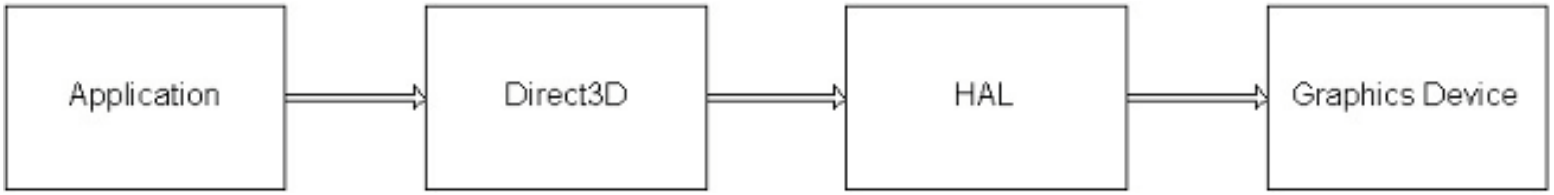
- Direct3D Initialization
- The Rendering Pipeline
- Drawing in Direct3D
- Color
- Lighting
- Texturing
- Blending
- Stenciling

Direct3D Initialization

- To learn how Direct3D interacts with graphics hardware
- To understand the role that COM plays with Direct3D
- To learn fundamental graphics concepts, such as how 2D images are stored, page flipping, and depth buffering
- To learn how to initialize Direct3D
- To become familiar with the general structure that the sample applications of this book employ

Direct3D Overview

- Direct3D is a low-level graphics API (application programming interface) that enables us to render 3D worlds using 3D hardware acceleration.



The REF Device

- Direct3D provides a reference rasterizer (known as a REF device), which emulates the entire Direct3D API in software.
- It is important to understand that the REF device is for development only

- D3DDEVTYPE_HAL
- D3DDEVTYPE_REF

Component Object Model (COM)

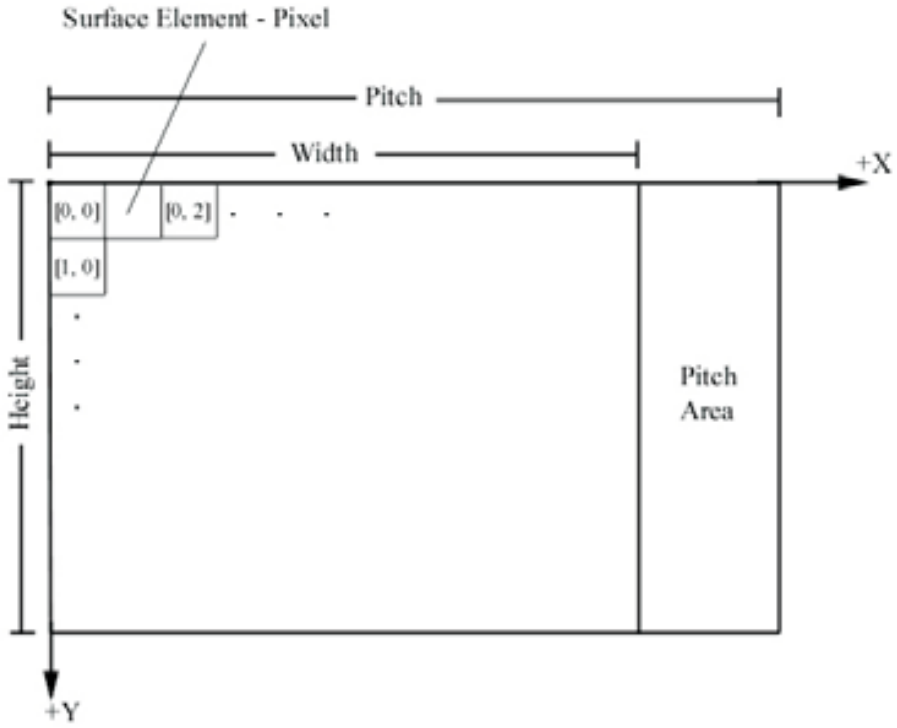
- Is the technology that allows DirectX to be language independent and have backward compatibility
- We obtain pointers to COM interfaces through special functions or the methods of another COM interface
- COM interfaces are prefixed with a capital **I**.
- For example, the COM interface that represents a surface is called **IDirect3DSurface9.**

Some Preliminaries

- Surfaces
- Multisampling
- Pixel Formats
- Memory Pools
- The Swap Chain and Page Flipping
- Depth Buffers
- Vertex processing
- Device Capabilities

Surfaces

- A *surface is a matrix of pixels that Direct3D uses primarily to store 2D* image data.



Surfaces

- IDirect3DSurface9
  - ✓ LockRect
  - ✓ UnlockRect
  - ✓ GetDesc

```
typedef struct  D3DLOCKED RECT {
    INT Pitch;   // the surface pitch
    void *pBits; // pointer to the start of the surface memory
} D3DLOCKED_RECT;
```

Multisampling

- Multisampling is a technique used to smooth out blocky-looking images

- D3DMULTISAMPLE_TYPE
  - ✓ D3DMULTISAMPLE_NONE
  - ✓ D3DMULTISAMPLE_1_SAMPLE
  - ✓ D3DMULTISAMPLE_16_SAMPLE

Pixel Formats

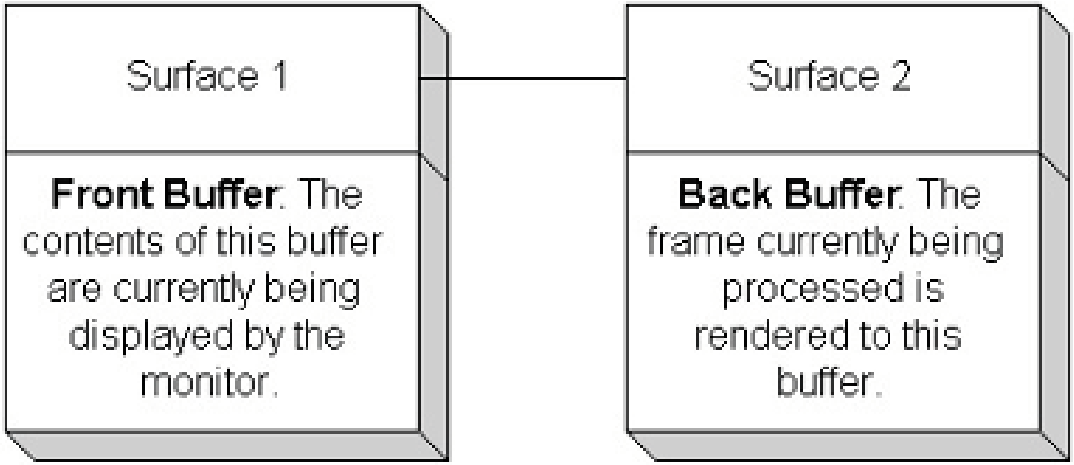
- D3DFORMAT
  - ✓ D3DFMT_R8G8B8
  - ✓ D3DFMT_X8R8G8B8
  - ✓ D3DFMT_A8R8G8B8
  - ✓ D3DFMT_A16B16G16R16F
  - ✓ D3DFMT_A32B32G32R32F
- The first three formats are common and supported on most hardware.

Memory Pools

- D3DPOOL
  - ✓ D3DPOOL_DEFAULT
  - ✓ D3DPOOL_MANAGED
  - ✓ D3DPOOL_SYSTEMMEM
  - ✓ D3DPOOL_SCRATCH

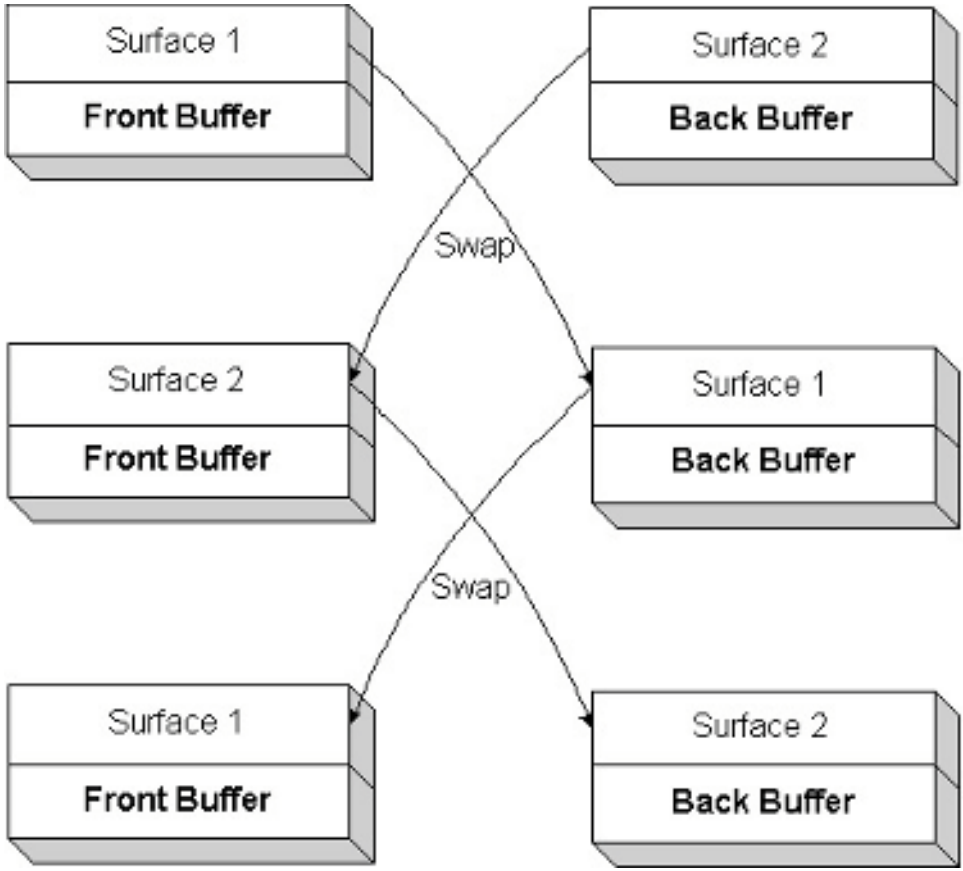
The Swap Chain and Page Flipping

- Swap chains is used to provide smooth animation between frames



Presenting

1. Render to back buffer
2. Present the back buffer
3. Go To (1)



Depth Buffers

- The depth buffer or z buffer is a surface that does not contain image data but rather depth information about a particular pixel.

Depth Buffers

- D3DFMT
  - ✓ D3DFMT_D32
  - ✓ D3DFMT_D24S8
  - ✓ D3DFMT_D24X8
  - ✓ D3DFMT_D24X4S4
  - ✓ D3DFMT_D16

Vertex processing

- Vertices are the building blocks for 3D geometry, and they can be processed in two different ways
  - *software vertex processing*
  - *hardware vertex processing*

- supports hardware vertex processing in hardware is to say that the graphics card supports transformation and lighting calculations in hardware.

Device Capabilities

- Every feature that Direct3D exposes has a corresponding data member or bit in the D3DCAPS9 structure.

```
bool supportsHardwareVertexProcessing;

// If the bit is "on" then that implies the hardware device
// supports it.
if( caps.DevCaps & D3DDEVCAPS HWTRANSFORMANDLIGHT )
{
     // Yes, the bit is on, so it is supported.
     supportsHardwareVertexProcessing = true;
}
else
{
     // No, the bit is off, so it is not supported.
     hardwareSupportsVertexProcessing = false;
}
```

Initializing Direct3D

- The process of initializing Direct3D can be broken down into the following steps:
  1. Acquire a pointer to an IDirect3D9
  2. Check the device capabilities (D3DCAPS9)
  3. Initialize an instance of the D3DPRESENT_PARAMETERS
  4. Create the IDirect3DDevice9