Graphics Programming with Direct X 9

Part I

(12 Week Lesson Plan)

Lesson 1: 3D Graphics Fundamentals

Textbook: Chapter One (pgs. 2 – 32)

Goals:

We begin the course by introducing the student to the fundamental mathematics necessary when developing 3D games. Essentially we will be talking about how 3D objects in games are represented as polygonal geometric models and how those models are ultimately drawn. It is especially important that students are familiar with the mathematics of the transformation pipeline since it plays an important role in getting this 3D geometry into a displayable 2D format. In that regard we will look at the entire geometry transformation pipeline from model space all the way through to screen space and discuss the various operations that are necessary to make this happen. This will include discussion of transformations such as scaling, rotations, and translation, as well as the conceptual idea of moving from one coordinate space to another and remapping clip space coordinates to final screen space pixel positions.

Key Topics:

- Geometric Modeling
  - 2D/3D Coordinate Systems
  - Meshes
    - Vertices
    - Winding Order
- The Transformation Pipeline
  - Translation
  - Rotation
  - Viewing Transformations
  - Perspective Projection
  - Screen Space Mapping

Projects: NONE

Exams/Quizzes: NONE

Recommended Study Time (hours): 5 - 7

Lesson 2: 3D Graphics Fundamentals II

Textbook: Chapter One (pgs. 32 – 92)

Goals:

Picking up where the last lesson left off, we will now look at the specific mathematics operations and data types that we will use throughout the course to affect the goals discussed previously regarding the transformation pipeline. We will examine three fundamental mathematical entities: vectors, planes and matrices and look at the role of each in the transformation pipeline as well as discussing other common uses. Core operations such as the dot and cross product, normalization and matrix and vector multiplication will also be discussed in detail. We then look at the D3DX equivalent data types and functions that we can use to carry out the operations discussed. Finally we will conclude with a detailed analysis of the perspective projection operation and see how the matrix is constructed and how arbitrary fields of view can be created to model different camera settings.

Key Topics:

- 3D Mathematics Primer
  - Vectors
    - Magnitude
    - Addition/ Subtraction
    - Scalar Multiplication
    - Normalization
    - Cross Product
    - Dot Product
  - Planes
  - Matrices
    - Matrix/Matrix Multiplication
    - Vector/Matrix Multiplication
    - 3D Rotation Matrices
    - Identity Matrices
    - Scaling and Shearing
    - Concatenation
    - Homogenous Coordinates
- D3DX Math
  - Data Types
    - D3DXMATRIX
    - D3DXVECTOR
    - D3DXPLANE
  - Matrix and Transformation Functions
    - D3DXMatrixMultiply
    - D3DXMatrixRotation{XYZ}
    - D3DXMatrixTranslation
    - D3DXMatrixRotationYawPitchRoll

        - D3DXVecTransform{...}
    - Vector Functions
        - Cross Product
        - Dot Product
        - Magnitude
        - Normalization
- The Transformation Pipeline II
    - The World Matrix
    - The View Matrix
    - The Perspective Projection Matrix
        - Field of View
        - Aspect Ratio

Projects:

Lab Project 1.1: Wireframe Renderer

Exams/Quizzes: NONE

Recommended Study Time (hours): 8 - 10

Lesson 3: DirectX Graphics Fundamentals I

Textbook: Chapter Two (pgs. 94 – 132)

Goals:

In this lesson our goal will be to start to get an overview of the DirectX Graphics pipeline and see how the different pieces relate to what we have already learned. A brief introduction to the COM programming model introduces the lesson as a means for understanding the low level processes involved when working with the DirectX API. Then, our ultimate goal is to be able to properly initialize the DirectX environment and create a rendering device for output. We will do this during this lesson and the next. This will require an understanding of the different resources that are associated with device management including window settings, front and back buffers, depth buffering, and swap chains.

Key Topics:

- The Component Object Model (COM)
  - Interfaces/IUnknown
  - GUIDS
  - COM and DirectX Graphics
- Initializing DirectX Graphics
- The Direct3D Device
  - Pipeline Overview
  - Device Memory
    - The Front/Back Buffer(s)
    - Swap Chains
  - Window Settings
    - Fullscreen/Windowed Mode
  - Depth Buffers
    - The Z-Buffer / W-Buffer

Projects:

Lab Project 2.1: DirectX Graphics Initialization

Exams/Quizzes: NONE

Recommended Study Time (hours): 8 – 10

Lesson 4: DirectX Graphics Fundamentals II

Textbook: Chapter Two (pgs. 132 – 155)

Goals:

Continuing our environment setup discussion, in this lesson our goal will be to create a rendering device for graphics output. Before we explore setting up the device, we will look at the various surface formats that we must understand for management of depth and color buffers. We will conclude the lesson with a look at configuring presentation parameters for device setup and then talk about how to write code to handle lost devices.

Key Topics:

- Surface Formats
  - Adapter Formats
  - Frame Buffer Formats
- Device Creation
  - Presentation Parameters
  - Lost Devices

Projects:

Lab Project 2.2: Device Enumeration

Exams/Quizzes: NONE

Recommended Study Time (hours): 8 - 10

Lesson 5: Primitive Rendering I

Textbook: Chapter Two (pgs. 156 – 191)

Goals:

Now that we have a rendering device properly configured, we are ready to begin drawing 3D objects using DirectX Graphics. In this lesson we will examine some of the important device settings (states) that will be necessary to make this happen. We will see how to render 3D objects as wireframe or solid objects and also talk about how to affect various forms of shading. Our discussion will also include flexible vertex formats, triangle data, and the DrawPrimitive function call. Once these preliminary topics are out of the way we will look at the core device render states that are used when drawing – depth buffering, lighting and shading, back face culling, etc. We will also talk about transformation states and how to pass the matrices we learned about in prior lessons up to the device for use in the transformation pipeline. We will conclude the lesson with discussion of scene rendering and presentation (clearing the buffers, beginning and ending the scene and presenting the results to the viewer).

Key Topics:

- Primitive Rendering
  - Fill Modes
  - Shading Modes
  - Vertex Data and the FVF
  - DrawPrimitiveUP
- Device States
  - Render States
    - Z – Buffering
    - Lighting/Shading/Dithering
    - Backface Culling
  - Transformation States
    - World/View/Projection Matrices
- Scene Rendering
  - Frame/Depth Buffer Clearing
  - Begin/End Scene
  - Presenting the Frame

Projects:

Exams/Quizzes: NONE

Recommended Study Time (hours): 5 – 7

Lesson 6: Primitive Rendering II

Textbook: Chapter Three (pgs. 194 – 235)

Goals:

In this lesson we will begin to examine more optimal rendering strategies in DirectX. Primarily the goal is to get the student comfortable with creating, filling and drawing with both vertex and index buffers. This means that we will look at both indexed and non-indexed mesh rendering for both static geometry and dynamic (animated) geometry. To that end it will be important to understand the various device memory pools that are available for our use and see which ones are appropriate for a given job. We will conclude the lesson with a discussion of indexed triangle strip generation and see how degenerate triangles play a role in that process.

Key Topics:

- Device Memory Pools and Resources
  - Video/AGP/System Memory
- Vertex Buffers
  - Creating Vertex Buffers
  - Vertex Buffer Memory Pools
  - Vertex Buffer Performance
  - Filling Vertex Buffers
  - Vertex Stream Sources
  - DrawPrimitive
- Index Buffers
  - Creating Index Buffers
  - DrawIndexedPrimitive/DrawIndexedPrimitiveUP
  - Indexed Triangle Strips/Degenerate Triangles

Projects:

Lab Project 3.1: Static Vertex Buffers
Lab Project 3.2: Simple Terrain Renderer
Lab Project 3.3: Dynamic Vertex Buffers

Exams/Quizzes: NONE

Recommended Study Time (hours): 8 – 10

Mid-Term Examination

The midterm examination in this course will consist of 40 multiple-choice and true/false questions pulled from the first three textbook chapters. Students are encouraged to use the lecture presentation slides as a means for reviewing the key material prior to the examination. The exam should take no more than 1.5 hours to complete. It is worth 35% of student final grade.

Lesson 7: Camera Systems

Textbook: Chapter Four (pgs. 238 – 296)

Goals:

In this lesson we will take a detailed look at the view transformation and its associated matrix and see how it can be used and manipulated to create a number of popular camera system types – first person, third person, and spacecraft. We will also discuss how to manage rendering viewports and see how the viewport matrix plays a role in this process. Once we have created a system for managing different cameras from a rendering perspective, we will examine how to use the camera clipping planes to optimize scene rendering. This will include writing code to extract these planes for the purposes of testing object bounding volumes to determine whether or not the geometry is actually visible given the current camera position and orientation. Objects that are not visible will not need to be rendered, thus allowing us to speed up our application.

Key Topics:

- The View Matrix
  - Vectors, Matrices, and Planes
    - The View Space Planes
    - The View Space Transformation
    - The Inverse Translation Vector
- Viewports
  - The Viewport Matrix
  - Viewport Aspect Ratios
- Camera Systems
  - Vector Regeneration
  - First Person Cameras
  - Third Person Cameras
- The View Frustum
  - Camera Space Frustum Plane Extraction
  - World Space Frustum Plane Extraction
  - Frustum Culling an AABB

Projects: NONE

Exams/Quizzes: NONE

Recommended Study Time (hours): 8 - 10

Lesson 8: Lighting

Textbook: Chapter Five (pgs. 298 – 344)

Goals:

In this lesson we will introduce the lighting model used in the fixed function DirectX Graphics pipeline. We begin with an overview of the different types of lighting (ambient, diffuse, specular, and emissive) that are modeled in real-time games. We will also talk about the specific light types (point/spot/directional) and how to setup their properties and configure the lighting pipeline to use them. This will include some discussion of the role of vertex normals and how to calculate them when necessary. In conjunction with lighting we must also discuss the concept of materials as they define how surfaces interact with the lights in the environment. We will see how to create them and set their properties to produce different results. We will conclude this lesson with a brief discussion of the advantages and disadvantages of using the fixed function vertex lighting pipeline as a means for setting the stage for more advanced lighting models that will be introduced in Part II of this course series.

Key Topics:

- Lighting Models
  - Indirect Lighting
    - Emissive/Ambient Illumination
  - Direct Lighting
    - Diffuse/Specular Light
- The Lighting Pipeline
  - Enabling DirectX Graphics Lighting
    - Enabling Specular Highlights
    - Enabling Global Ambient Lighting
  - Lighting Vertex Formats and Normals
  - Setting Lights and Light Limits
- Light Types
  - Point/Spot/Directional
- Materials
  - Colors, Specular and Power
  - Material Sources

Projects:

Lab Project 5.1: Dynamic Lights
Lab Project 5.2: Scene Lighting

Exams/Quizzes: NONE

Recommended Study Time (hours): 10 - 12

Lesson 9: Texture Mapping I

Textbook: Chapter Six (pgs. 346 – 398)

Goals:

In this lesson students will be introduced to texture mapping as a means for adding detail and realism to the lit models we studied in the last lesson. We begin by looking at what textures are and how they are defined in memory. This will lead to a preliminary discussion of mip-maps in terms of memory format and consumption. Then we will look at the various options at our disposal for loading texture maps from disk or memory using the D3DX utility library. Discussion of how to set a texture for rendering and the relationship between texture coordinates and addressing modes will follow. Finally we will talk about the problem of aliasing and other common artifacts and how to use various filters to improve the quality of our visual output.

Key Topics:

- Texture Memory Pools
  - Texture Formats
    - Validating Texture Formats
    - Surface Formats
- MIP Maps
- Loading Textures
- Setting Textures
- Texture Coordinates
- Sampler States
  - Texture Addressing Modes
    - Wrapping/Mirroring/Bordering/Clamping/MirrorOnce
  - Texture Coordinate Wrapping
  - Texture Filtering
    - Magnification/Minification
    - Point/Bilinear/Trilinear/Anisotropic

Projects:

Lab Project 6.1: Simple Texturing

Exams/Quizzes: NONE

Recommended Study Time (hours): 8 - 10

Lesson 10: Texture Mapping II

Textbook: Chapter Six (pgs. 399 – 449)

Goals:

This lesson will conclude our introduction to texture mapping (advanced texturing will be discussed in Part II of this series). We will begin by examining the texture pipeline and how to configure the various stages for both single and multi-texturing operations. Then we will take a step back and examine texture compression and the various compressed formats in detail as a means for reducing our memory requirements. Once done, we will return to looking at the texture pipeline and see how we can use transformation matrices to animate texture coordinates in real time to produce simple but interesting effects. Finally, we will conclude with a detailed look at the DirectX specific texture and surface types and their associated utility functions.

Key Topics:

- Texture Stages
  - Texture Color
  - Texture Stage States
- Multi-Texturing and Color Blending
- Compressed Textures
  - Compressed Texture Formats
    - Pre-Multiplied Alpha
  - Texture Compression Interpolation
  - Compressed Data Blocks – Color/Alpha Data Layout
- Texture Coordinate Transformation
- The IDirect3DTexture Interface
- The IDirect3DSurface Interface
- D3DX Texture Functions

Projects:

Lab Project 6.2: Terrain Detail Texturing
Lab Project 6.3: Scene Texturing
Lab Project 6.4: GDI and Textures
Lab Project 6.5: Offscreen Surfaces

Exams/Quizzes: NONE

Recommended Study Time (hours): 10 – 12

Lesson 11: Alpha Blending

Textbook: Chapter Seven (pgs. 451 – 505)

Goals:

In this lesson we will examine an important visual effect in games: transparency. Transparency requires that students understand the concept of alpha blending and as such we will talk about various places alpha data can be stored (vertices, materials, textures, etc.) and how what various limitations and benefits are associated with this choice. We will then explore the alpha blending equation itself and look at how to configure the transformation and texture stage pipelines to carry out the operations we desire. We will also examine alpha testing and alpha surfaces for the purposes of doing certain types of special rendering that ignores specific pixels. We will conclude our alpha blending discussion with a look at the all important notion of front to back sorting and rendering, examining various algorithms that we can use to do this. Finally, we will wrap up the lesson with an examination of adding fog to our rendering pipeline. This will include both vertex and pixel fog, how to set the color for blending, and the three different formulas available to us (linear/exponential/exponential squared) for producing different fogging results.

Key Topics:

- Alpha Components
  - Vertex Alpha – Pre-Lit/Unlit Vertices
  - Material Alpha
  - Constant Alpha + Per-Stage Constant Alpha
  - Texture Alpha
- The Texture Stage Alpha Pipeline
- Frame Buffer Alpha Blending
- Transparent Polygon Sorting
  - Sorting Algorithms and Criteria
    - Bubble Sort/Quick Sort/Hash Table Sort
- Alpha Surfaces
- Alpha Testing
- Fog
  - Enabling Fog and Setting the Fog Color
  - Vertex/Pixel Fog
  - Fog Factor Formulas
    - Linear/Exponential/Exponential Squared

Projects:

Lab Project 7.1: Vertex Alpha
Lab Project 7.2: Alpha Testing
Lab Project 7.3: Alpha Sorting
Lab Project 7.4: Texture Splatting

Exams/Quizzes: NONE

Recommended Study Time (hours): 10 - 12

Lesson 12: Exam Preparation and Course Review

Textbook: NONE

Goals:

In this final lesson we will leave the student free to prepare for and take their final examination. Multiple office hours will be held for student questions and answers.

Key Topics: NONE

Projects: NONE

Exams/Quizzes: NONE

Recommended Study Time (hours): 15 - 20

Final Examination

The final examination in this course will consist of 50 multiple-choice and true/false questions pulled from all of the textbook chapters. Students are encouraged to use the lecture presentation slides as a means for reviewing the key material prior to the examination. The exam should take no more than two hours to complete. It is worth 65% of student final grade.