1. 3D 그래픽 기초(3D Graphics Fundamentals)

(1) 3D 그래픽

① 컴퓨터 구조(Computer Architecture)

최신 컴퓨터는 3D 그래픽 처리를 할 수 있는 강력한 그래픽 카드(Graphic card, 비디오 카드(Video card))를 가지고 있다. 우리는 그래픽 카드의 기능을 사용한 3D 게임 프로그램을 작성하기 위하여 Direct3D를 이해하고 활용하는 것을 배운다. 윈도우 10 운영체제에서 Direct3D 12 API를 기반으로 하며 프로그래밍 언어는 C++을 사용한다.

② 게임 프로그래밍(Game programming)

실세계의 일반적인 문제를 컴퓨터 프로그램으로 해결할 때의 과정은 문제에 대한 이해와 분석, 문제의 표현, 알고리즘 작성, 프로그래밍 언어를 사용하여 구현하는 것이다. 게임을 컴퓨터 프로그램으로 개발하는 과정도 유사하게 게임의 설계, 게임의 표현, 알고리즘 작성, 프로그래밍 언어를 사용하여 구현하는 것이다. 컴퓨터 프로그램의 작성에서 중요한 것은 문제의 표현, 알고리즘 작성, 구현 과정이 모두 수학적인 방법(궁극적으로 숫자로)으로 이루어져야 한다는 것이다.

③ 2D 게임과 3D 게임의 구분

- 2D 게임이란?
- 3D 게임이란?
- 2D 게임과 3D 게임을 구별할 수 있는가? 구별할 수 있다면 어떻게, 무엇으로?

④ Direct3D 게임 프로그래밍

Direct3D 게임 프로그래밍은 2학기 과정으로 구성된다. 1학기 과정에서는 다음과 같은 기초적인 내용을 다룬다. 2학기 과정에서는 Direct3D의 고급 내용과 애니메이션 등을 다룬다.

- 3D 그래픽 파이프라인(Graphics Pipeline)
  - 변환(Transformation)
  - 조명 처리(Lighting)
  - 텍스쳐 처리(Texturing)
- 3D 그래픽을 위한 수학
  - 벡터(Vector)
  - 행렬(Matrix)
- 게임 프레임워크(Game Framework)
  - 윈도우 프로그래밍(Windows Programming)
  - 게임 프레임워크에 대한 이해와 구현
- Direct3D 파이프라인(Pipeline)
  - Direct3D 그래픽 파이프라인
- 쉐이더 프로그래밍(Shader Programming)

- 쉐이더 언어(HLSL: High Level Shader Language)
- 쉐이더(Vertex Shader, Pixel Shader, Geometry Shader, …)

이 과정을 이수하기 위하여 윈도우 프로그래밍(Window Programming), C++ 프로그래밍, 그리고 벡터와 행렬에 대한 수학적 기초가 필요하다.

⑤ 기초 개념

3D 게임을 개발하기 위하여 다음과 같은 처리를 하는 프로그램을 작성해야 한다.

- 사용자 입력(User Input)
- 자원 관리(Resource Management)
- 그래픽 로딩과 렌더링(Loading and Rendering Graphics)
- 스크립트 해석과 실행(Interpreting and Executing Scripts)
- 음향 처리(Playing Sound Effects)
- 인공지능(Artificial Intelligence)

이 과정에서는 3D 그래픽 처리에 대한 내용을 집중적으로 다룰 것이다. 일반적으로 3D 그래픽 처리를 담당하는 프로그램의 요소를 렌더링 엔진(Rendering Engine)이라고 한다. 렌더링 엔진은 여러 가지 게임 엔진의 핵심 모듈(Module)이며 렌더러(Renderer)라고 부르기도 한다. 렌더러는 가상적인 게임 세계의 3차원 표현을 모니터 화면(스크린, Screen)에 2차원 영상으로 그려내는 역할을 한다. 이러한 처리를 하기 위하여 수학적인 3차원 표현과 수학적인 처리 방법이 필요하다. 렌더링(Rendering)이란 수학적으로 표현된 3차원 세상을 수학적인 처리 방법을 통하여 2차원 영상(Image)으로 그리는 것을 말한다.

⑥ 그림 그리기 알고리즘(Painter's Algorithm)

3D 렌더링의 과정은 실세계에서 화가가 그림을 그리는 과정과 아주 유사하다. 실세계에서 화가는 3차원 세상을 눈으로 보고 종이에 그림을 그린다. 종이에 그림을 그리는 것은 종이를 구성하는 영역에 색칠은 하는 것이다. 단, 제대로 그림을 그렸다면 우리는 종이에 그려진 2차원 이미지를 보고 원래의 3차원 세상을 잘 느낄 수 있을 것이다.

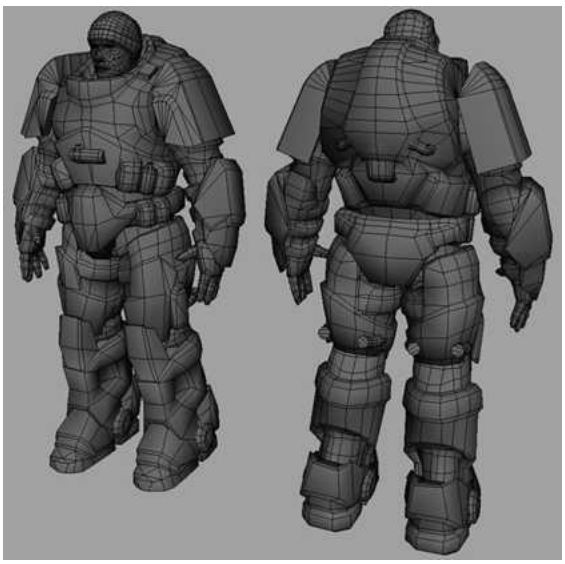
3D 렌더링의 과정은 3차원으로 표현된 게임 세상을 가상적인 카메라를 통하여 화면에 2차원 영상를 그려야 하며, 게임 플레이어는 이 모니터 화면(영상)을 보고 3차원 게임 세상을 잘 느낄 수 있어야 한다.

⑦ 기하학적 모델링(Geometric Modeling)

메쉬(Mesh)는 물체의 외관을 표현하기 위한 연결된 다각형(Polygon)들의 집합이다. 다각형(Polygon)은 연결된 선분들의 집합이고, 선분은 연결된 점들의 집합이다. 면(Face)은 메쉬를 구성하는 각 다각형이다. 가장 간단한 다각형은 삼각형이고 모든 다각형은 삼각형의 집합으로 나눌 수 있으므로 메쉬를 삼각형들의 집합으로 정의할 수 있다. 직육면체는 6개의 직사각형 면을 가지고 하나의 직사각형은 2개의 직각삼각형으로 나눌 수 있으므로 직육면체의 외관은 12개의 연결된 삼각형으로 표현할 수 있다.

우리는 렌더링할 대상 객체의 외관(Outer Hull) 또는 피부(Skin)에만 관심이 있다. 왜냐하면 실세계에서 어떤 3차원 물체의 외관(겉)만 볼 수 있으므로 객체의 보이지 않는 부분은 다루지 않아도 된다.

메쉬는 렌더링할 대상 객체의 외관을 표현하는 삼각형들의 집합이다. 삼각형은 꼭지점 3개로 표현할 수 있으므로 메쉬를 객체의 외관을 표현하는 구조화된(연결된) 점들의 집합으로 정의할 수 있다. 이렇게 3차원 물체의 외관을 메쉬 즉, 점들의 집합으로 표현하는 것을 기하학적 모델링(Geometric modeling)이라 하고, 메쉬를 기하학적 모델, 간단하게 모델, 또는 모델 메쉬라고 한다. 메쉬를 구성하는 점들은 정점(Vertex)이라고 한다. 실제 모델에는 메쉬를 구성하는 점들 뿐 아니라 다른 정보들이 포함된다. 일반적으로 메쉬를 포함하는 모델의 기하학적 정보는 렌더링을 하는 과정에서 바뀌지 않는다. 수학적으로 3차원 세상의 한 점을 표현하기 위하여 실수 3개가 필요하므로 모델이란 3차원 물체의 외관을 표현하는 실수들의 집합(배열)이라 생각할 수 있다. 이러한 실수들의 집합을 시각적으로 렌더링하면 다음 그림과 같을 것이다.



보통 이러한 모델 메쉬는 3D MAX 또는 Maya와 같은 3D 모델링 도구를 사용하여 제작되어 파일의 형태로 저장된다. 게임 프로그램 또는 게임 엔진은 이러한 모델 파일을 읽어 화면에 렌더링한다. 3D 모델링 도구를 사용하여 제작한 모델 파일의 내용을 자세히 살펴보면 궁극적으로 실수들의 집합일 것이다.

모델 또는 메쉬의 모든 면이 한꺼번에 보이는 것은 아니다. 직육면체의 경우 6개의 직사각형 면에서 한 번에 최대 3개까지의 면을 볼 수 있다. 어떤 순간에 보이는 면을 정면(Front face)이라 하고 보이지 않는 면을 은면(후면, Back face, Hidden face)이라 한다.

⑧ 좌표계(Coordinate System)

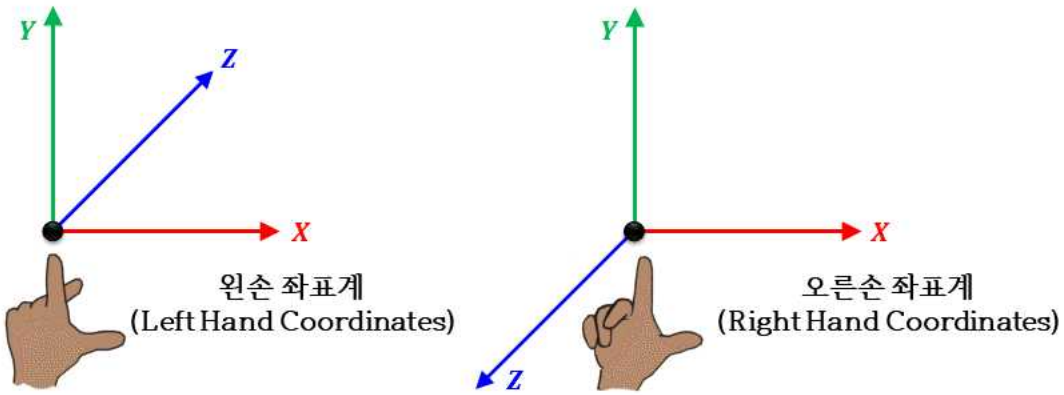
메쉬를 기하학적으로 표현하기 위해서는 좌표계가 필요하다. 가장 이해하기 쉽고 일반적인 좌표계는 직교좌표계(Cartesian coordinate system)이다. 2차원 직교좌표계는 직교하는 두 개의 좌표축과 좌표축의 교점에 해당하는 좌표계의 원점(Origin)으로 정의할 수 있다.

모니터 디스플레이 화면은 2차원 직교좌표계로 표현할 수 있다. 모니터 디스플레이 화면을 정의하는 좌표계를 화면좌표계 또는 화면좌표계(Screen Coordinates System)라고 한다. 그러나 수학 또는 실생활에서 사용하는 일반적인 직교좌표계와는 다르게 화면좌표계의 좌표축의 방향과 원점이 다르게 정의된다. 디스플레이 화면의 좌측 상단은 화면좌표계의

원점이 되고 좌표축의 방향은 원점에서 오른쪽으로 가면서 x-축이 증가하고, 원점에서 아랫방향으로 가면서 y-축이 증가한다.

⑨ 3차원 좌표계

3차원 공간을 표현하기 위하여 직교좌표계를 사용한다. 3차원 직교좌표계는 직교하는 3개의 좌표축과 좌표축의 교점인 원점으로 정의할 수 있다. 그래픽에서 일반적으로 사용하는 3차원 좌표계는 다음 그림과 같이 왼손좌표계 또는 오른손좌표계 중 하나이다. Direct3D에서는 왼손좌표계를 사용한다. 왼손좌표계와 오른손좌표계의 차이는 z-좌표축의 방향이 다르다는 것이다.



3차원 좌표계에서 한 점은 (x, y, z)의 형태로 표현할 수 있다. 그리고 3차원 좌표계에서 점, 다각형, 메쉬를 표현하기 위한 자료구조는 다음과 같이 C-언어를 사용하여 표현할 수 있다.

```
struct Vertex
{
    float       x;
    float       y;
    float       z;
};

struct Polygon
{
    UINT        nVertices;
    Vertex      *pVertices;
};

struct Mesh
{
    UINT        nFaces;
    Polygon     *pFaces;
};
```

⑩ 모델 좌표계(Model Space)와 월드 좌표계(World Space)

메쉬를 표현하기 위해 3차원 좌표계는 모델 좌표계(Model space)와 월드 좌표계(World space)로 구분할 수 있다. 모델 좌표계는 모델 메쉬의 점을 표현하기 위한 좌표계이다. 각

모델마다 자체적인 별도의 좌표계를 갖고 있다고 가정한다. 보통 모델 메쉬의 중심이 모델 좌표계의 원점이 되며 메쉬의 각 점들은 이 원점에 상대적인 좌표로 표현된다. 객체의 위치와 방향을 표현하는 좌표계를 객체 좌표계(Object space)라고 한다. 객체 좌표계는 보통 지역 좌표계(Local space)라고도 한다. 모델 좌표계의 원점과 객체 좌표계의 원점이 같으면 보통 모델 좌표계와 객체 좌표계는 같다. 모델을 표현하기 위해 모델 좌표계를 사용하는 이유는 모델 좌표계로 표현된 모델을 동일한 외관을 가진 여러 객체들이 공유하기 쉽기 때문이다.

월드 좌표계는 게임 세계에 존재하는 객체들의 위치와 방향을 표현하기 위한 좌표계이다. 월드 좌표계는 게임 세계 전체를 하나의 통일된 좌표계로 표현하여 배치하기 위한 좌표계이며 모든 객체들에 적용될 수 있는 전역 좌표계(Global Coordinates System)이다.

객체 또는 게임 객체들은 월드 좌표계를 사용하여 위치와 방향이 표현되고 모델은 모델 좌표계를 사용하여 표현된다. 렌더링의 과정에서 게임 객체를 렌더링하면 게임 객체의 외관을 표현한 모델이 그려지므로 게임 객체는 모델의 인스턴스(Instance)라고 할 수 있다.

⑪ 메쉬와 텍스쳐 매핑(Texture Mapping)

대부분의 실세계 게임 객체의 모델을 완벽하게 다각형의 집합으로 표현할 수 없다. 실세계 게임 객체의 겉 표면은 연속적으로 울퉁불퉁할 수 있고, 미세한 질감을 표현하기 위하여 작은 다각형을 많이 사용하면 렌더링의 시간이 많이 걸리기 때문에 적당한 개수의 다각형을 사용하여 모델을 표현하는 것이 일반적이다. 그리고 다각형 표면의 질감을 표현하기 위하여 렌더링을 할 때 다각형의 표면에 2D 이미지(Image)를 입히는(그리는) 방법을 사용한다. 이러한 과정을 텍스쳐 매핑이라고 하며 사용하는 2D 이미지를 텍스쳐 맵(Texture map) 또는 텍스쳐라고 한다. 3D 그래픽에서 맵(Map)은 2D 이미지를 의미하며 매핑은 2D 이미지 사용하여 어떤 처리를 하는 것을 의미한다. 다음은 3D 그래픽에서 사용하는 여러 가지 맵의 예이다.

- 높이 맵(Height Map)
- 텍스쳐 맵(Texture Map)
- 법선 맵(Normal Map)
- 범프 맵(Bump Map)
- 조명 맵(Light Map)

⑫ 게임 월드 단위(Game World Units)

개발자는 게임 월드의 크기와 단위를 결정하기 위해 그래픽 아티스트와 협력해야 한다. 즉, 월드 좌표계의 1단위와 모델 좌표계의 1단위를 가급적 일치하는 것이 필요하다. 모든 객체가 일관된 크기로 만들어진다면 문제가 없다.

⑬ 은면 제거(Back Face Culling)

은면 제거는 렌더링을 할 때 관찰자(카메라)가 볼 수 없는 면을 그리지 않는 것이다. 은면이란 다각형의 면이 관찰자를 향하지 않는 면을 의미한다. 즉, 다각형의 면이 관찰자에게 보이지 않는 면이 은면이다. 은면은 결과적으로 화면에 보이지 않을 것이므로 렌더링의 대상에서 제외하는 것이 렌더링의 속도를 높일 것이다. 그러므로 3D 렌더링 과정에서 모델의

다각형을 그리기 전에 다각형을 구성하는 각 면이 은면인 가를 빠르게 판단해야 한다.

다각형을 구성하는 정점들을 나열하는 순서를 와인딩 순서(Winding order)라고 하며, 은면 제거를 위하여 와인딩 순서를 사용한다. 와인딩 순서는 메쉬를 구성하는 선분(Edge)들이 어떻게(어떤 순서로) 연결되는 가를 나타낸다. 와인딩 순서는 시계방향(Clockwise)과 반시계방향(Anticlockwise) 중 하나를 사용한다. 시계방향 와인딩 순서는 어떤 면을 정면에서 바라볼 때 그 면을 구성하는 모든 정점들이 시계방향으로 순서대로 나열되는 것을 의미한다. 와인딩 순서에서 정점의 시작은 문제가 되지 않고 방향이 중요하다. 모델을 구성하는 모든 면(다각형)의 와인딩 순서는 같아야 한다. Direct3D에서 모델을 구성하는 다각형들은 기본적으로 시계방향 와인딩 순서를 갖는 것으로 가정한다.

직육면체의 경우 6개의 직사각형의 정점들을 각 직사각형이 보일 때를 기준으로 시계방향을 나열하여 모델을 표현했다면, 어떤 순간에 보이지 않는 은면에 해당하는 직사각형을 구성하는 정점들은 반시계방향이 된다. 즉, 모델에 표현된 정점들의 나열 순서가 시계방향인 면들은 전면이 되고 반시계방향인 면들은 은면이 된다. 객체의 전면과 은면은 객체 또는 카메라가 회전을 하면 바뀔 수 있음에 주의해야 한다.

⑭ 씬(Scene, 장면)과 렌더링

씬이란 게임 월드 자체 또는 게임 월드에서 현재 화면에 그려져야 하는 부분(카메라에 현재 보이는 게임 객체들)을 의미한다. 게임 월드는 게임 객체(메쉬)들의 구조화된 집합이며 배경, 이펙트, 조명, 카메라 등도 게임 객체로 취급한다.

씬을 렌더링하는 것은 게임 월드의 게임 객체들을 그리는 것이다. 이것은 결과적으로 게임 객체의 외관을 표현하는 모델의 모든 다각형들을 그리는 것이고, 화면에서 다각형을 구성하는 모든 픽셀들을 그리는 것(색칠하는 것)이다. 이러한 과정은 크게 2가지 단계를 연속적으로 거치게 된다. 첫 번째 단계에서 모델의 다각형들을 구성하는 3D 점들을 화면의 2D 픽셀들로 바꾸는 것이 필요하다. 이렇게 다각형의 3D 모델 좌표를 2D 화면 좌표(픽셀)로 변환하는 과정을 변환(Transformation)이라고 한다. 두 번째 단계에서 모델의 다각형에 해당하는 화면의 픽셀의 색상을 결정하여 색칠한다. 각 픽셀의 색상을 결정하기 위하여 조명 계산(Lighting) 또는 텍스쳐 매핑 등을 사용한다.

렌더링을 흔히 T&L(Transformation and Lighting)이라고 한다.

⑮ 화면 좌표계에서 그리기

변환 과정에서 모델의 다각형들을 구성하는 각 정점들을 화면의 2D 픽셀 좌표로 바꾸었다면 모델의 각 다각형에 대응되는 픽셀들을 선분으로 이어서 화면에 다각형을 그릴 수 있다. 예를 들어, 화면 좌표계에서 다각형을 그리기 위하여 윈도우 API의 MoveTo(), LineTo() 함수를 사용할 수 있다.

⑯ 변환 파이프라인(Transformation Pipeline)

3D 모델 좌표를 2D 화면 좌표(픽셀)로 변환하는 과정을 변환 파이프라인이라고 한다. 변환 파이프라인은 기본적으로 다음과 같이 4개의 구별되는 순차적인 단계(Stage)로 구성된다. 모델의 한 정점을 화면의 한 점(픽셀)으로 변환할 수 있으면 모든 정점들을 변환하는 것은 반복을 하면 될 것이다. 그러므로 변환 파이프라인에서 모델의 한 정점을 화면의 한 점(픽셀)으로 변환하는 과정을 살펴보자.

- 월드 변환(World Transform) 또는 월드 좌표 변환
  정점(모델 좌표계)을 월드 좌표계로 변환하는 과정이다.
- 카메라 변환(View Transform) 또는 카메라 좌표 변환
  월드 좌표계의 점을 카메라 좌표계로 변환하는 과정이다.
- 투영 변환(View Transform) 또는 투영 좌표 변환
  카메라 좌표계의 점을 투영 좌표계로 변환하는 과정이다.
- 화면 변환(Screen Transform) 또는 화면 좌표 변환
  투영 좌표계의 점을 화면 좌표계로 변환하는 과정이다.

변환 파이프라인을 3D 메쉬 표현을 2D 표현으로 변환하여 컴퓨터 화면에 그릴 수 있도록 하는 함수(Function)라고 이해하자. 변환 파이프라인은 2D 화면에서 3D 게임 세상을 느낄 수 있도록(Illusion) 변환해야 한다. 2D 화면에서 3D 게임 세상을 느낄 수 있으려면 원근감의 표현이 되어야 한다. 카메라에 가까운 물체는 크게 보이고 멀리 있는 물체는 작게 보여야 한다. 인간은 실세계를 보고 그림을 그릴 때 이 과정을 거친다.