다이렉트 X

#3. 다이렉트 X를 이용한 3차원 그래픽 프로그래밍 기초

이강훈
서울대학교 전기컴퓨터공학부
3MAP 연구실

Part I : 기본 개념

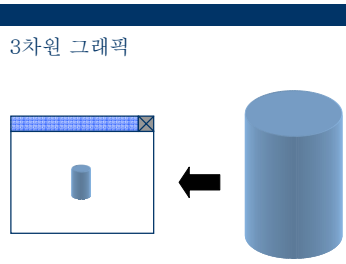
- 3차원 그래픽의 기본 개념
  - 3차원 모델
  - 카메라
  - 빛
  - 렌더링

3차원 그래픽

- 2차원 그래픽과 3차원 그래픽
  - 2차원 그래픽
    - 화가가 캔버스 위에 그림 그리는 작업
  - 3차원 그래픽
    - 사진 작가가 카메라로 대상을 촬영하는 작업

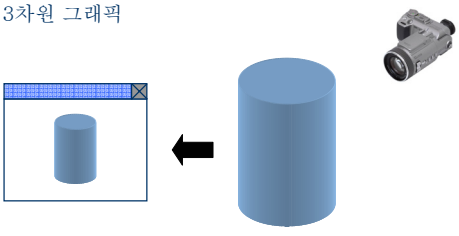
3차원 그래픽

- 3차원 그래픽



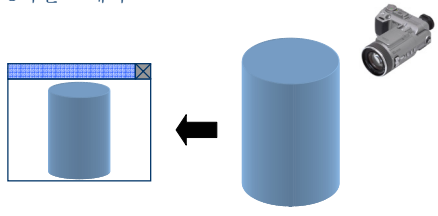
3차원 그래픽

- 3차원 그래픽



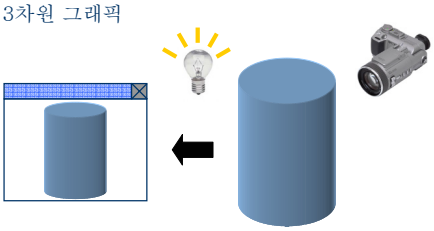
3차원 그래픽

- 3차원 그래픽



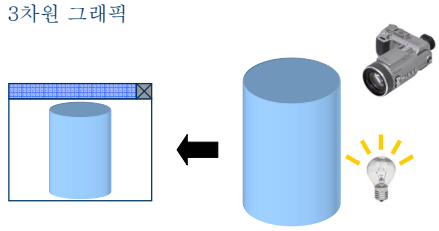
3차원 그래픽

- 3차원 그래픽



3차원 그래픽

- 3차원 그래픽



3차원 그래픽

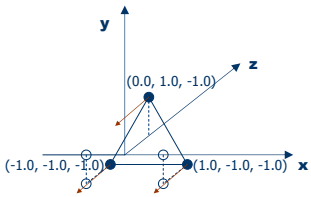
- 3차원 그래픽
  - 최종 장면을 결정짓는 세 가지 요소
    - 물체(촬영할 대상)
    - 빛
    - 카메라

3차원 그래픽

- 3차원 그래픽
  - 프로그래머가 해야 할 세 가지 작업
    - 물체를 배치한다
    - 빛을 배치한다
    - 카메라를 배치한다
    - 찍는다!!!

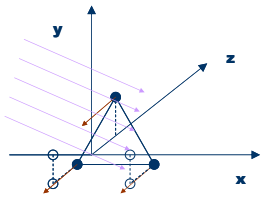
3차원 그래픽

- 1단계: 물체를 배치한다
  - 2차원 그래픽과 마찬가지로 버텍스와 프리미티브로 구성



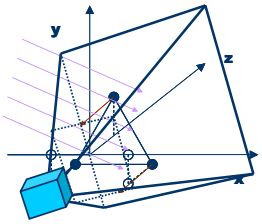
3차원 그래픽

- 2단계: 빛을 배치한다
  - 빛의 위치, 방향, 색깔 등을 결정



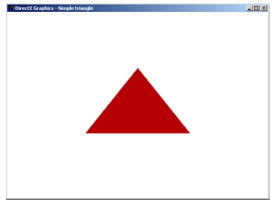
3차원 그래픽

- 3단계: 카메라를 배치한다
  - 카메라의 위치, 방향, 폭 등을 결정



3차원 그래픽

- 4단계: 찍는다!!!
  - 2차원 그래픽의 render() 함수와 동일



3차원 그래픽

- 코딩
  - 2차원 그래픽과 거의 비슷한 틀을 가짐
  - 세부적인 내용보다는 전체적인 구성에 대한 이해

Part II : 빛

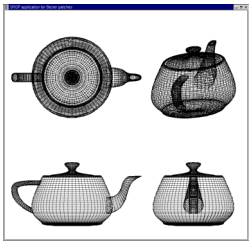
- 빛
  - 다이렉트X의 기본 3차원 객체 사용하기
    - 주전자
  - 빛 넣기
    - 빛의 종류
      - 선 광원
      - 점 광원
      - 스팟 광원
    - 빛과 재질
    - 하이라이트 효과

다이렉트X의 기본 3차원 객체 사용

- D3DX 모양 그리기 함수
  - D3DXCreateBox
  - D3DXCreateCylinder
  - D3DXCreatePolygon
  - D3DXCreateSphere
  - ***D3DXCreateTeapot***
  - D3DXCreateText
  - D3DXCreateTorus

다이렉트X의 기본 3차원 객체 사용

- D3DXCreateTeapot



다이렉트X의 기본 3차원 객체 사용

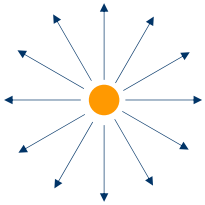
- 실습: 주전자 넣기
  1. 주전자 담을 변수 선언
     ID3DXMesh* d3dx_teapot = 0;
  2. 주전자 만들기
     D3DXCreateTeapot(d3d9_device, &d3dx_teapot, NULL);
  3. 주전자 그리기
     d3dx_teapot->DrawSubset(0);
  4. 주전자 휴지통에 버리기
     if(d3dx_teapot) { d3dx_teapot->Release(); }

다이렉트X의 기본 3차원 객체 사용

- 결과

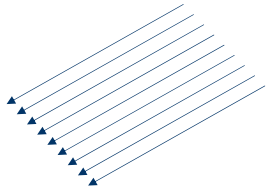
빛 넣기

- 빛의 종류
  - 점 광원(Point light)



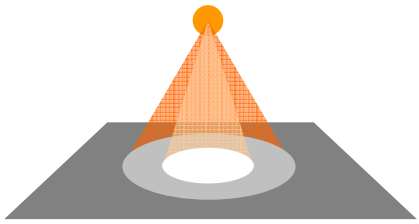
빛 넣기

- 빛의 종류
  - 선 광원(Directional light)



빛 넣기

- 빛의 종류
  - 스팟 광원(Spotlight)



빛 넣기

- 빛 구조체

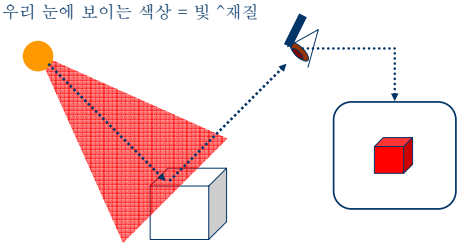
```
typedef struct _D3DLIGHT9 {
    D3DLIGHTTYPE Type;
    D3DCOLORVALUE Diffuse;
    D3DCOLORVALUE Specular;
    D3DCOLORVALUE Ambient;
    D3DVECTOR Position; D3DVECTOR Direction;
    float Range; float Falloff;
    float Attenuation0; float Attenuation1; float Attenuation2;
    float Theta; float Phi;
} D3DLIGHT9;
```

빛 넣기

- 실습: 선 광원 추가해보기
  1. D3DLIGHT9 구조체의 내용을 채운다
     - 빛의 종류 : D3DLIGHT_DIRECTIONAL
     - 빛의 색상 : D3DXCOLOR(r, g, b, a)
     - 빛의 방향 : D3DXVECTOR3(x, y, z)
  2. 완성된 D3DLIGHT9 구조체를 첫 번째 빛으로 설정한다
     `d3d9_device->SetLight( 0, &light );`
  3. 첫 번째 빛을 켠다
     `d3d9_device->LightEnable( 0, TRUE );`

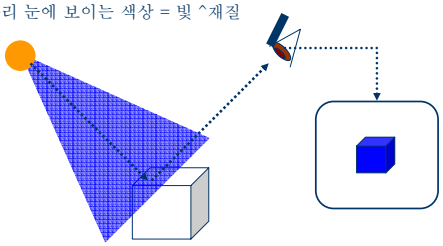
빛 넣기

- 빛과 재질
  - 우리 눈에 보이는 색상 = 빛 ^재질



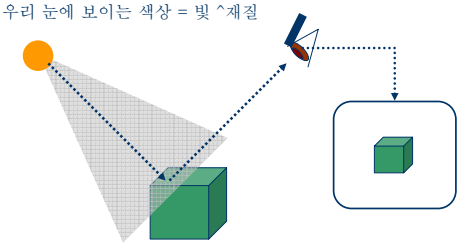
빛 넣기

- 빛과 재질
  - 우리 눈에 보이는 색상 = 빛 ^재질



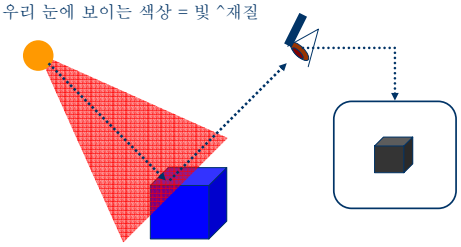
빛 넣기

- 빛과 재질
  - 우리 눈에 보이는 색상 = 빛 ^재질



빛 넣기

- 빛과 재질
  - 우리 눈에 보이는 색상 = 빛 ^재질



빛 넣기

- 재질 구조체

```
typedef struct _D3DMATERIAL9 {
    D3DCOLORVALUE Diffuse;
    D3DCOLORVALUE Ambient;
    D3DCOLORVALUE Specular;
    D3DCOLORVALUE Emissive;
    float Power;
} D3DMATERIAL9;
```

빛 넣기

- 실습: 선 광원 추가해보기 (계속)
  4. D3DMATERIAL9 구조체의 내용을 채운다
     - 재질의 색상 : D3DXCOLOR(r, g, b, a)
  5. 완성된 D3DMATERIAL9 구조체를 재질로 설정한다

```
d3d9_device->SetMaterial( &material );
```

직접 해 볼 것!!

- 해보기 1 : 점 광원 추가해보기
  - D3DLIGHT9 구조체의 내용만 바꿔준다
    - 빛의 종류 → 바뀐다 : D3DLIGHT_POINT
    - 빛의 색상 → 그대로
    - 빛의 방향 → 없앤다
    - 빛의 위치 → 추가 : D3DXVECTOR3(x, y, z)
    - 빛의 범위 → 추가 : Range
    - 빛의 감쇠도 → 추가 : Attenuation0~2

직접 해 볼 것!!

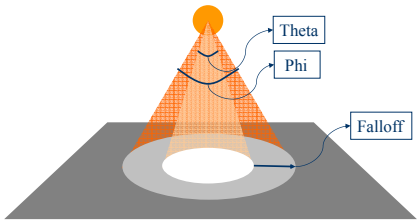
- 해보기 2 : 하이라이트 효과 만들어보기
  1. 하이라이트 효과를 켠다

```
d3d9_device->SetRenderState( D3DRS_SPECULARENABLE, TRUE );
```

  2. D3DMATERIAL9 구조체의 Power 변수 조절

직접 해 볼 것!!

- 해보기 3 : 스팟 광원 추가해보기



직접 해 볼 것!!

- 해보기 4 : 여러 개의 빛
  - 빛의 인덱스를 바꿔가며 구조체를 할당한다

```
D3DLIGHT9 light0, light1;
// 구조체 내용 채우기
d3d9_device->SetLight( 0, &light0 );
d3d9_device->SetLight( 1, &light1 );
```

  - 모든 빛을 켠다

```
d3d9_device->LightEnable( 0, TRUE );
d3d9_device->LightEnable( 1, TRUE );
```

Part III : 카메라

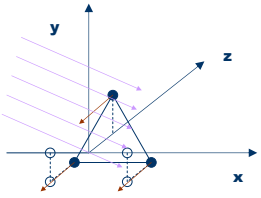
- 카메라
  - 카메라의 내부 설정
  - 카메라의 외부 설정
  - 카메라 애니메이션

카메라

- 내부 설정과 외부 설정
  - 내부 설정: 렌즈 조작
  - 외부 설정: 카메라 배치

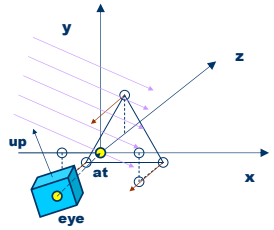
카메라

- 내부 설정과 외부 설정
  - 1단계: 3차원 객체 및 빛 배치



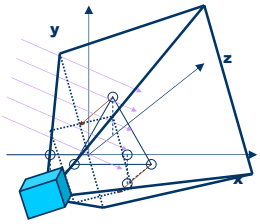
카메라

- 내부 설정과 외부 설정
  - 2단계: 카메라 외부 설정



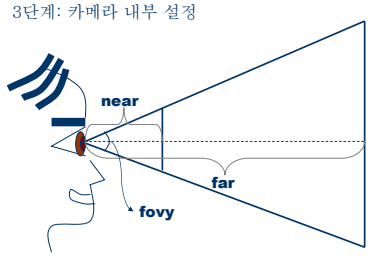
카메라

- 내부 설정과 외부 설정
  - 3단계: 카메라 내부 설정



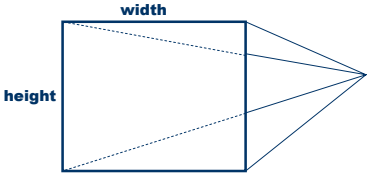
카메라

- 내부 설정과 외부 설정
  - 3단계: 카메라 내부 설정



카메라

- 내부 설정과 외부 설정
  - 3단계: 카메라 내부 설정
    - aspect_ratio = height / width



카메라

- 실습: 개념 이해
  - 기본 카메라 배치 : 외부 설정

```
D3DXMATRIX matView;
D3DXMatrixLookAtLH(
    &matView,
    &D3DXVECTOR3(eye_x, eye_y, eye_z),      // 카메라의 위치
    &D3DXVECTOR3(at_x, at_y, at_z),         // 대상의 위치
    &D3DXVECTOR3(up_x, up_y, up_z)          // 카메라 상향 벡터
);
d3d9_device->SetTransform( D3DTS_VIEW, &matView );
```

카메라

- 실습: 개념 이해
  - 기본 카메라 배치 : 내부 설정

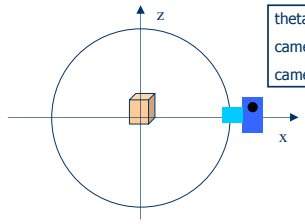
```
D3DXMATRIX matProjection;
D3DXMatrixPerspectiveFovLH(
    &matProjection,
    fovy,                   // 시야 각도
    aspect_ratio,           // 시야의 가로/세로 비율
    z_near,                 // 시야의 최소 거리
    z_far                   // 시야의 최대 거리
);
d3d9_device->SetTransform( D3DTS_PROJECTION, &matProjection );
```

카메라

- 실습: 개념 이해
  - 외부 설정 변경
    - 카메라 위치 변경
    - 대상 위치 변경
  - 내부 설정 변경
    - 시야의 범위 변경
    - 시야의 가로/세로 비율 변경
    - 시야의 최소/최대 거리 변경

카메라

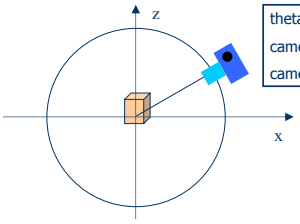
- 실습: 애니메이션
  - 물체 주위를 회전하는 카메라



theta = 0˚
camera_x = distance * cosf(0˚);
camera_z = distance * sinf(0˚);

카메라

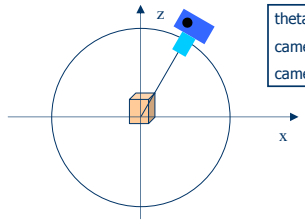
- 실습: 애니메이션
  - 물체 주위를 회전하는 카메라



theta = 30˚
camera_x = distance * cosf(30˚);
camera_z = distance * sinf(30˚);

카메라

- 실습: 애니메이션
  - 물체 주위를 회전하는 카메라



theta = 60˚
camera_x = distance * cosf(60˚);
camera_z = distance * sinf(60˚);

카메라

- 실습: 애니메이션
  - 전역 변수 선언
    - camera_x, camera_z (camera_y???)
    - theta, distance
  - WM_TIMER
    - theta 값 증가 / 창 invalidate
  - 주의 사항
    - 메시지 처리 부분에서 직접 render() 함수를 호출하지 않는다!!!
    - setupCamera() 함수를 render() 함수 내부에서 호출한다!!!
    - D3DXToRadian() : 도(degree) 단위 → 레디안(radian) 단위

직접 해 볼 것!!!

- 해 보기
  - 키보드 인터페이스를 이용한 카메라 조작
    - 물체 주위를 회전하는 카메라 코드 확장
    - WM_KEYDOWN 메시지 처리
    - 화살표 좌/우 방향키: 회전 각도 조절 (theta)
    - 화살표 상/하 방향키: 거리 조절 (distance)