3D Game Programming 03

afewhee@gmail.com

- Graphic Pipe Line
- Frame Work
- Input System
- Extension Utility
- 실습

1. Graphic Pipe Line

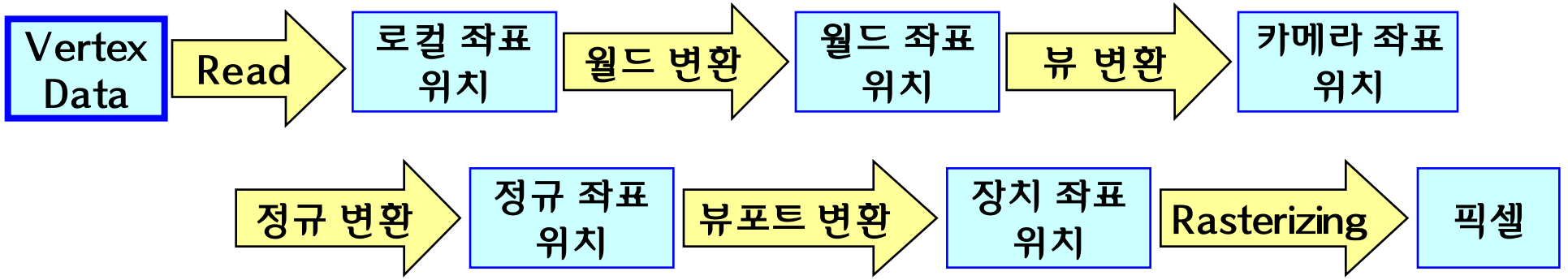
- Graphic Pipe Line = Rendering Pipe Line
  - 3차원 기하 정보를 2차원 픽셀로 변환하는 과정
  - 2단계로 진행
    - 정점 처리 과정(Vertex Processing)
    - 픽셀 처리 과정(Pixel Processing)

- 정점 처리 과정 (Vertex Processing)
  - 그래픽에 대한 기하 정보(Geometry)를 이용해서 픽셀 처리과정에 필요한 픽셀을 생성
  - 정점의 변환 → Rasterizing 과정을 거침

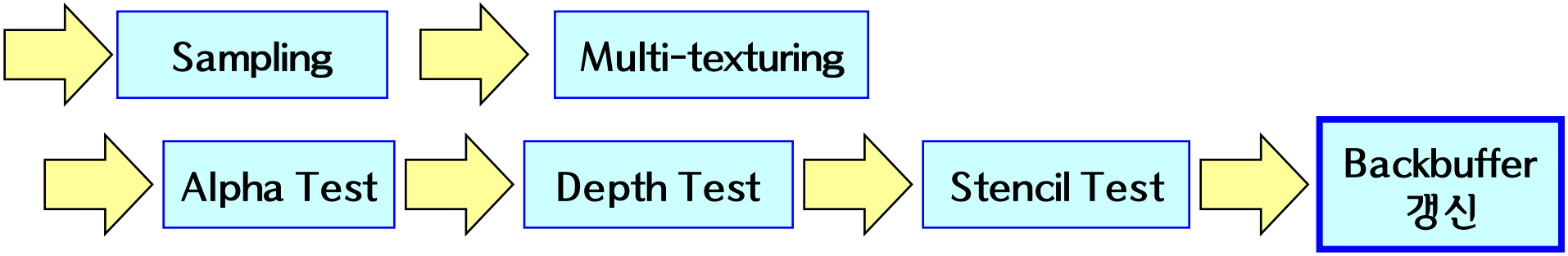
- 정점의 변환
  - 월드 변환 (World Transform)
    - 3D 장면 연출을 위해서 Geometry Object를 월드 좌표계로 변환
  - 뷰잉 변환 (Viewing Transform)
    - 월드 좌표계의 Geometry Object를 카메라 공간 좌표계로 변환
  - 정규 변환(Norma Transform) = 투영 변환 (Projection Transform)
    - 장치에 의존하지 않는 정규화로 카메라 공간의 Geometry의 정보를 투영 평면으로 변환.
    - 카메라의 View Frustum 안에 있는 정점은 정규 변환을 거치면 [-1,1] 사이의 범위를 가짐
      ➔Direct3D (-1,-1,0) ~ (1,1,1), OpenGL (-1,-1,-1) ~ (1,1,1)
  - 뷰포트 변환 (Viewport Transform)
    - 해당 장치(Device) 윈도우로 변환 ➔ 장치 의존 변환
    - 클리핑 수행

- Rasterizing 과정
  - 뷰포트 변환을 거친 기하정보를 이용해서 픽셀을 만듦

Vertex Processing

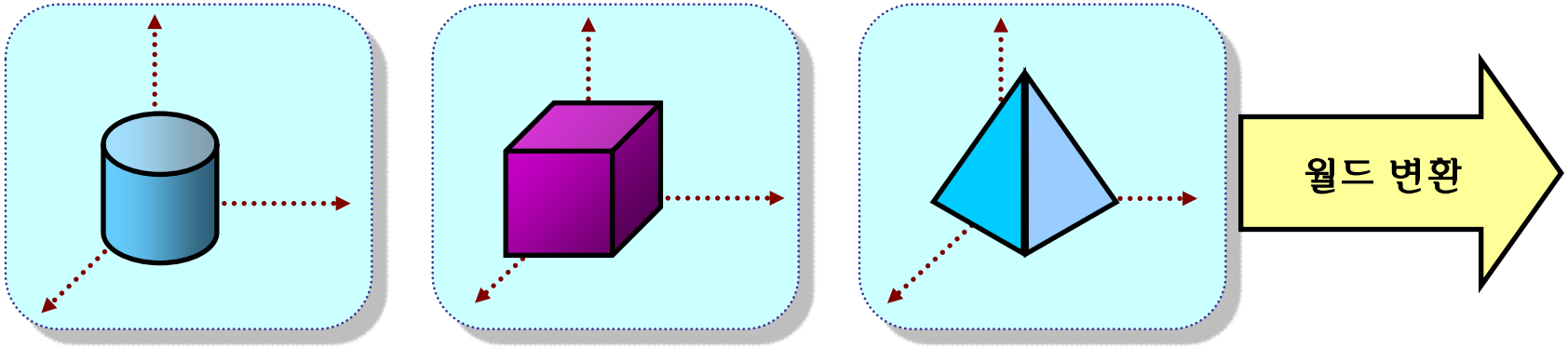


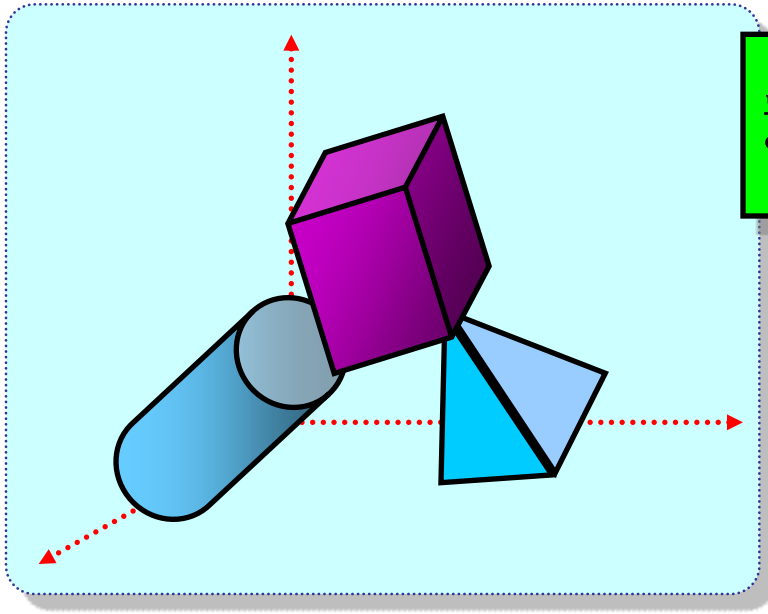
Pixel Processing



- 월드 변환 (World Transform)

각각의 로컬 좌표계(모델 좌표계)에 만들어진 Geometry





모델 좌표계에서 만들어진 Geometry는 크기, 회전,
이동 변환을 거쳐 월드 좌표계의 정점으로 변환

모델 좌표계: Geometry Object를 구성하기 위한 좌표계

월드 좌표계: 3D 장면을 연출 하기 위한 좌표계

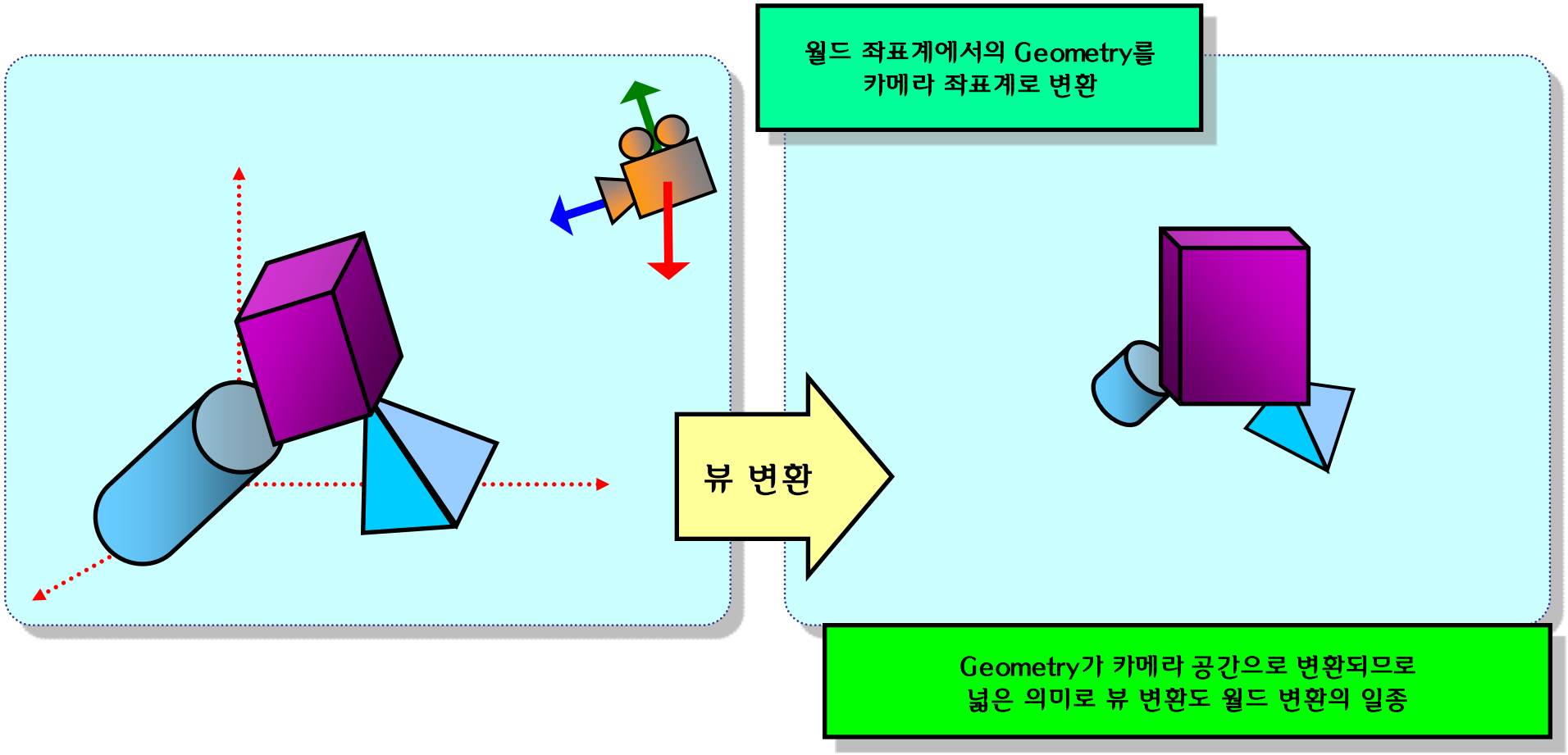
월드 변환 행렬 적용

- 정점의 위치 자체를 이동하는 경우
  - D3DXVec3TransformCoord()
  - D3DXVec3TransformNormal()
  - 장점: 정점의 위치는 곧, 월드 좌표계의 위치와 일치
  - 단점 같은 모델을 사용하는 경우라면 정점의 수 만큼 변환에 대한 연산 필요 ➔ 하드웨어 가속을 사용 못함

- 장치(Device)의 출력에만 영향을 주는 경우
  - 렌더링 상태 머신의 값 설정
  - pDevice->SetTrasform(D3DTS_WORD, &World_Matrix)
  - 장점: 하나의 모델을 가지고 월드 행렬만 바꾸어서 장치에 여러 번 출력 ➔ 하드웨어 가속
  - 단점: 정점들의 위치는 절대 변하지 않아 해당 Geometry의 Scaling, 회전, 위치 값을 가지고 있어야 함

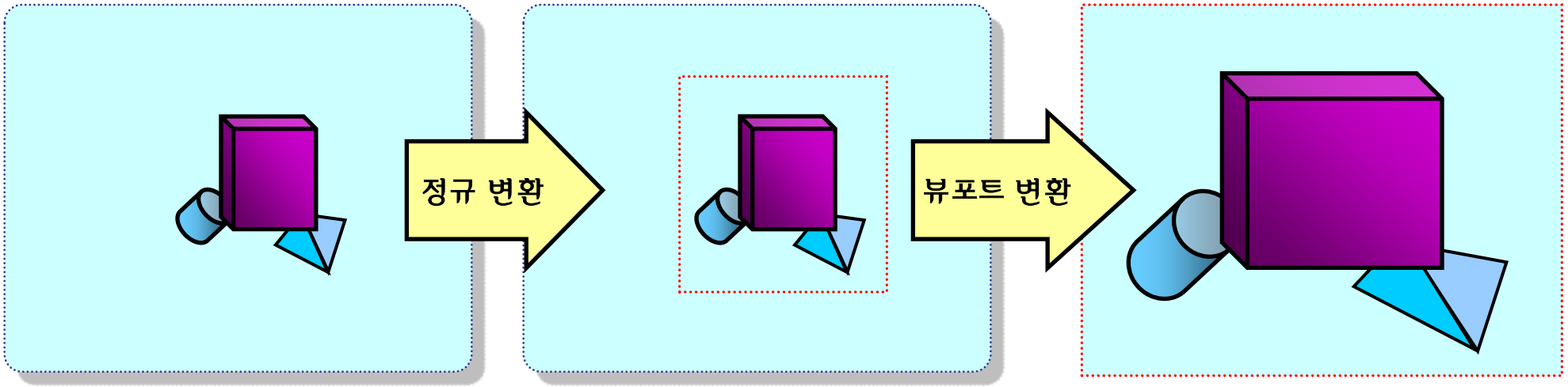
- 뷰잉 변환 (View Transform) ➔ Camera 좌표계로 변환

- 뷰 변환 행렬 적용
  - 정점의 위치 자체를 이동: D3DXVec3TransformCoord()
  - 상태 머신 설정: pDevice->SetTrasform(D3DTS_VIEW, &View_Matrix)

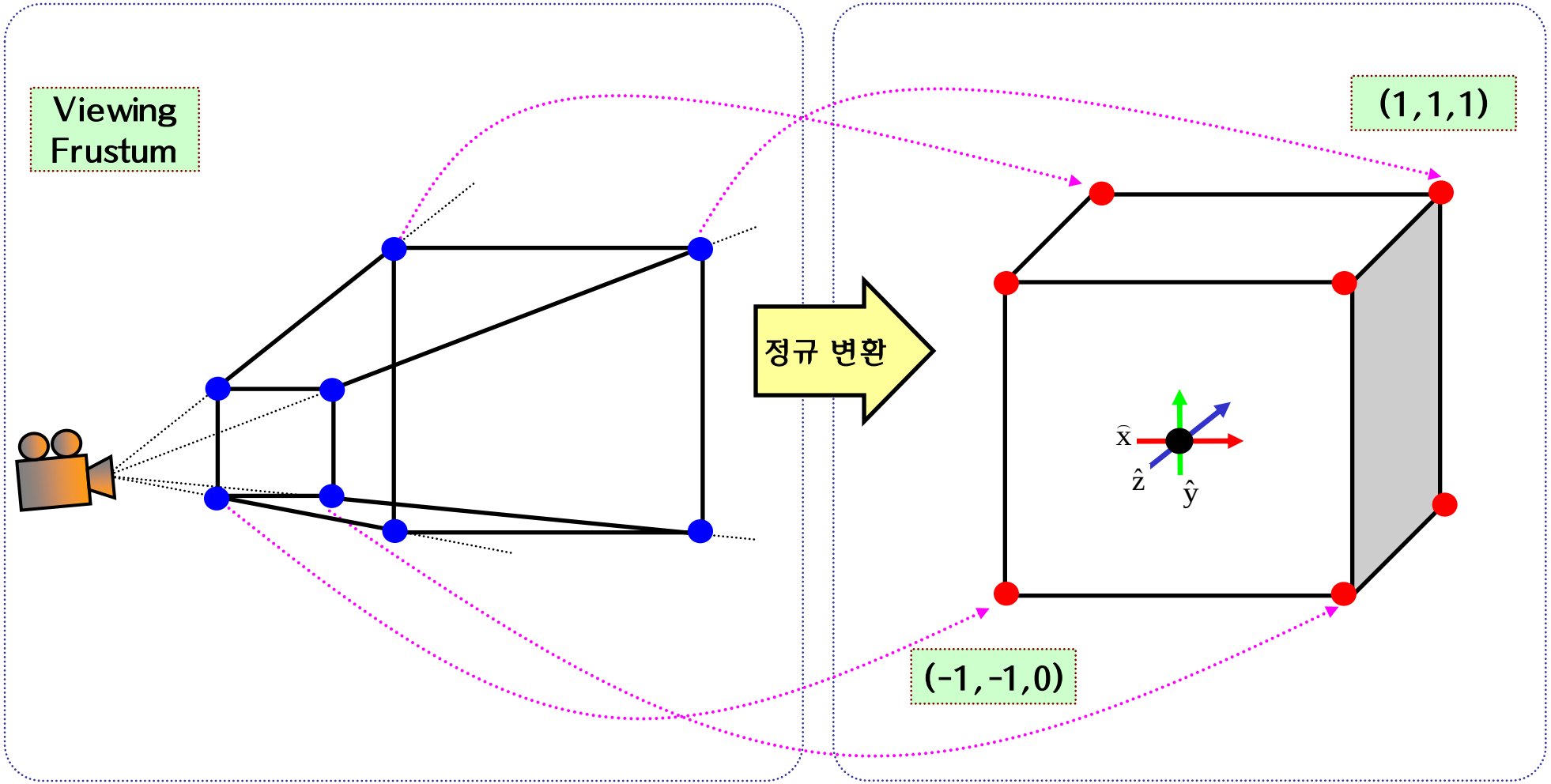


- 정규 변환 (View Transform) ➔ 투영 변환(Projection Transform)
  - Camera 좌표계의 Geometry를 투영 평면으로 변환
  - 투영행렬을 적용하면 카메라의 뷰 체적(View Volume) 안의 Geometry는 [-1,1] 범위의 값을 가짐
  - 장치에 독립
  - pDevice->SetTrasform(D3DTS_PROJECTION, &Projection_Matrix)

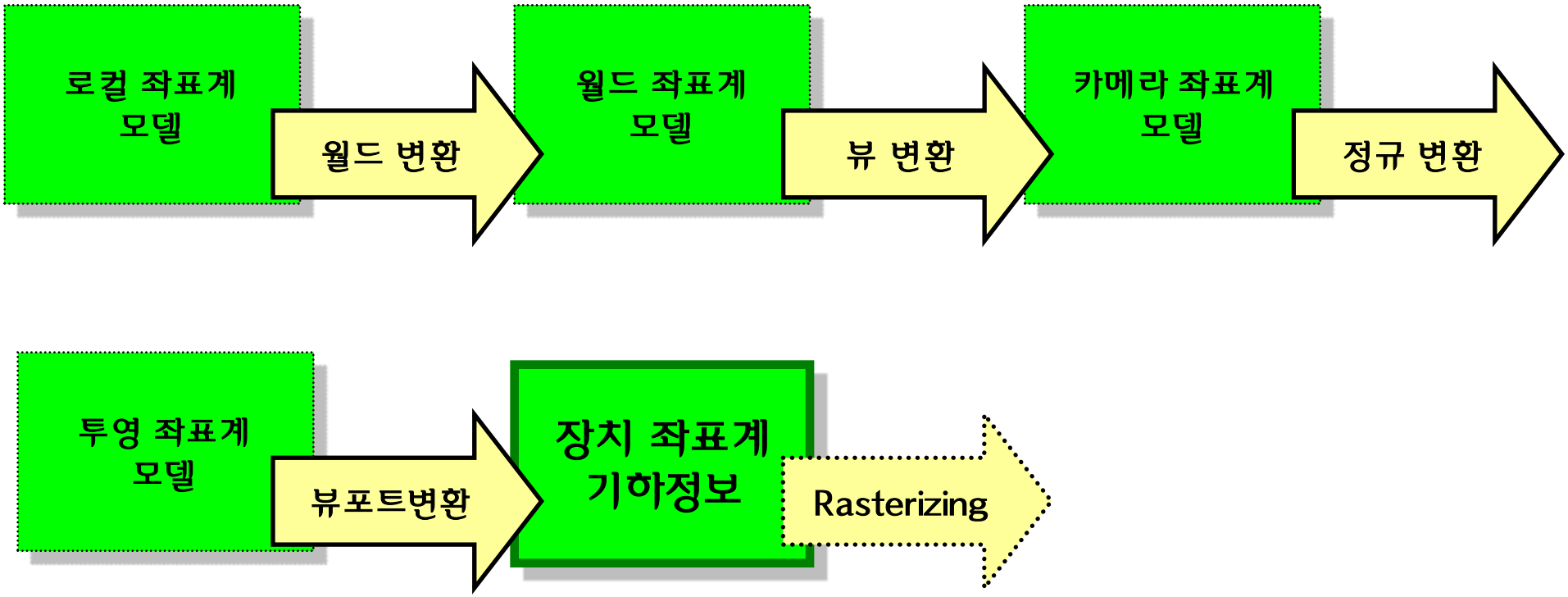
- 뷰포트 변환
  - 해당 윈도우의 클리핑 영역에 맞게 변환
  - Direct3D는 뷰포트를 설정하지 않으면 Device의 가로, 세로 폭이 해당 화면의 폭과 높이로 설정
  - pDevice->SetViewport()



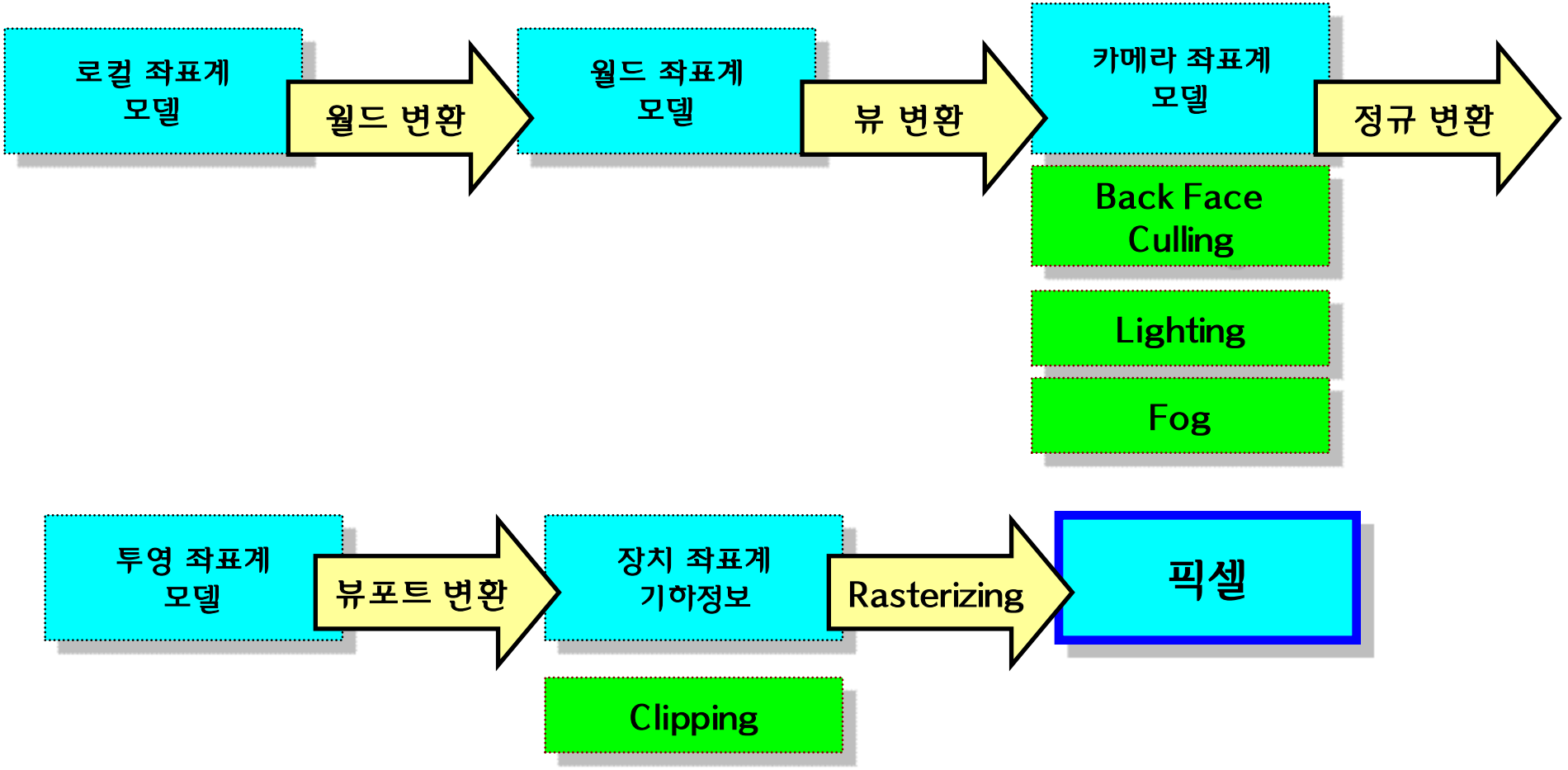
Direct3D의 뷰 체적(View Volume)과 정규변환



Graphic Pipe Line의 Geometry 변환

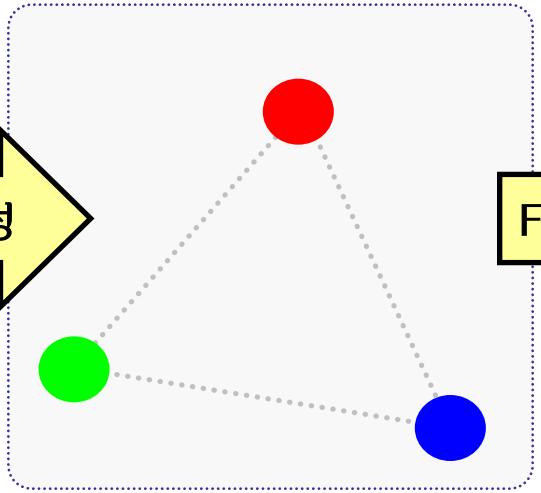


Graphic Pipe Line의 Vertex Processing 병행 작업



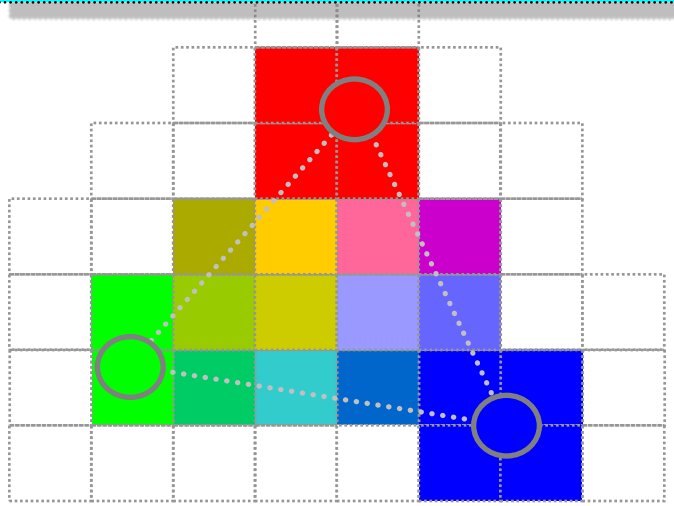
Rasterizing

Primitive 구성

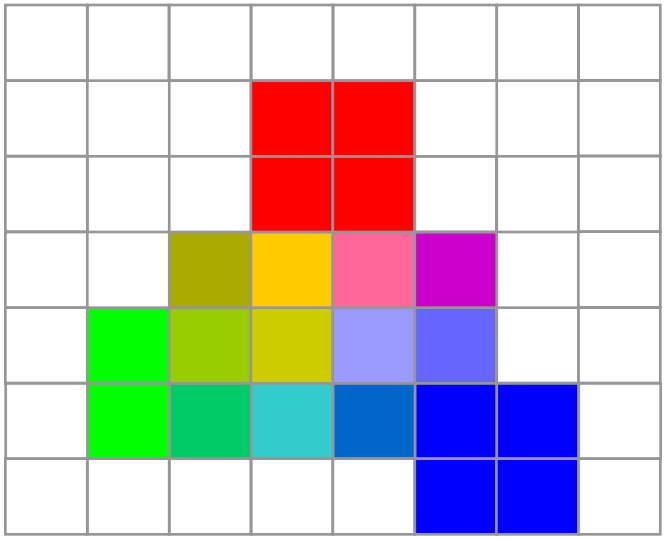


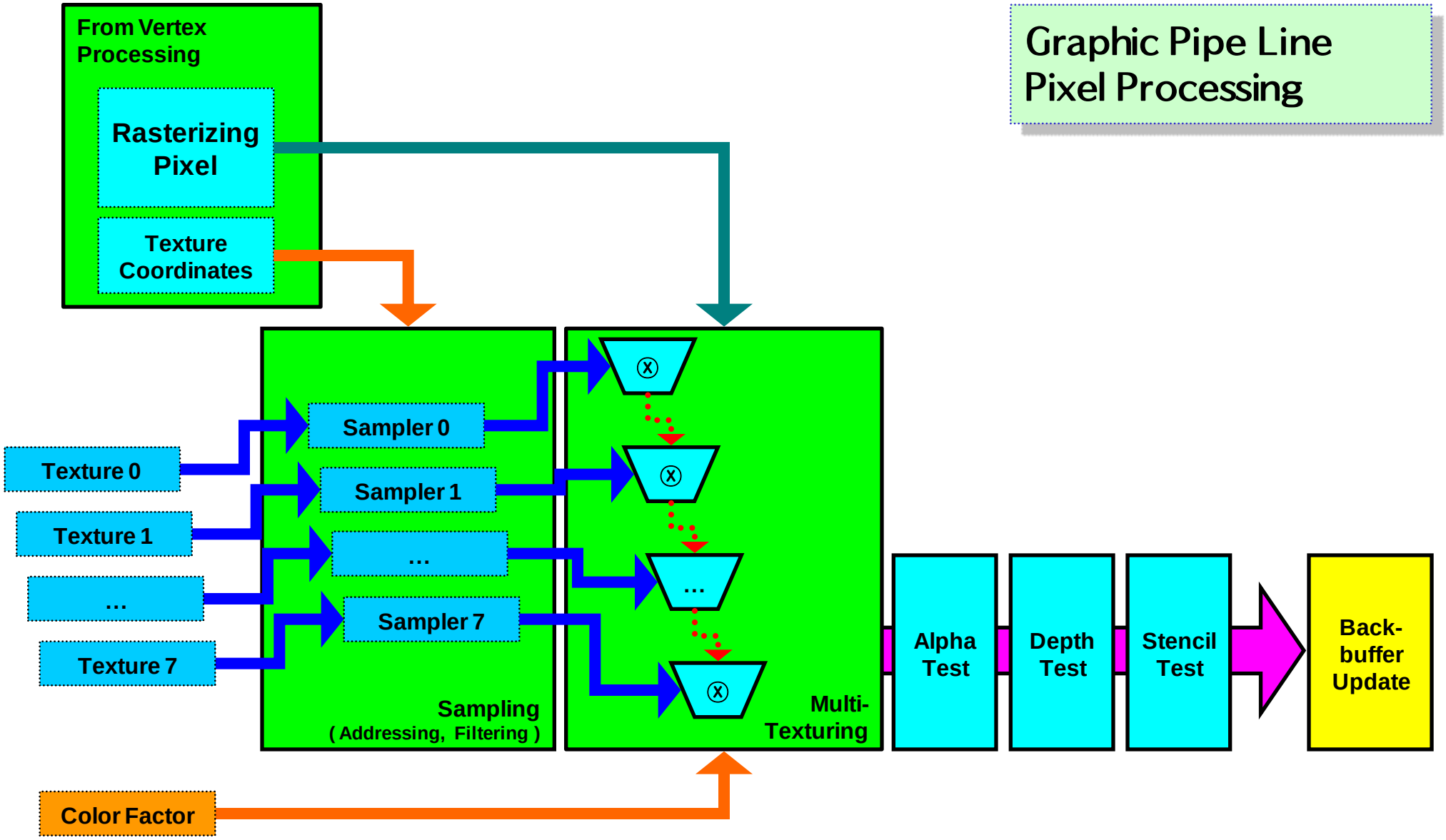
Fragment 구성

Pixel 구성을 위한 Sampling: 필요에 따라
Anti-aliasing을 위해 Super Sampling이 적용

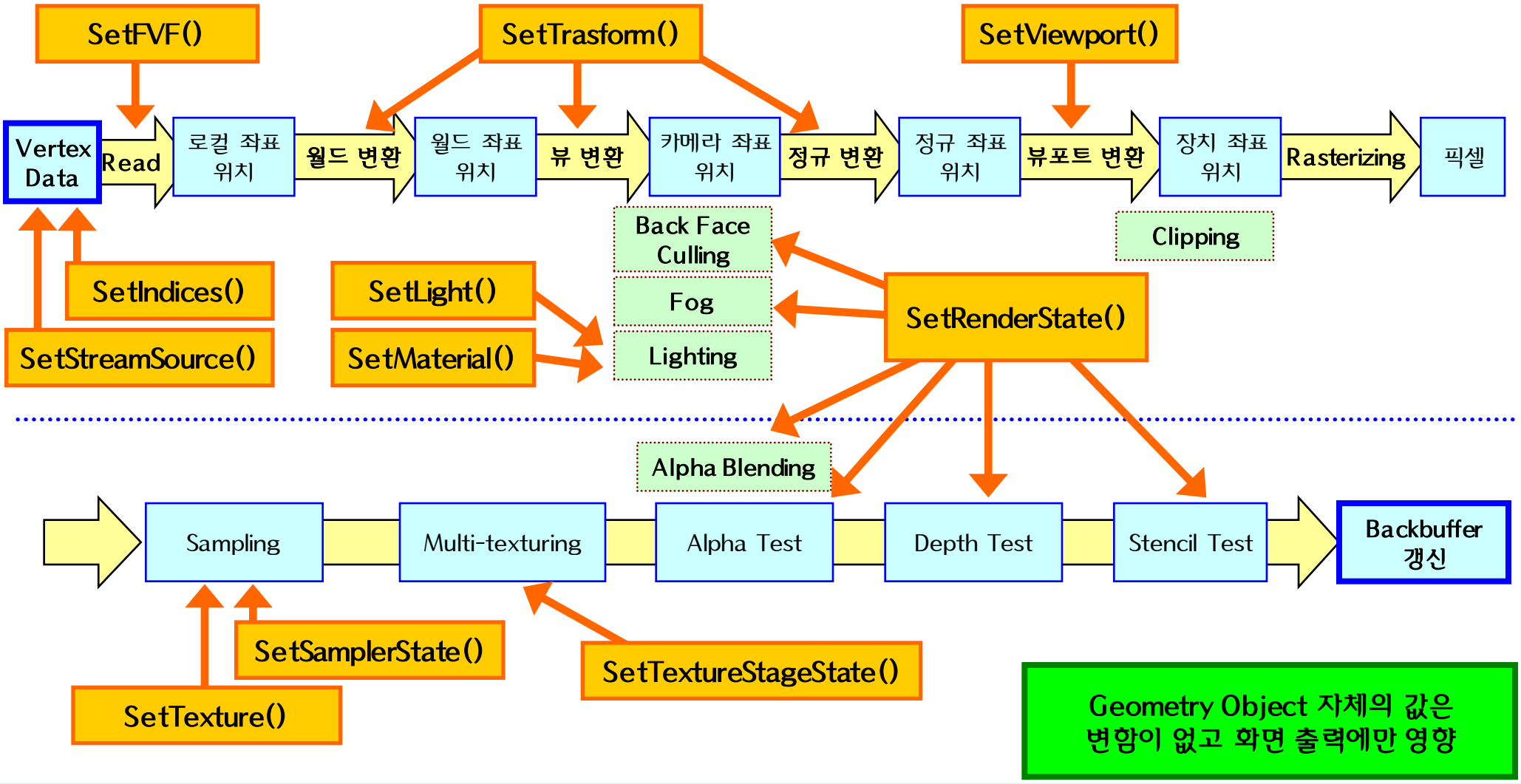


Pixel 구성





Direct3D 렌더링 가상 머신 상태 설정 함수



- Sample Frame Work
  - Wizard code 변경 ➔ Sample 코드 참조

- API Input
  - Keyboard: GetAsyncKeyState()&0x8000, GetKeyboardState()&0x80
  - Mouse: GetCursorPos(), ScreenToClient()
  - Wheel Event
    - Message Procedure
    - #define _WIN32_WINNT 0x0400
    - WM_MOUSEWHEEL

- Direct Input
  - DirectX 7.0에 완성 ➔ 8.0이후 거의 변함 없음
- Direct Input 객체들
  - Direct Input
    - LPDIRECTINPUT8 m_pDInput;
  - Direct Input Device
    - Keyboard, Mouse, Joystick 모두 공통 적용
      - LPDIRECTINPUTDEVICE8 m_pKeyboard;
      - LPDIRECTINPUTDEVICE8 m_pMouse;
      - LPDIRECTINPUTDEVICE8 m_pJoyStick;
- D Input 프로그램 순서
  - Direct Input 생성
  - 디바이스 생성
  - 데이터 포멧 설정
  - 협력 레벨(Cooperative Level) 설정
  - 데이터 가져오기

- Direct Input 인스턴스 생성
  - DirectInput8Create(...)
- Device 생성
  - pDirectInput->CreateDevice(GUID_NUMBER, &pDevice,...)
  - GUID_NUMBER:
    - 키보드: GUID_SysKeyboard
    - 마우스: GUID_SysMouse
    - 조이스틱: GUID_Joystick
- Data Format 설정
  - pDevice>SetDataFormat(&DataFormat);
  - Data Format
    - extern 키워드로 정의되어 있음
    - 키보드: c_dfDIKeyboard
    - 마우스: c_dfDIMouse
    - 조이스틱: c_dfDIJoystick

- 협력 레벨 설정(Set Cooperative Level)
  - pDevice->SetCooperativeLevel(해당 작업 윈도우 핸들, Flags);
  - Flags
    - '|' 연산자로 결합해서 사용
    - DISCL_EXCLUSIVE: 장치에 대한 배타적 독점권 행사. 디바이스에 설정된 해당 윈도우의 이벤트를 막음
    - DISCL_NONEXCLUSIVE: 장치에 대한 비 배타적. 가장 일반적인 방법
    - DISCL_FOREGROUND : 해당 윈도우(hWnd)가 활성화 될 때만
    - DISCL_BACKGROUND : 해당 윈도우(hWnd)의 상태에 상관 없이 동작
    - DISCL_NOWINKEY 해당 윈도우(hWnd)의 상태에 상관 없이 동작

- Data 가져오기
  - 데이터 가져오기: pDevice->GetDeviceState(size, buffer);
  - size, buffer
    - 키보드: 256, unsigned char[256];
    - 마우스: sizeof(DIMOUSESTATE), DIMOUSESTATE
    - 조이스틱: sizeof(DIJOYSTATE), DIJOYSTATE
  - 가져오기가 실패하면 데이터를 가져올 수 있도록 디바이스를 조정
    - pDevice->Acquire();

- Keyboard
  - DIK_ ...로 시작
    - ex) DIK_A, DIK_LEFT....
  - VK_ ...와 일치하지 않음

- Mouse
  - 마우스 위치
    - 마우스의 IX, IY, IZ는 이전과 상대적인 값 ➔ 마우스의 위치는 IX, IY, IZ를 누적해서 적용
  - 마우스 버튼
    - L, R, M button: rgbButtons[]의 값 ➔ 순서 주의

- 2D
  - ID3DXSprite
  - ID3DXFont
  - ID3DXLine

- 3D
  - ID3DXMesh: 모델을 구성하는 Mesh를 저장
  - ID3DXBuffer: 데이터 버퍼, 쉐이더 코드의 에러 메시지 등의 저장에 사용

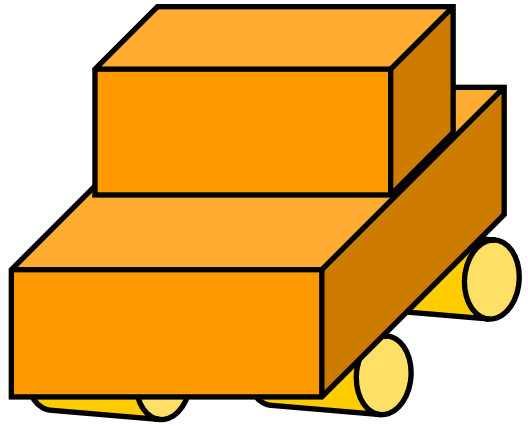
- Shape Drawing Functions
  - ID3DXMesh 객체를 생성
  - 육면체: D3DXCreateBox()
  - 원뿔 및 원통: D3DXCreateCylinder()
  - 다면체: D3DXCreatePolygon()
  - 구: D3DXCreateSphere()
  - 주전자: D3DXCreateTeapot()
  - 고리(링): D3DXCreateTorus()

- Rendering

```
pDevice->SetTransform(D3DTS_WORLD, &Wolrd_Matix);
pMesh->DrawSubset(0);
pDevice->SetTransform(D3DTS_WORLD, &Wolrd_Matix_Identity);
```


- 2D게임을 작성했던 코드의 Direct3D 생성과 해제 부분을 DirectX 에서 제공하는 코드로 전환 하시오.

- Direct3D의 Extension Utility를 이용해서 박스 2, 원기둥 4개로 구성된 자동차를 완성하시오.



- Extension Utility의 구를 가지고 월드 행렬을 이용해서 태양, 지구, 달의 운동을 표현해 보시오.