Real-Time Rendering 4th

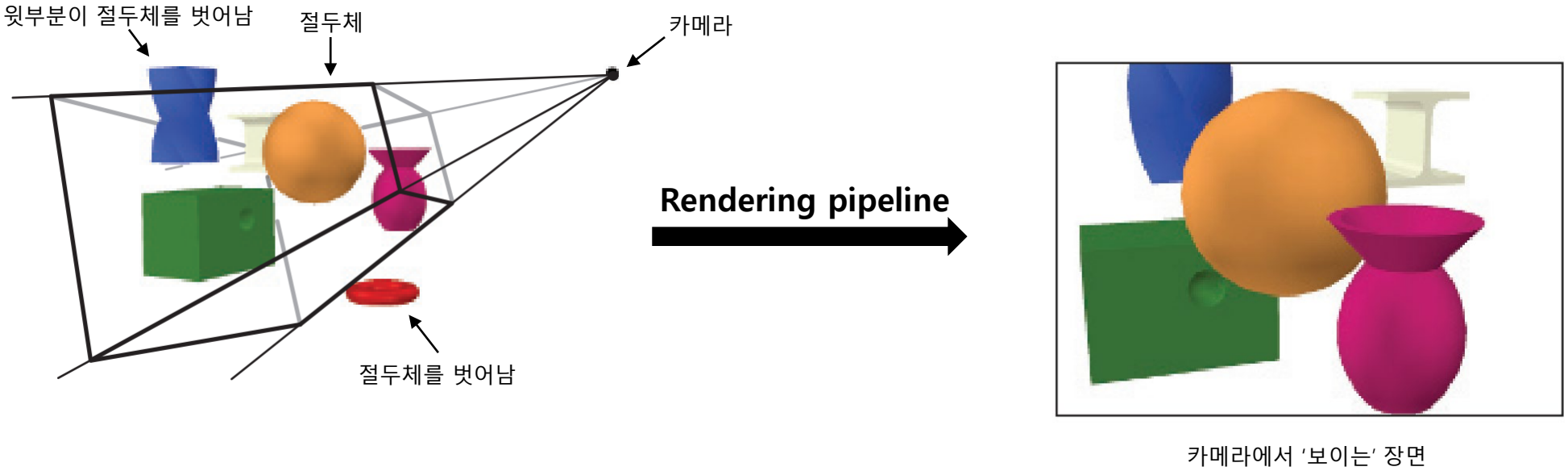
Ch. 2

The Graphics Rendering Pipeline

Ch.2 The Graphics Rendering Pipeline

렌더링 파이프라인? 11p 참조

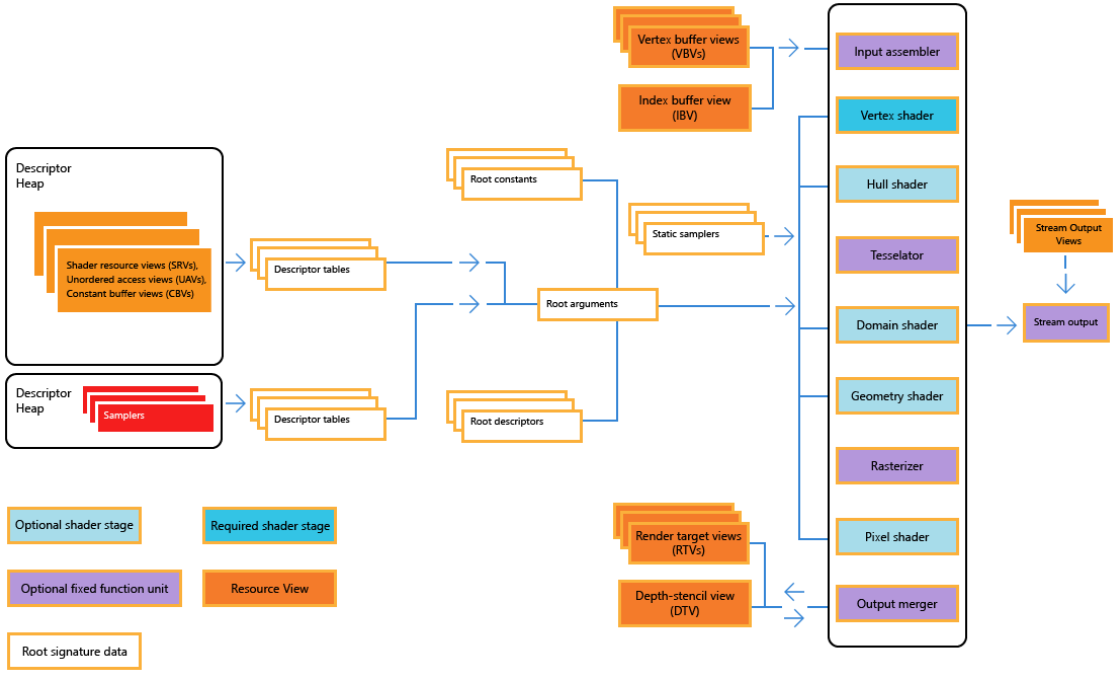
- 실시간 그래픽스의 핵심 구성요소로, 가상의 카메라, 물체, 광원 등이 주어진 3차원 환경을 2차원 이미지로 제작, 생성하는 하나의 방법이다.
- 물체의 모양, 위치는 그 물체의 기하학적 구조와 환경의 특성, 그리고 그 상황을 담아내는 카메라의 위치에 영향을 받는다.
- 물체의 색상과 같은 모습은 물질의 속성, 광원, 텍스쳐(표면에 적용된 이미지), 그리고 셰이딩 공식에 의해 결정된다.



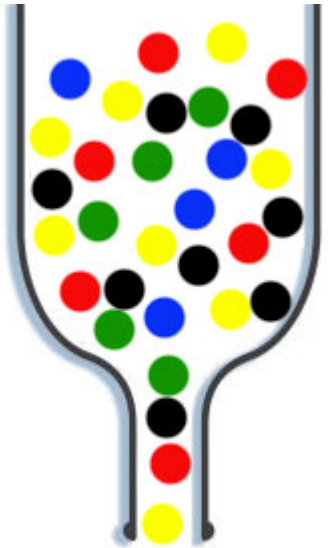
카메라에서 '보이는' 장면

특징 12p 참조

- 렌더링 파이프라인은 병렬로 처리되기 때문에 성능 면에서 이득을 볼 수 있다.
- 다른 과정과 처리시간이 현저하게 차이가 나는 과정에서 병목 현상이 발생할 수 있다.
- 구현 방법이 다양하다.



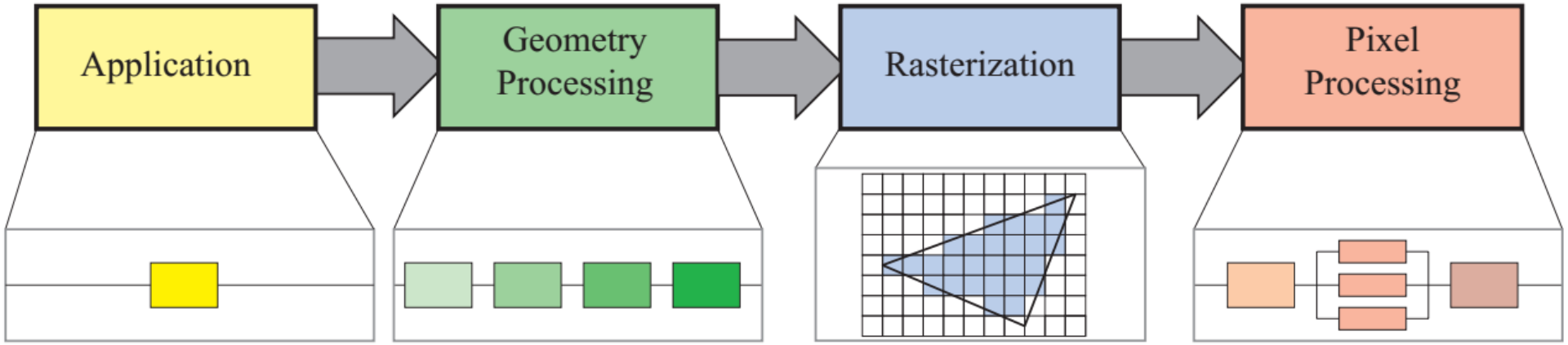
Direct3D 12 그래픽스 파이프라인



병목 현상

구조 개론 13p 참조

- Application stage : 응용프로그램(application)에 의해 동작하는 단계로, 일반적으로 범용 CPU에 의해 프로세스가 실행된다. (물리 시뮬레이션, 모델 애니메이션, 전역 가속 알고리즘 등..)
- Geometry processing stage : 프로그램 상에 존재하는 물체들 등에 대한 변환, 투영등의 모든 기하 처리들이 진행되는 단계로, GPU에 의해 처리된다.
- Rasterization stage : 이 단계에선 일반적으로 3개의 버텍스를 하나의 삼각형이라 간주하고, 그 삼각형이 그려질 모든 픽셀들을 고려하여 이들을 다음 단계로 넘긴다.
- Pixel stage : 프로그램이 픽셀 단위로 실행되며, 해당 픽셀의 색상을 정하고, 그것이 가시적인지에 대한 깊이 테스트를 수행한다. 또한 새로 계산된 색상을 이전의 색상과 섞는 것과 같은 작업 역시 할 수 있다.



응용프로그램 단계 13p~14p 참조

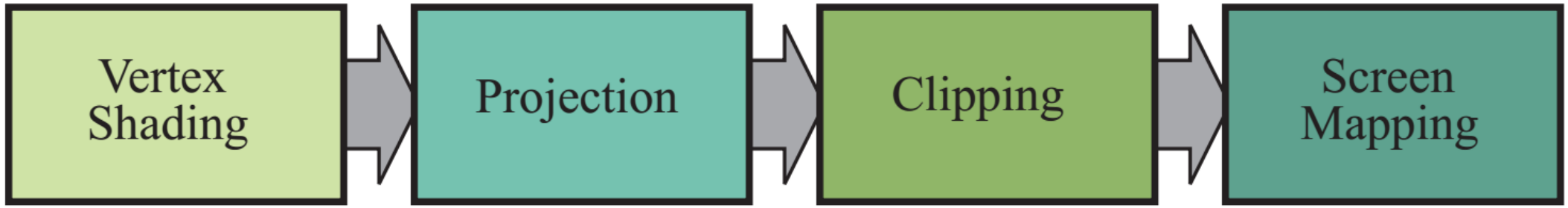
- 주로 CPU(일부는 GPU) 상에서 실행되며, 구현, 수정이 용이하다. 또한, 이후 스테이지에 영향을 미칠 수 있다.
- 일부는 Compute shader라 불리우는 별도의 방법을 사용해 GPU에서 처리 될 수 있다.
- 성능 향상 때문에 몇몇 프로세서 코어들에 의해 병렬로 처리 되는데, CPU 설계에선 이를 superscalar 구조라고 부른다.
- 렌더될 도형(점, 선, 삼각형 등의 렌더링 프리미티브)들을 다음 단계인 기하 처리 단계로 전송하는데, 이 부분이 본 단계에서 가장 중요한 작업이다.

Compute Shader
GPU에서 처리되는 프로그램으로,
렌더링 파이프라인과는 별개로 동작한다.

Superscalar
명령어 파이프라인을 여러 개를 두어
명령어를 동시에 실행하는 기능이다.

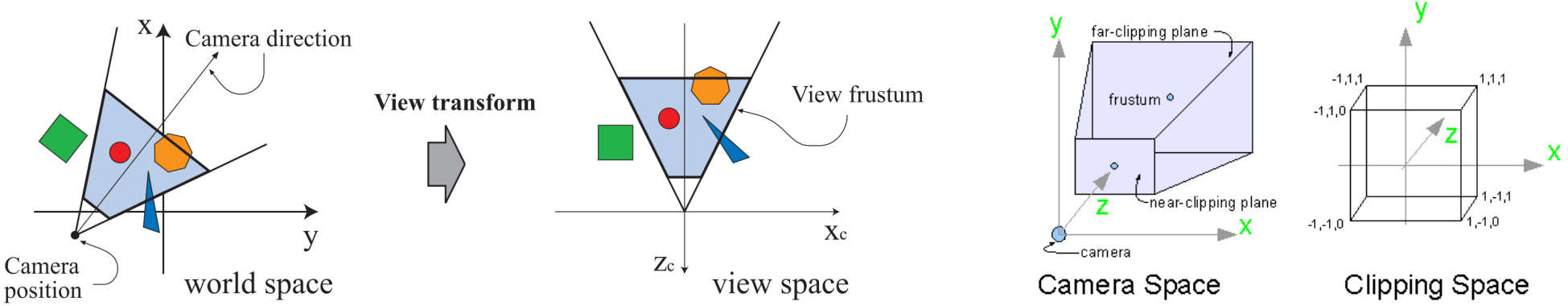
기하 처리 단계 14~15p 참조

- GPU상의 스테이지로, 대부분의 삼각형, 버텍스의 조작을 책임진다.
- 이 단계는 버텍스 셰이딩, 투영, 클리핑, 화면 맵핑으로 다시 분할된다.



기하 처리 단계 – 버텍스 셰이딩 15p~18p 참조

- 이 과정에선 버텍스의 위치를 계산하는 것과 법선, 텍스쳐 좌표와 같은 버텍스 출력 데이터를 정하는 두 가지 주요 작업이 수행된다.
- 먼저, 모델 고유의 공간인 로컬 좌표계에서 월드 좌표계로 변환한다.
- 투영과 클리핑을 위해, 뷰 변환을 통해 뷰 공간으로 변환한다.
- 투영(직교 or 원근) 변환을 통해 절두체를 NDC(Normalized Device Coordinates)로 변환한다.
- 텍스쳐 좌표, 음영 등은 삼각형 단위로 보간되어 다음 단계로 전송된다.

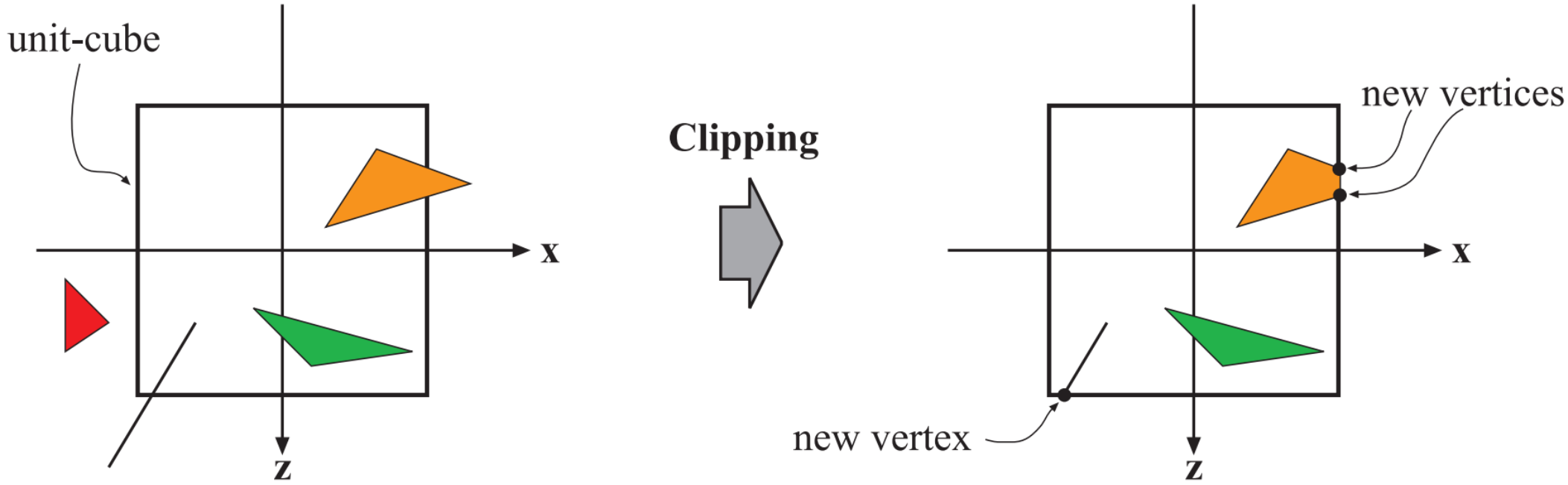


기하 처리 단계 – 선택적 정점 처리 18p~19p 참조

- 테셀레이션, 기하 셰이딩, 스트림 출력이 속하는 곳으로, 하드웨어의 지원 여부와 프로그래머의 의도에 따라 선택적으로 사용된다.
- 테셀레이션 : 필요에 따라 삼각형의 개수를 늘려 물체의 퀄리티(특히 곡면)를 높이는 방법으로, Hull Shader, Tessellator, Domain Shader의 과정을 거친다.
- 기하 셰이딩 : 다양한 종류의 프리미티브를 취하고, 새로운 정점을 생성 할 수 있다는 점에서 테셀레이션과 같다. 단, 이 생성은 범위와 출력 프리미티브 유형이 제한되어있다.
- 스트림 출력 : 앞서 처리된 정점을 나머지 파이프라인으로 전달하는 과정이다. 또한, 배열에 출력하는 것도 가능하다. 이 데이터는 이후 패스에서 CPU 혹은 GPU 자체에서 사용이 가능하다.

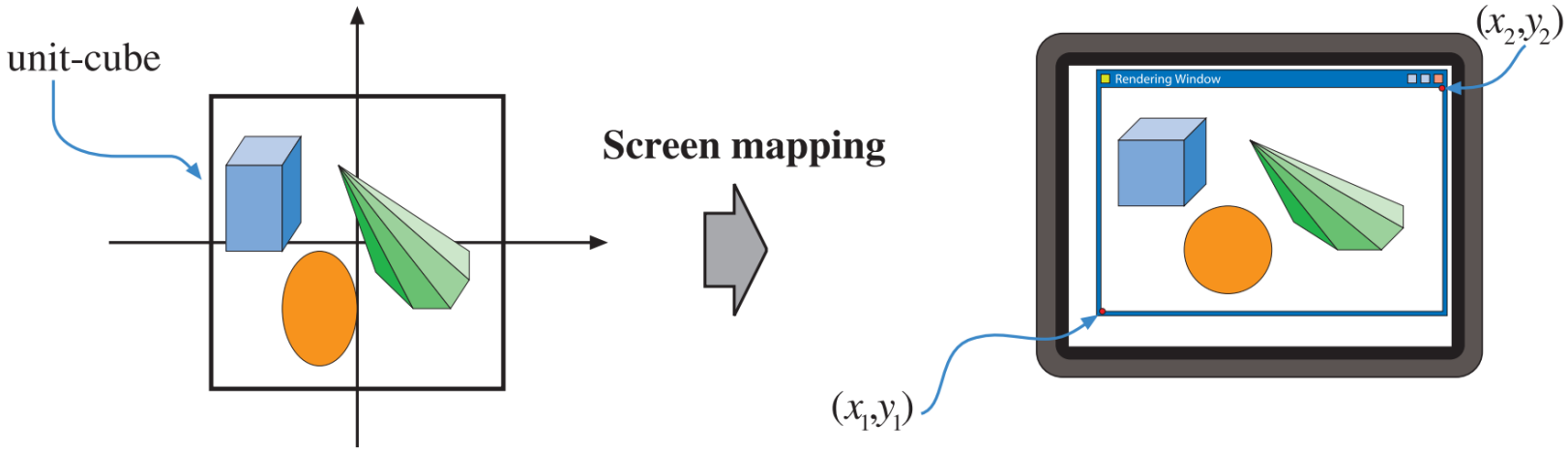
기하 처리 단계 – 클리핑 19p~20p 참조

- 동차 좌표(x, y, z, w)를 사용하여 뷰 볼륨 내부에 속해있는 폴리곤을 외부에 있는 폴리곤과 분리하는 작업이다.



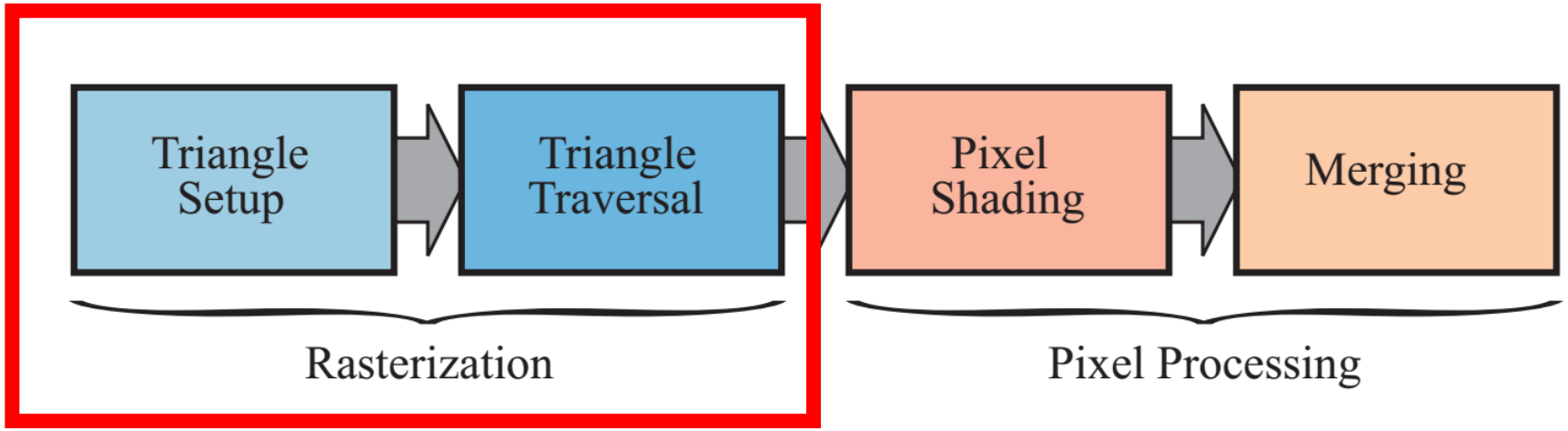
기하 처리 단계 – 화면 맵핑 20p~21p 참조

- 뷰 볼륨 내부의 클리핑 된 프리미티브만이 전달된다. 이 때 좌표는 여전히 3차원으로, 각 프리미티브의 x, y 값은 화면 좌표계에 맞게 변환된다.
- 또, z값 역시 최대최소 z값에 맞춰 보간되어 래스터라이저 단계에 보내진다.



래스터화 21p~22p 참조

- 화면 좌표계로 변환된 2차원 정점을 화면의 픽셀로 변환한다.
- 벡터 처리와 픽셀 처리의 동기화 지점으로 생각하면 된다.
- Triangle Setup : 삼각형에 대한 미분, 변 방정식 및 기타 데이터가 처리된다. 이는 기하 처리 단계에서 제공된 다양한 쉐이딩 데이터의 보간을 맡는 Triangle Traversal 단계에서 사용 될 수 있다.
- Triangle Traversal : 렌더될 삼각형이 그려질 픽셀들을 탐색하는 과정이다. 삼각형의 세 꼭짓점에 의해 보간된 데이터를 생성될 프래그먼트 속성으로 지정한다.



픽셀 처리 22p~ 참조

- 픽셀, 샘플 단위의 계산이 해당 픽셀에 대해 수행되는 단계다.
- Pixel Shading : 보간된 데이터들을 입력으로 하여 픽셀 별 셰이딩 계산을 진행한다. 결과는 다음 단계로 넘길 한 개 이상의 색상이다.
- Merging : 픽셀 셰이딩 단계서 계산된 색상과 그 색상이 칠해질 버퍼에 있는 색상을 혼합하는것은 이 단계서 처리한다. 알파블렌딩, Z 값 비교 등..