WEBGL #4

Three.js

04 28, 2015 - 이준호

오늘 볼 내용은...

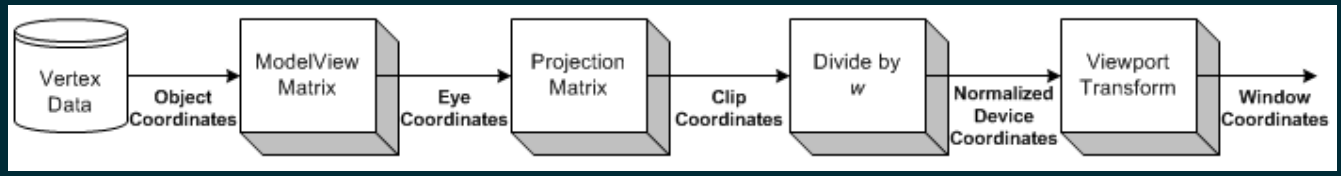
- CPU version - http://ligo.kr/po1
- GPU version - http://ligo.kr/gr6

WHY CPU?

- GPU 가속없이 CPU에서 동작하는 자바스크립트만으로 3D엔진을 구축 해보면
- 최신 GPU 가속을 이용한 3D 구축 원리를 이해할 수 있고,
- 더 복잡한 3D 엔진들을 쉽게 이해할 수 있게됨.
- 참고: http://ligo.kr/vwz
- 코드: view source
- 개인적으로는 GPU의 고마움을 알게 되었음. :)

CORE

- Camera, Mesh, Device
- Device의 render 메소드에서 Camera 기준의 View Matrix와 Projection Matrix를 처리.



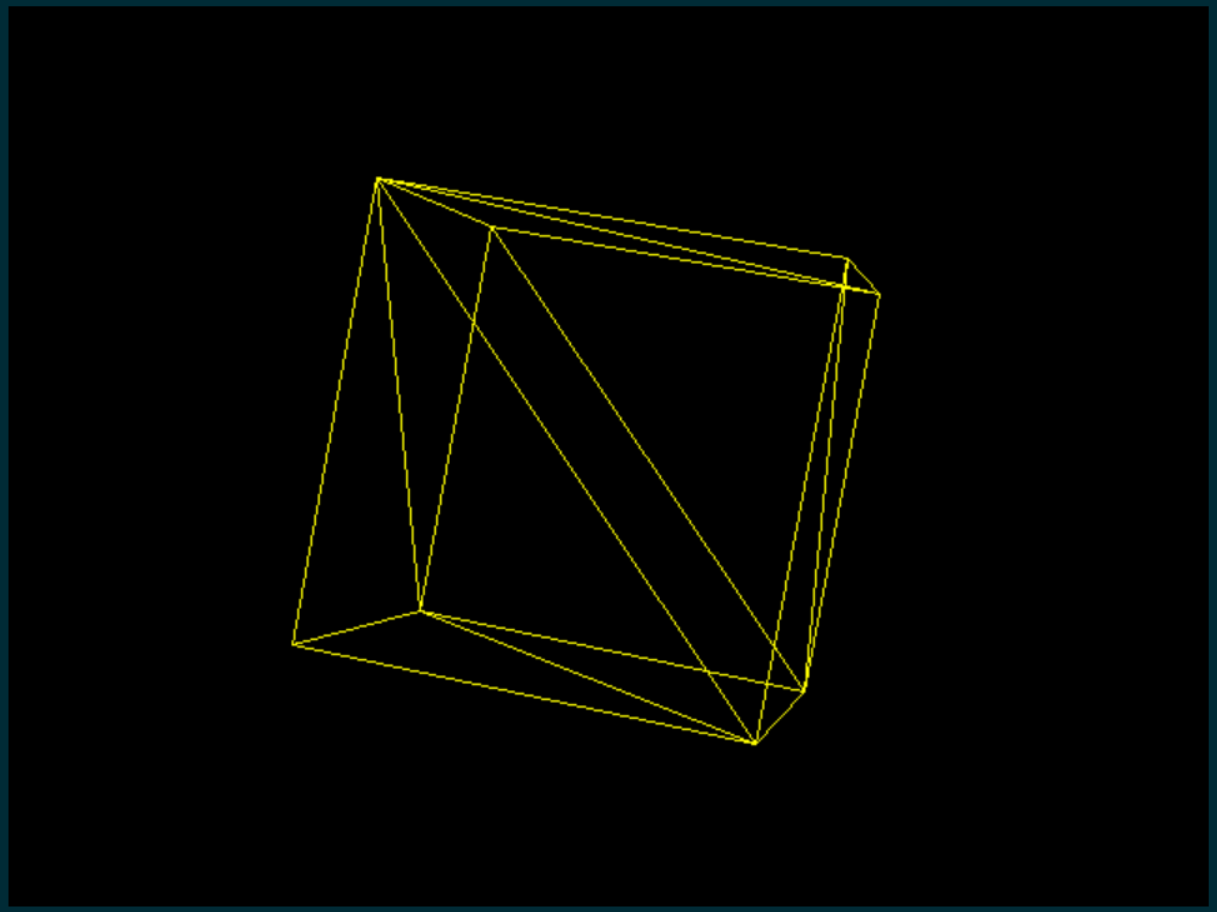
- Rendering loop: clear => render => present

```
var transformMatrix = worldMatrix * viewMatrix * projectionMatrix
```

DRAW TRIANGLE

- Geomatry의 기본은 삼각형
- 모든 Mesh는 삼각형의 집합
- Face: 삼각형을 구성하기 위해 필요한 3개의 Vertex의 인덱스
- Vertex를 사용하여 삼각형을 그리고 Mesh를 구성하는 것으로 모든 오브젝트를 표현할 수 있음

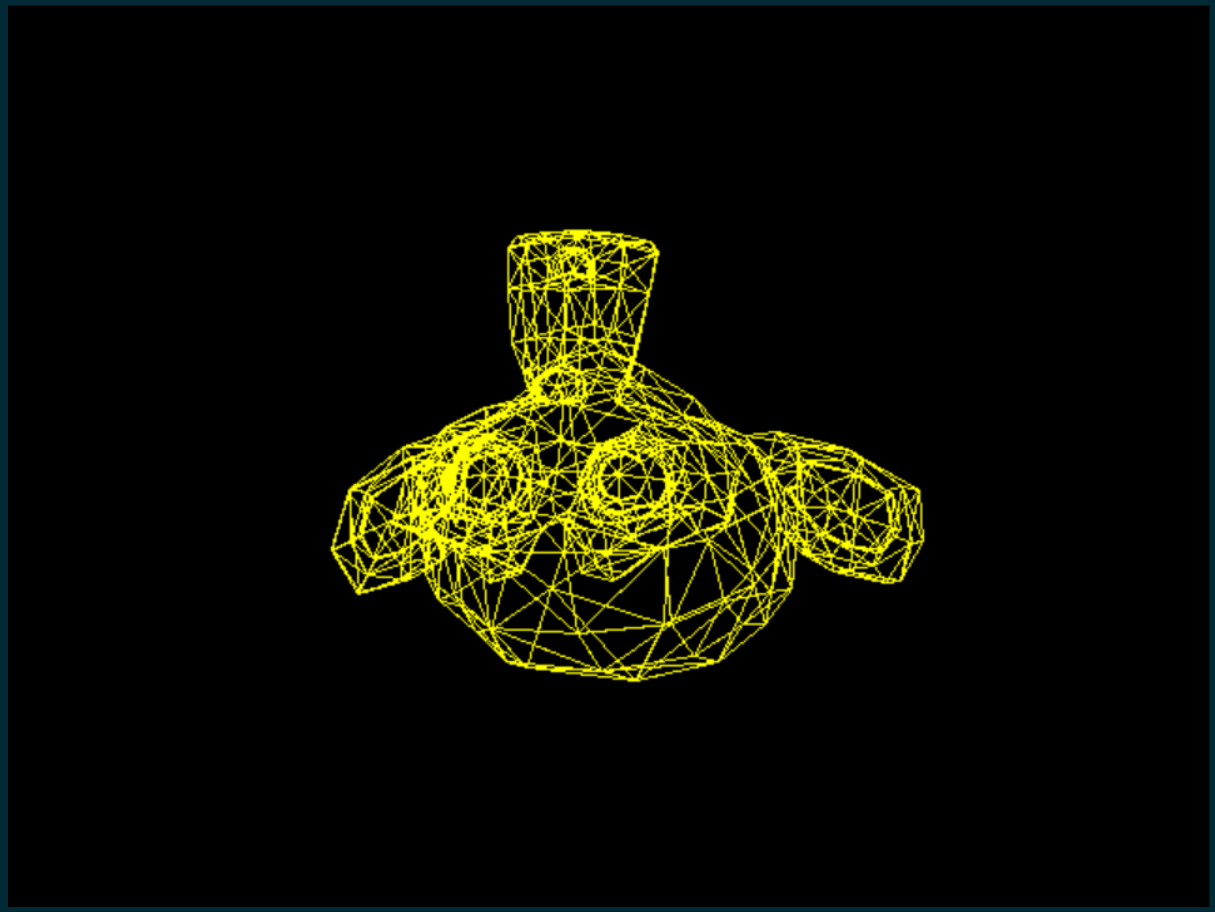
DRAW TRIANGLE



MODELING DATA

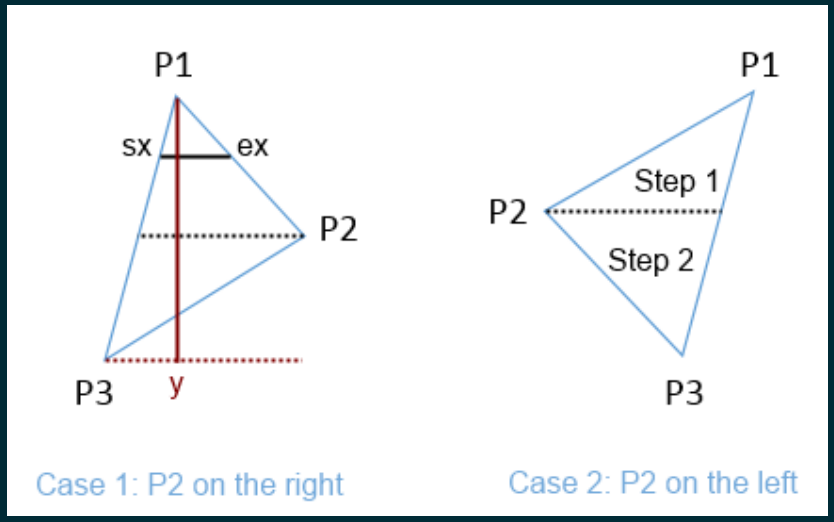
- 복잡한 오브젝트를 그리기 위해서는 무수한 삼각형을 그려야 하는데...
- 일일이 Vertex 좌표를 작성하는 것은 불가능
- 모델링 툴에서 모델링하여 export한 데이터를 로드하여 사용한다.
- 모델링 툴도 많고 로더도 많고 포맷도 많고...
- 파싱은 나름. - 필요한 데이터만 파싱하여 사용하자.

MODELING DATA



삼각형 채우기

- resterization 알고리즘은 매우 많다. 심지어 각자 제작하기도
- CPU던 GPU던 가장 많은 부하가 걸리는 부분
- processScanLine
  - 위에서 아래로, 좌측에서 우측으로
  - 두 가지 패턴의 삼각형을
  - 두 가지 케이스로 나누어서 채운다.



Z-BUFFER

- 최종적으로 점을 채우기 전에
- 뒤에 있는 점을 앞에 있는 점이 가리고 있으면 그리지 않는 처리
- 점을 찍기 전에 z-buffer와 비교하면 간단히 처리 가능
- z-buffer를 저장하기 위한 depthbuffer를 생성

GOURAUD SHADING

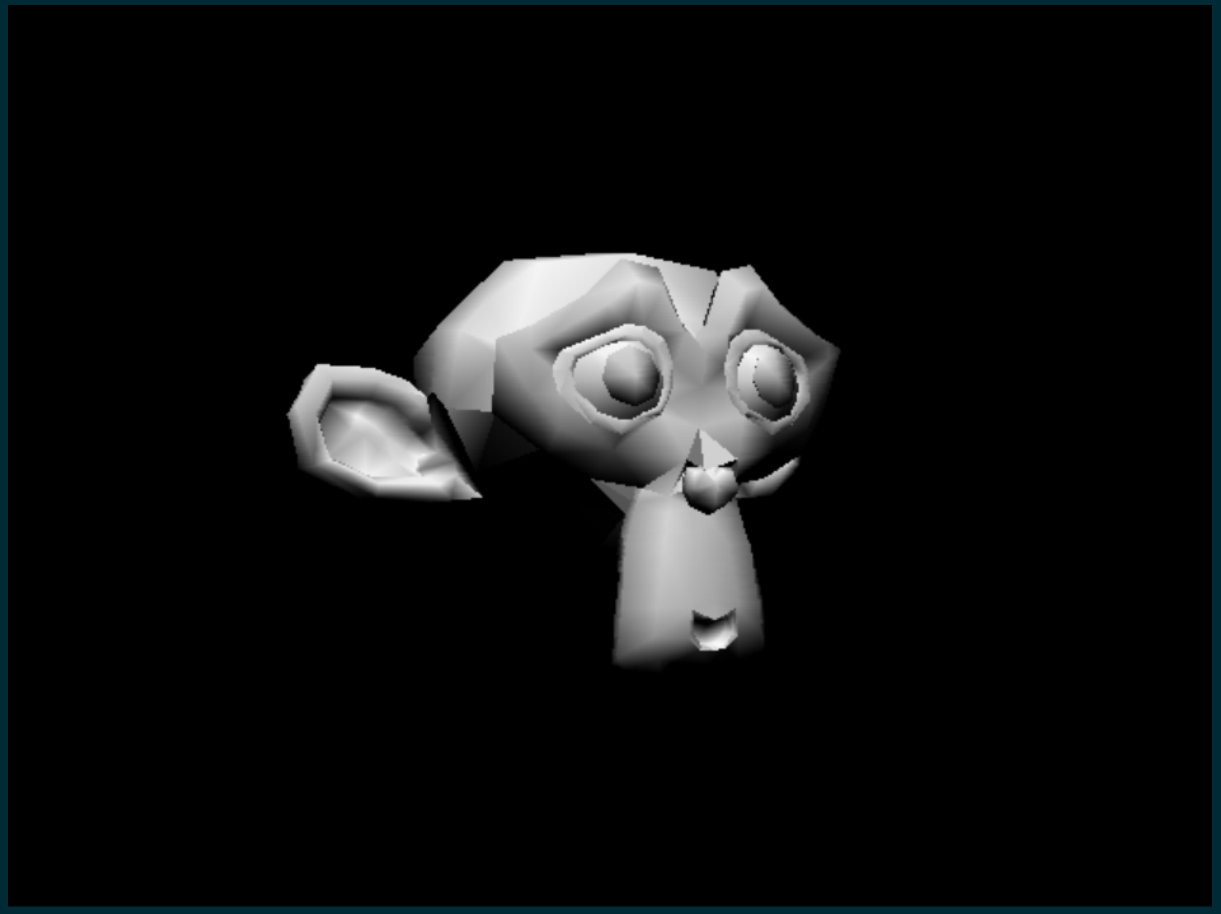
- 사실성을 높이기 위해 Light와 Face 사이의 각도를 이용.
- Face normal vector와 light vector 사이의 각에 따라 0과 1 사이의 값을 곱
- Blender에서 확인 가능
- Vertex에 Normal을 저장
- 각 Vertex의 Normal을 사용하여 Vertex 사이를 보간

```
color * Math.max( 0, cos(angle) )
```

주요 메소드

Project	World Matrix를 사용해서 3D상의 좌표를 가지는 Vertex를 생성
DrawTriangle	Vertex를 받아서 ComputeNDotL 메소드를 통해 NDotL을 연산한 결과를 가지고 ProcessScanLine을 호출
ComputeNDotL	Normal과 Light 사이의 cosine 각도를 연산
ProcessScanLine	DrawTriangle로 부터 전달된 NDotL 값을 사용해 color를 결정.

GOURAUD SHADING



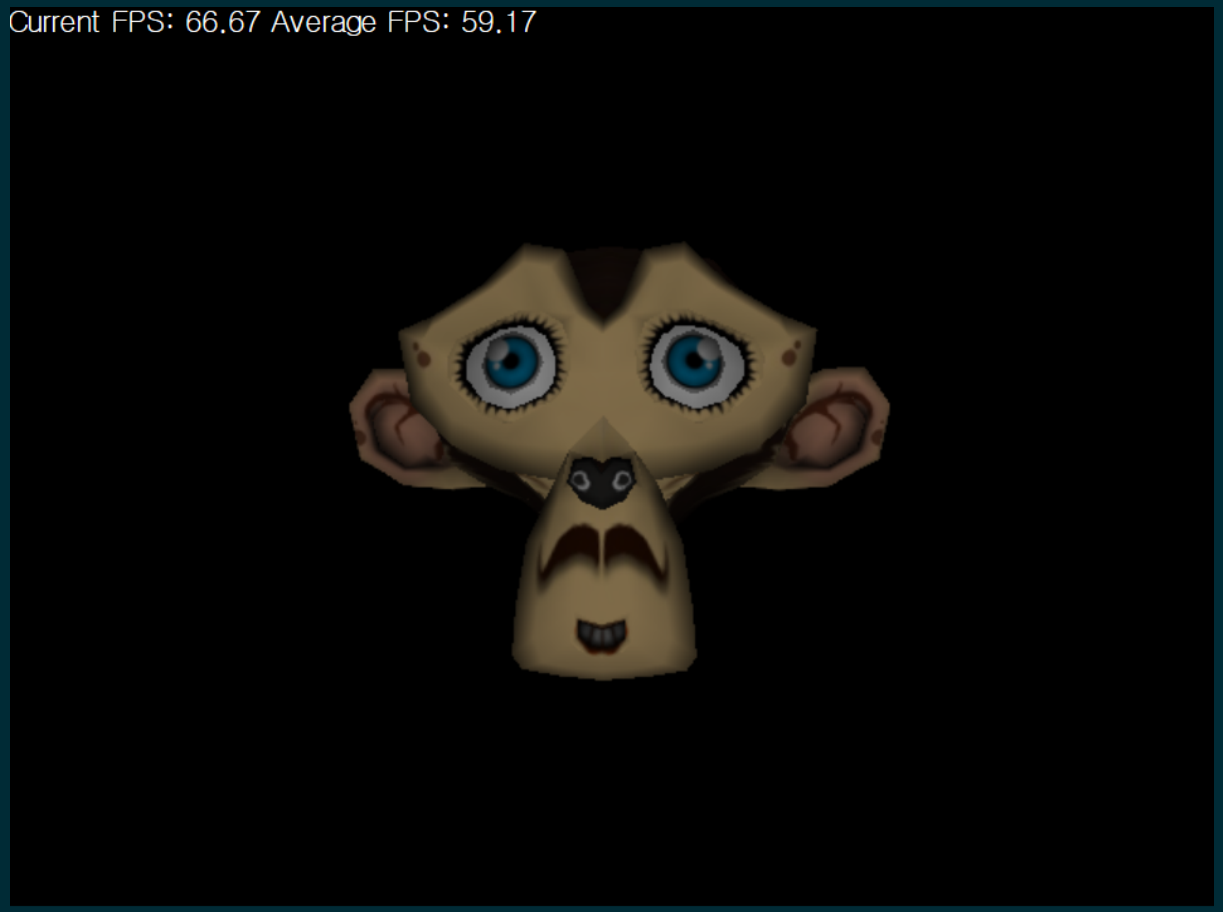
TEXTURE

- Blender에서 export한 UV좌표를 사용
- 두 Vertex 사이를 보간하는 처리가 핵심
- UV좌표란: Vertex에 사용하는 Texture 상의 2D 좌표
- 이미지를 로드하고 u, v 좌표를 받아 이미지 좌표 상의 컬러를 반환하는 map을 구현

TEXTURE

THREE.JS

Custom Loader



소스 코드를 직접 보겠습니다. :)

THANK YOU