렌더링 메쉬의 면들을 그리는거.

메쉬-그물망연결 = 모델(그래픽) = 객체

3d겉모양표현하기위해 2차원 다각형들 서로 이어서.

메쉬(Mesh) 🡪실수(점)들의 집합. 삼각형집합.

game에서 제일 중요한 거: 프레임 유지. 프레임이 널뛰는게 제일 구림. 프레임데이터가 너무 높으면 게임이 빈약.

너무 낮은건 게임이 복잡하고 처리할 것들이 많음. 게임세상을 줄여야 함. 🡪 렌더링 과정 중에 알고리즘적으로 더 좋게 만들 수 없나, 쓸 데 없는 거 그리는 거 없나.

🡪최적화

은면 – 보이지 않는 면

전면 – 보이는 면

구분 어떻게? 면이 향하는 방향. 카메라의 위치에 따라 바뀜.

점을 표현하기 위해 좌표계가 필요,

구 같은 건 식으로 만들 수 있는데 사람모양 같은 건 방정식으로 안 만들어짐

왼손 좌표계 업, 룩 (보는 방향), 라이트

순서화: 기준이 있음

좌표계가 있으면 스리디 모델의 위치방향표시가능

GameObject{ \*mesh데이터, 위치, 방향 }

Mapping , map 🡪 2D 이미지

범프맵 🡪 물체의 요철을 표현

0313

모델좌표-월드좌표 정하는 거 중요

좌표계 두개 일치시키는 것도 중요. 텍스쳐의 크기도 중요.

모델<--메쉬🡨점<--실수

보이지 않는 면 안 그리면 프레임 up.

다렉에서 폴리곤의 정점 나열하는 방법(와인딩 오더) 시계방향

보이는 면은 🡪 시계

뒤에 있는 면은 폴리곤들의 와인딩 오더가 반시계방향으로 바뀌면 뒤에 있음.

클래스🡪객체(오브젝트)로 만든다.

렌더링 transpomaion 제일 먼저.

3d렌더링 폴리곤을 그리는 거 아니고 T&L==TRANSFORMATION & LIGHTING

라이팅: 스리디 물체표면색은 조명에 따라 달라짐. 픽셀의 색은 조명의 영향도 많이 받음.

픽셀의 색 결정. 우리가 사는 세상에서 물체표면의 색을 정하는데 가장 중요한 건 조명.

빛이 없으면 물체표면 까망임.

모핑 (얼굴근육을 움직이는 거?)?

파이프라인. 어떤 일들이 순차적으로 일어난다. 중간에 다른거 못끼어듬. 꼭 필요한 절차들.

변환 파이프라인: 쓰리디 처리 과정에서 트랜스포메이션 파이프라인단계 써둔 거.

* 모델좌표🡪 월드변환 🡪 카메라변환 🡪 투영변환 🡪 화면변환 🡪 화면좌표
* 정점 🡪 프리미티브 🡪 프래그먼트 🡪 픽셀

+) 파이프라인: 정점 🡪 레스터라이저 🡪 픽셀 🡪 출력병합기

+) 파이프라인을 거쳐야 렌더링이 된다. 그럼 3D렌더링하는 과정이 정점~픽셀인가 그럼 파이프라인에서 후에 출력병합기는 뭐지

트랜스폼 한다 = 스리디로 표현된 모델의 정점들을 화면에 픽셀 투디 좌표로 변환하는 과정. 이를 찾아내는 수학적인 처리과정. 화면에 투디 픽셀좌표들로 바꿈. 코드🡪변환파이프라인

왜 첫번째 입력이 정점? 메쉬는 폴리곤들의 집함. 폴리곤은 정점들의 집합. 메쉬를 트랜스포메이션해서 화면에 그리기 위해서 폴리곤을 트랜스포메이션을 함. 반복적으로. 폴리곤을 트랜스폼한다는 건 폴리곤의 모든 정점들을 반복적으로. -->정점을 트랜스폼.

변환과정에서 입력할 때 정점하나 넣는거 이해하셈…모델좌표계로 표현된 정점하나.

* 이해해보자

<변환 파이프라인: 내가 이해한 버젼>

1. 정점 (모델좌표계) 🡪 월드변환: 모델 골반 000에 피벗을 두고 모델링을 함. 모델좌표계. 그걸 내 게임세상에 띄우고 싶다. 이때 필요한 거가 월드변환.
2. 월드 좌표계 🡪 카메라 변환: 내가 바라보는 방향으로 화면을 렌더링하는거. 예를 들어 삼인칭게임이면 모델 뒤에 카메라 있음. 이거에 맞춰서 렌더링하는거 월드좌표계🡪 카메라좌표계.
3. 카메라 좌표계 🡪 투영변환: 내가 표현하고 싶은 건 3d인데 모니터는 2d화면이니까 그거에 맞게 변환.
4. 투영 좌표계 🡪 화면 변환: 띄울 화면크기에 맞게 화면 좌표 변환.
5. 화면 좌표계

<수업 때 중구난방으로 적어둔것>

월드변환. 모델 좌표계를 게임 월드 좌표계로 바꾸는 과정. 월드좌표계로 주어진 점은 카메라 좌표계로 카메라좌표에 있는 애들까지 다 스리디. 이걸 옮기려면 2차원으로 바꾸는 거. 투영변환.

카메라=뷰

1. 월드좌표변환

* 평행이동(Translation)

오브젝트 짱짱 많으면 🡪 겉모양이 같은 데이터들을 만들 땐 \* 🡪인스턴싱.

겉모양이 같은 여러 개의 객체가 하나의 메쉬. 데이터를 공유하도록 만드는 거

🡪공부여기부터.

회전에 필요한 거. 회전 축이 주어져야 함. 회전축이란 걸 피봇이라는 용어로도 부름. 몇도 돌지 각도도 필요함. 회전을 하고 트랜스 하느냐 트랜스 먼저하고 회전을 먼저 하느냐는 결과도 다르고 좌표도 다름. ))컴그에서 해봄

변환 = 행렬의 곱으로 변환들을 합성할 수 있어! ))컴그에서 뭔가 함수 써봄

모델좌표 화전하면 자전 될 수도 있음. 근데 반드시 자전이 되지는 않을 수도 있음

게임 오브젝트에 대한 포인터는 물체의 겉모양. 위치정보를 위한 xyz, 방향을 위한 회전각도xyz

오일러각도표현.

x회전 피치 y회전 yaw z회전 roll

크기변화는 프레임을 갉아먹음. Scale +)스케일하면 행렬 요소하나하나를 다 곱해야해서.

온라인(리얼타임)

오프라인(스태틱, 컴파일타임)

* 그 차이가 뭐야! 미리 계산할 수 있는 것은 미리 해라. 게임프로그램이 실행되기 이전에!

모델을 몇 개를 만들던 프레임엔 영향 없어! 모델 하나 만들어 놓고 새로운 걸 계속 만들어낸다면 프레임이 느려진다.

프로그램 만드는 거 계산하는 거 아님. 미리 테이블에 넣어 놓고. 60분이ㅡ 1초마다 미리 어떤 정보를 만들어 낼 수 있느냐 없느냐. 요소들을 잘 구분해 둬야함. 그러면 프레임레이트 떡락하는 일은 적음. 미리! 미리! 미리! 미리 할 수 있는 요소들을 만들어라. 게임 실행할 때 그때 바로 게임이 시작되는게 아니라 로딩시간 거침. 하지만 게임도중에 만들어야 하는 정보도 있음? 그 시간에 만드는 것도 프레임 레이트랑은 관계가 ?

그걸 분리하는 것도 오프라인. 엔진을 쓰면서도 망하는 이유는 이걸 구분해주지 않으니까.

세상을 줄이면 겜은 노잼임. 한프레임에서 필수적으로 해야 되는 거 잘 골라내야 함.

게임세상 그려내는 거 이해하면. 똑같이 처리. 카메라가 눈에 해당하는 거면 모니터. ~~

렌즈가 원형모양.

카메라가 볼 수 있는 영역에서 보이는

카메라가 렌더링을 한다. 씬에있는 모든 오브젝트가 다 보인다 생각하고 다 그려! 가 아니고 그러면 프레임레이트 안나오니까 나올 수 있는 것만 찾아서 그려.

내뒤에잇는객체는 카메라 좌표계 기준으로. z좌표가 음수면 안 그림. 카메라 로컬좌표계 기준으로.

카메라를 중심인거처럼 좌표를 바꾸는게 카메라 좌표변환. 즉 뷰 트랜스폼. 카메라를 월드ㅂ좌표계의 원점으로 평행이동. 월드좌표계에서 x축으로 한칸 아래로 두칸 z축은 세칸움직이면 원래 있었던 자리로 감. 월드좌표 000을 바꾼다는건 카메라를 월드좌표계 원점으로 이동시킨거. 그러면 카메라가 세상의 중심. 세상으 ㅣ중심은 카메라. 카메라 좌표변환할 때 회전먼저하면 이모든 점들이 월드좌표라 공전이됌. 카메라변환할땐 카메라를 원점으로 가따놓고 회전해야함 반대로. – 붙여서. )?

객체의 모든정점아니고 위차만 바꿔야하니까. 위치만 803이니까 4뺌.

카메라가 움직이면 그만큼 반대로 해야됌. 실제 스리디에서는 카메라가 있을 때 선분의 좌표가 둘다 – 면 안 그림

투영이 여러가지있는데. 페인터알고리즘은 원근감 있게 그려얗행

카메라가 가진 가상적인 면? 눈으로 투영이됌. 카메라 좌표에서 x랑y는 바귐. 우리가 그림을 그렸을 때 원근감. 원근감을 살리는 그런 알고리즘이 들어가 있음.

프러스텀. 절두체. 음수면카메랑[ㅔ안보인다고하면 좋은디 평면을 더 오른쪽에 두면 안보임.