KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BIL3008 BİLGİSAYAR GRAFİKLERİ-I
DIRECTX 12 DERS NOTLARI

BURKAY DURDU

2016-2017 Bahar Dönemi

DirectX 12

OnInit() ⇒ ShowWindow önce 1 kere koşuyor. Çeşitli nesneler üretiliyor. Bazı matrisler tanımlanıyor. Bufferlar tanımlanıyor.

OnUpdate() ⇒ Her bir fromede koşar. onRender() 'dan önce koşar. Genellikle çizim yapacağımız cisme transformasyon uygulanır. Öteleme, Döndürme, büyütme, Küçültme yapılır.

OnRender() ⇒ Her bir fromede koşar. Güncellenen cisimleri çiziyoruz burda sadece çizim komutları var.

Shaders.hlsl → Ekran kartında koşan programlar var.

```
cbuffer SceneConstantBuffer : register(b0)
{
    matrix World;
    matrix View;
    matrix Projection;
}
```

cbuffer → key

```
Struct VSOutput
{
  float4 position : SV_POSITION;
  float4 color : COLOR;
}
```

float4 → 4 tane parametre alabilen bir tür sanki 4 tane structure değişken tutar gibi programa özgü bir anahtar.

```
                    ↱Vertex Shader
VSOutput VSMain(float3 position:POSITION, float4 color:COLOR)
{
    VSOutput result;

    result.position = mul(float4(position, 1), World);
    result.position = mul(result.position, View);
    result.position = mul(result.position, Projection);

    result.color = color;
    return result;
}
                    ↱Pixsel Shader
float4 PSMain(VSOutput input) : SV_TARGET
{
    return input.color;
}
```

mWorld matrisi cisimleri rotation, scaling, translate gibi çeşitli transformasyonları yapmamızı sağlıyordu.

View matrisi hangi noktadan gözlemleniyor hangi noktaya bakıyoruz alt vektörü nedir.

PSMain'de ekran koordinatları düşürülmüş. cismi burdaki color a setliyoruz.

Vertex Shader'a yollucağımız World, view projection matrislerini constant buffer aracılığı ile yolluyacağız

3 tane değişkenimiz olucak

constant buffer türetmek için =>ComPtr<...> m_constantBuffer;
buffer'a yazıcağımız veriyi tutan structure=>SceneConstantBuffer m_constantBufferData;
Buffer'ın başlangıç adresini tutan pointer=>UINT8* m_pCbvDataBegin = NULL;

View Eye, At, Up matrislerini barındırıyor.

```
g_World = XMMatrixIdentity();
XMVECTOR Eye = XMVectorSet(0.0f, 0.0f, -5.0f, 0.0f);
XMVECTOR At =
         Up =
g_View = XMMatrixLookAtLH(Eye, At, Up);
g_Projection = XMMatrixPerspectiveFovLH(XM_PIDIV4,
                                        1280/(FLOAT)720, 0.01f, 100.0f);
m_constantBufferData.mWorld = XMMatrixTranspose(g_World);
//                    .mView = //          (g_View);
//                    .mProjection = //    (g_Projection)
```

İlk örnekte rotation yapacak şekilde world güncelliyoruz.

```
OnUpdate(){
    rotation += 0.03;
    g_World = XMMatrixRotationY(rotation);
    m_constantBufferData.mWorld = XMMatrixTranspose(g_World);
```

↳ Neyi kopyalayacağımızı

```
    memcpy(m_pCbvDataBegin, &m_constantBufferData,
           sizeof(m_constantBufferData));
```

↓ ekran kartında ilgili constant bufferın başlangıç adresini parametre olarak veriyoruz

↓ byte cinsinden boyutu

OnInit in içerisinde tanımladığımız root signature ile cpp tarafında yaptığımız vertex shader için constantbuffer structurın aynısı ekran kartında gidecek olan kısma root signature ile bildiriyoruz onunla eşliyoruz.

CD3DX12_ROOT_PARAMETER1 rootParameters[1];
1 tane constant buffer var bu yüzden 1 tane root Prometresi açtık.

rootParameters[0].InitAsConstantBufferView(0, 0 - - -
ShaderRegister
0
1

cbuffer SceneConstantBuffer : register(b0)
b0 → 0 olduğunu söylüyor.
14 tane buffe tanımlamanıza izin veriyor.

render ederkende constant buffer setliyoruz.

m-commandList → SetGraphicsRootConstantBuffersView(0, [ ) constantbuffe ismi

root Signature in parametresi [0]

m-constantBuffer_changesEveryFrame→ (1, [1])

m_constantBuffer_NeverChanges→

register(b0) → buffer 0 demek

0-13 arasında 14 tane buffer tanımlama imkanı veriyor.

2 tane Constant buffer
Linde world matrisi diğerlerinde projection ve view olsun

```
struct SceneConstantBuffer_ChangesEveryFrame
{
    XMMATRIX mWorld;
};

struct SceneConstantBuffer_NeverChanges
{
    XMMATRIX mView;
    XMMATRIX mProjection;
};
```

Değişkenlerinde kopyası çıkıcak.

```
ComPtr<ID3D11Resource> m_constantBuffer;
SceneConstantBuffer_ChangesEveryFrame m_constantBufferData_ChangesEveryFrame;
UINT8* m_PCbvData_Begin_ChangesEveryFrame = NULL;
```

bu everyFrame için burdan birde neverchanges Frame için yapılcak.

Onlinitin içerisinde 2 tane constantbuffer tanımlanması gerekiyor.

```
{                    {

}                    }
```

İlk değer atamalarda g_world everyframe veriyoruz.
g_view neverchanges veriyoruz.
g_projection

bunların içinde tanımladık çünkü

→ onUpdate() içerisinde sadece world matrisi güncellenicek

→ rootSignature yeri bir root parametresi ekliyeceğiz.

--- rootParameters[2];

register buffer numaraları.

rootParameters[0] ____ (b ____

" [1] ____ (1 ____

hlsl içerisine 2 tane buffer tanımlıcaz

```
cbuffer Scene_____ : register(bb)
{
   matrix World;
}

cbuffer Scene_____ : register(b1)
{
   matrix View;
   matrix Projection;
}
```

render içerisinde setlicez bunları bağlıcaz yani cpp içerisinde tanımlanan structureları hlsl ile olarak tanımlıcaz

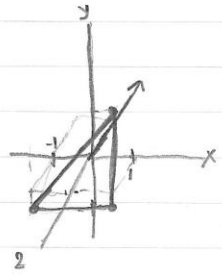
```
m-commandList → SetGraphicsRootConstantBufferView (0,
m-constatBuffer_ChangesEveryFrame → _____ );
                 "                  "          (1,
m-constantBuffer_NeverChanges →  _____ );
```

Update kısmında changesEveryFrameleri memcpy diyerek bellege atıyoruz.

neverchanges atmadık ilk değer atanılarında onu eklicez bellege.

memcpy(m_pcbvDataBegin_NeverChanges, &m_constantBufferData_NeverChanges, sizeof(m_constantBuffer_NeverChanges));

{ XMFLOAT3(1.0f, 1.0f, 1.0f), ——— },
{ " (1.0f, -1.0f, 1.0f), ——— },
{ " (-1.0f, -1.0f, 1.0f), ——— }

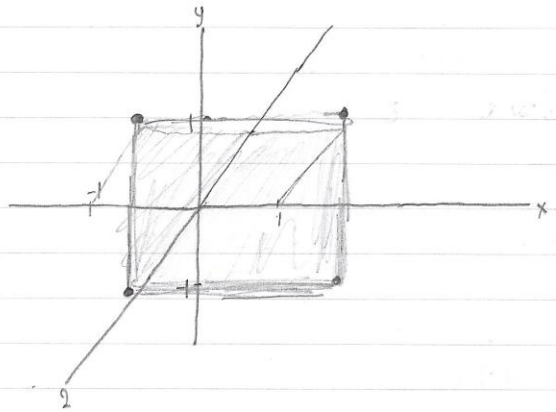


Dik üçgen oluşturduk ---

Kare yapmak için

color

EXMFLOAT3 (1.0f, 1.0f, 1.0f) ; ——— },
{ " (-1.0f ; -1.0f, 1.0f) ; },
{ " (-1.0f, 1.0f, 1.0f), }



→ bundan 1 kare daha yapıcaksak x değerlerini 3 br ekleyerek sağ tarafa öteleyerek göstericeğiz. 12 köşenoktası oluyor.

burda bir constantbuffer daine 256 byte kaydırarak bir cisim daha kopyayız yani bir buffer açmak yerine

burda 2 tane kare dusturmak yerine 1 tane olusturup öteleyerek digerini göstermeye calisicaz.

Vertex buffer'a dokunmadan world matrisi güncelleyerek yapicaz.

aynı constantbuffer a öteleyerek ekleme yapicaz. 256 byte'da 1 yazabilirsin diyor.

onUpdate()

g_world = XMMatrixIdentity(); //sabit ilkC

m_constantBufferData_chagesEveryFrame.mWorld = XMMatrixTranspose(g_world);

memcpy(m_pCbvDataBegin_chagesEveryFrame, &m_constantBufferData_chagesEveryFrame, sizeof(m_constantBuffer... pointer'dı Data_chagesEveryFrame));

g_world = XMMatrixTranslation(3,0,0); // x ekseninde 3br translate yaptık
m_constantBuffer ____ (g_world);

memcpy(" +256, & ____));

256 byte öne yaz.

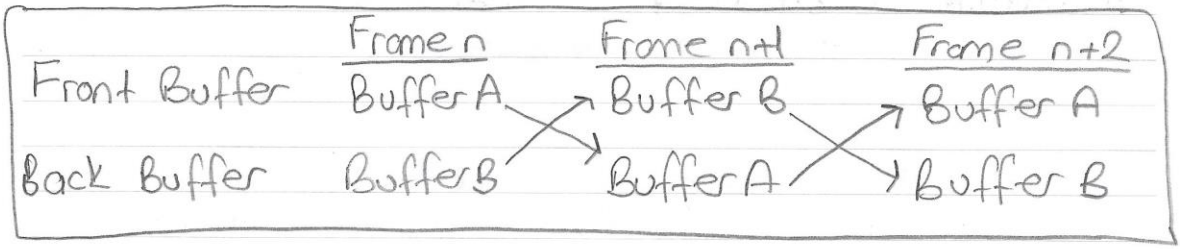
Bellekte yaptık simdi draw kısmında bunu nasıl göstericez.

m_commandList->SetGraphicsRoot ConstantBufferView(0, m_constantBuffer_chagesEveryFrame->GetGPUVirtualAddress() +256));

m_commandList->DrawInstanced(6,1,0,0);
tekrar Draw yaptık

GPU'da ki adresinden 256 byte sarakindeki buffer'ı yazdırdedik bu bufferda da translate edilmiş matris var 5 birim x ekseninde bunu yazdırmış olduk.

Bu nasıl oldu swapchain nesnesi sayesiyle oldu.



Çizim yaptığımız 2 tane ekran belleği var. O anda ekranda görüntülenen buffer'ın ismi Front Buffer. Herhangi bir n. frame'de, ekranda front buffer görüntülenirken n+1. frame yazılan buffer'da da back buffer deniyor.

front buffer n+1 gelince back buffer'dan alarak bu işlene swapchain nesnesi diyoruz.

Ekranda bir cisim görüntülenirken arkada bir diğer frame'de gösterilicek nesneyi yazıyoruz. Swapchain nesnesinin avantajı

onRender içerisinden populateCommandList() çağırarak back buffer'a çizim işlemlerini yapıyoruz.
bu cisimleri sağ sol ortaya çiziyoruz back buffer'da sonra bunları ne zaman front buffer'a swap yapıyoruz

onRender()

ThrowIfFailed(m_swapChain->Present(1,0));
bu komut ile swap yapıyoruz.

PopulateCommandList() return yapınca bu çağırılıyor bu çağırılana kadar istediğimiz kadar öteleyip çizebiliriz

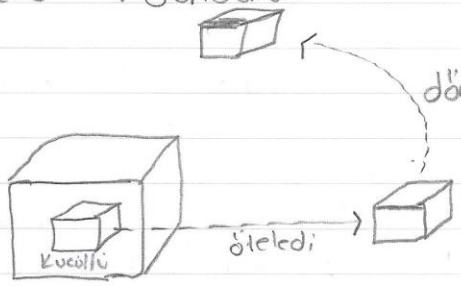
Transformasyon

3 tane var → Rotation → Dönme
Translation → Ötelene
Scaling → Ölçekleme

scaling(0.3f, 0.3f, 0.3f) 3 ekseninde de 0.3 oranında küçülttük

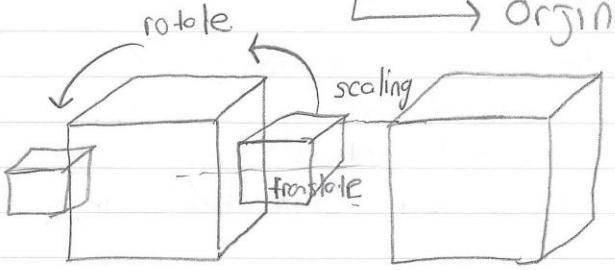
scaling * Translate * Rotate

Küçültür → öteler → döndürür.



translate * scaling * rotate

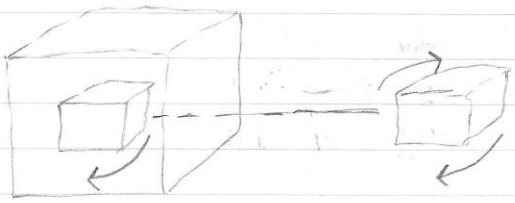
→ orjine göre yapar.



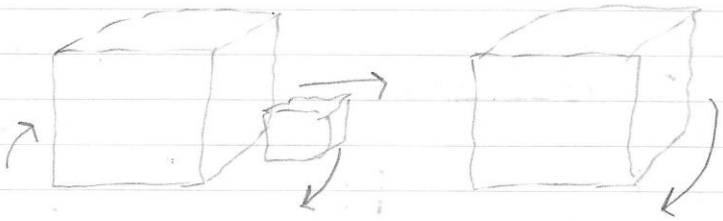
→ Scaling küçültürken orjine yaklaştırır uzaktaysa orjine göre küçültür bu yüzden yaklaşmış görürüz.

Birinci küpe yaklaşır döner.

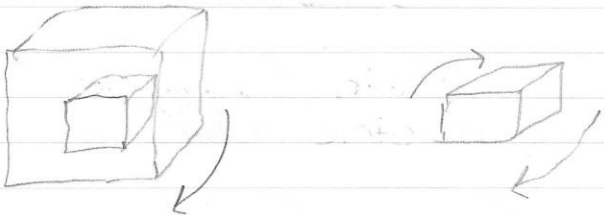
g_World = mScale * mRotate * mTranslate;



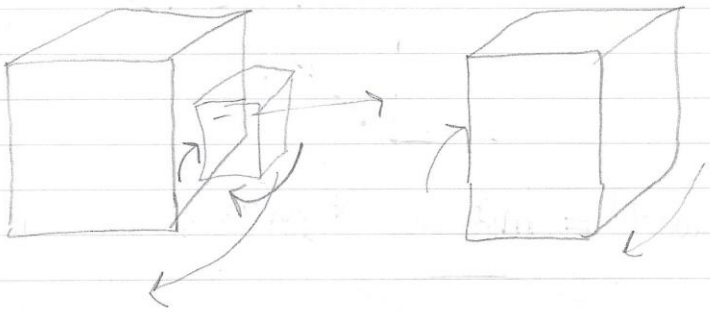
g_World = mTranslate * mRotate * mScale;

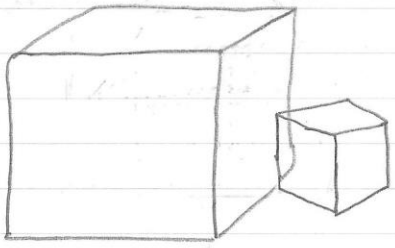


g_world = mRotate * mScale * mTranslate;

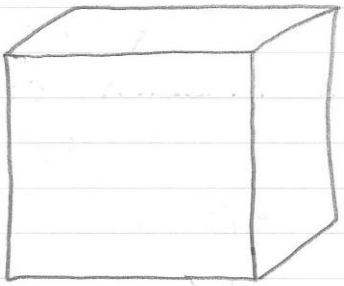


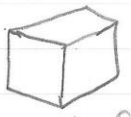
g_world = mRotate * mTranslate * mScale;





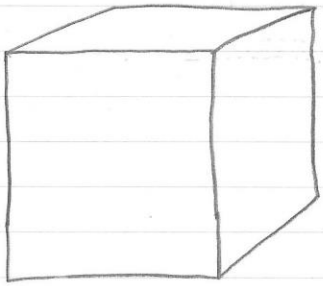
g_world = mTranslate * mScale * mRotate;
g_world = mTranslate * mRotate * mScale;





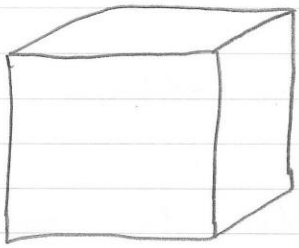
etrafında döner

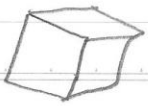
g_world = mScale * mRotate * mTranslate
g_world = mRotate * mScale * mTranslate



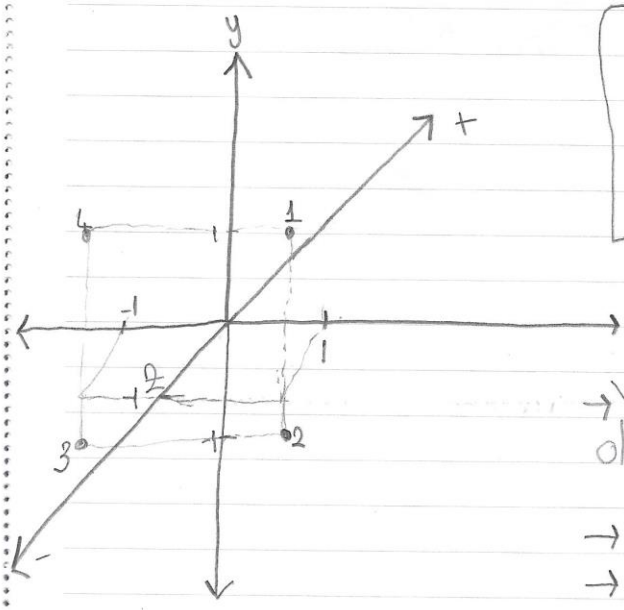
etrafında döner.

g_world = mRotate * mTranslate * mScale;





g_world = mScale * mTranslate * mRotate



→ XMMatrixRotation#(); Cismi # eksenine göre döndürür.
→ XMMatrixTranslation(); cismi öteler.
→ XMMatrixScaling(); cismi ölçekle

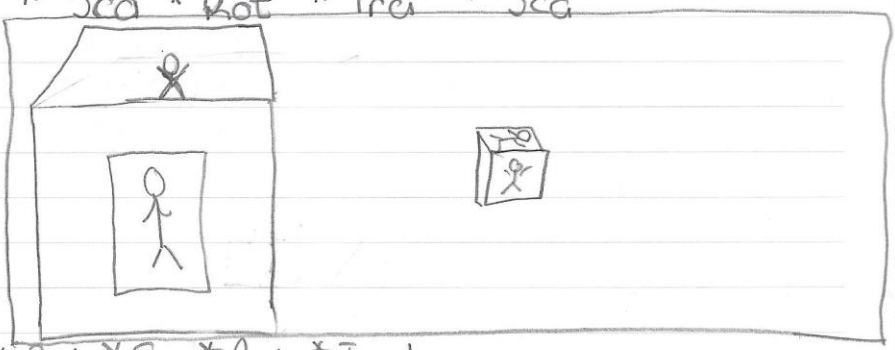
→ View matrisi hangi vektörlerden oluşur.

→ Eye: Bakış noktasının konumu
→ At: Bakılan nokta (Bakış doğrultusu)
→ Up: Yukarı doğrultu

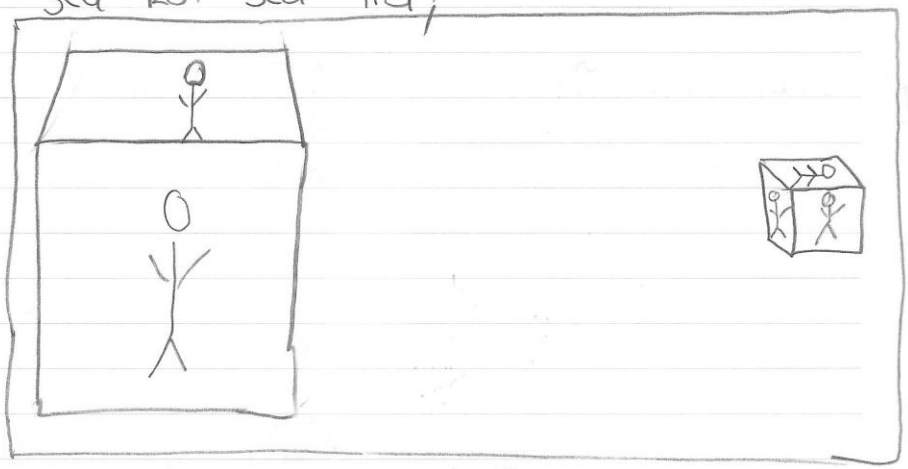
SwapChain'e ait Present() fonksiyonu ne işe yarar.
→ Backbuffer içeriğini ekranda görüntüle

Backbuffer'a çizim görevi hangi nesneye aittir.
→ M_commandList

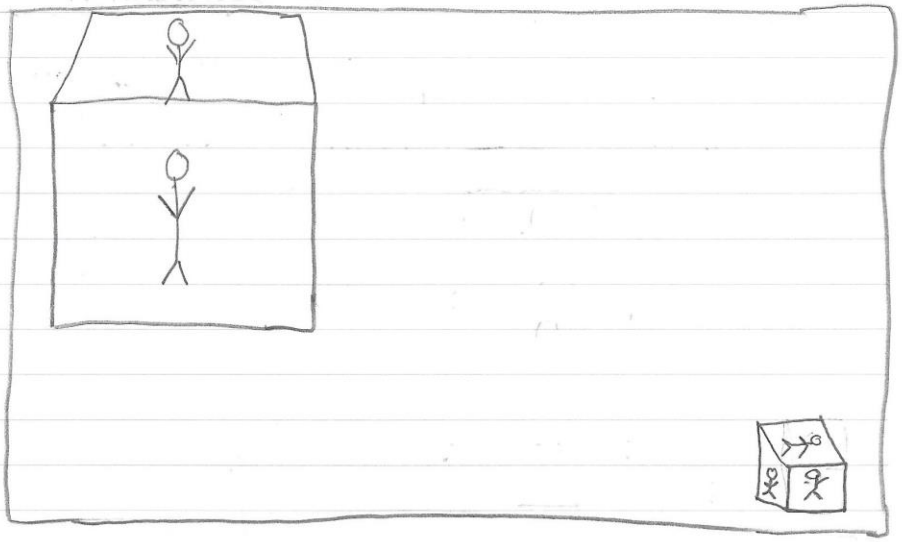
⇒ World = Rot * Sca * Rot * Tra * Sca



⇒ World = Sca * Rot * Sca * Rot * Tra ;
⇒ World = Rot * Sca * Rot * Sca * Tra ;

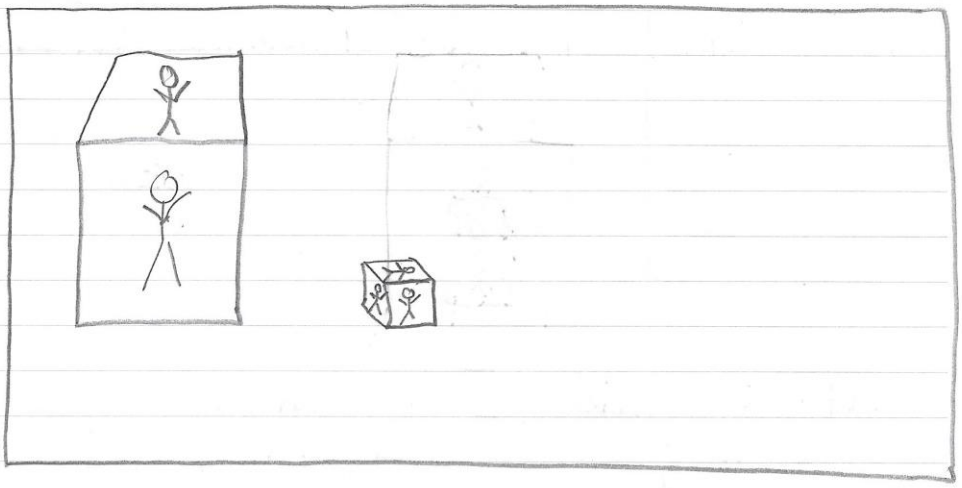


⇒ World = Sca * Rot * Sca * Tra * Rot

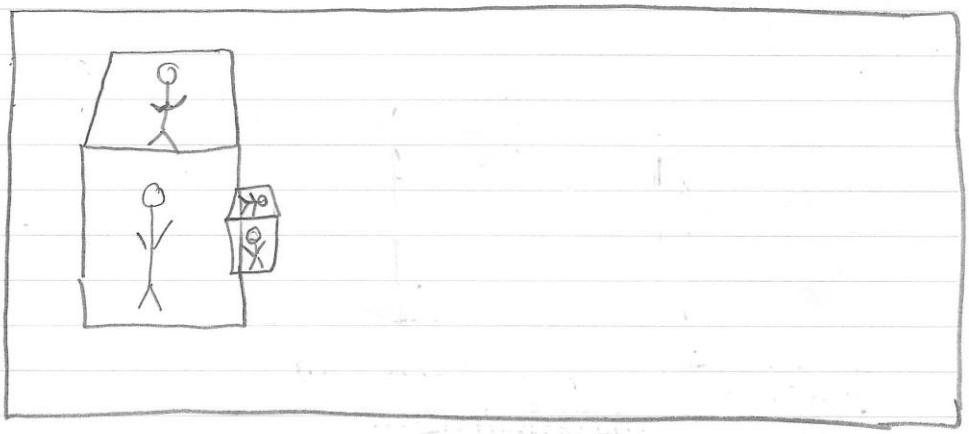


hem etrafında döner
hemde büyük küp
etrafında

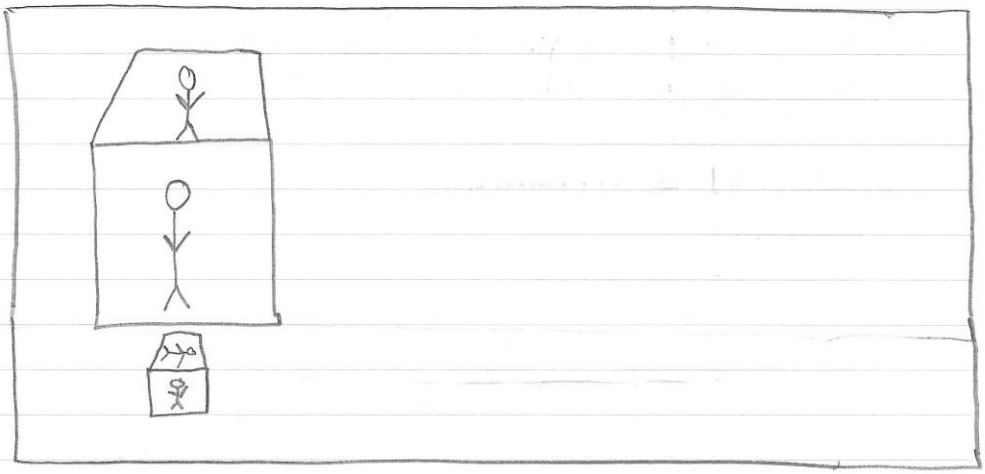
World = Rot * Sca * Tran * Rot * Sca;
World = Rot * Sca * Tran * Sca * Rot;
World = Sca * Rot * Tran * Sca * Rot;
World = Sca * Rot * Tran * Rot * Sca;



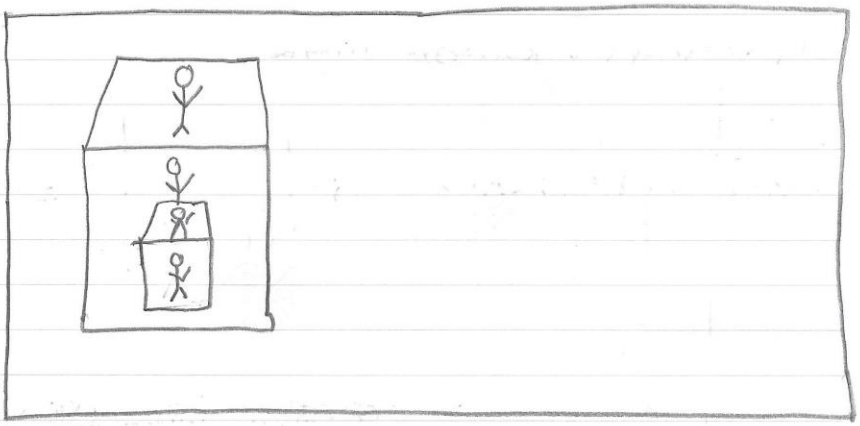
World = Rot * Tran * Sca * Rot * Sca;



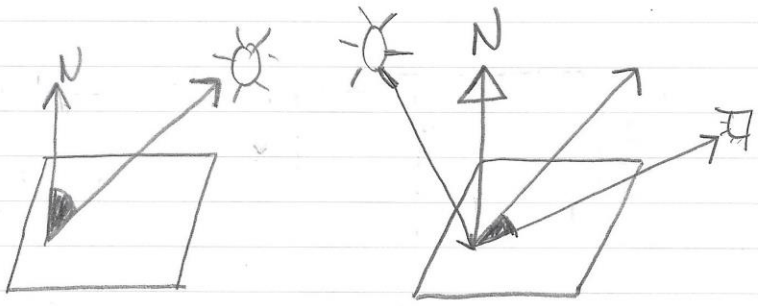
World = Sca * Tran * Rot * Sca * Rot



World = Tran * Sca * Rot * Sca * Rot ;
world = Tran * Rot * Sca * Rot * Sca ;



Lighting =>



diffuse Speculer

Burda 2 tane Pixsel Shader kosuyor.
→ PS_Phong → yesil zeminin rengini
→ PS_Solid → ışık kaynagının renginf ayarlıyor.

positionW → world
3 boyutlu uzayda

```
PS_Phong (VSOutput input)
//Diffuse
float3 toLight    = normalize (LightPos - input.positionW);
float  dotEyeNorm = dot(toLight, input.normal);
float4 diffuseColor = max(dotEyeNorm * MeshColor, 0);

//Specular

float3 fromLight = normalize (input.positionW - LightPos);
float3 toEye     = normalize (EyePos - input.positionW);
float3 reflected = fromLight + 2*dot(fromLight, input.normal)*input.normal;

float dotEyereflected = dot (toEye, reflected);
float4 specularColor = max(pow (dotEyereflected, 32.0f)*LightColor,0);
```

↓ Işığın zemindeki yayılması cemberin geri gibi ama ters orantılıdır. Küçülse daha fazla yayılır.

→ Cisim rengini belli kayaşı ile çarpıyoruz.

```
return 0.2 * MeshColor + 0.5 * diffuseColor +
       0.3 * specularColor;
```

OnInit() → buffer'ların sellediği kısım.

```
D3D12_GRAPHICS_PIPELINE_STATE_DESC psoDescPhong{};
psoDescPhong.VS = ____ BYTECODE(VertexShader.Get());
psoDescPhong.PS =      (pixelShaderPhong.Get());
```

birden fazla pixsel shader tanımlayıp nede veriyoruz
hlsl içinde 2 tane pixsel shader var PS_Phong ve
PS_sdiddiye bunları veriyoz

FABER-CASTELL

STATE_DESC psoDescSolid = psoDescPhong;

yukarıda tanımlanan Phong için bütün veriler bu structure'da.

ve...

psoDescSolid.PS = ____BYTECODE(pixelShaderSolid.Get());

burda sadece .PS setliyoruz.

Populate list burası ekrana basmak için.

m_commandList->SetPipelineState(

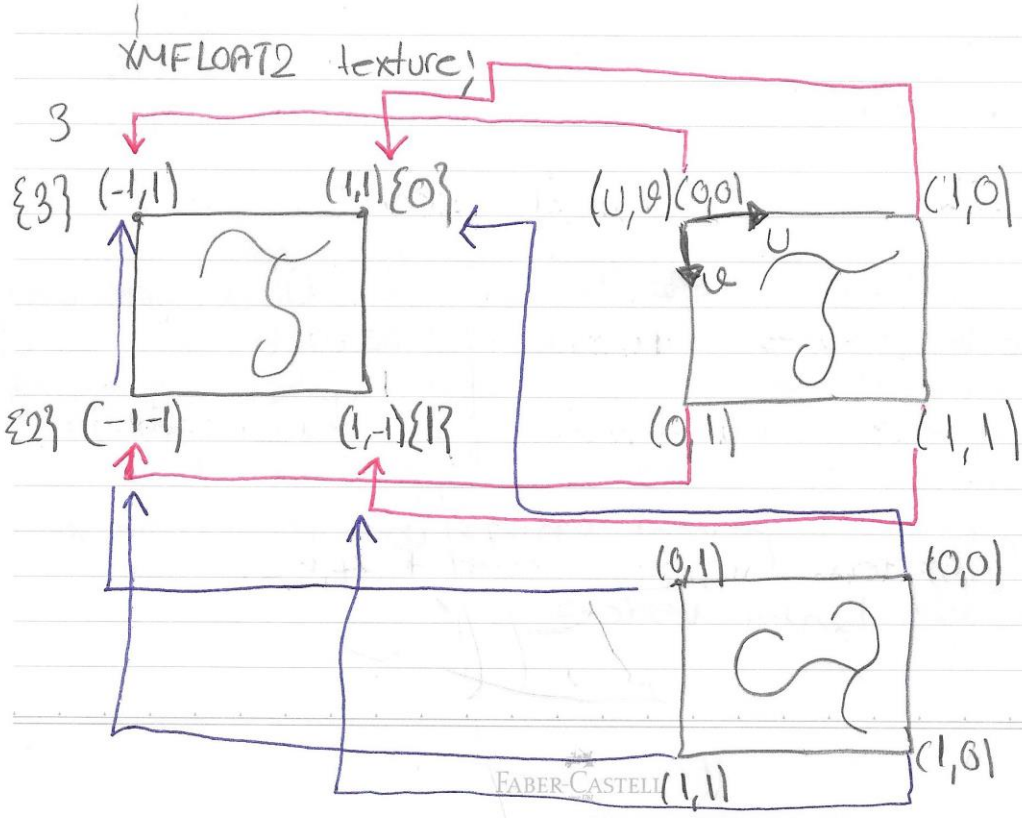
m_pipelineStatePhong.Get());

burda setliyor burası

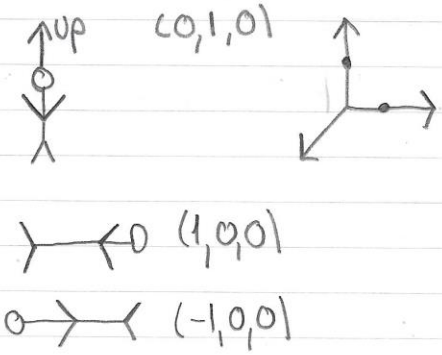
Texture Mapping

Vertex {

XMFLOAT2 texture;



up vektörü bakış noktamızın dik olan.



hlsl de

textureMap.Sample(sampleLinear, input.tex);

resimi nasıl sellicini

cpp sallodigimiz text, u, v koordinatları geliyor buraya.

Stenciling =>

→ iki tane buffer'ımız var.

→ back buffer la aynı boyutta olan stencil buffer var.

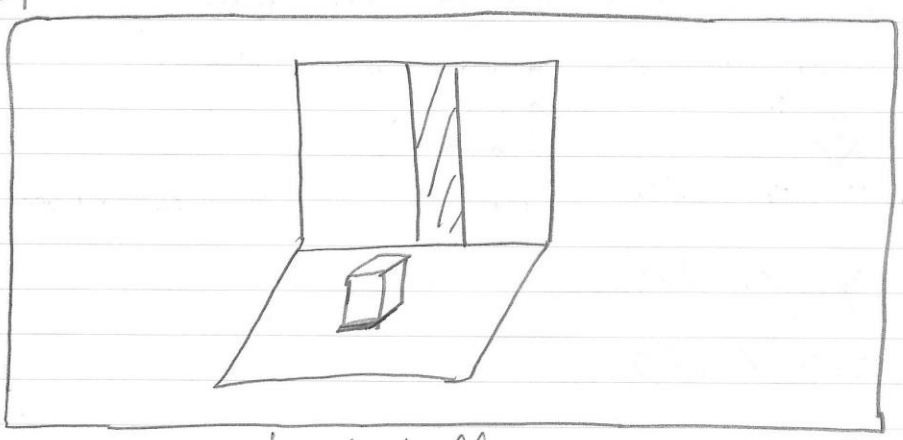
→ stencil buffer 0-255 pixsel değer veriyosun.

→ Hebir pixsele değer veriyosun.

→ back buffeda hebir pixselin (RGB Alfa) değerleri tutuluyor.

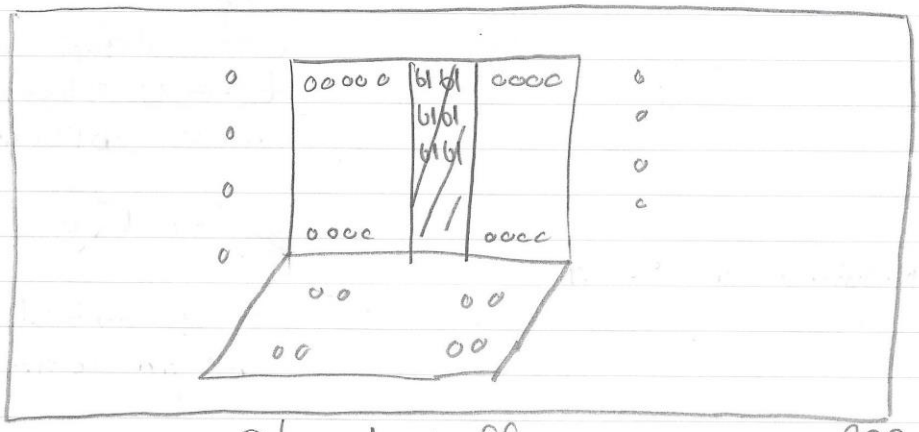
→ stencil buffeda integer değerler tutuyoruz.

(R,G,B)



back buffer 1280 x 720

(0,—255)



Stencil buffer 1280 x 720

back buffer renkleri basıyoruz 0-1

PopulateCommandList ile clearRender komutuyla

Depth buffer → t değeri en küçük değeri olan pixsel renk basıyor.

↳ RT renkleri basılıyor mu? 0-1 değerleri seçilir.

Stencil buffer da 0 seçiliyor. tüm pikseller.

GMSetStencilRef(61);

→ Cisimleri çizerken aynayı sonra bırakıyoruz.

→ m_commandList → GMSetStencilRef(61);
→ aynadaki pixselleri 61 setliyoruz.

→ 3 boyutlu ortamdaki nelerin yansımasını çizmek istersek aynadan.

Küp, gölge, zemin

bu cisimlerin yansımasını çizerken stencil buffer'a bak ve bu buffer'da bulunan 61 değeli pixsellere bu yansımaları çiz diyoruz.

2 tane PipelineState var.
PSG yazar.
- markMirrors → stencil buffer aynayı temsil eden Pix 61.
- drawReflections → cisimlerin yansımalarını daha önceden stencil buffer'a 61 yazılan pixsellere çizilmesine yarıyor.

StencilFunc → stencil buffer üzerinde koşan fonksiyon. pixsellerin hepsi için koşar.

(stencilRef, ilgiliPixsel) → bool. 'dür

Koşulcu fonksiyonu 61, 0 karşılaştırır, bit nedir olsun 61 yazmak istiyoruz. bu yüzden Always'ı setledik.

Eğer bu true döndürürse fonksiyon koşar fark. StencilPassOp bu da replace yapar. stencil buffer ilgili pixsellerini stencil ref. değiştir diyor. burlar gidip 61 yazdırıyor. bu settingleri aynayı çizerken yapılması için aynayı setlerken yapıyoruz. -reflectionDSS;

→ Cisimlerin yansımalarını çizerken daha önceden 61 yaz. pixsellere yansımaları çiz diyoruz.

pos. tex Norm
8 14 24

→ Front face. StencilFunc = _____ FUNC_EQUAL
ilgili değele esitse fark döner. bl pixellede esit dur ve yansımayı orya alıoiz.

StencilPASSOP = ____ KEEP Kou zoten bl.

→ diğerinde REPLACE'di bl kopydiyordık simdi KEEP Kodı zoten bl 'di.

→ İkiside bellek alanı back buff stenci buff

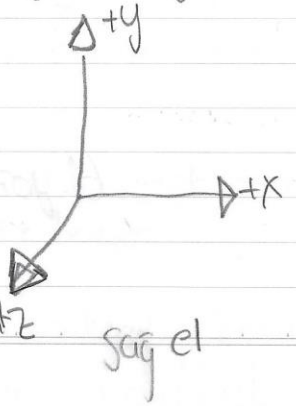
maya Tool

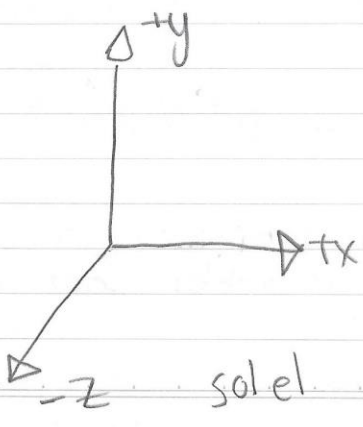
- Move Tool
- Rotate Tool
- Scale Tool
- Extrude Tool seçtiğin nesnenin kopyasını alıkoyar.

→ maya → mb mayabinary

obj kaydediyoruz direx

Maya Kartezyen Koordinat



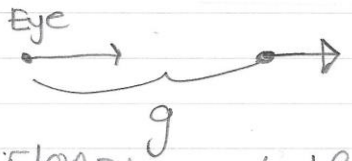


FABER-CASTELL

FireTankMissile

Ro_Tank_Missle = Eye + XMVectorSet(0, -1.65, 0, 0)
Rd_Tank_Missle = XMVector3Normalize(At - Eye);

XMVECTOR: inificPosition = Ro_Tank_Missile + g * Rd_TankMiss



XMFLOAT4 initialPosition_F4; XMStoreFloat4(&initial Position_F4, initialPosition);
g_World_Missile = g_world_Tank; // tank ile aynı dönsün
XMFLOAT4X4 g_World_Missile_4x4; XMStoreFloat4(&g_world_Missle_4x4, g_world_Missle);

TraceTankMissile //Memi gönderildi
float t_walls = IntersectTriangle(Ro_Tank_Missile, Rd_Tank_Missile _____)
If(t_walls >=0 && t_walls<1){ //Çapışma.
FireTankMissile = true;
TraceTank Missile = false
}

Zoom özelliği =>

Zoom out

g_Projection = XMMatrixPerspectiveFovLH(XM_PIDIV4,
1280/(FLOAT)720, 0.01f, 1000.0f);
m_constantBufferData.mProjection = XMMatrixTranspose(
g_Projection);

Zoom in

(XM_PI/12, 1280 ----)

Özgürce gezinme

W → ilerlerken

XMVECTOR Ro = Eye;
XMVECTOR Rd;

Rd = XMVector3Normalize(At - Eye)
float t_walls = IntersectTriangle(Ro, Rd ---)
if(t_walls > 10 || t_walls == 0)
moveBackForward += speed;

S → gerilerken

Rd = XMVector3Normalize(Eye - At)

moveBackForward -= speed;