KARADENİZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
BİLGİSAYAR GRAFİKLERİ LABORATUARI

Pürüzlü Yüzey Üretimi

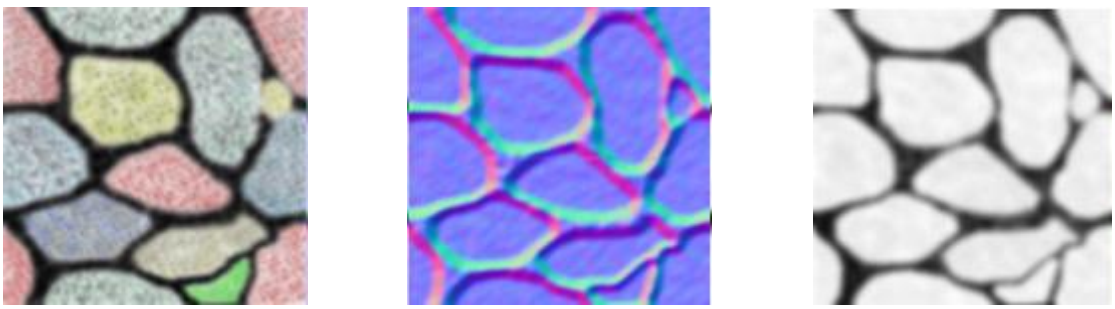
1. Giriş

Yüzeylerin özel bir doku kaplama yapılarak pürüzlü görünmesini sağlayan 2 temel yöntem vardır : 1.Bump Mapping, 2.Parallax Mapping.

Bump Mapping yönteminde Şekil.1(a)'da kaplanacak dokudaki renk değişimlerinden elde edilen (b)'deki "**Normal Map**" dokusu kullanılır. Bu normal mapdeki **(R,G,B)** değerleri yüzeyin normalinin **(X,Y,Z)** değerleri olarak alınır. Yüzeyin diffuse ve specular renk bileşenleri yüzey normaline bağlı olarak hesaplandığından **(0,1,0)** gibi tek bir normal değerine sahip düzlemsel bir yüzeye bump mapping yöntemine göre doku kaplandığında herbir piksel için normalmap dokusundan okunan farklı normallerle hesaplanan renk değerleri sanki yüzlerce farklı poligona sahip pürüzlü bir yüzey varmış hissi verecektir.

Parallax Mapping yöntemi **(R,G,B,A)** renk bileşenlerinden **A-**alpha parlaklık bileşeninden elde edilen Şekil.1(c)'deki "**Height Map**" dokusu ile yüzeyin yüksekliğini değiştirerek görüntüler ve böylece tümsekler/çukurlar oluşur. Bump mappingden farklı olarak yüzeyin koordinatları y-ekseni boyunca piksel mertebesinde gerçekten artar/azalır. Dolayısyla elde edilen pürüzlü yüzeylerdeki tümsekler/çukurlar, bump mapping yöntemine nazaran daha fazladır. Hatta Parallax mapping yönteminde bu derinlik değerini ayarlamak bile mümkündür.

Deneyde **Jason Zink**'in, "**A Closer Look At Parallax Occlusion Mapping**" başlıklı makalesi ve örnek HLSL programından yararlanılarak geliştirilen DirectX 11 uygulaması ile yukarıda bahsedilen özel doku kaplama yöntemleri incelenecektir.



(a) (b) (c)

Şekil 1: (a)'daki dokudan elde edilen (b) Normal Map ve (c) Height Map dokuları

Bilindiği gibi DirectX uygulamalarında **.cpp** uzantılı program dosyasında çizilecek grafiği oluşturan poligonlara ait koordinat, normal, doku bilgileri ve **World, View, Projection** matrisleri setlenir. Bu bilgiler kullanılarak en son ekrana çizilecek şekle ait renk değerleri **.fx** uzantılı HLSL programındaki Vertex ve Pixel shader fonksiyonları aracılığıyla hesaplanır. Uygulamada üzerine doku kaplanacak yüzey düzlemsel olduğundan **.cpp** programında iki üçgen ile temsil edilmiş ve normal, doku koordinatları gerektiği gibi setlenmiştir.

2. Parallax Mapping Yöntemi

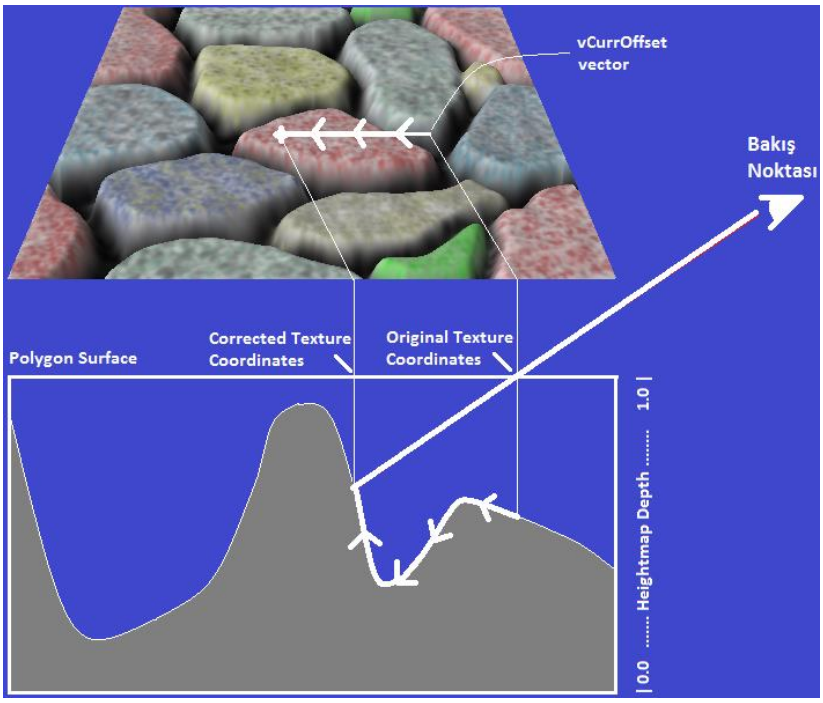
Parallax mapping yöntemi ile pürüzlü yüzey üretilirken ilk işlem bakış noktasından **eye** ışını yollamaktır. Bu ışının yüzeye kaplanacak dokuda hangi renk ile kesiştiğinin belirlenebilmesi için **3D** uzaydan **2D (u,v)** doku uzayına bir dönüşüm yapılmalıdır. Bu dönüşüm **Normal (0,1,0)**, **Tangent (1,0,0)** ve bu ikisinin vektörel çarpımları ile hesaplanan **Binormal (0,0,-1)** vektörlerinden elde edilen matris ile ışının doğrultusu çarpılarak HLSL programındaki Vertex shader fonksiyonunda yapılır. **2D** doku uzayına izdüşürülen ışının dokuda kesiştiği koordinatlardaki rengin alpha değerine ve gerekirse ışının izdüşüm doğrultusu **vCurrOffset** boyunca başka alphalara bağlı olarak **Parallax Mapping** yönteminin nasıl gerçeklendiği aşağıda verilen Pixel shader kodu ve Şekil.2 üzerinden anlatılacaktır:

while döngüsünden önce **SampleGrad** fonksiyonu **IN.texcoord + vCurrOffset** parametresi ile çağrılmıştır. Burda **IN.texcoord** doku koordinatına eklenen **vCurrOffset**, Şekil.2'den de görüldüğü gibi doku üzerinde ilerlemede kullanılan doğrultu vektörüdür. Hesaplanması dolaylı yoldan **eye** vektörüne dayanır. **SampleGrad** fonksiyonunun sonundaki **.a** ifadesi ile **NormalHeightMap** isimli dokudan **fCurrSampledHeight a**lpha değeri okunur. Bu değer ile bakış noktasından yollanan **eye** ışınının, başlangıç değeri **1**'e setlenmiş yüksekliği **fCurrRayHeight** karşılaştırılır. **fCurrSampledHeight < fCurrRayHeight** yani dokudan okunan alpha, ışının yüksekliğinden küçük olduğu müddetçe **fCurrRayHeight** değeri **fStepSize** kadar azaltılır. Işının yüksekliği değiştikçe yeni okunan alphaların değerleri Şekil.2'deki Height Map eğrisi ile temsil edilmiştir. **fStepSize * vMaxOffset** ile **vCurrOffset** vektörü güncellenerek yeni **fCurrSampledHeight** değeri okunur. Dokudan okunan alpha değeri ışının yüksekliğinden **>=** olduğunda döngüden çıkılır. Bu nokta aynı zamanda ışının Height Map eğrisiyle kesiştiği noktadır. Dokuda bu noktaya karşılık gelen renk ekrana basılarak parallax mapping yöntemi gerçeklenmiş olur. Şekilde doku kaplanacak yüzey **polygon surface** ile temsil edilmiştir. Pürüzlü yüzeyin oluşması, ışının **polygon surface** ile kesiştiği nokta değil, alpha bileşeninden elde edilen height map eğrisi ile kesiştiği noktaya karşılık gelen doku koordinatlarındaki **vFinalColor** renginin görüntülenmesine dayanmaktadır.

```
fCurrSampledHeight = NormalHeightMap.SampleGrad( samLinear,
                                    IN.texcoord + vCurrOffset, dx, dy ).a;

while ( fCurrSampledHeight < fCurrRayHeight )
{
      fCurrRayHeight      -= fStepSize;
      vCurrOffset         += fStepSize * vMaxOffset;
      fCurrSampledHeight   = NormalHeightMap.SampleGrad( samLinear,
                                    IN.texcoord + vCurrOffset, dx, dy ).a;
}
```

```
float2 vFinalCoords  = IN.texcoord + vCurrOffset;
float4 vFinalColor   = ColorMap.SampleGrad( samLinear, vFinalCoords, dx, dy );
float4 vFinalNormal  = NormalHeightMap.SampleGrad(samLinear, vFinalCoords, dx, dy );
vFinalNormal         = vFinalNormal * 2.0f - 1.0f;
float3 vAmbient      = vFinalColor.rgb * 0.3f;
float3 vDiffuse      = vFinalColor.rgb * max(0.0f, dot( L, vFinalNormal.xyz )) * 0.7f;
vFinalColor.rgb      = vAmbient + vDiffuse;
OUT.color            = vFinalColor;
```



Şekil 2: Alpha eğrisi ve ışının yüksekliğine bağlı olarak piksel renginin bulunması

3. Bump Mapping Yöntemi

Giriş bölümünde de bahsedildiği gibi **Bump Mapping** yöntemi yüzey normalini "normal map" denilen özel bir dokudaki renk değerleri olarak almaya dayanır. Kaynak kodlardaki **`Textures`** klasöründe bulunan dokulara dikkat edilirse HLSL'deki **`ColorMap`** için **`*_colormap.dds`**; **`NormalHeightMap`** için de **`*_normalmap.dds`** gibi iki doku vardır. **`*_colormap.dds`** dokusundan hangi **`vFinalColor`** renginin okunacağına parallax mapping yöntemine göre yukarıda anlatıldığı gibi karar verilir. **`*_normalmap.dds`** dokusunu hem parallax mapping hem de bump mapping yöntemi kullanır. Gerçekte **`*_heightmap.dds`** gibi bir doku yoktur. Çünkü parallax mapping **`*_normalmap.dds`** 'nin **`(R,G,B,A)`** bileşenlerinden **`A-`**alphayı; bump mapping de yukarıdaki kod parçasından da görüldüğü gibi **`(R,G,B)`**'yi kullanır ve yeni yüzey normali **`vFinalNormal`** olarak alır. Böylece bump mapping yöntemi de gerçeklenmiş olur. Her bir piksel için farklı yüzey normali kullanılması her bir pikselin için farklı diffuse ve specular renk değeri hesaplanacağı anlamına gelir. Böylece yüzeyde pürüzler varmış gibi görülür. **`*_normalmap.dds`**'deki **`(R,G,B)`** değerleri **`[0..1]`** arası değiştiğinden **`3D`** uzayda normalize edilmiş **`[-1..1]`** aralığına map etmek için **`vFinalNormal * 2.0 - 1.0`** işlemi yapılmıştır.

Örnek programda klavyenin sağ/sol tuşları **`fStepSize`** değişkenini; yukarı/aşağı tuşu da yükseklik değerini arttırıp/azaltmaktadır. Ayrıca **"`Q`"**, **"`W`"** ve **"`E`"** tuşları ile değişik dokular arasında geçiş yapmak mümkündür.

Şekil 3: Specular renk bileşeni eklenmiş görüntü

4. Deney Hazırlığı

Örnek program yalnızca ambient ve diffuse renk bileşenlerini hesaplamaktadır. **`ParallaxMapping11.fx`**'e gerekli kodları yazarak bunlara Şekil.3'teki gibi specular renk bileşenini ekleyiniz. Bakış noktasına doğru olan vektör olarak **`E`** vektörünü kullanınız. Işık kaynağından gelen vektör olarak da **`L`** vektörünü kullanınız.

5. Deney Soruları

1. Yüzey normali olarak **`vFinalNormal`** yerine yüzeyin kendi normali **`N`** kullanılacak şekilde kodu güncelleyiniz ve öncekinden farklı yönlerini açıklayınız. Örneğin iki görüntü arasındaki specular renk farklılıkları neden oluşmuştur?
2. Yüzeyin kendi normali **`N`** kullanılırken yukarı tuşuna sürekli basılıp yükseklik sıfırlandığında oluşan görüntüyü yorumlayınız. Yükseklik sıfır iken **`vFinalNormal`**'e göre çizilen görüntü ile normal **`N`** alındığında oluşan arasındaki farkın sebebi nedir?
3. Örnek program hem parallax hem de bump mapping yöntemini gerçeklemektedir. Sadece bump mapping yönteminin etkisini veya sadece parallax mapping yönteminin etkisini görmek için hangi değişiklikleri yapmak gerekir?

6. Deney Tasarımı ve Uygulaması

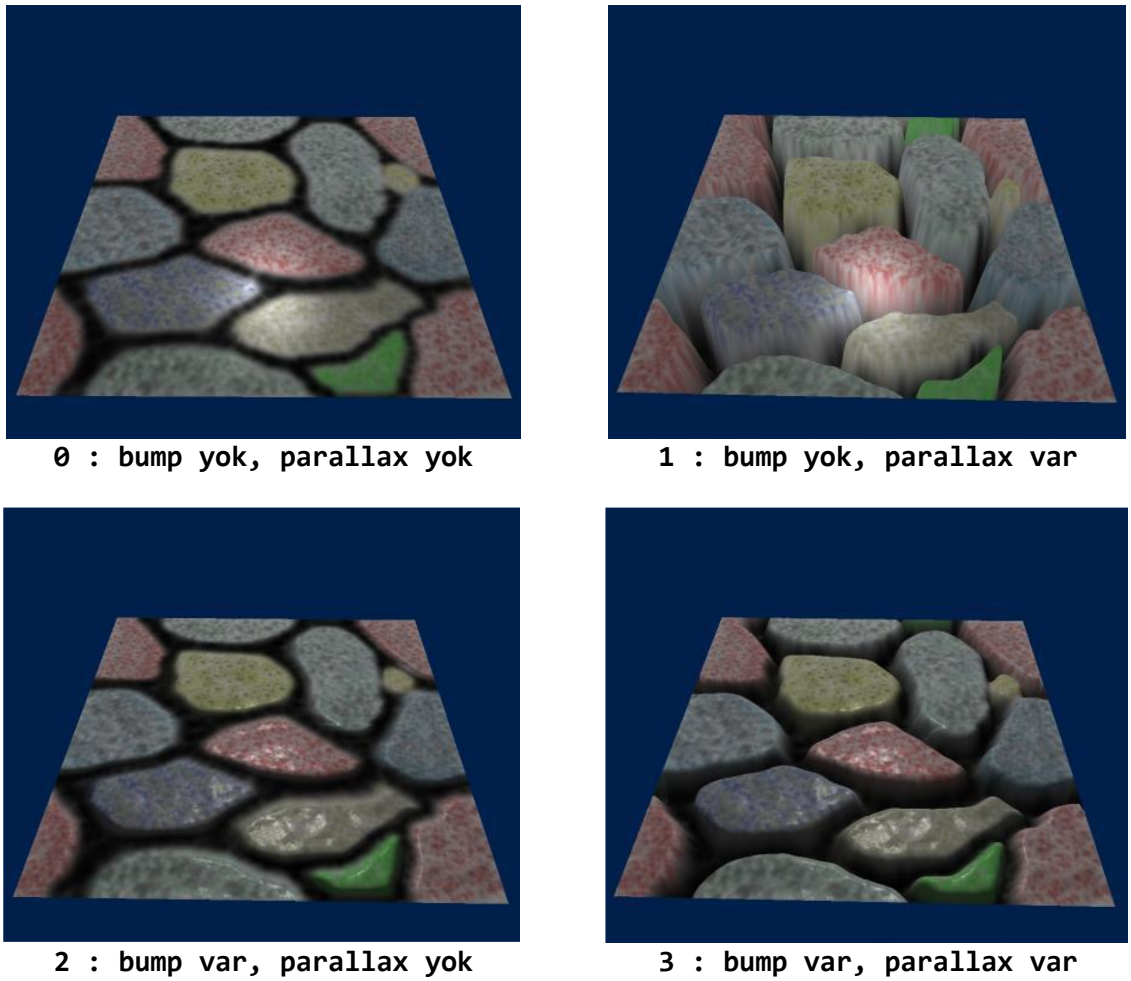
"**`0`**", "**`1`**", "**`2`**" ve "**`3`**" tuşlarına basıldığında Şekil.4'teki görüntüler elde edilecek şekilde programı güncelleyiniz. Görüntülerde :

"**`0`**" : bump mapping **yok**, parallax mapping **yok**,
"**`1`**" : bump mapping **yok**, parallax mapping **var**,
"**`2`**" : bump mapping **var**, parallax mapping **yok**,
"**`3`**" : bump mapping **var** parallax mapping **var**

olarak özetleyebileceğimiz etkileri görüyoruz.

İpucu → **.cpp** programında 24. satırdaki `ConstantBuffer` adlı struct'a **`int key`** değişkeni ekleyiniz. 201. satırdaki **`WndProc()`** içinde **`case 'E':`** 'den sonra 4 tane daha **`case`** ekleyiniz ve basılan tuşa göre **`key`**'i setleyiniz. **`.fx`** programının 12. satırında da **`.cpp`**'dekinin benzeri bir `ConstantBuffer` vardır. Buna da bir **`int key`** değişkeni ekleyiniz. **`.cpp`**'deki **`key`**'in değerinin **`.fx`**'teki eşdeğerine otomatik aktarıldığını varsayınız. Basılan tuşun değerini tutan **`key`**'e göre **`PS()`** adlı pixel shaderda 100. satırdaki **`fParallaxLimit`** değişkenini **`0`**'a setleyerek parallax mapping etkisini kaldırabilirsiniz. Bump mapping etkisini kaldırmak için de 200. satırdaki **`vFinalNormal`** değişkenini yine **`key`**'e göre setleyiniz. Program default olarak hem bump hem de parallax etkisini gerçeklediğinden **`key=3`** için **`.fx`**'te kod yazmanız gerekmiyor. Fakat **`.cpp`**'de basılan tuşa göre **`key=3`** setlemesini yapmalısınız.

Kaynak kodların olduğu klasörde Şekil.4'teki bump/parallax mapping modlarını içeren **`bpModes.mp4`** isimli video paylaşılmıştır.



Şekil 4: bump/parallax mapping modları

7. Deney Raporu

Deney Raporunu, **`Rapor.docx`** adlı şablon belgeye göre takım adına hazırlayıp **Deney Hazırlığı** çalışması **`ParallaxMapping11.fx`** ile **deney saatine kadar** (takım adına biriniz) dersin Moodle Sayfasına yükleyiniz.