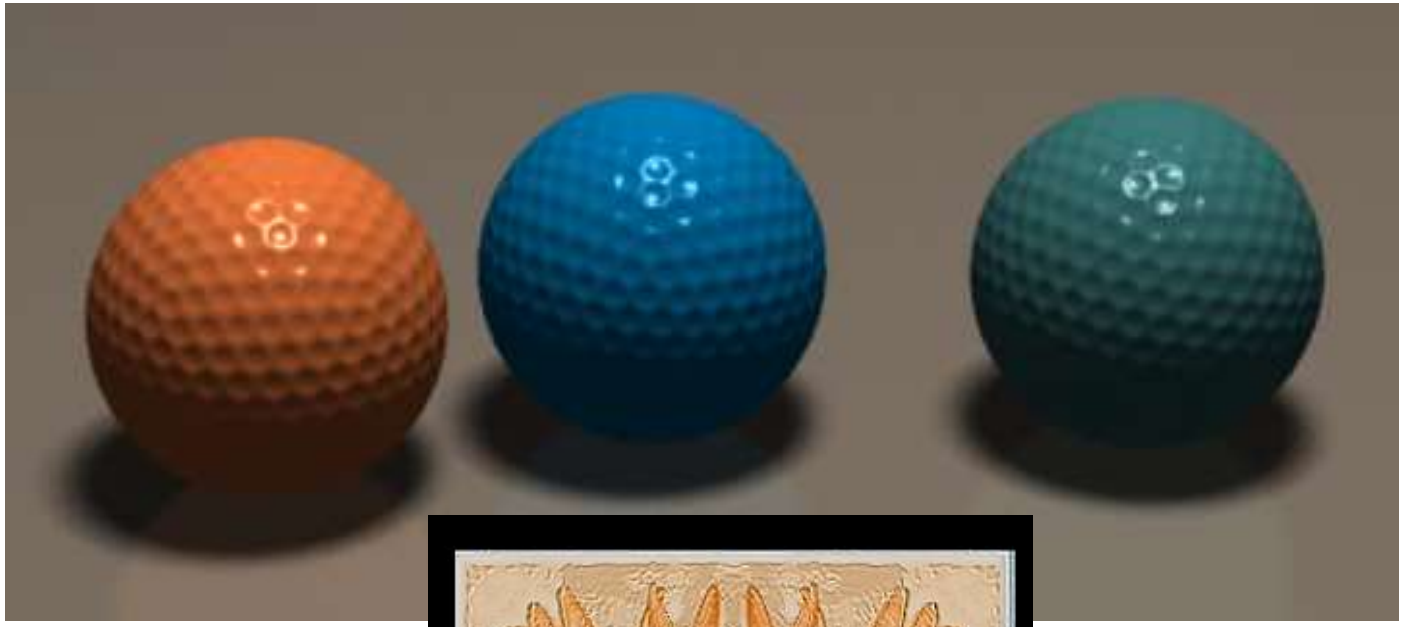
Iškilumų žemėlapiai

T120B167 Žaidimų grafinių specialiųjų efektų kūrimas ir programavimas

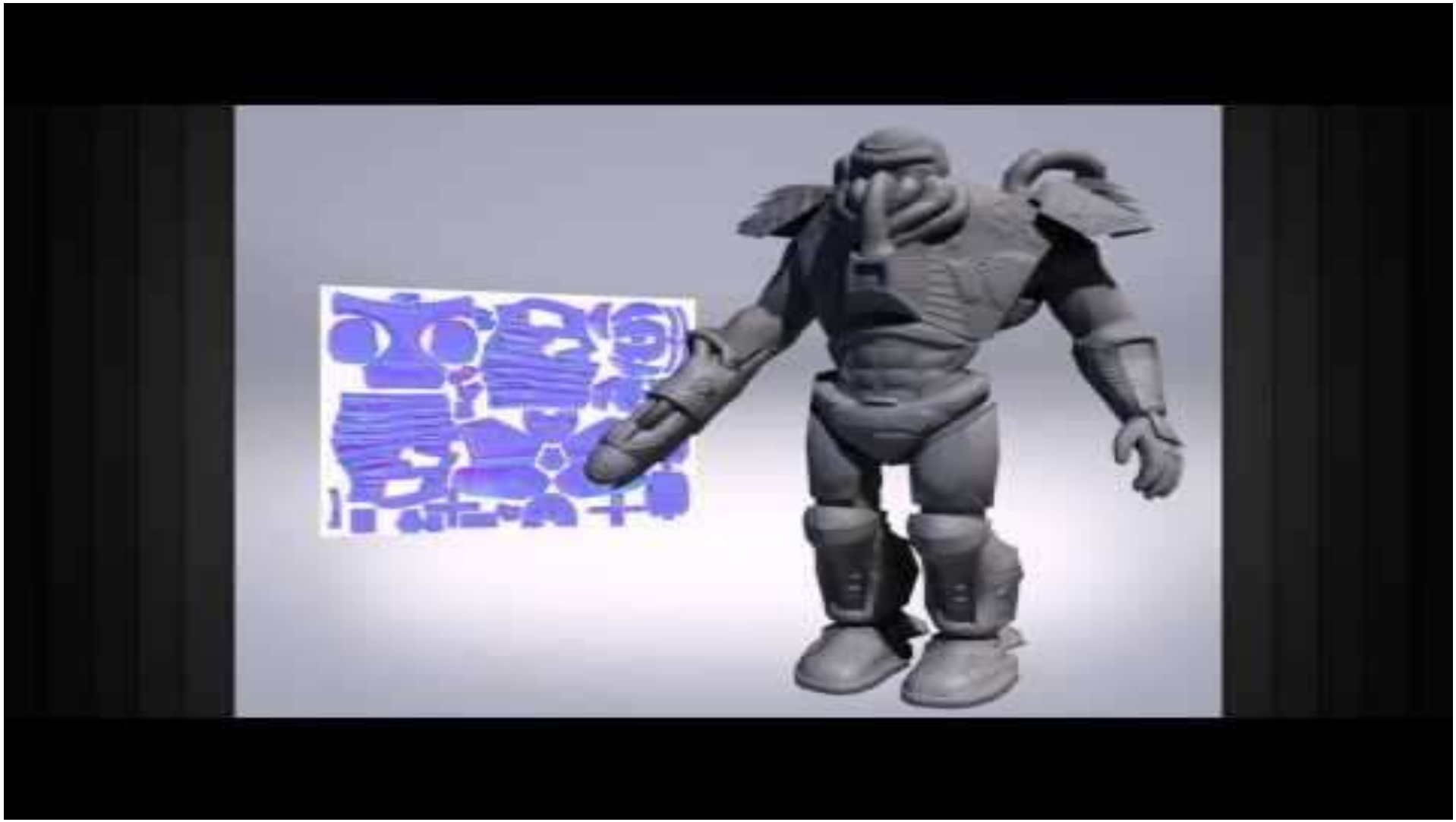
Rytis Maskeliūnas
Skype: rytmask
Rytis.Maskeliunas@ktu.lt

Paskaitos tema

MULTIMEDIJOS INŽINERIJOS KATEDRA



Normalių žemėlapis



https://www.youtube.com/watch?v=yHzIx41eiD4

Kas tai?

- Tiek normalių, tiek iškilumų žemėlapiai turi bendrą tikslą – imituoja detalesnį 3D paviršių nei jis yra iš tiesų;
- Atrodo kad paviršius turi daugybę mažų elementų ir nėra visiškai plokščias;
- Modifikuojamas tik kiekvieno pikselio šeideriavimas, todėl objektas neduos papildomų šešėlių (nėra nuo ko) nei užstos kitą objektą;
- Kameros jus išduos. Tam tikru kampu pasukus kamerą matysite kad paviršius visgi nėra toks jau detalus...

Kaip tai veikia?

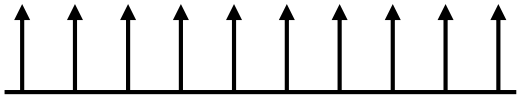
MULTIMEDIJOS INŽINERIJOS KATEDRA

- Abiem atvejais modifikuojamas normalės kampas (kryptis statmena paviršiui), taip įtakojant kaip šeideriuojamas pikselis.
- Dažnai naudojami kaip sinonimai, bet..... kai kuo jie skiriasi
- **Iškilumų žemėlapiuose šalia tekstūros saugomas ryškumas (angl., intensity),** t.y. reliatyvus pikselių aukštis nuo kameros žiūrėjimo kampo. Atrodo, kad pikseliai yra patraukti norimu atstumu link paviršiaus normalių krypties. Galima naudoti greyscale paveikslus arba RGB tekstūros ryškumo vertes. Tai senesnis, klasikinis būdas, galima pasidaryti „rankomis“ (pvz., photoshop).
- **Normalių žemėlapiuose saugoma kryptis,** t.y. paveikslo RGB vertėse nurodoma normalių kryptis. Būdas yra tikslesnis, nes čia imituojama, kad pikselis gali būti „atitrauktas“ bet kuria kryptimi (nei tik linija į kamerą). Trūkumas – sunkiau pasigaminti, dažnai yra generuojami iš aukštesnės raiškos objektų.

Iškilumų žemėlapis

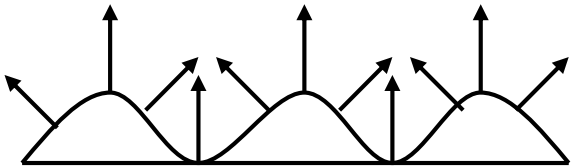
MULTIMEDIJOS INŽINERIJOS KATEDRA

- Paviršius

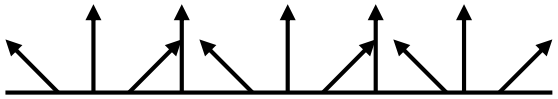


- Iškilumų žemėlapis

- „Tikslas“

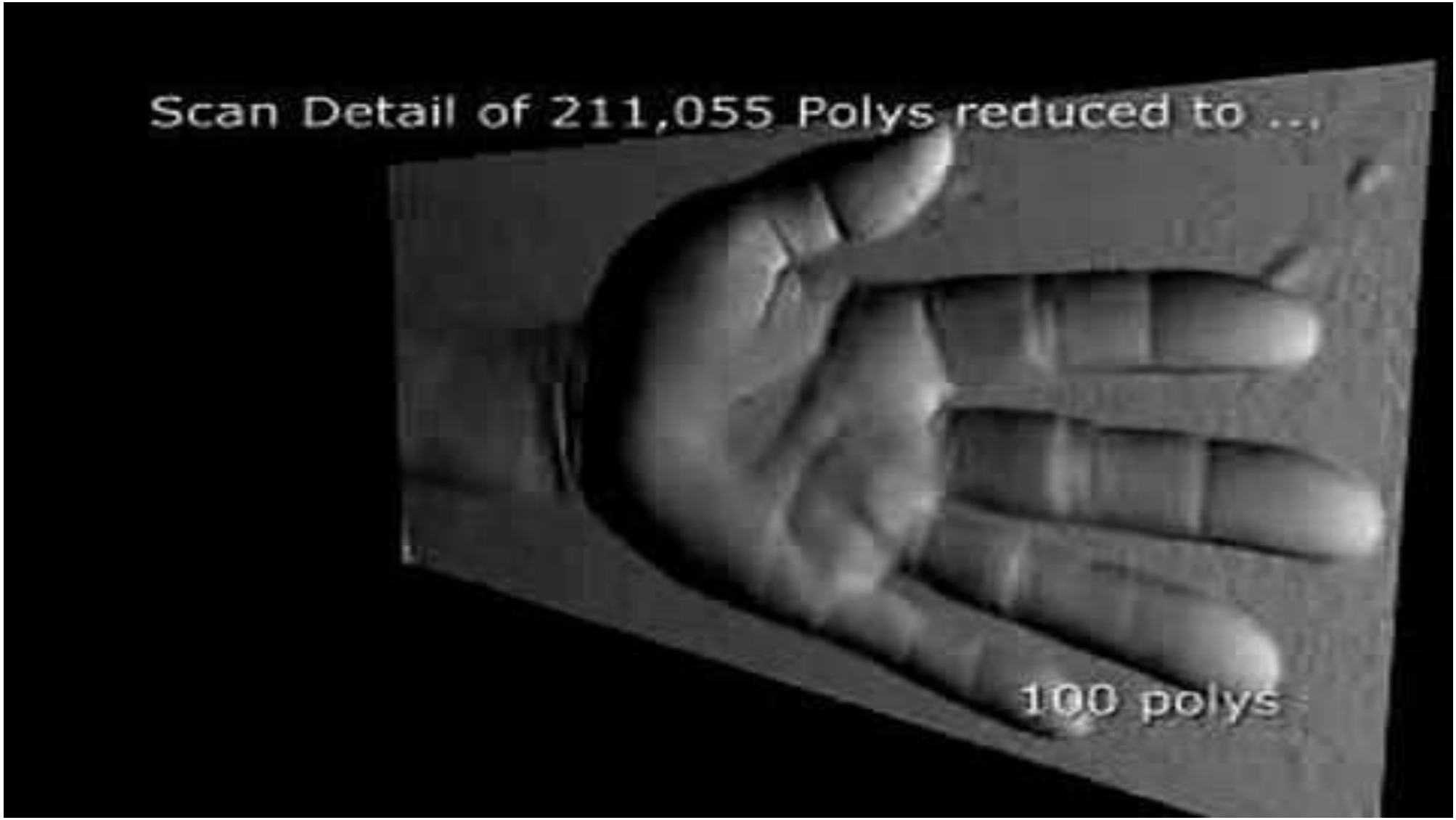


- „Kas gaunasi“



Nauda apčiuopiama!

MULTIMEDIJOS INŽINERIJOS KATEDRA



https://www.youtube.com/watch?v=RSmjxcAhkfE

Iškilumų žemėlapis

Privalumai aiškus:

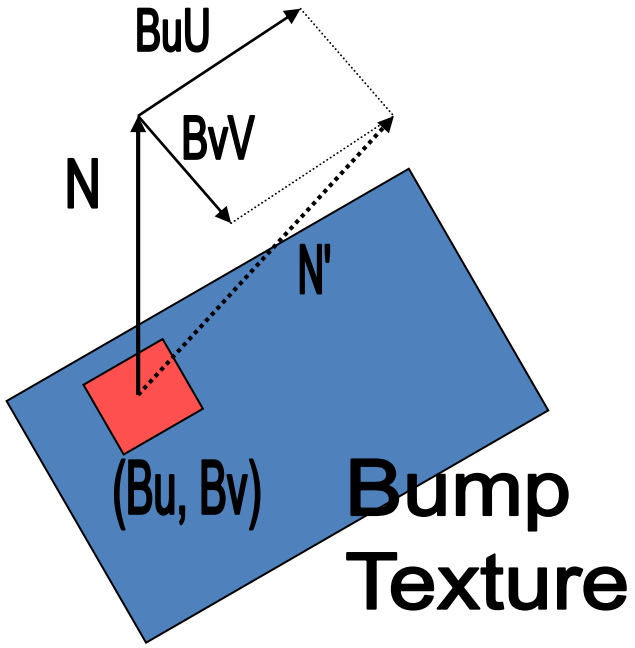
- Mažiau modeliavimo laiko, paprastesni modeliai (mažiau realių iškilumų ir tuo pačiu poligonų);
- Vienas žemėlapis gali būti panaudotas dideliam plotui;
- Lengviau pagaminti nei normalių žemėlapius;

Trūkumai:

- Iš tiesų paviršiaus geometrija nekeičiama, todėl tam tikru kampu galima aptikti optinę apgaulę.
- „Iškilumai“ savaime nesukuria šešėlių (nebent gaminti atskirą šešėlių algoritmą)

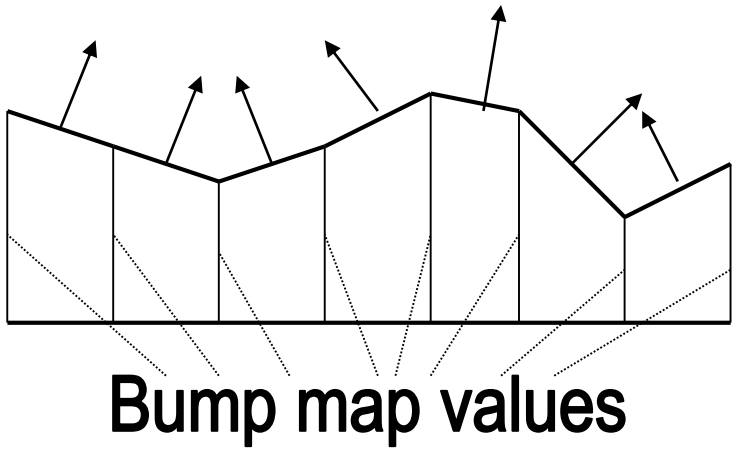
Iškilumų žemėlapis

- Iškilumų žemėlapyje saugoma
  - Atskaitinis (angl., offset) vektorių žemėlapis:
    - 2 vertės, b_u ir b_v saugomos kiekvienoje pozicijoje
    - b_u ir b_v naudojamos nuo normalės vektoriaus atskaitos u ir v kryptimis



Iškilumų žemėlapis

- Iškilumų žemėlapyje saugoma
  - Aukščio žemėlapis:
    - 1 vertė, h saugoma kiekvienoje pozicijoje
    - h pagal ją nustatomas aukštis
    - h vertės naudojamos išvesti b_u ir b_v vertes imant skirtumus tarp kaimynų u ir v kryptimis
    - Bu ir Bv taikomos kaip ir atskaitinio vektoriaus žemėlapyje



Iškilumų žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Kada geriau nenaudoti iškilumų žemėlapio?
  - Jeigu nėra spindinčių akcentų, judančių šviesų ir pats objektas nejudės – geriau naudoti šviesų žemėlapyje, o jame saugoti kaip atrodys šviesa ant „iškilumų“

Iškilumų žemėlapis

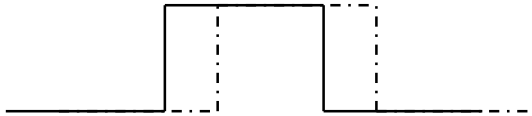
- Iškilumų žemėlapių tipai:
  - Emboss (reljefiniai) Bump Mapping
  - Dot Product Bump Mapping (DOT3) arba NORMALIŲ žemėlapiai
  - Environment (aplinkos) Map Bump Mapping (EMBM)

Reljefinis iškilumų žemėlapis

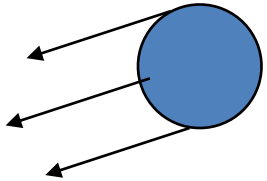
- Principas remiasi reljefiniu 2D vaizdų apdorojimu angliškai vadinamu *embossing*
- Difuzinės šviesos lygtis: $I_d = I_i (L \bullet N)$
  - Tikrasis iškilumų žemėlapis „reguliuoja“ N kiekvienam pikseliui
  - Reljefinis iškilumų žemėlapis aproksimuoja (L•N)
- Atskaitai nuo paviršiaus naudojamas aukščio dydis (angl., heightfield)
  - Pirmoji jo dedamoji vaizduoja nuožulnumą (slope): *m*
  - *m* naudojamas padidinti ar pamažinti bazinę difuzijos vertę:
  - *(Bazinė difuzijos vertė + m)* aproksimuoja *(L•N)* per pikselį

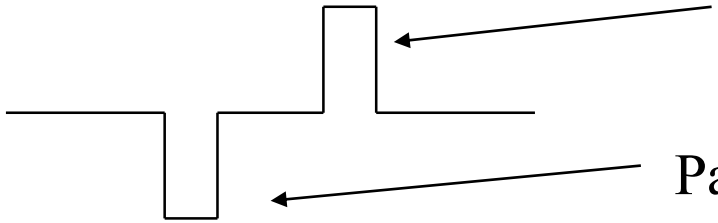
Reljefinis iškilumų žemėlapis

Originalus iškilumas *(H0)*



Originalus iškilumas *(H0)* persidengiantis su antru iškilumu *(H1)* link šviesos šaltinio





Originalus iškilumas atimamas iš antrojo *(H1-H0)*

Reljefinis iškilumų žemėlapis

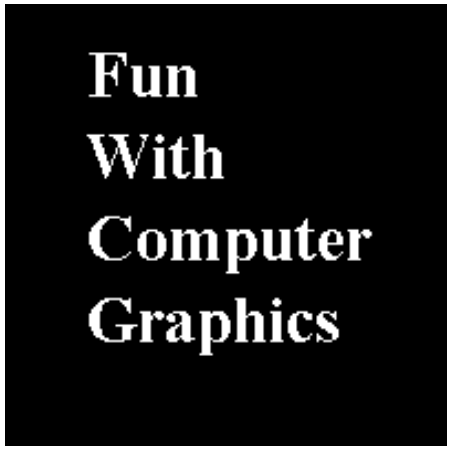
- Reljefuojant:
  - Matomas aukštis *H0* taške *(u,v)*
  - Matomas aukštis *H1* taške link šviesos šaltinio *(u+Δu, v+Δv)*
  - Originalus aukštis *H0* atimamas iš *H1*
  - Skirtumas vaizduoja nuolydį *m=H1-H0*

Reljefinis iškilumų žemėlapis

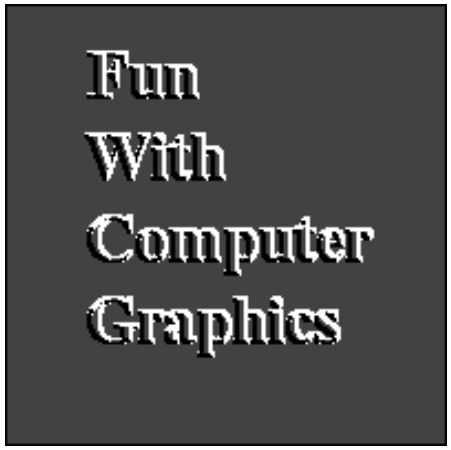
Algoritmas toks:

1. Renderiuojamas paviršius su aukščiu apibrėžiamu difuzine juodai balta tekstūra (monochrome)
2. Viršūnėlių koordinatės (u, v) pastumiamos link šviesos
3. Paviršius renderiuojamas su pastumtu aukščiu kaip difuzinė tekstūra, atimant iš pirmo ciklo rezultato → taip gaunamas reljefinis efektas.
4. Paviršius dar kartą surenderiuojamas be aukščio, apšviečiamas difuzine šviesa ir šeideriuojamas, o gautas vaizdas pridedamas prie relfjefavimo rezultato.

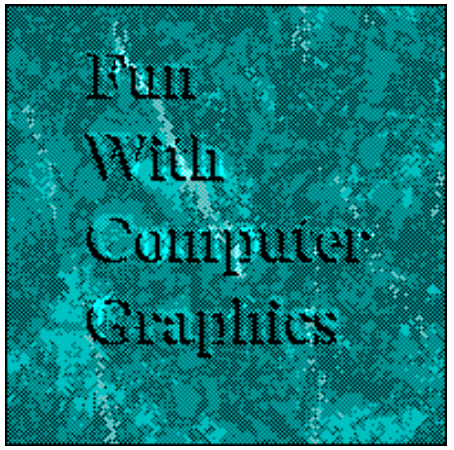
Reljefinis iškilumų žemėlapis



Aukštis



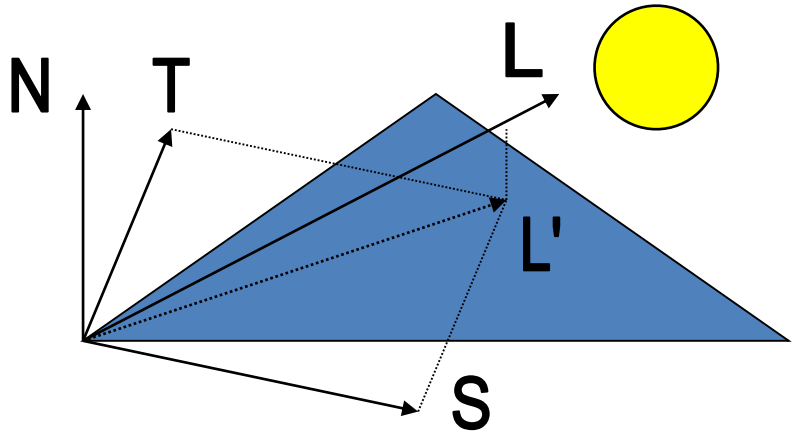
Reljefuotas



Pridėta difuzija

Reljefinis iškilumų žemėlapis

- Sunkiausia yra nustatyti kiek pastumti viršūnių vertes antrame žingsnyje.
- Reikia nustatyti šviesos kryptį reliatyviai paviršiui
  - Tam šviesa transformuojam iš globalios į viršūnėlių tangentinę erdvę.
- Tokia tangentinė koordinačių sistema apibrėžiama:
  - Paviršiaus Normale n, ties nagrinėjama viršūnėle
  - Paviršiaus vektoriumi s, ties tekstūros ašimi u
  - Paviršiaus vektoriumi t, ties kita tekstūros ašimi v



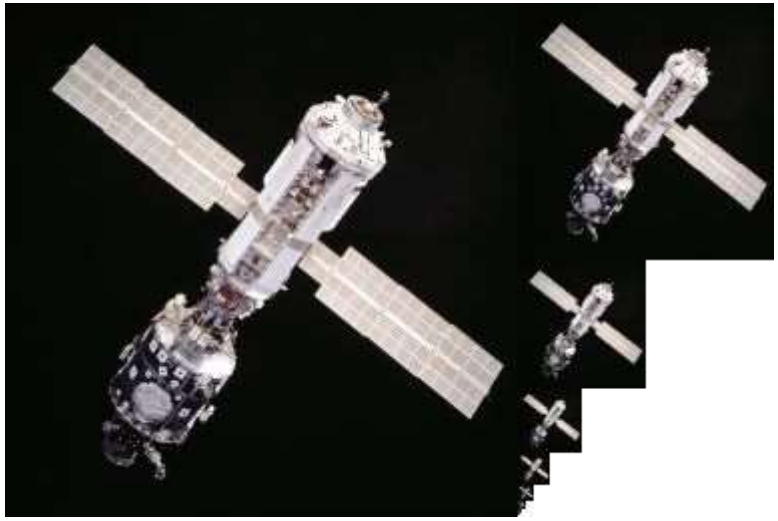
Reljefinis iškilumų žemėlapis

S_x	S_y	S_z	0
T_x	T_y	T_z	0
N_x	N_y	N_z	0
0	0	0	1

- Ši matrica naudojama transformuoti šviesų vektorių (vektorius iš viršūnėlės į šviesą) į tangentinę erdvę.
- Gautas šviesos vektorius projektuojamas ST erdvėje → L’
- L’ x ir y koordinatės naudojamos tekstūros koordinačių (u, v) pastūmimui link šviesos šaltinio.
- Matrica skaičiuojama viršūnėlėms siekiant nustatyti (u, v) poslinkius kas kiekvieną viršūnėlę.

Reljefinis iškilumų žemėlapis

- Apribojimai:
  - Tinka tik difuziniams paviršiams – negalima suformuoti atspindžių
  - Kai šviesa yra tiesiai virš paviršiaus nesimato jokio iškilumo (nėra kur pastumti)
  - Algoritmas „nemoka“ vaizduoti iškilumų esančių už šviesos
  - Negalima naudoti Mipmap filtravimo:
  - Kas yra mipmap? ->



Aplinkos žemėlapio iškilumų žemėlapiai...

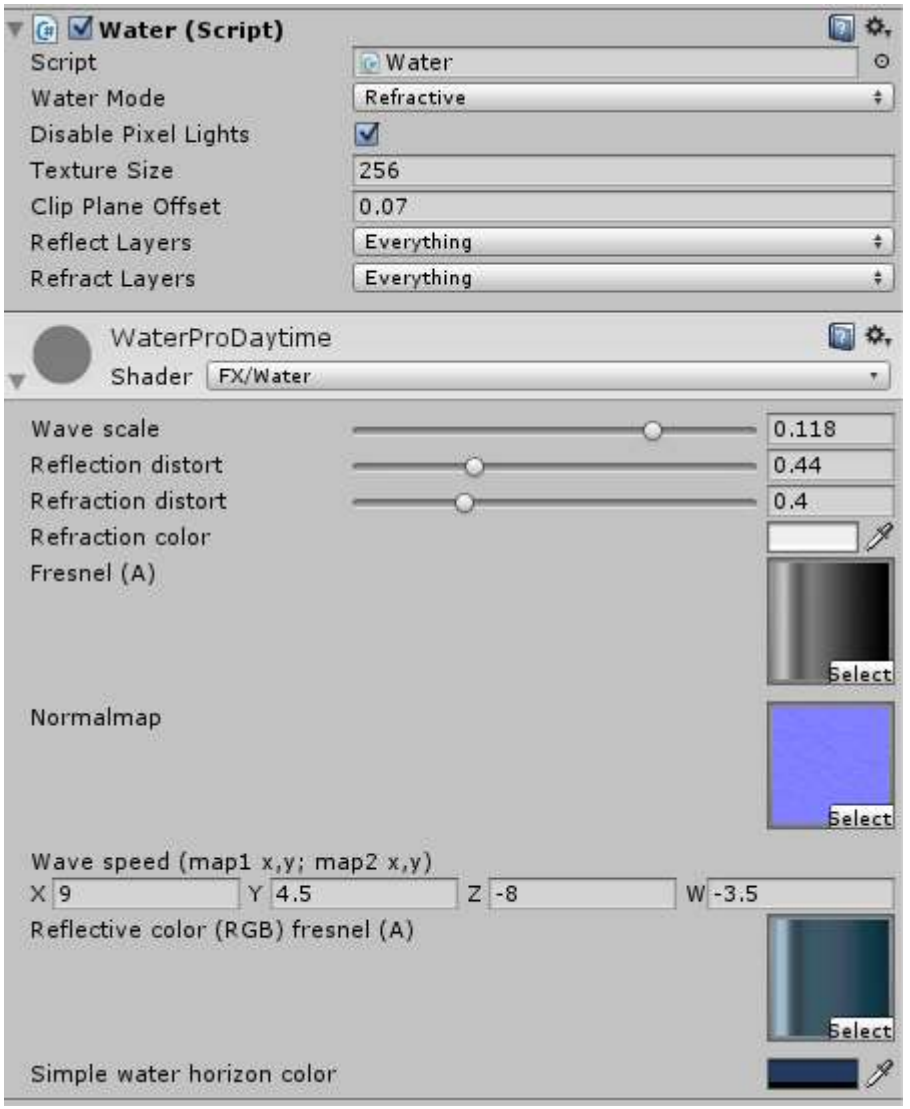
- Angl. Environment Map Bump Mapping
- Naudojami vaizduoti iškilumus blizgiuose atspindinčiuose paviršiuose.
- Tikslas iškraipyti aplinkos žemėlapio koordinates per u ir v diferencialus iš iškilumų žemėlapio tekstūros.
  - Gaunamas modifikuotas atspindžio (R Reflection) vektorius kuris iškraipo atspindimą vaizdą.
    - Dažniausiai naudojama vandenyje matomiems atvaizdams modeliuoti.

Aplinkos žemėlapio iškilumų žemėlapiai... Žr. į vandenį

- Assets>Import Package>Effects

Properties:

_WaveScale	Range: Wave scale
_ReflDistort	Range: Reflection distort
_RefrDistort	Range: Refraction distort
_RefrColor	Color: Refraction color
_Fresnel	Texture: Fresnel (A)
_BumpMap	Texture: Normalmap
WaveSpeed	Vector: Wave speed (map1 x,y; map2 x,y)
_ReflectiveColor	Texture: Reflective color (RGB) fresnel (A)
_HorizonColor	Color: Simple water horizon color
_ReflectionTex	Texture: Internal Reflection
_RefractionTex	Texture: Internal Refraction



Perkėlimų žemėlapis

- Iškilumų žemėlapiuose aukštumo dydis (Heighfield) naudojamas paviršiaus normalių korekcijai
- Perkėlimo (Displacement) žemėlapiuose pačios viršūnėlės yra perkeliamos per duotąjį aukštį palei paviršiaus normalių kryptį.
- Šis metodas labai imlus HW resursams.

Perkėlimų žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Principas pirmą kartą praktiškai panaudotas Pixar's Renderman animacijos renderiavimo pakete (ne real-time).

- Real-time reikia:
  - Siųsti mažo poligonų skaičiaus modelį į HW
  - Siųsti aukščio žemėlapį į HW
  - HW turi pagaminti teseliuotą paviršių koreguojant naujai sukurtas viršūnėles pagal duotus aukštumus
- Būtinas specialus HW ir API (>=DX9)

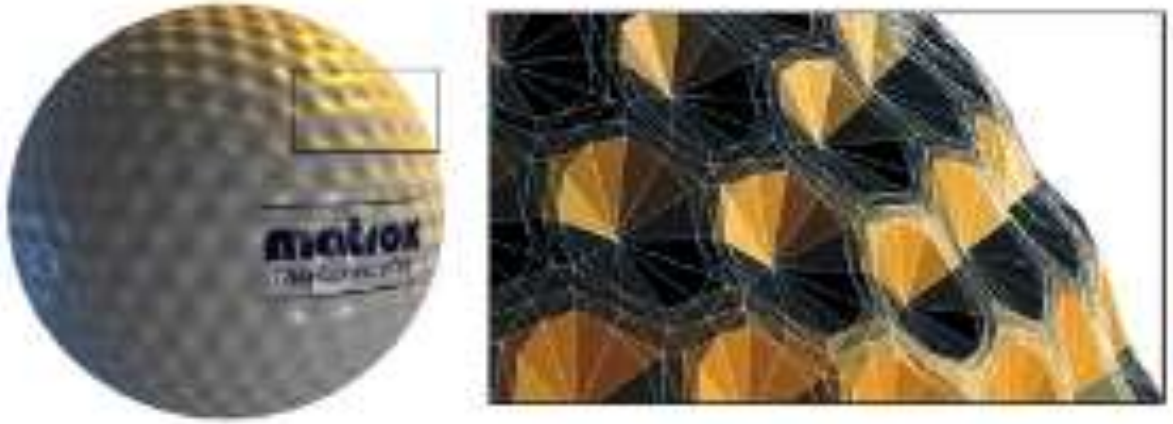
Perkėlimų žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Pagrindinis privalumas – iškilumai matomi bet kokiu kampu!

Grand Canyon rendered using Hardware Displacement Mapping.

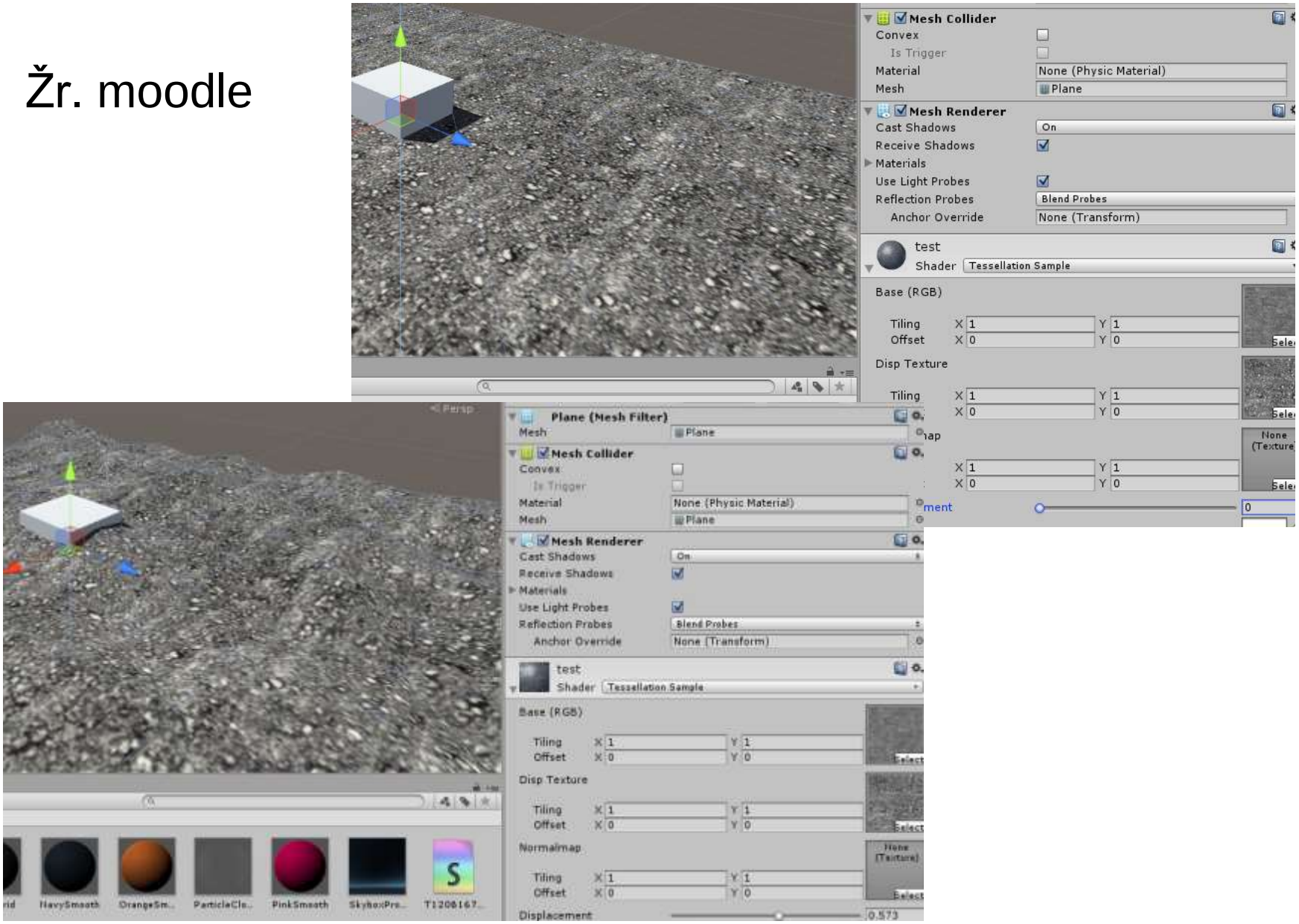
Displacement map for Grand Canyon



Unity3D pavyzdys

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Žr. moodle



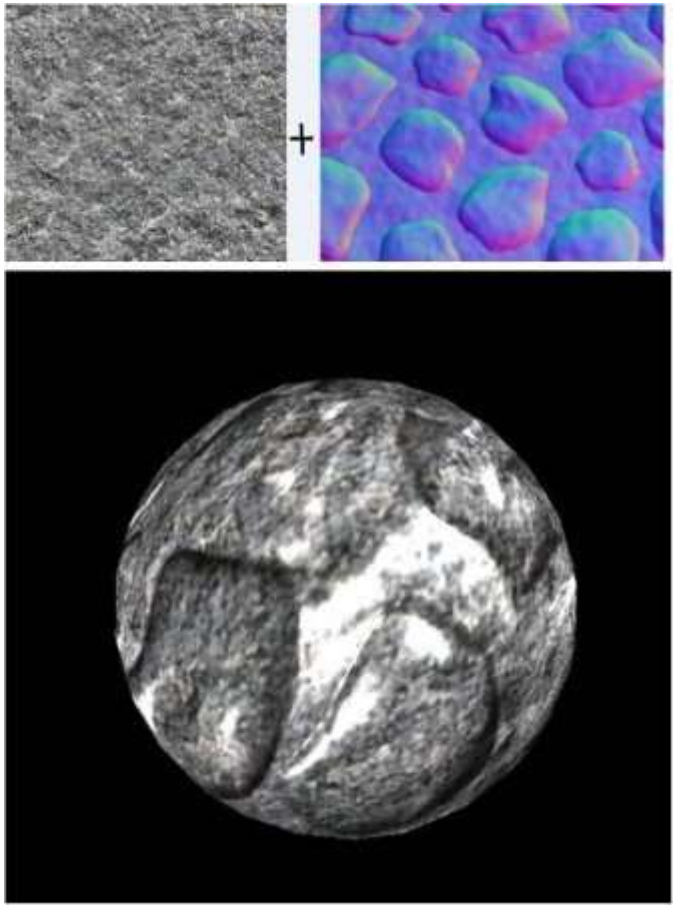
Normalių žemėlapis

- Dažniausiai naudojamas metodas
- Dažnai vadinamas “DOT3” metodu
- Tokiam žemėlapyje saugoma ne aukščiai ar atskaitos, bet pačios paviršiaus normalės
  - Kiekviename tekselyje yra 3 vertės: (x, y, z)
    - [-1..+1] yra suženklintą į [0..255] kiekvienai koordinatei
- Norint suskaičiuoti normalių žemėlapį reikia dviejų tekstūrų: vienos spalvų žemėlapiui (pvz., akmens ar plytos raštas) ir normalių žemėlapio kuriame būtų sužymėtos visų paviršiaus normalių kryptys.
- Vietoje šviesos skaičiavimo naudojant viršūnėlių normales, šviesa skaičiuojama naudojant normales saugomas normalių žemėlapyje.

Normalių žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Pavyzdyje per šeiderį apjungta akmens tekstūra ir normalių žemėlapis ant sferos lygiu paviršiumi.
- Kad tai yra optinė apgaulė išduoda sferos kraštai, nes iš tiesų objektas nėra fiziškai iškraipytas.



Normalių žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Šviesų šaltinių vietos yra transformuojamos į viršūnių tangentinę erdvę kiekvienoje viršūnėje
  - Panašiai kaip ir reljefinio žemėlapio atveju, bet nėra galutinės projekcijos į st plokštumą
- Šie šviesos šaltinio vektoriai yra interpoliuojami per paviršių
  - Spalvos ir gylio vertės
- Gaunam šviesos šaltinio vektorių kiekvienam pikselyje (iš minėtų etapų) ir normalių vektorių kiekvienam pikselyje (iš normalių žemėlapio)
  - Skaičiuojam (L•N) kiekvienam pikseliui

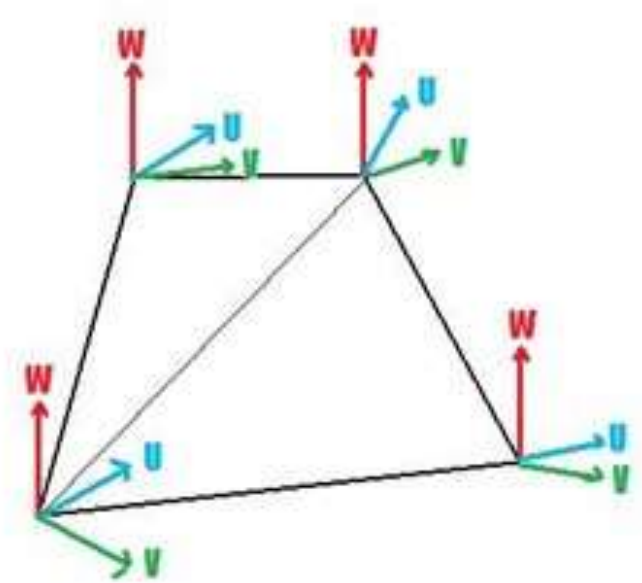
Normalių žemėlapis

- Dažnai daroma, kad šviesos šaltinio vektorius yra viršūnėlės „spalva“.
- Toliau naudojama įprastinė spalvos interpoliacija, kuria šviesos šaltinio vektorius paskleidžiamas per pikselius
  - Šviesos vektoriai ir spalvos sudaryti iš 3 komponentų .
- Panaudojama speciali tekstūros apjungimo funkcija, kurios metu paviršiaus interpoliuotos „spalvos“ (šviesos vektoriai) apjungiami su tekstūros žemėlapio vertėmis (normalėmis).
  - Ta funkcija vadinasi DOT3 ir ją palaiko visas HW palaikantis DX.

Normalių žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Kas yra tangentinė koordinačių sistema?
- Kadangi šviesos vektorius yra manipuliuojamas objekto arba pasaulio erdvėje, reikia tą vektorių transformuoti į tą pačią ervę kaip ir normalės, normalių žemėlapyje.
- Iš modelio failo paimami tangentai (stačiojo trikampio smailiojo kampo Δ yra statinio, esančio prieš šį kampą, ir kito statinio santykis) ir jie perduodami šeideriui.
- Šeideris pagal šią info suskaičiuoja matricą, kuri toliau naudojama šviesų skaičiavimui tangentinėje erdvėje.
- Tam reikia įjungti specialų požymi VS Content processor medyje: Generate tangent: TRUE



Build Action	Compile
Content Importer	X File - XNA Framework
Content Processor	Model - XNA Frame
Color Key Color	255; 0; 255; 255
Color Key Enable	True
Default Effect	BasicEffect
Generate Mipmap	True
Generate Tangent	True
Premultiply Textu	True
Premultiply Verte	True
Resize Textures to	False

Normalių žemėlapis

- Kaip pridėti tekstūras?
- Tekstūra iš principo gali būti bet koks suderinamas grafinis failas: .jpg, .bmp, .png,
- Norint pasigaminti paprastą normalių žemėlapį reikia dviejų tekstūrų (pačių tekstūrų (spalvų) ir normalių žemėlapių).
- HLSL tai daroma naudojant tekstūrų skaitytuvus: Texture sampler
- Jame saugomi įvairūs parametrai, pvz., kaip filstruosite tekstūrą (tolimesniuose pvz. bus naudojamas trilinear filtravimas), kaip bus manipuliuojama su tekstūrų žemėlapio U,V koordinatėmis (pavyzdžiui padaromas tekstūros atspindys) ir kt.
- Tam sukuriamas atitinkamas kintamasis (sampler).

```
texture2D ColorMap;
sampler2D ColorMapSampler = sampler_state
{
    Texture = <ColorMap>;
    MinFilter = linear;
    MagFilter = linear;
    MipFilter = linear;
};
```

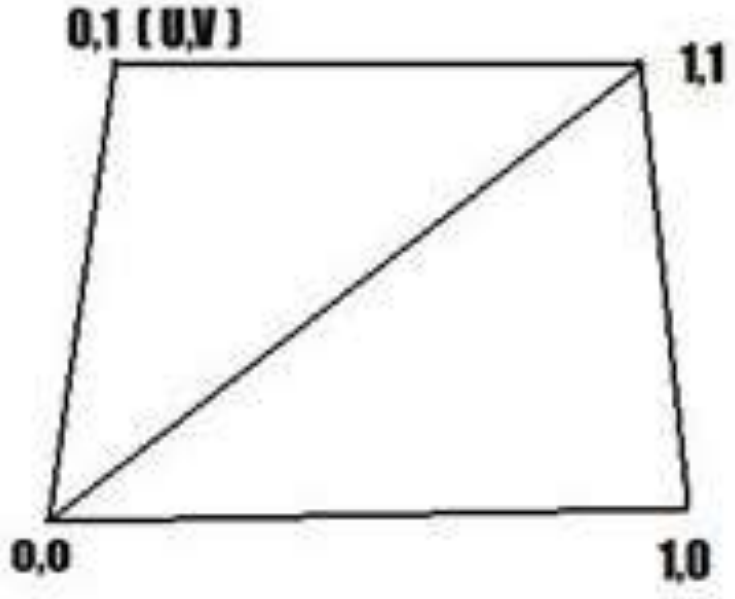
Normalių žemėlapis

- Kadangi naudojam pikselių šeiderį „pririšti“ tekstūrą prie objekto, galime sukurti vektorių kuriame saugosime spalvos informaciją. Spalva gali būti 3 (RGB) arba 4 (RGBA) kanalų.
  - `float4 Color;`
- `Color` kintamajame reikia nustatyti spalvą lygią spalvai tekstūroje, koordinatėje UV.
- HLSL tai daroma naudojant funkciją `tex2D( s, t );` kur `s` yra sampleris (sampler), `t` yra tekstūros koordinatė (pikselio kurį skaičiuojame).
  - `float4 color = tex2D(ColorMapSampler, input.TexCoord);`

Normalių žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Tekstūra apibrėžiama 2D koordinatėmis (U,V), kurios saugomos kiekvienoje 3D modelio viršūnėlėje. Jos naudojamos tekstūros pririšimui prie objekto ir yra nuo 0.0 iki 1.0 **(žr. paskaitą apie „žemėlpaius“)**



- Dėka tekstūros koordinačių, tekstūras galima priskirti skirtingoms modelio vietoms (pvz., akies rainelę prie akių, lūpas prie burnos ir t.t.).

Normalių žemėlapis

- Apšvietimui specular šviesa naudojama kaip ir ankstesnių lab. darbų atveju, tačiau normalės naudojamos iš tekstūros (bet ne iš viršūnėlių).
- Antra – naudojame tangentinę erdvę (bet ne objekto erdvę), o normalės naudojamos šviesų skaičiavime imamos iš normalių žemėlapio.
- Apibrėžiam tekstūras ir tekstūros samplerį:

```
texture2D ColorMap;
sampler2D ColorMapSampler = sampler_state
{
    Texture = <ColorMap>;
    MinFilter = linear;
    MagFilter = linear;
    MipFilter = linear;
};

texture2D NormalMap;
sampler2D NormalMapSampler = sampler_state
{
    Texture = <NormalMap>;
    MinFilter = linear;
    MagFilter = linear;
    MipFilter = linear;
};
```

Normalių žemėlapis

- Prie viršūnių šeiderio įėjimo struktūros pridedam tekstūros koordinates. Nepamirštam normalės, binormalės ir tangento (generuojama modelyje).

```
struct VertexShaderInput
{
    float4 Position : POSITION0;
    float2 TexCoord : TEXCOORD0;
    float3 Normal : NORMAL0;
    float3 Binormal : BINORMAL0;
    float3 Tangent : TANGENT0;
};
```

- Tekstūros koordinatė naudojama pikselių šeiderio įėjime, todėl pridedama viršūnių šeiderio išėjimo struktūroje.

```
struct VertexShaderOutput
{
    float4 Position : POSITION0;
    float2 TexCoord : TEXCOORD0;
    float3 View : TEXCOORD1;
    float3x3 WorldToTangentSpace : TEXCOORD2;
};
```

Normalių žemėlapis

- Viršūnių šeideryje reikia užpildyti WorldToTangentSpace parametrą, siekiant transformuoti normales ir šviesų skaičiavimus į atitinkamą erdvę.
- Kad gauti WorldToTangentSpace dauginame kiekvieną komponentą su pasaulio (World) matrica.

```
VertexShaderOutput VertexShaderFunction(VertexShaderInput input,float3 Normal : NORMAL)
{
    VertexShaderOutput output;

    float4 worldPosition = mul(input.Position, World);
    float4 viewPosition = mul(worldPosition, View);
    output.Position = mul(viewPosition, Projection);
    output.TexCoord = input.TexCoord;

    output.WorldToTangentSpace[0] = mul(normalize(input.Tangent), World);
    output.WorldToTangentSpace[1] = mul(normalize(input.Binormal), World);
    output.WorldToTangentSpace[2] = mul(normalize(input.Normal), World);

    output.View = normalize(float4(EyePosition,1.0) - worldPosition);

    return output;
}
```

Normalių žemėlapis

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Sutransformavus poziciją.
- Toliau kuriam 3x3 matricą, WorldToTangentSpace, kuri naudojam pasaulio erdvės vertimui į tangentinę erdvę.
- Šviesos skaičiavimui gauname transformuotą poziciją ir šviesų bei kameros vektorius paremtus tangentinės erdvės matrica.
- Turint šiuos dydžius pereiname prie pikselių šeiderio, kur reikia „nuskaityti“ spalvą ir spalvų žemėlapio bei normalę iš normalių žemėlapio.
- Atlikus šias operacijas galima skaičiuoti fono, difuzinę ir atspindžio šviesas pagal normales iš normalių žemėlapio.

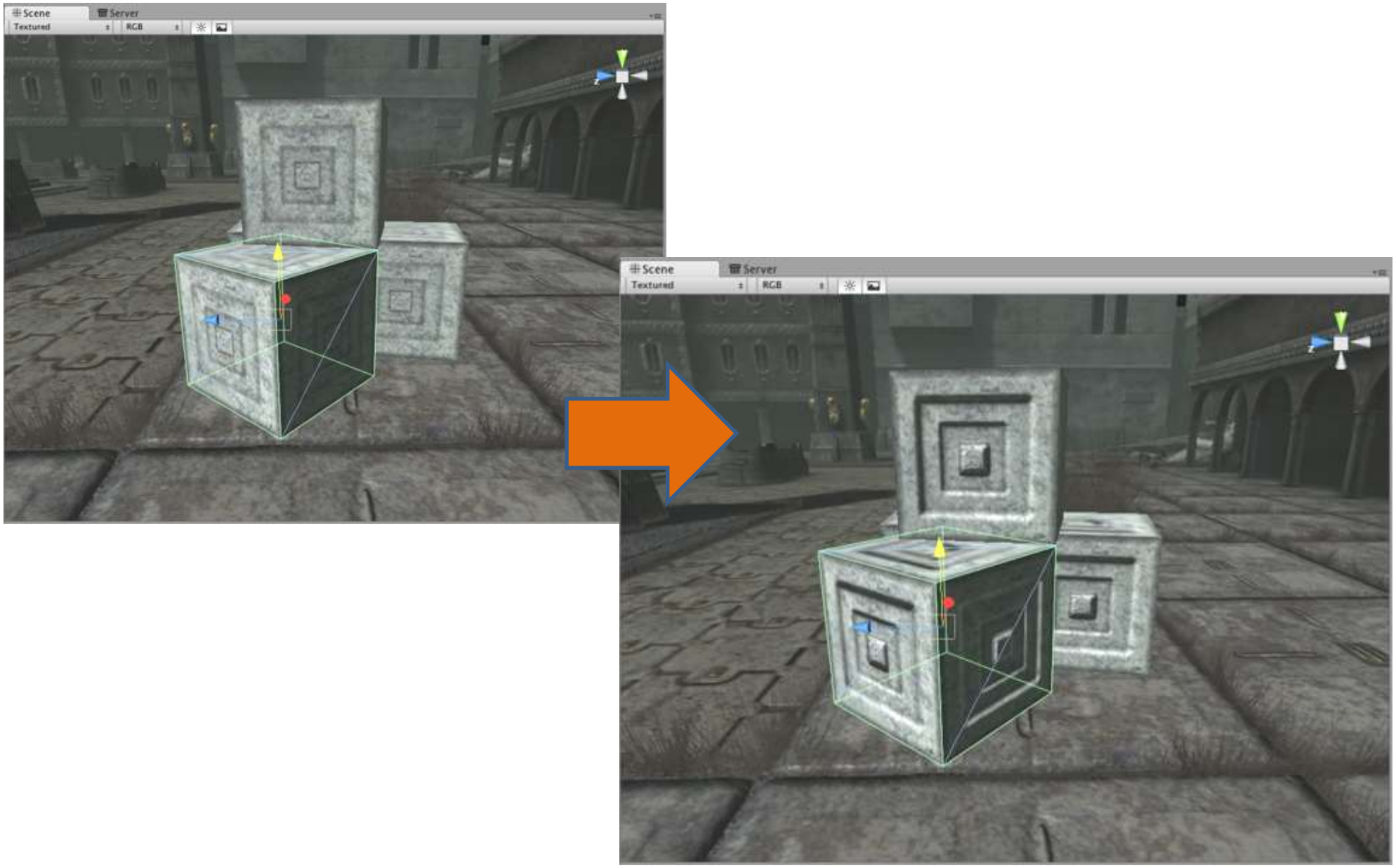
Normalių žemėlapis

- Pikselių šeiderio kodas.
- Pakeičiam kaip kuriamos normalės ir pridedam spalvą iš tekstūros naudojant tex2D(s,t) funkciją, kuri gražina spalvą esančią *t* pozicijoje, tekstūroje *s*.
- Tas pats daroma su normale, tik nurodoma kad intervalas yra 0-1, o ne –1 +1.
- Pabaigoje pridedama spalvos vertė prie foninės, difuzinės ir atspindžio šviesų.

```
float4 PixelShaderFunction(VertexShaderOutput input) : COLOR0
{
    float4 color = tex2D(ColorMapSampler, input.TexCoord);

    float3 normalMap = 2.0 *(tex2D(NormalMapSampler, input.TexCoord)) - 1.0;
    normalMap = normalize(mul(normalMap, input.WorldToTangentSpace));
    float4 normal = float4(normalMap,1.0);

    float4 diffuse = saturate(dot(-LightDirection,normal));
    float4 reflect = normalize(2*diffuse*normal-float4(LightDirection,1.0));
    float4 specular = pow(saturate(dot(reflect,input.View)),32);

    return  color * AmbientColor * AmbientIntensity +
            color * DiffuseIntensity * DiffuseColor * diffuse +
            color * SpecularColor*specular;
}
```

Unity3D implementacija

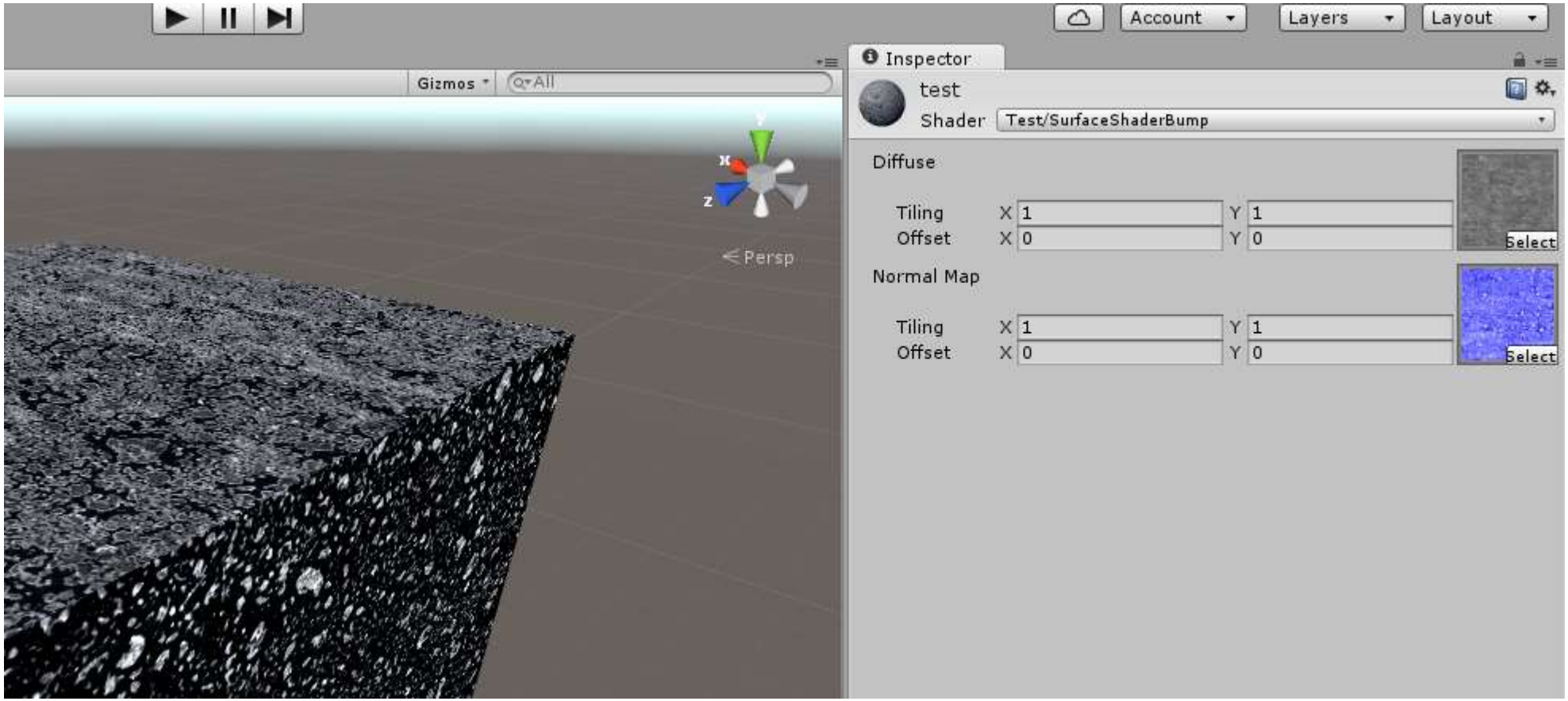
MULTIMEDIJOS INŽINERIJOS KATEDRA



Simple iškilumų šeideris

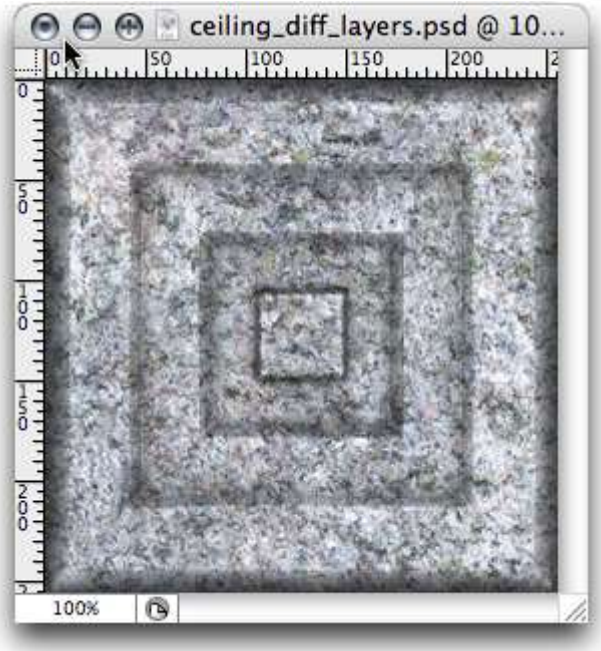
MULTIMEDIJOS INŽINERIJOS KATEDRA

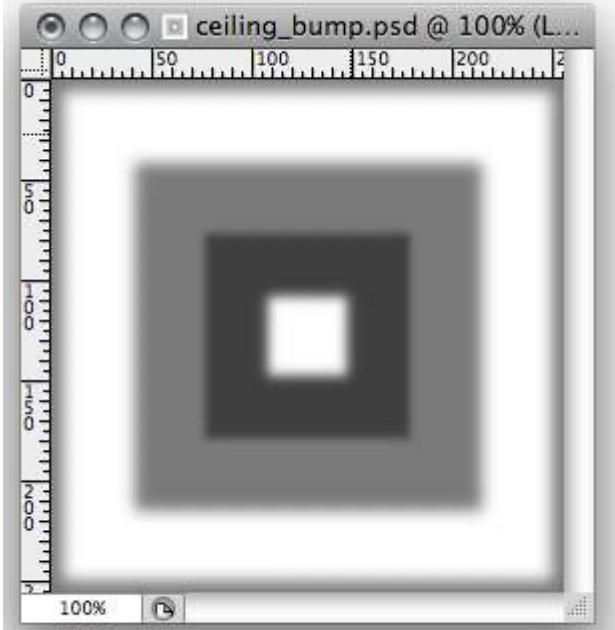
- Žr. moodle

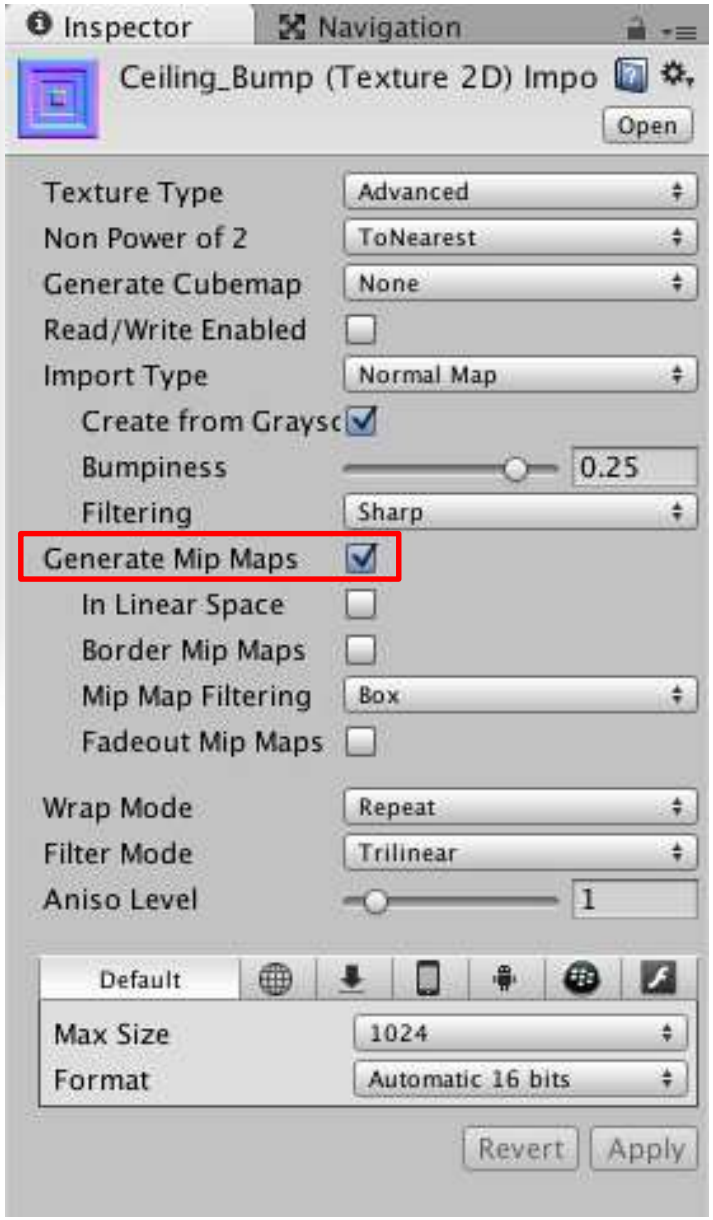


DIY. Pasileidžiam „Photoshop‘ą“

MULTIMEDIJOS INŽINERIJOS KATEDRA

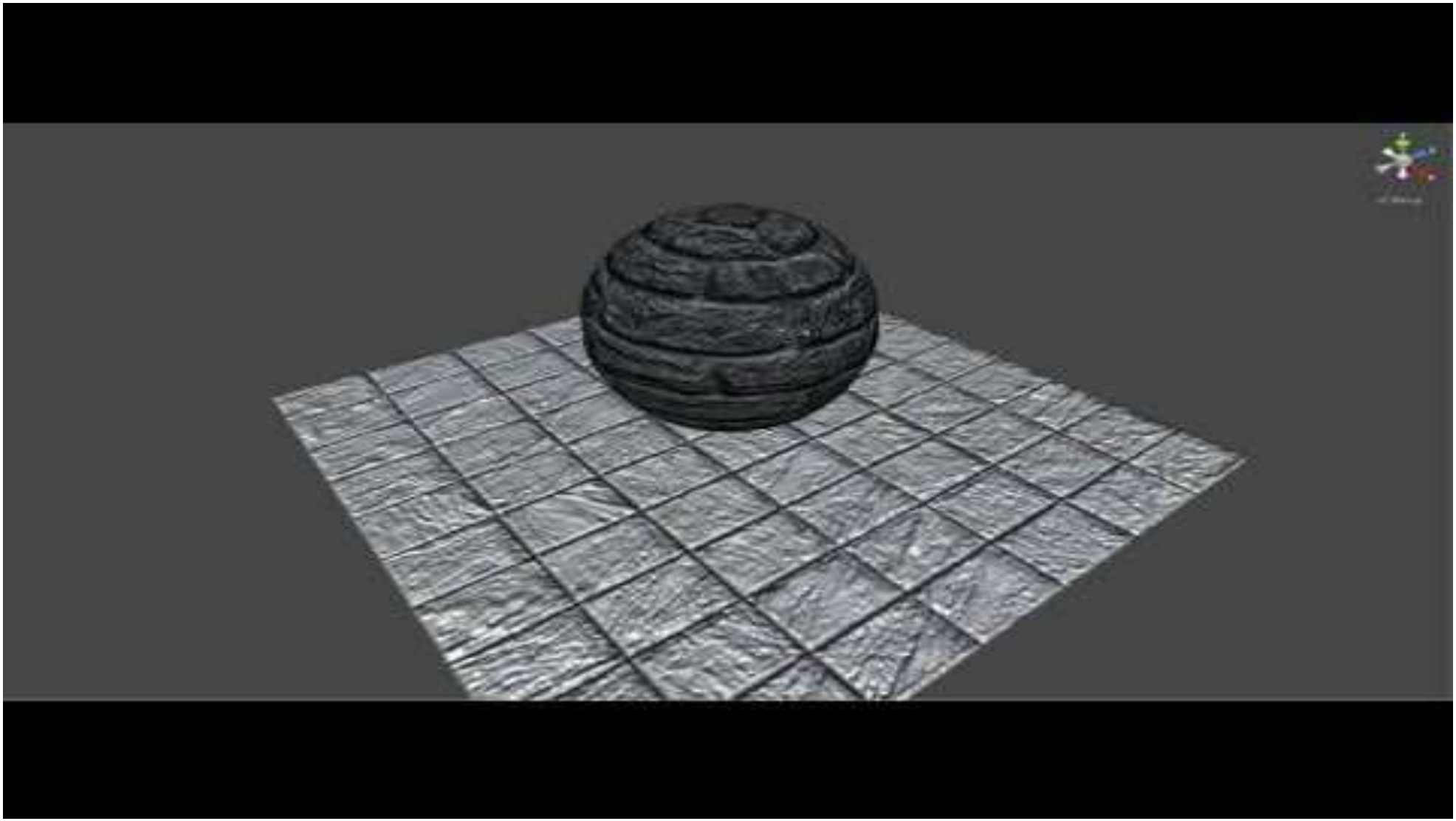






Tutorial‘as nelankantiems paskaitų

MULTIMEDIJOS INŽINERIJOS KATEDRA



https://www.youtube.com/watch?v=3CNeusagQmg