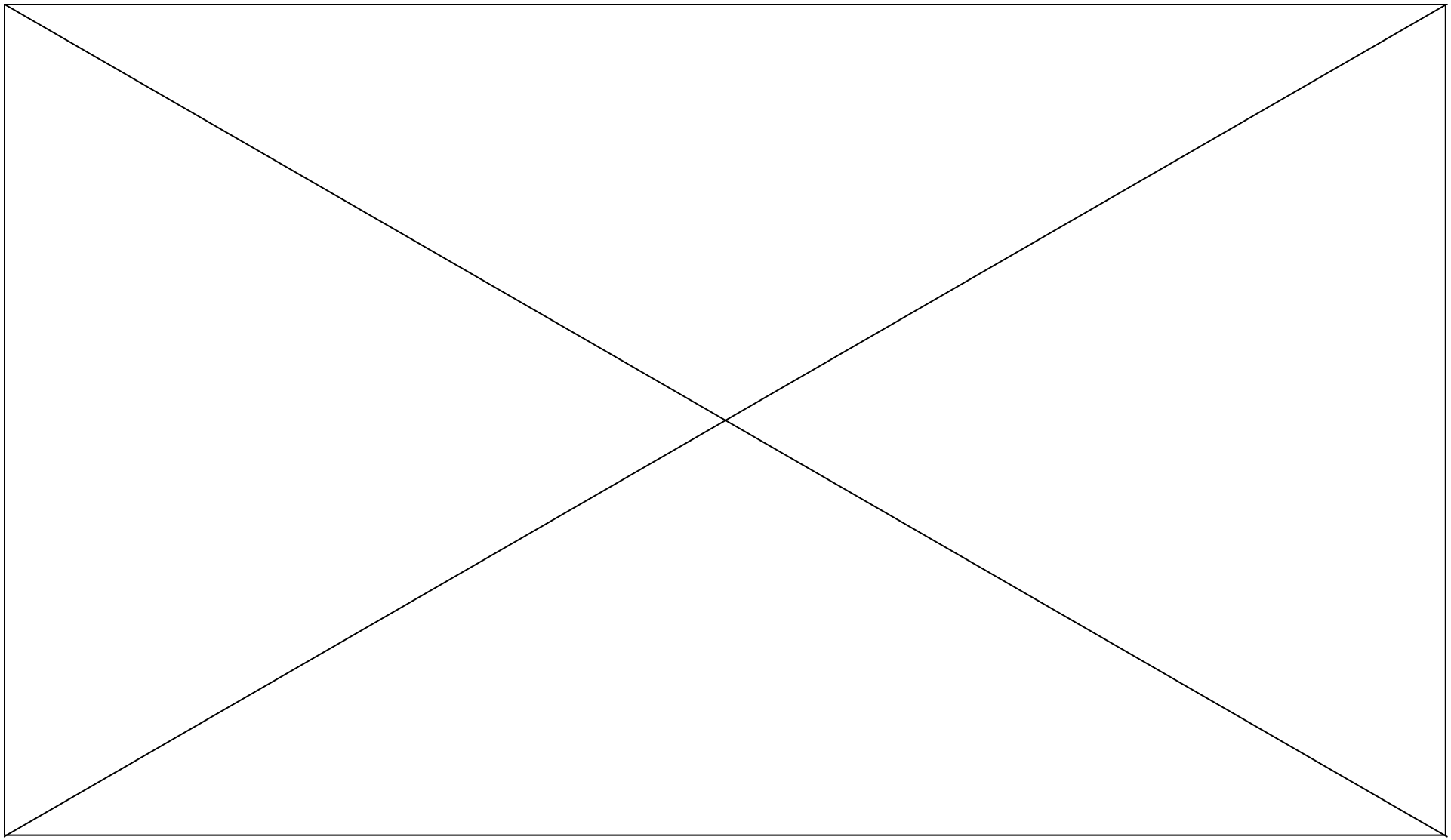
Specialieji efektai 1

T120B167 Žaidimų grafinių specialiųjų efektų kūrimas ir programavimas

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

Paskaitos tema

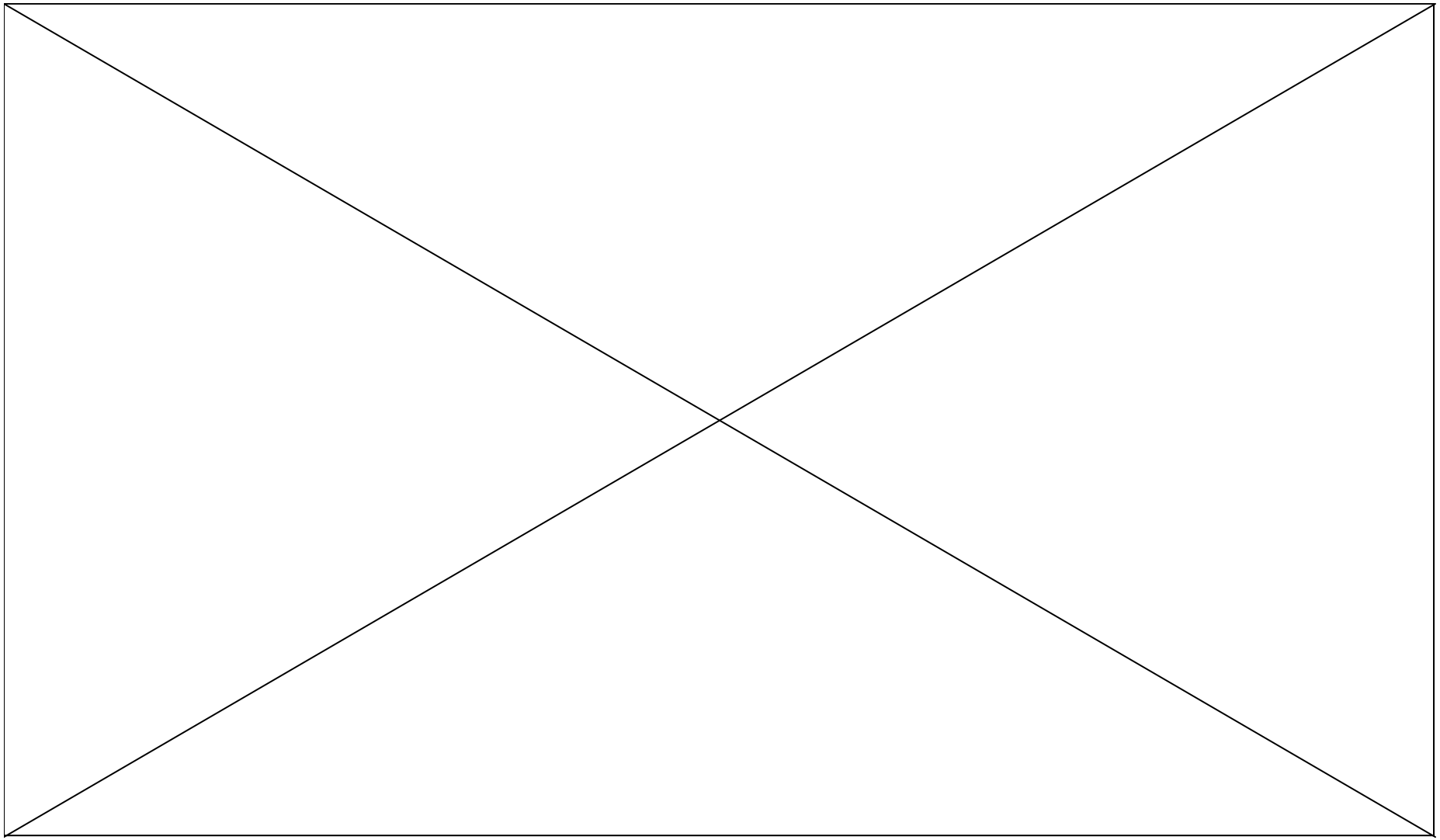
MULTIMEDIJOS INŽINERIJOS KATEDRA



http://www.youtube.com/watch?v=eksLqBKh5OY

Už brangiai

MULTIMEDIJOS INŽINERIJOS KATEDRA



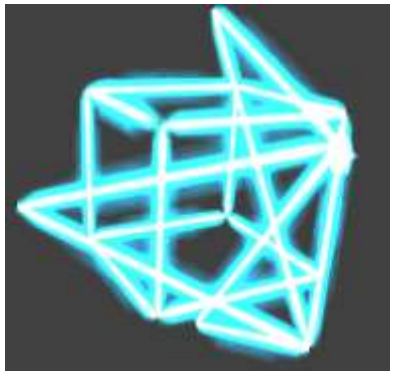
http://www.youtube.com/watch?v=IJ9OFwFxxt4

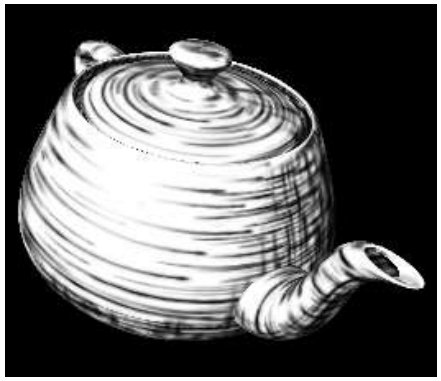
Powered by Shader

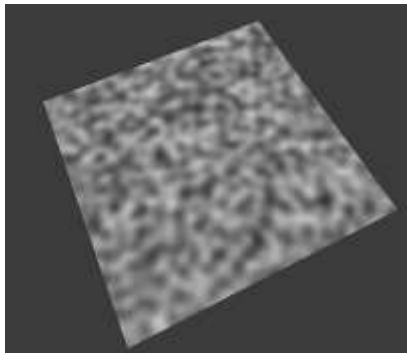
MULTIMEDIJOS INŽINERIJOS KATEDRA

- Per-pixel lighting
- Motion blur
- Volume / Height fog
- Volume lines
- Depth of field
- Fur shading
- Reflection / Refraction
- NPR
- Shadow
- Linear algebra operators
- Perlin noise
- Quaternion
- Sparse matrix solvers
- Skin bone deformation
- Normal map
- Displacement map
- Particle shader

Procedural Morphing
Water Simulation

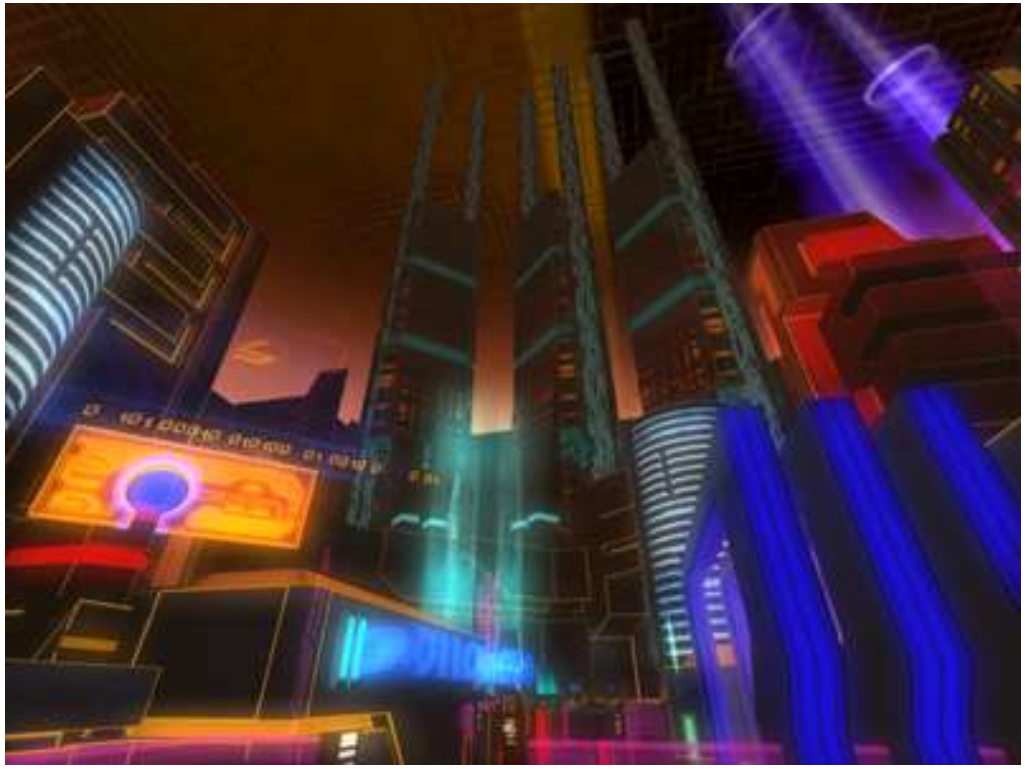






Spindėjimo (Glow) efektas

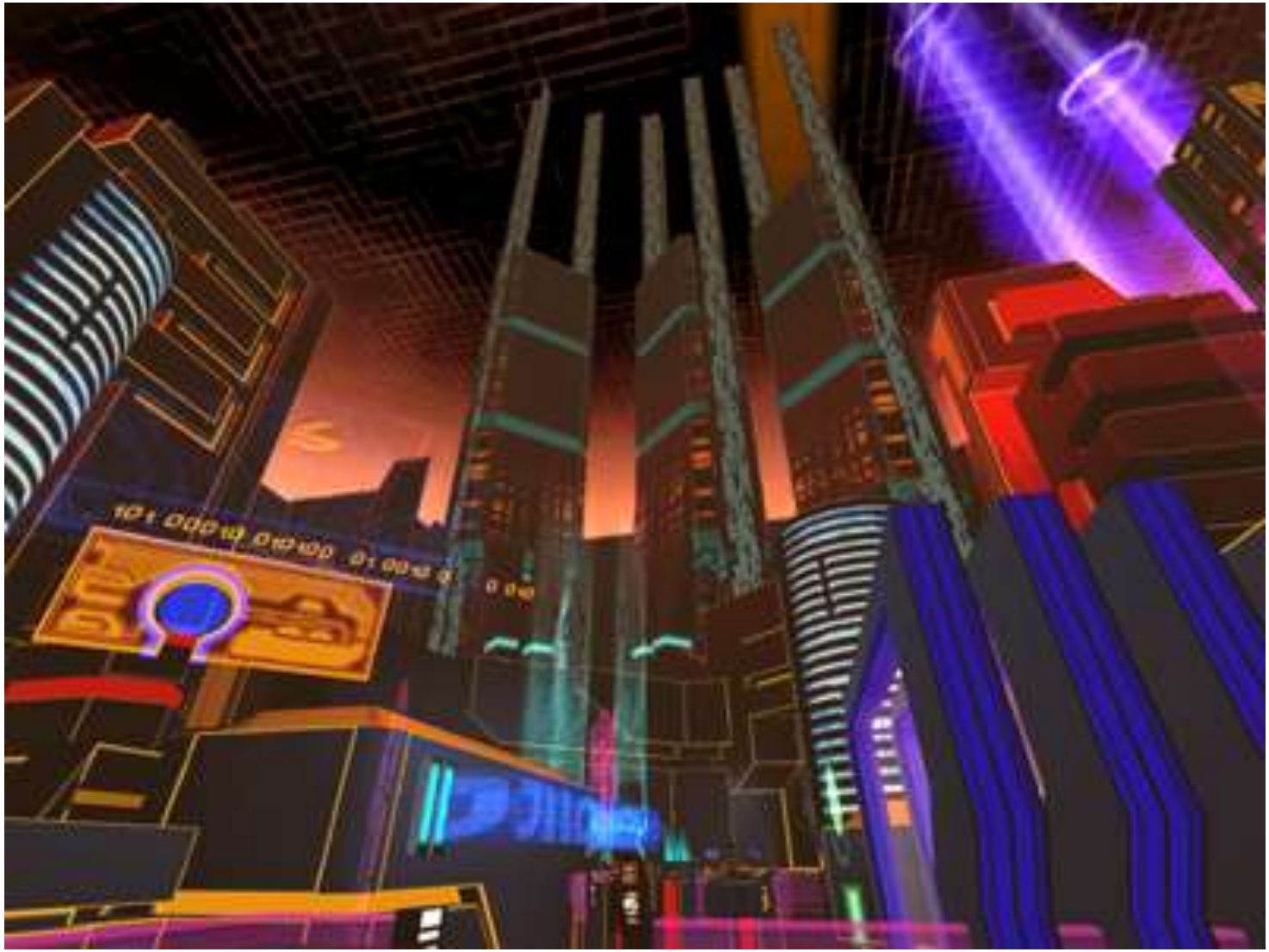
- Spindesiai kompleksinėse scenose
- Greitai veikia
- Naudojamas HDR efektui
- Gali būti keleto spalvų (multicolor)
- Lengva programiškai valdyti



“Tron2.0”

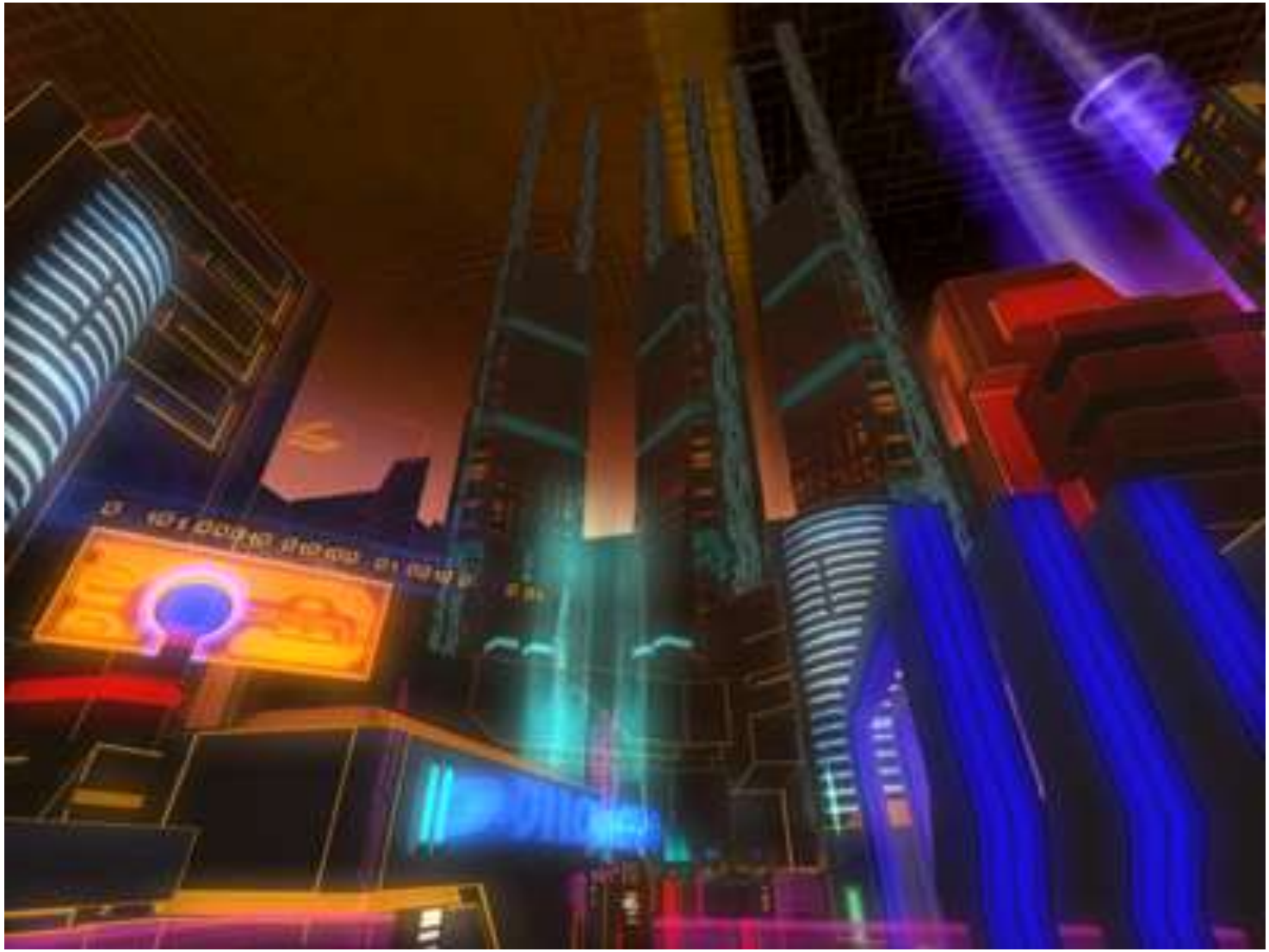
Be Glow

“Tron2.0” courtesy of Monolith & Disney Interactive



Su Glow

“Tron2.0” courtesy of Monolith & Disney Interactive



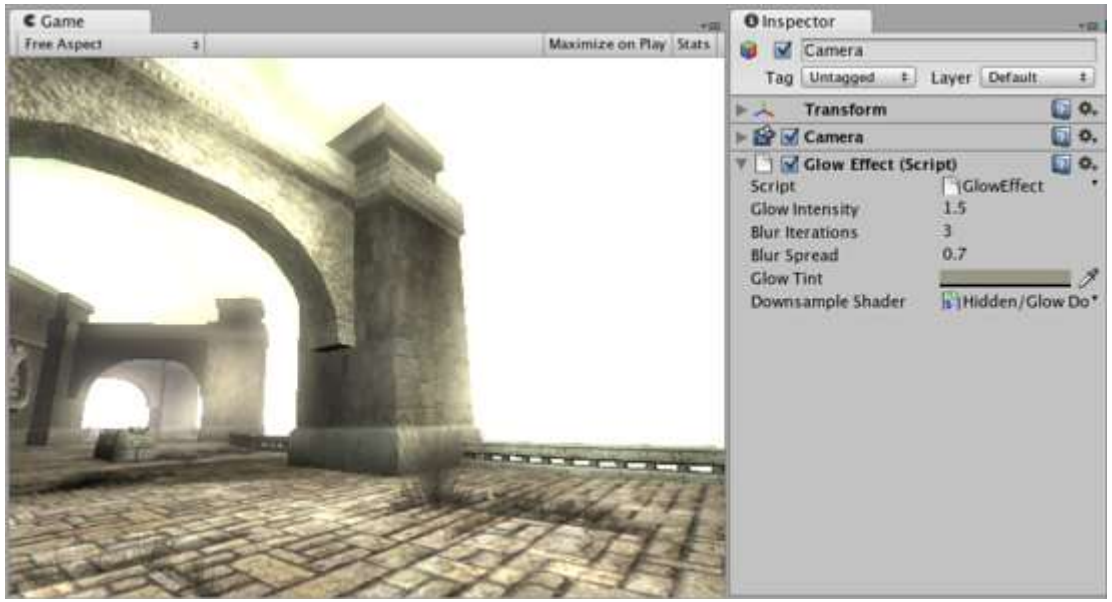
Kur naudoti glow?

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Kur norime „paryškinti“ scenos objektus
  - Subtilus spindesis gali išryškinti dominatį objektą
  - Atspindžiai: vanduo, spindintys objektai
  - Atmosfera: neono šviesos, dūmų sukuriamas šviesos iškraipymas
- Nereikia HDR asetų ar sudėtingų tekstūrų!

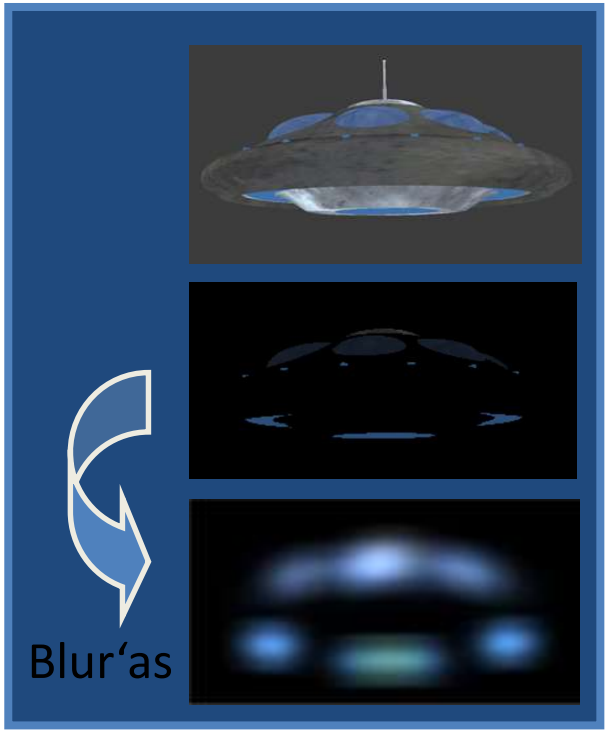
- **BANDOME**

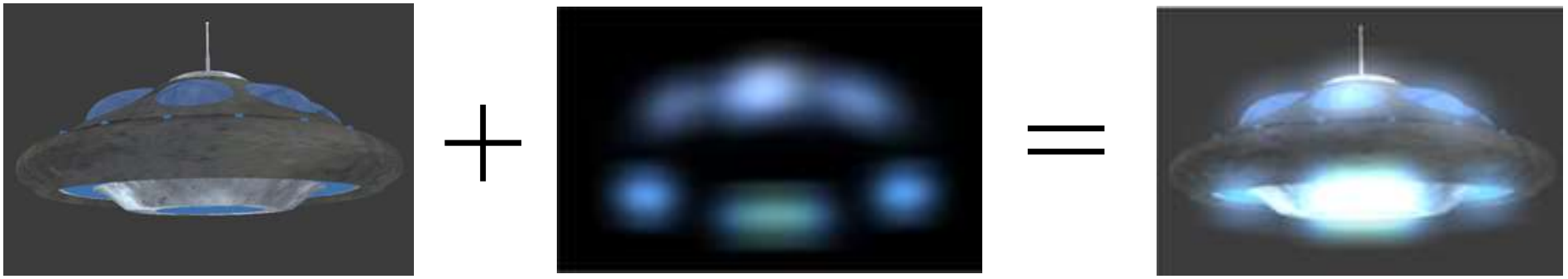
(Effects kolekcija)



Kaip tai veikia

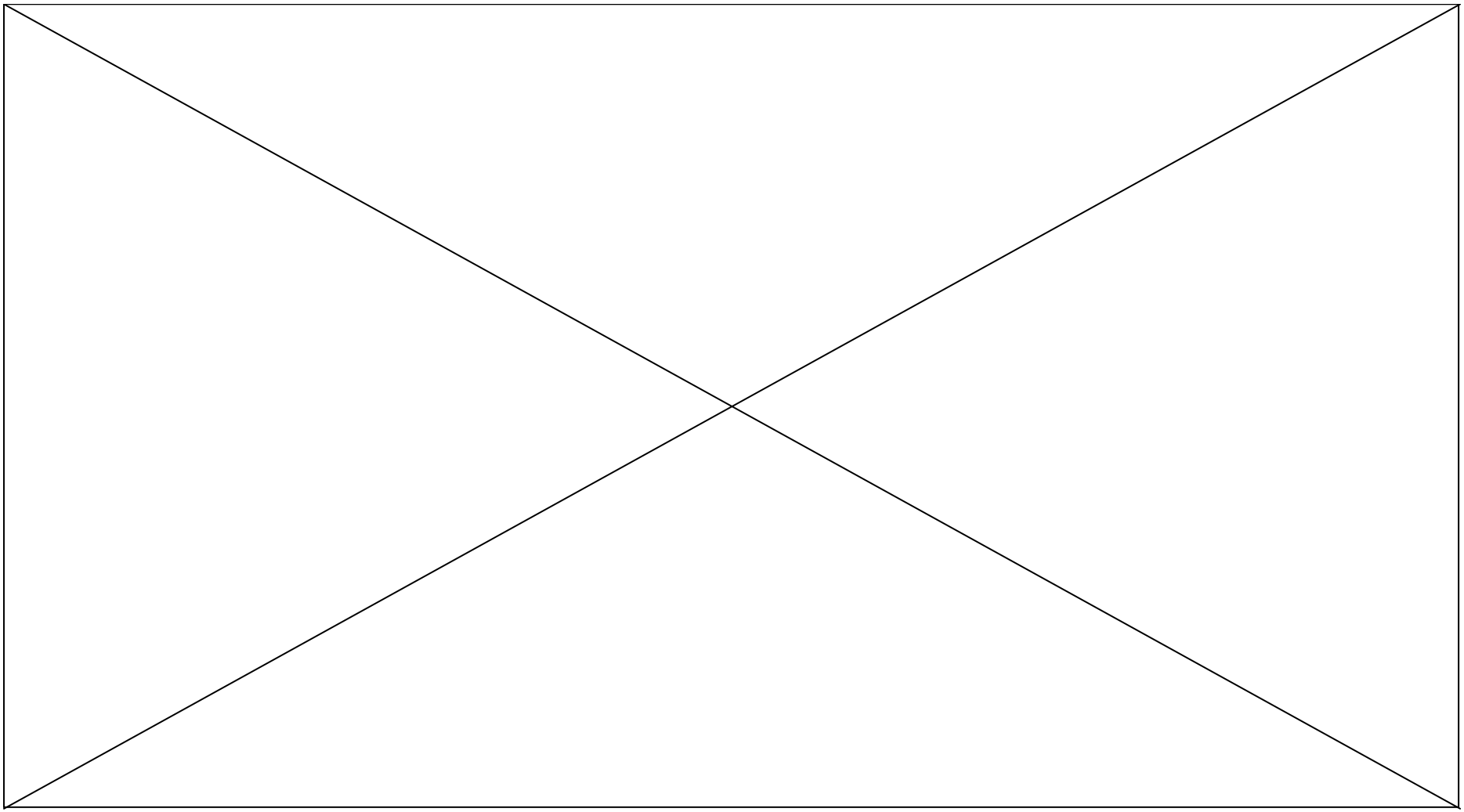
- Turim kažkokį modelį
  - Renderiuojam į backbuffer
- Renderiuojam dalis kurios yra spindesio šaltinis
  - Renderiuojam į offscreen tekstūrą
- Blur‘inam tekstūrą
- Pridedam tą tekstūrą prie scenos:





Kaip tai veikia

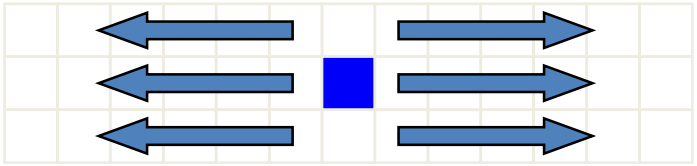
MULTIMEDIJOS INŽINERIJOS KATEDRA



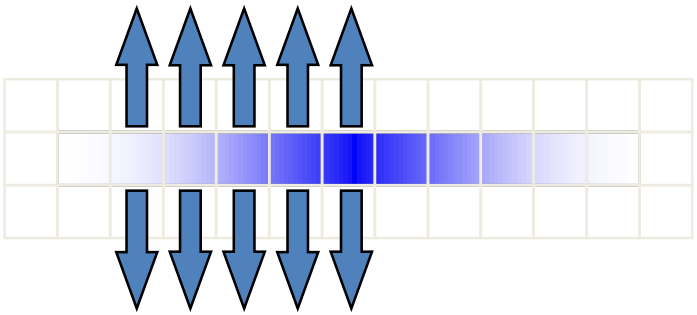
http://www.youtube.com/watch?v=YhMYJXzqX7U

Efektyvus blur

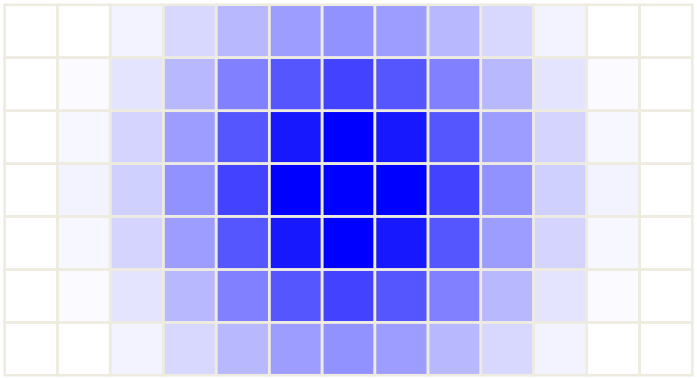
- Variacijos



Blur‘inam šaltinį horizontaliai



Blur‘inam šaltinį vertikaliai



Rezultatas

Šaltinis

Horizontalus Blur

Vertikalus Blur

Reziumė

- Taupome CPU apkrovą!
  - Renderiuojam į GPU video atminties tekstūras
- Minimizuojami renderiavimo ciklo pokyčiai
- „Fill rate“ ribos
  - Minimizuojama „fill“ kaina
  - Žemos rezoliucijos glow apdorojimas
  - Galima spindesio tekstūrą išdidinti taip padengiant visą ekraną
- Visą sceną galima sublurinti iškarto
  - Galima suskaidyti jei norima detalizuoti (skirtingais efektais tam tikras dalis ir kt.)

Specifiniai Glow šaltiniai

MULTIMEDIJOS INŽINERIJOS KATEDRA

Model with diffuse texture, t0.rgba

t0.a * t0.rgb = glow source

- Pradedam su standartiniu modeliu
- Nurodom kokios vietos spindės
  - texture Alpha * texture RGB = glow source color
  - Arba sukuriam atskirą glow geometriją

Renderiuojam Glow šaltinį į tekstūrą

Tikslas

- Tekstūros renderis gali būti žemesnės rezoliucijos nei visas galutinis renderiuojamas vaizdas
  - Kuo žemesnė rezoliucija – tuo didesnis aliasingas
    - Gali atsirasti mirgėjimai

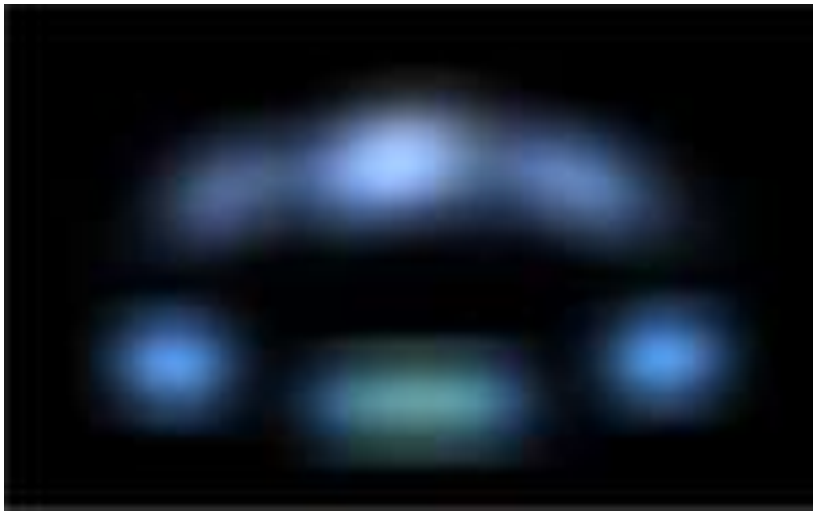
Žemos rezoliucijos tekstūra

- Žema rezoliucija galima pagerinti našumą
- Kiekvienas glow tekselis gali padengti 2, 3, 4 etc. Ekrano pikselių
  - Pvz: Blurinam 40x40 tekselių plotą
  - Gaunam 160x160 pikselių glow plotą

Blurinam siekiant sukurti Glow tekstūrą

MULTIMEDIJOS INŽINERIJOS KATEDRA

Renderiuojama tekstūra



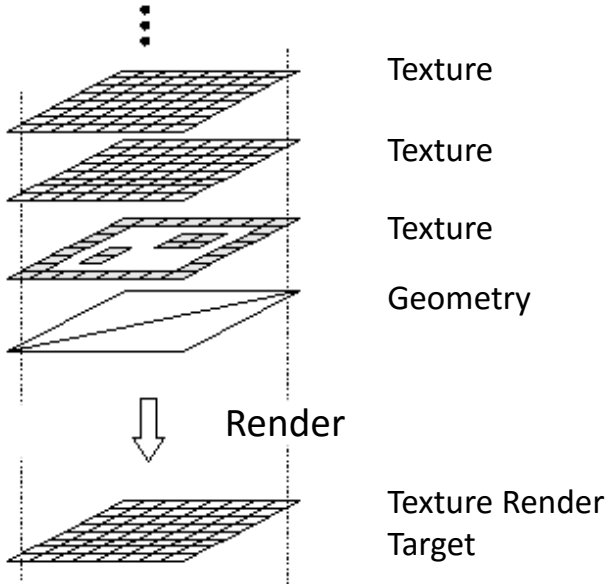
Mūsų tikslas

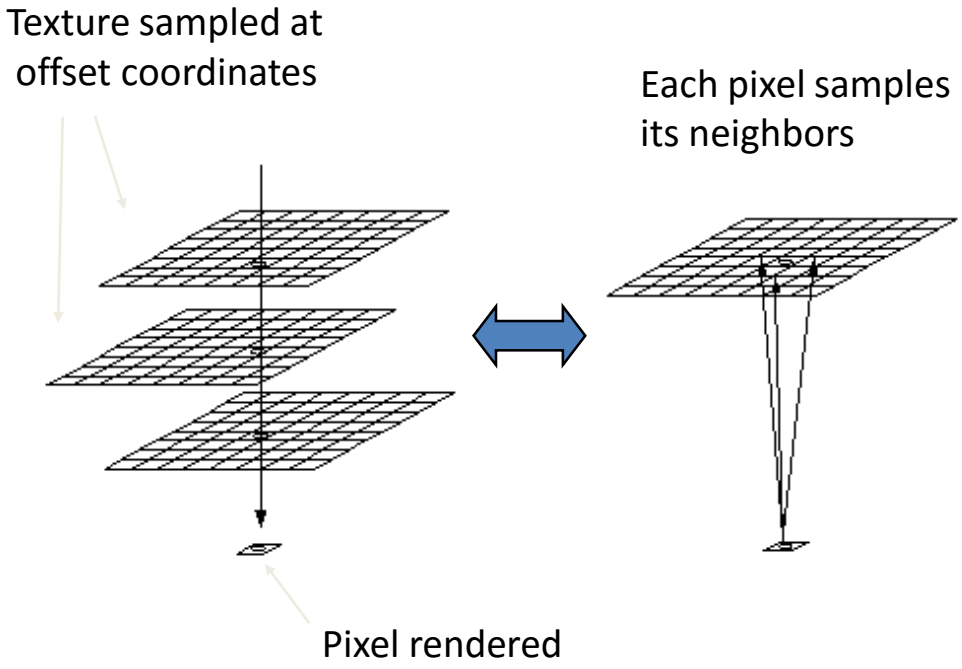
- GPU render-to-texture
- Pixel samples pagal „kaimynus“

Kaip blurinti?

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Naudojant atskaitą „pagal kaimyną“ (Neighbor sampling) su vertex ir pixel šeideriais
- Paprasta geometrija su keliomis tekstūros koordinatėmis
- Kiekviena tekstūros koordinatė atskaitoma gretimą pikselį





- Arba rankytėmis „photoshope“...

BLUR Per HLSL?

- Reikia paimti greta esančių pikselių vidurkį ir suskaičiuoti spalvą, kokia turi būti atvaizduota. Paprasčiausiam blur efektui įvedama atstumo (distance) kintamasis, kuris naudojamas tekstūros koordinatės pagal kurią nurodome ką blurinti modifikavimui (imamas nuo nurodyto taško pikselis iš viršaus kairės, viršaus dešinės, apačios kairės ir apačios dešinės. Jie sudedami ir padalinama iš 4):

```
Color  = tex2D( ColorMapSampler, float2(Tex.x+BlurDistance, Tex.y+BlurDistance));
```

```
Color += tex2D( ColorMapSampler, float2(Tex.x-BlurDistance, Tex.y-BlurDistance));
```

```
Color += tex2D( ColorMapSampler, float2(Tex.x+BlurDistance, Tex.y-BlurDistance));
```

```
Color += tex2D( ColorMapSampler, float2(Tex.x-BlurDistance, Tex.y+BlurDistance));
```

BLUR Per HLSL?

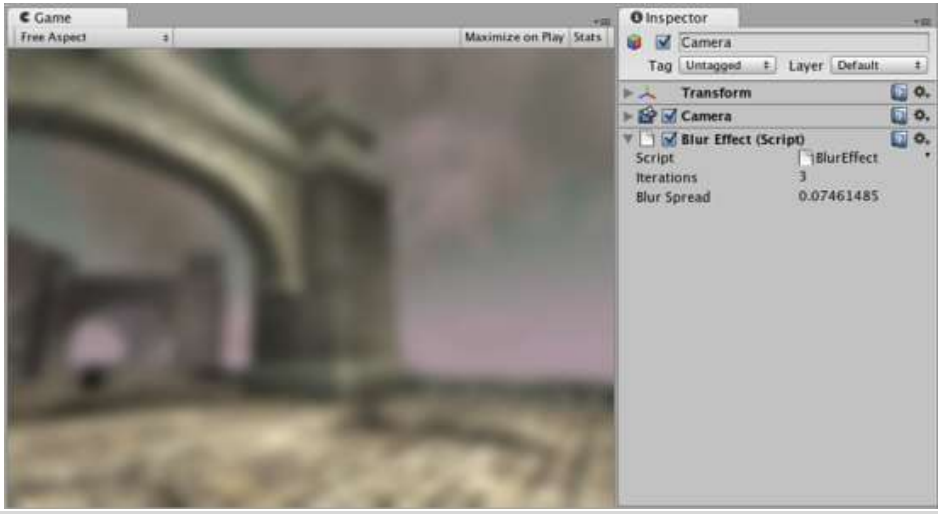
MULTIMEDIJOS INŽINERIJOS KATEDRA

```
// Blur'o kiekis (kaip toli nuo tekselio reikia ieškoti kaimynų?)
float BlurDistance = 0.002f;

// Prikabinam tekstūrą prie objekto
sampler ColorMapSampler : register(s0);

float4 PixelShader(float2 Tex: TEXCOORD0) : COLOR
{
float4 Color;

// Paimamams tekselis iš ColorMapSampler naudojant modifikuotas tekstūros koordinates ir
pridedama prie spalvos

Color  = tex2D( ColorMapSampler, float2(Tex.x+BlurDistance, Tex.y+BlurDistance));
Color += tex2D( ColorMapSampler, float2(Tex.x-BlurDistance, Tex.y-BlurDistance));
Color += tex2D( ColorMapSampler, float2(Tex.x+BlurDistance, Tex.y-BlurDistance));
Color += tex2D( ColorMapSampler, float2(Tex.x-BlurDistance, Tex.y+BlurDistance));

// Dalinam spalvą iš tiek kartų kiek kartų sudėjote, šiuo atveju 4
Color = Color / 4;

// gaunam sublurintą spalvą
return Color;
}
```

Pridedame glow prie scenos

- Pridedame glow naudojant du trikampius padengiančius jūsų vaizdą
- Reguliuojam ryškumą per alpha
- **BANDOME**

(Effects kolekcija)

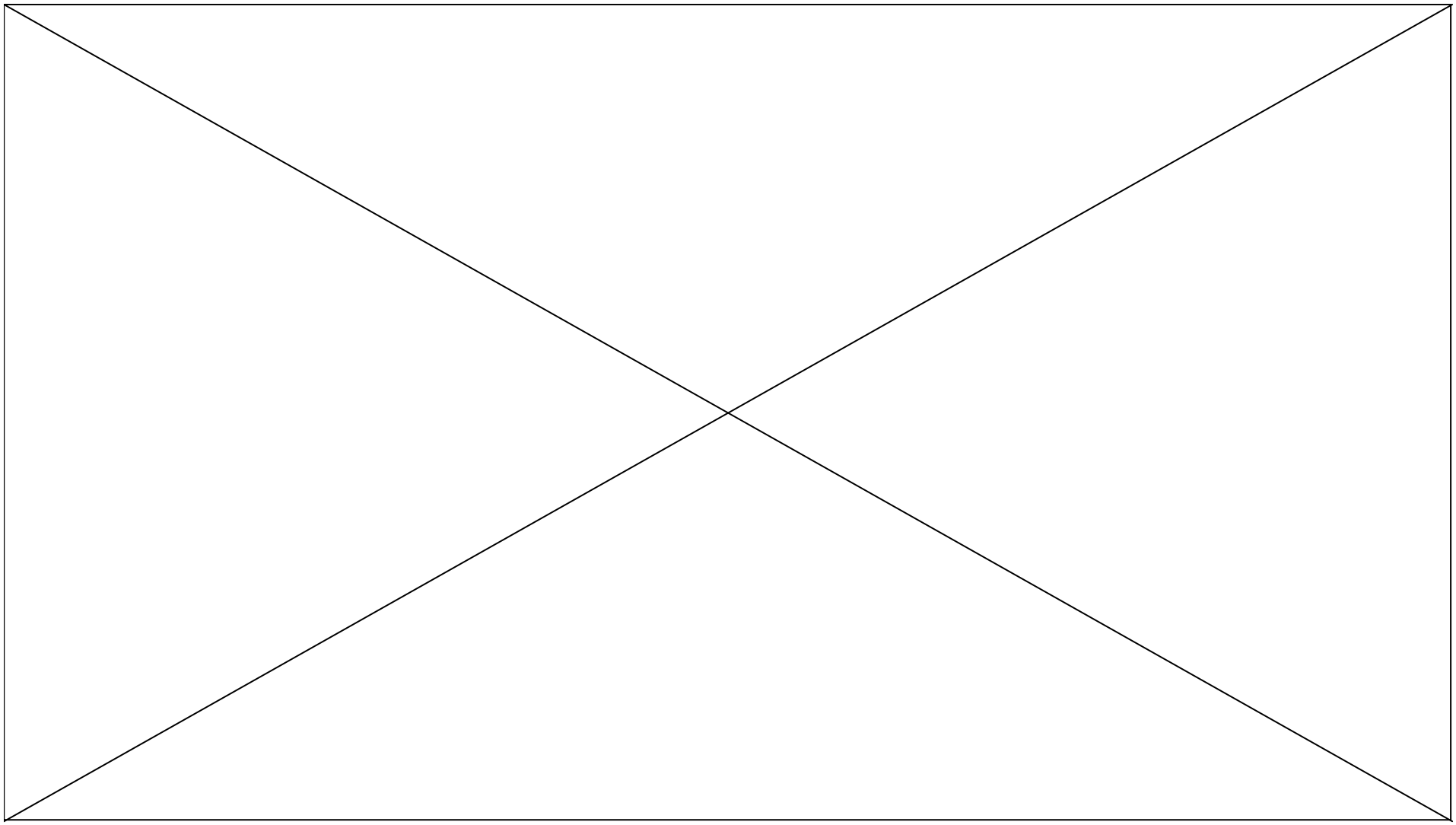


Motion Blur (judesio blur‘as)

- Kas yra motion blur?
  - Greitai judantys objektai atrodo sulieti judėjimo kryptimi
- Kas sukuria motion blur?
  - Realybėje tai yra kameros išlaikymo problema, tiesiog pajudinkite fotoaparatą fotografuodami arba fotografuokite telefonu ir greitai mojuokite ☺
- Kam jo reikia?
  - Pridedam realizmo, „filmo“ vaizdą, ypač populiaru žaidimuose bet ir dėl to, kad:
  - 24fps su motion blur atrodo geriau nei 60fps be ☺

Motion Blur vs Nothing

MULTIMEDIJOS INŽINERIJOS KATEDRA



http://www.youtube.com/watch?v=Wi3tmYZIJ-M

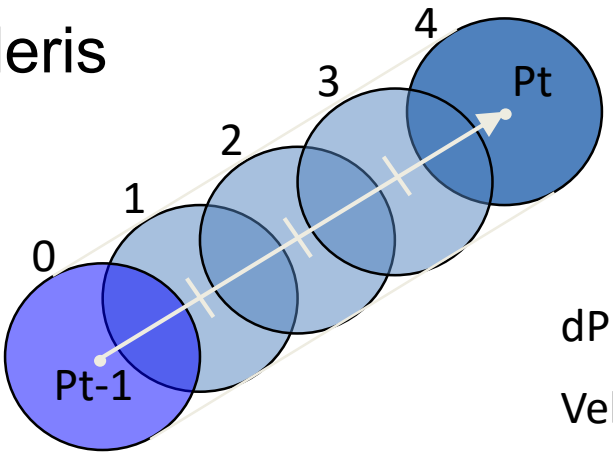
Image Space Motion Blur

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Visiškai „teisingą“ motion blur renderiavimą daryti paprastai yra neefektyvu, nes reikia daryti:
  - Temporal supersampling (accumulation/T-buffer)
  - Distributed ray tracing
- „Uodegos“ paskui judančius objektus tai nėra tikrasis motion blur‘as
- Paprastai visi dirba su Image space (2.5D) motion blur
  - Didelė greitaveika
  - Sublurina paveikslą pagal objekto judėjimo parametrus
  - Išlaiko paviršiaus detalumą
  - Nelabai gerai veikia kai objektai persidengia ☺

Algoritmas

- 1. Renderiuojam sceną į tekstūrą
  - Esamu laiku
- 2. Skaičiuojam kiekvieno pikselio judėjimo greitį
  - Naudojam vertex šeiderį
  - Susiskaičiuom esamą poziciją – buvusią poziciją
- 3. Renderiuojam sublurintą sceną
  - Naudojamas pikselių šeideris



dP = Pt – Pt-1

Velocity = dP / dt

Psample = P + dP * u

Nsamples = 5

Greičių skaičiavimas

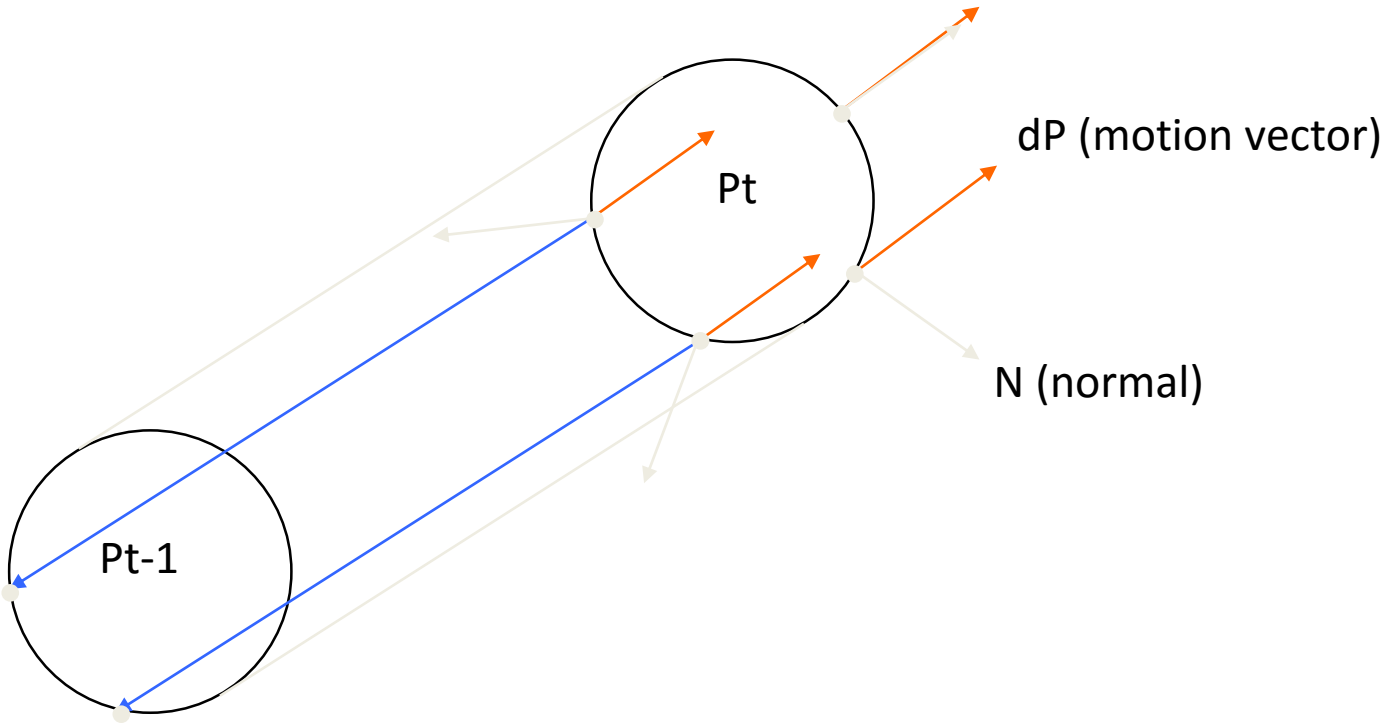
- Reikia žinoti kiekvieno pikselio judėjimo greitį matomoje ekrano erdvėje
- Lengvai skaičiuojama vertex šeideryje
  - Transformuojamas kiekvienas verteksas į lango koordinates pagal esamą ir buvusią transformacijas
  - Deformacijoms reikia padvigubinti visus skaičiavimus
  - Velocity = (current_pos – previous_pos) / dt
- Greitis yra interpoliuojamas per trikampius
- Galima renderiuoti į float/color buferį, ar naudoti tiesiogiai

Greičių skaičiavimas

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Problema: greitis objekto silueto išorėje = 0 (= nėra blur‘o)
- Sprendimas: reikia „ištempti“ geometriją tarp buvusios ir esamos pozicijų
- Reikia palyginti normalės kryptį su judesio vektoriumi per dot funkciją
- Jei normalė „žiūri“ judesio kryptimi, reikia transformuoti viršūnėlę pagal esamą transformaciją, kitu atveju transformuoti pagal buvusią transformaciją

Geomtrijos „ištempimas“



Vertex šeiderio kodas


```
struct a2v {
  float4 coord;
  float4 prevCoord;
  float3 normal;
  float2 texture;
};
struct v2f {
  float4 hpos     : HPOS;
  float3 velocity : TEX0;
};

v2f main(a2v in,
         uniform float4x4 modelView,
         uniform float4x4 prevModelView,
         uniform float4x4 modelViewProj,
         uniform float4x4 prevModelViewProj,
         uniform float3   halfWinSize,
         )
{
  v2f out;

  // transform previous and current pos to eye space
  float4 P = mul(modelView, in.coord);
  float4 Pprev = mul(prevModelView, in.prevCoord);

  // transform normal to eye space
  float3 N = vecMul(modelView, in.normal);
```

```
  // calculate eye space motion vector
  float3 motionVector = P.xyz - Pprev.xyz;


  // calculate clip space motion vector
  P = mul(modelViewProj, in.coord);
  Pprev = mul(prevModelViewProj, in.prevCoord);

  // choose previous or current position based
  // on dot product between motion vector and normal
  float flag = dot(motionVector, N) > 0;
  float4 Pstretch = flag ? P : Pprev;
  out.hpos = Pstretch;

  // do divide by W -> NDC coordinates
  P.xyz = P.xyz / P.w;
  Pprev.xyz = Pprev.xyz / Pprev.w;
  Pstretch.xyz = Pstretch.xyz / Pstretch.w;

  // calculate window space velocity
  float3 dP = halfWinSize.xyz * (P.xyz - Pprev.xyz);

  out.velocity = dP;
  return v2f;
}
```

Motion Blur šeideris

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Kreipiasi į scenos tekstūrą daug kartų priklausomai nuo judesio vektoriaus
- Rezultatas yra pasverta (svarbumo prasme) atskaitų suma (weighted sum of samples)
  - Gali naudoti vienodus svorius (box filter), Gaussian ar judesio pabaigą (rampa)
- Semplų skaičius priklauso nuo judesio kiekio
  - Laikoma, kad 8 samplai yra gerai, 16 dar geriau
  - Daugiau semplų valgo našumą (mažina fps)
- Iš esmės greičio informacija panaudojama sukurti kažkią aproksimaciją tarp kadrų

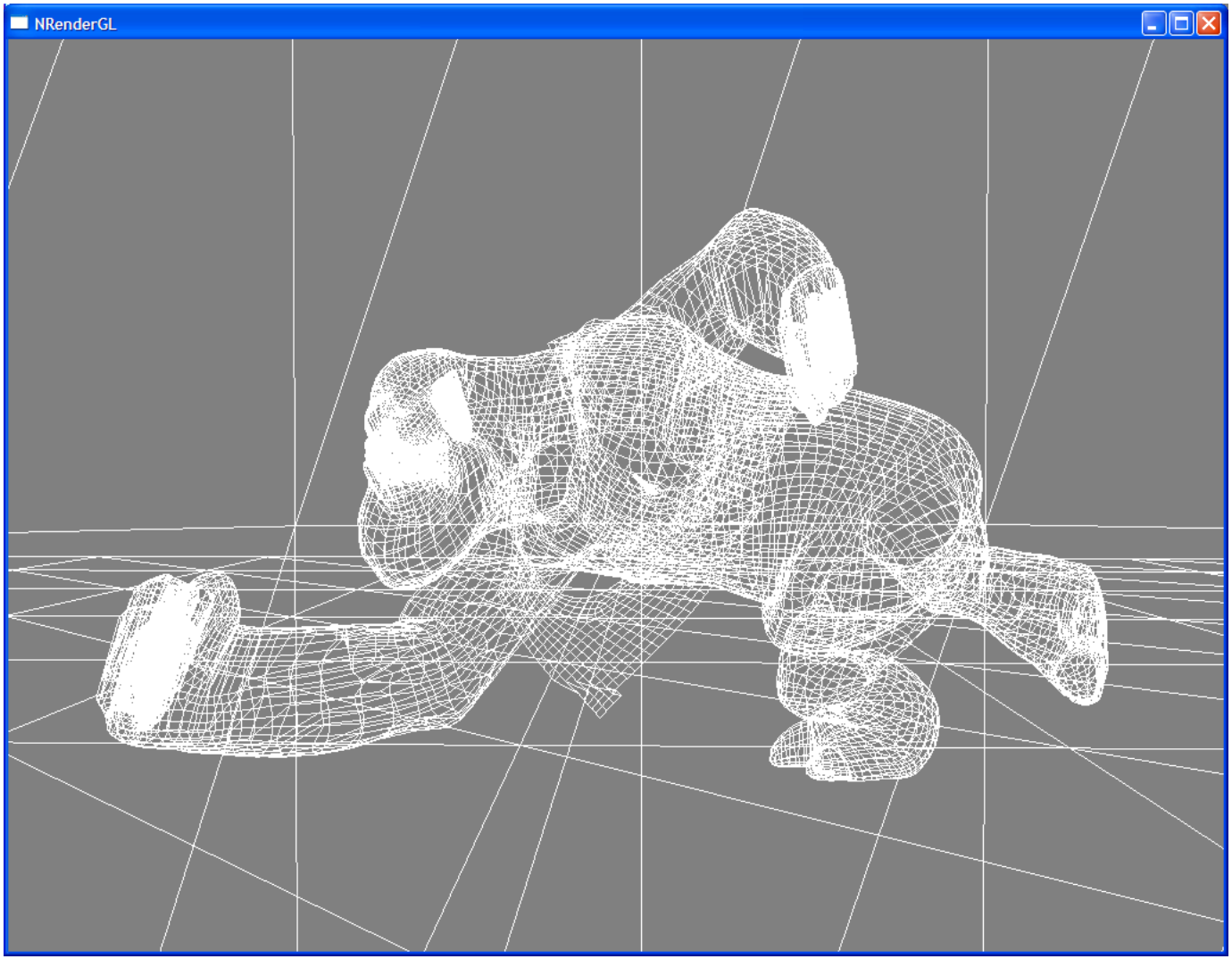
Motion Blur šeiderio kodas


```
struct v2f {
  float4 wpos     : WPOS;
  float3 velocity : TEX0;
};
struct f2f {
  float4 col;
};

f2fConnector main(v2f in,
                  uniform samplerRECT sceneTex,
                  uniform float blurScale = 1.0
                  )
{
  f2f out;
  // read velocity from texture coordinate
  half2 velocity = v2f.velocity.xy * blurScale;

  // sample scene texture along direction of motion
  const float samples = SAMPLES;
  const float w = 1.0 / samples; // sample weight

  fixed4 a = 0;                  // accumulator
  float i;
  for(i=0; i<samples; i+=1) {
    float t = i / (samples-1);
    a = a + x4texRECT(sceneTex, in.wpos + velocity*t) * w;
  }
  out.col = a;
}
```

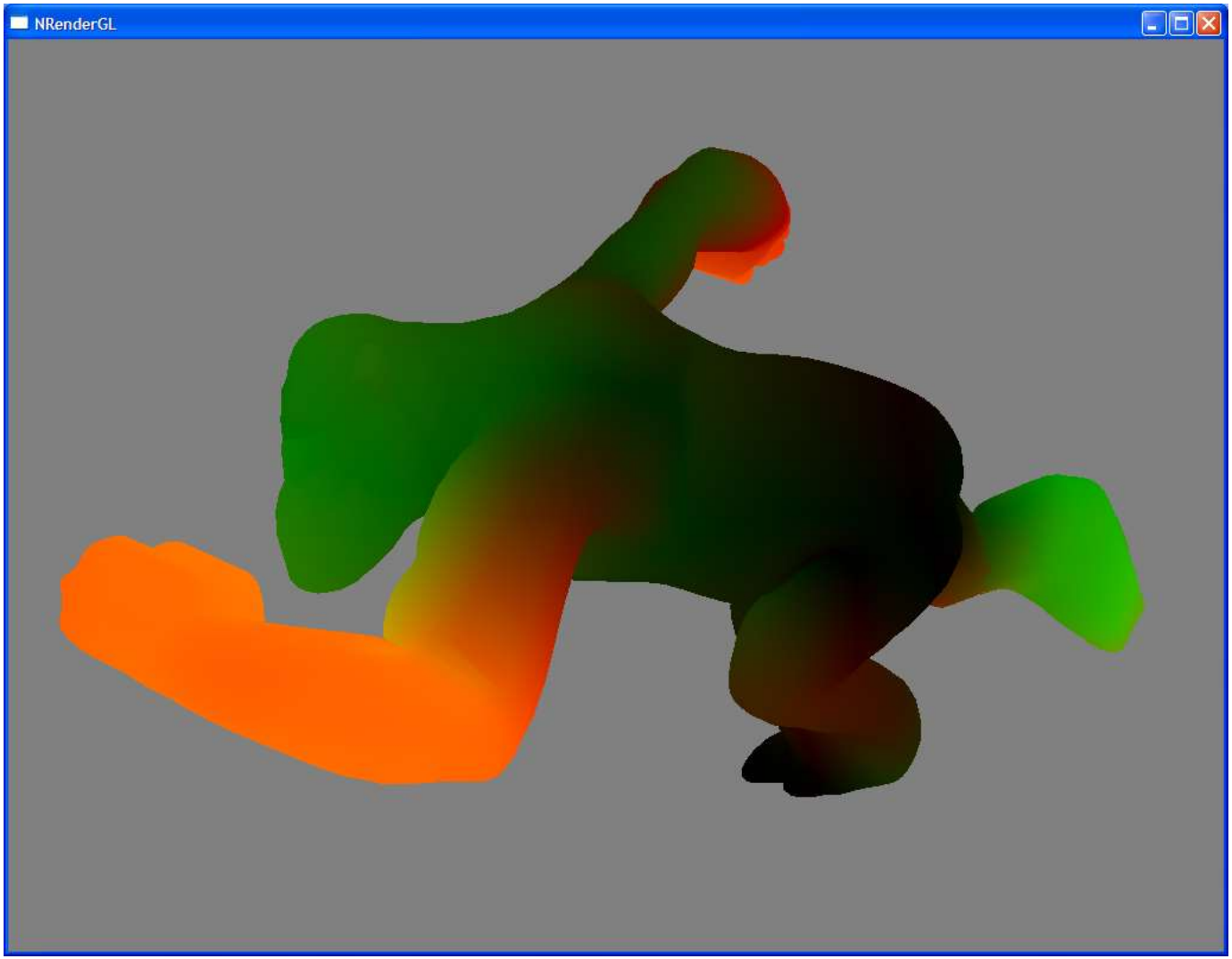
Originalus vaizdas

Ištampyta geomtrija

MULTIMEDIJOS INŽINERIJOS KATEDRA

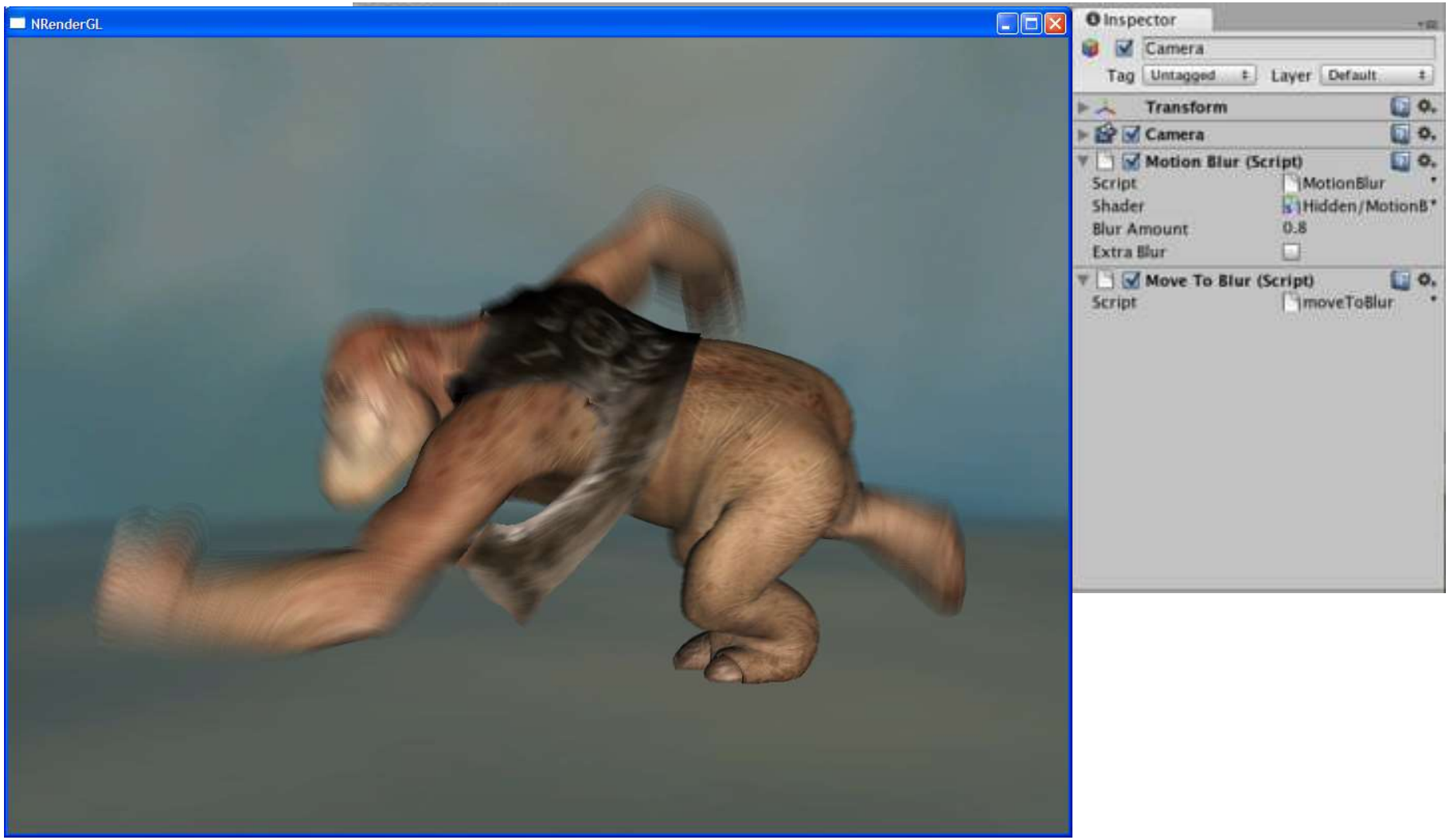


Greičio vizualizacija



Motion Blur‘u apdorotas vaizdas

MULTIMEDIJOS INŽINERIJOS KATEDRA

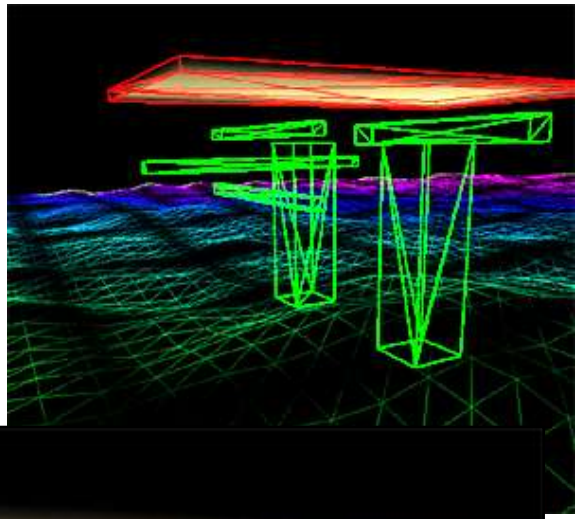


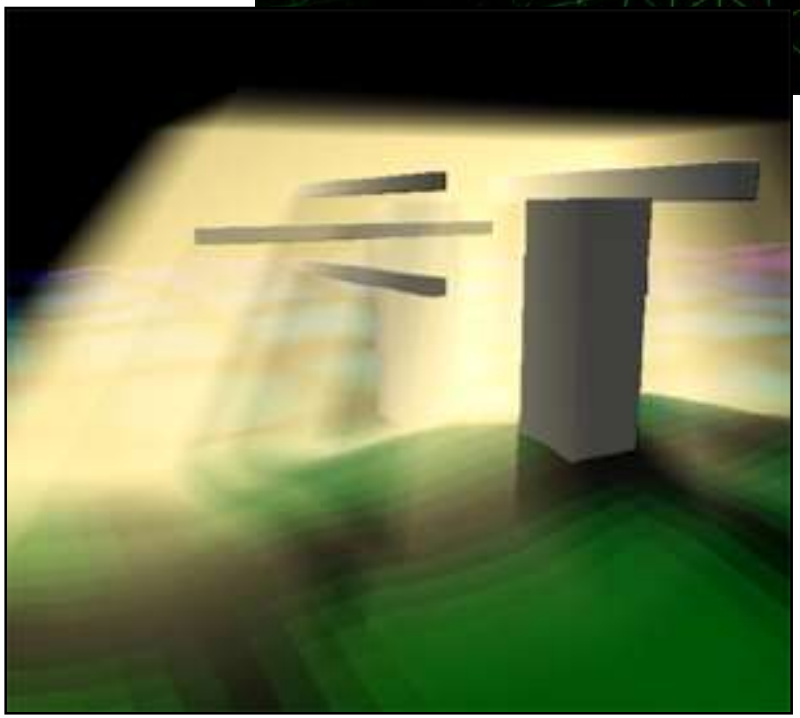
- **BANDOME (Effects kolekcija)**

Tūrinis rūkas (Volume Fog)

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Naudojami poligoniniai griaučiai (hull)
- Tikras tūrinis (volumetric) efektas
- Lengva redaguoti
- Galima animuoti
- Teigiami ir neigiami tūriai
- Greita okliuzija ir persikirtimai





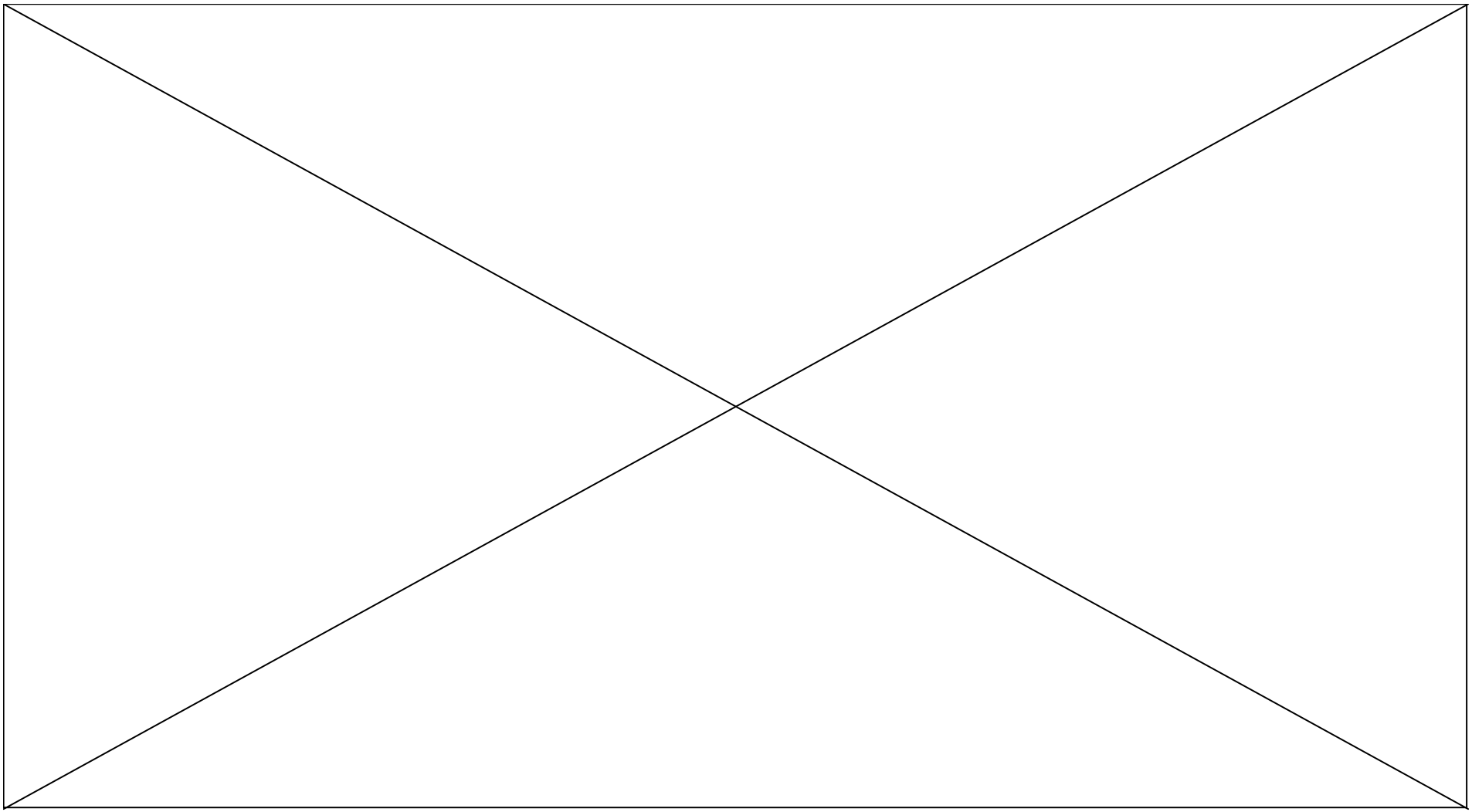
- Plačiau:
- http://developer.download.nvidia.com/SDK/9.5/Samples/DEMOS/Direct3D9/src/FogPolygonVolumes3/docs/FogPolygonVolumes3.pdf

Praktiškai

Concept art

In-game

Gyvai

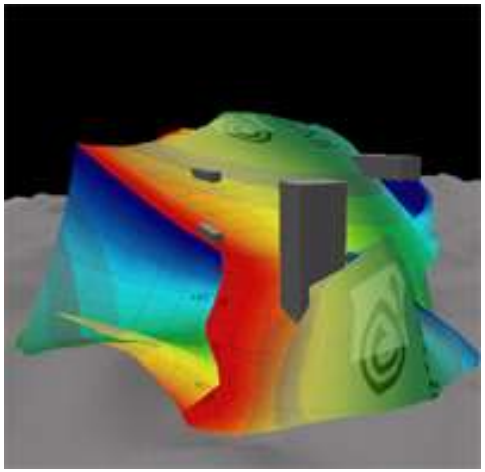
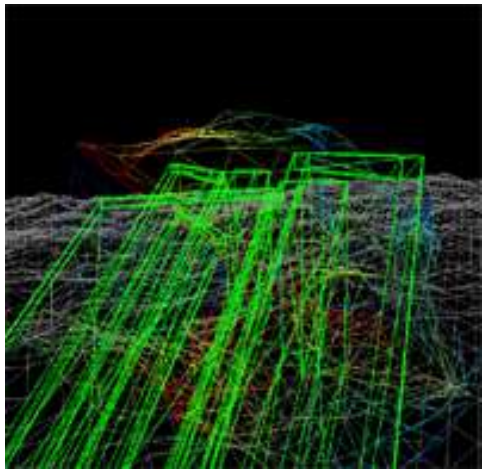
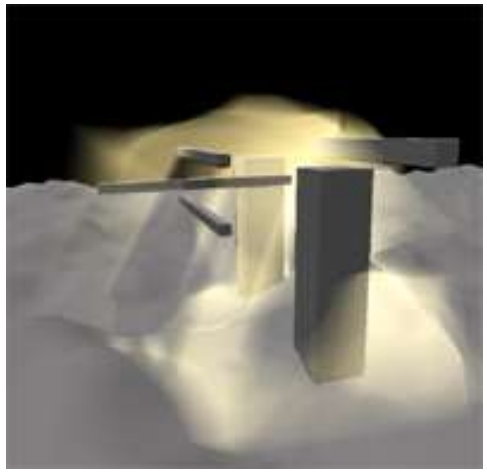


http://www.youtube.com/watch?v=RDND5hwS08k

Tūriniai griaučiai (volume hulls)

MULTIMEDIJOS INŽINERIJOS KATEDRA

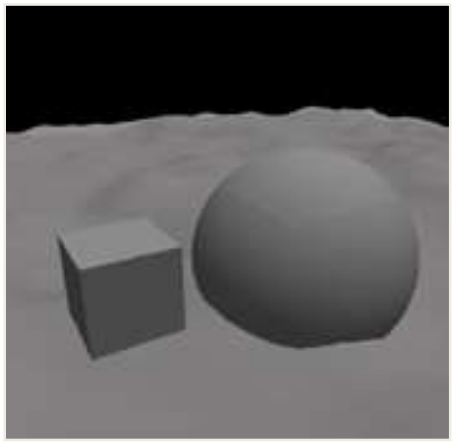
- Įprastinės poligonų figūros
  - Galite naudoti esamus objektus
  - Nereikia naujų objektui vertex ir pixel duomenų
  - Tik vienas kintamasis reguliuojantis „tirštumą“ ir 3 mažos bendrinės tekstūros
  - Galima naudoti stencil shadow volume geometriją

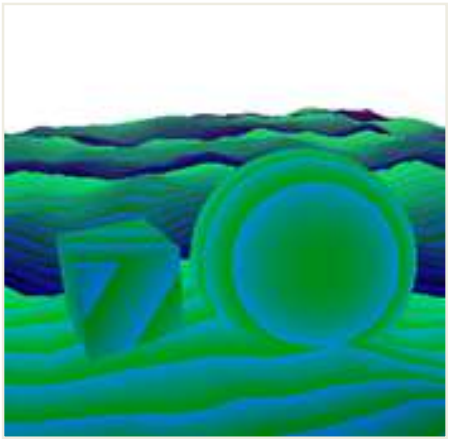




Kaip veikia?

- Renderiuojama į offscreen tekstūras
- Vietoje objekto „spalvos“ renderiavimo – renderiuojamas objekto gylis kiekvienam pikseliui (kaip darėte šešėlių atveju)
- Užkoduojamas gylis kaip RGB spalva

Objektai



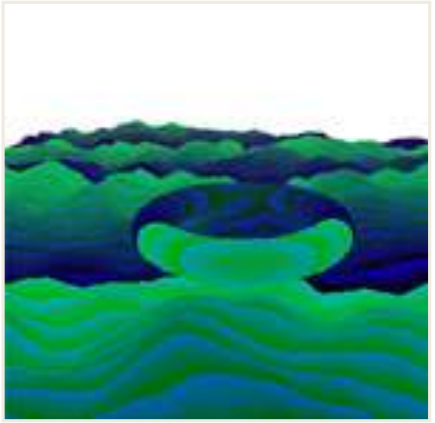


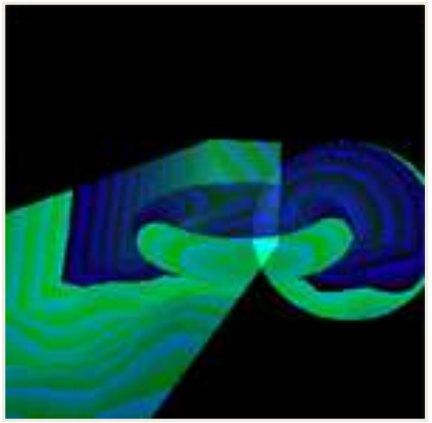
RGB koduojams gylis

- Gyliai naudojami „tirštumo“ suskaičiavimui kas kiekvieną pikselį

Algoritmas

1. Renderiuojam pasirinktą objektą į backbuffer
   - Standartiškai (be pakeitimų)
2. Render mūsų objekto gylį kuris perkirs rūko objektus (fog volumes)
   - Į ARGB8 texture, “S”
   - RGB-encoded depth.
3. Renderiuojam tūrinio rūko „ galinius vaizdus“ (fog volume backfaces)
   - Į ARGB8 texture, “B”
   - Sumuojami gyliai
   - Tekstūra sankirtoms, “S”

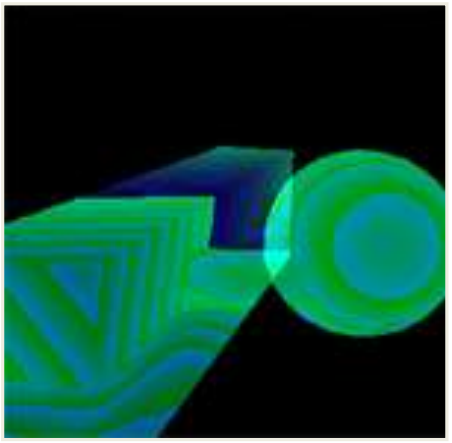


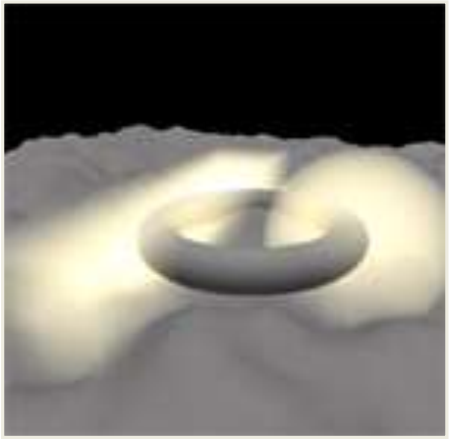


Algoritmas

MULTIMEDIJOS INŽINERIJOS KATEDRA

3. 4. Renderiuojam tūrinio rūko „priekinius vaizdus“ (fog volume frontfaces)
   - Į ARGB8 tekstūrą, “F”
   - Sumuojami gyliai
   - Tekstūra sankirtoms, “S”
5. Renderiuojamas per backbuffer
   - Atskaitos “B” ir “F”
   - Skaičiuojamas „tirštumas“ kiekvienam pikseliui
   - Skaičiuojama spalvų rampa
   - Tirštumas verčiamas į spalvas
   - Surišama su scena
   - 7 instrukcijos ps_2_0 šeideryje

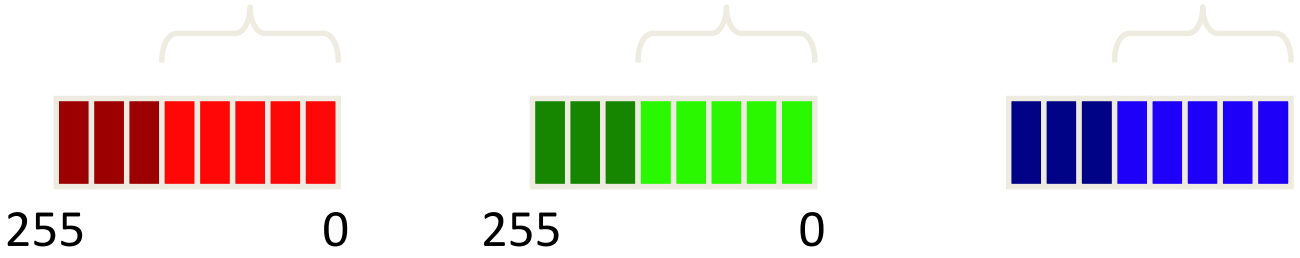




RGB gylio kodavimas

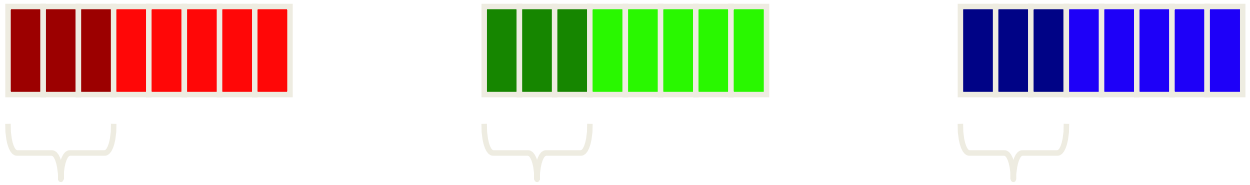
MULTIMEDIJOS INŽINERIJOS KATEDRA

- Naudojama “L” „žemų“ (low) bitų kiekvienam spalvos kanalui
  - Pvz. 5 bitai iš R, G, ir B spalvų duoda 3*L bitų tikslumą



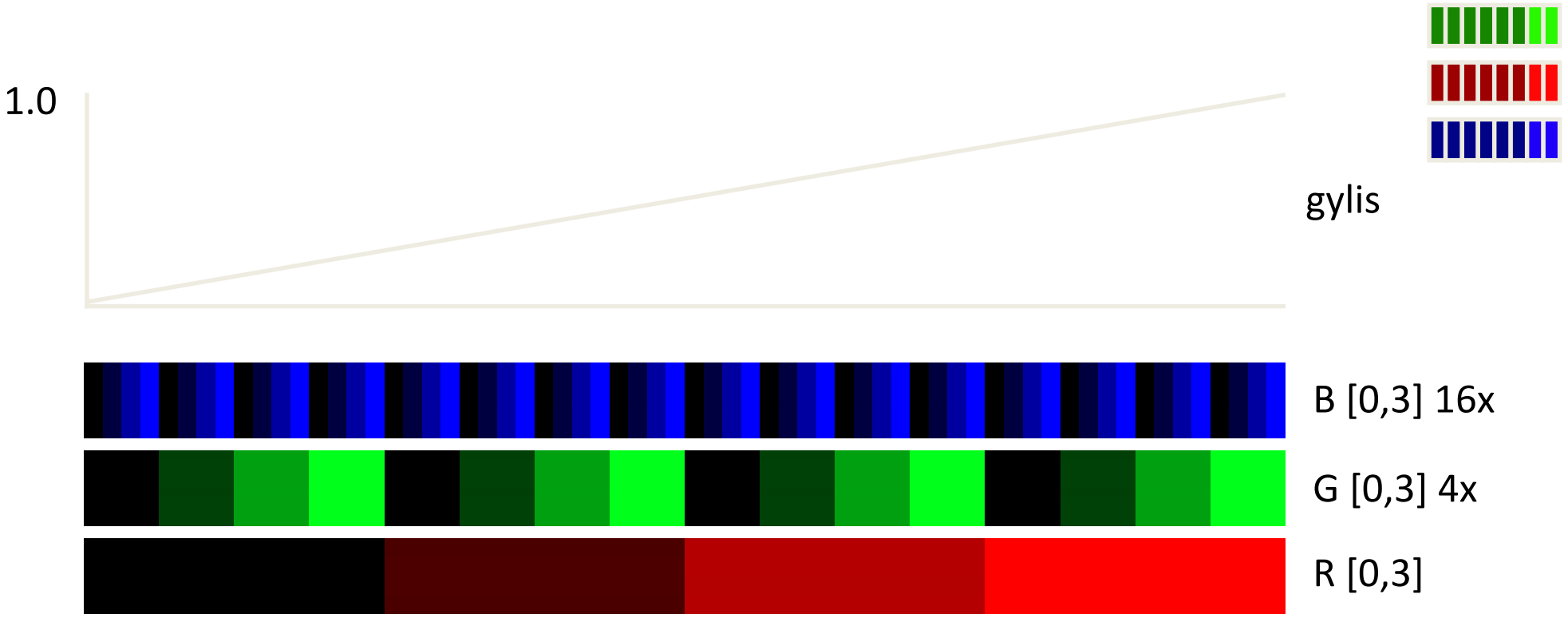
- (8 – L) „aukštų“ (high) bitų saugojimui, “H”
  - $2^{(8-L)}$ gylio vertes galima pridėti iki prisipildymo (overflow)
  - L=5 leidžia pridėti 8 vertes

RGB gylio kodavimas



- (8 – L) „aukštų“ (high) bitų saugojimui, “H”
  - $2^{(8-L)}$ gylio vertes galima pridėti iki persodrinimo (saturation)
  - L=5 leidžia pridėti 8 vertes

RGB kodavimo diagram
L=2



- Vienas 6-bit gylis sunaudoja tik [0,3] iš [0,255]
- Vertės [4,255] naudojamos pridedant gylius

RGB-Encoding

MULTIMEDIJOS INŽINERIJOS KATEDRA

- Vertex šeideris skaičiuoja gylį iš [0,1]

```
DP4 r1.x, V_POSITION, c[CV_WORLDVIEWPROJ_0]
DP4 r1.y, V_POSITION, c[CV_WORLDVIEWPROJ_1]
DP4 r1.z, V_POSITION, c[CV_WORLDVIEWPROJ_2]
DP4 r1.w, V_POSITION, c[CV_WORLDVIEWPROJ_3]
```

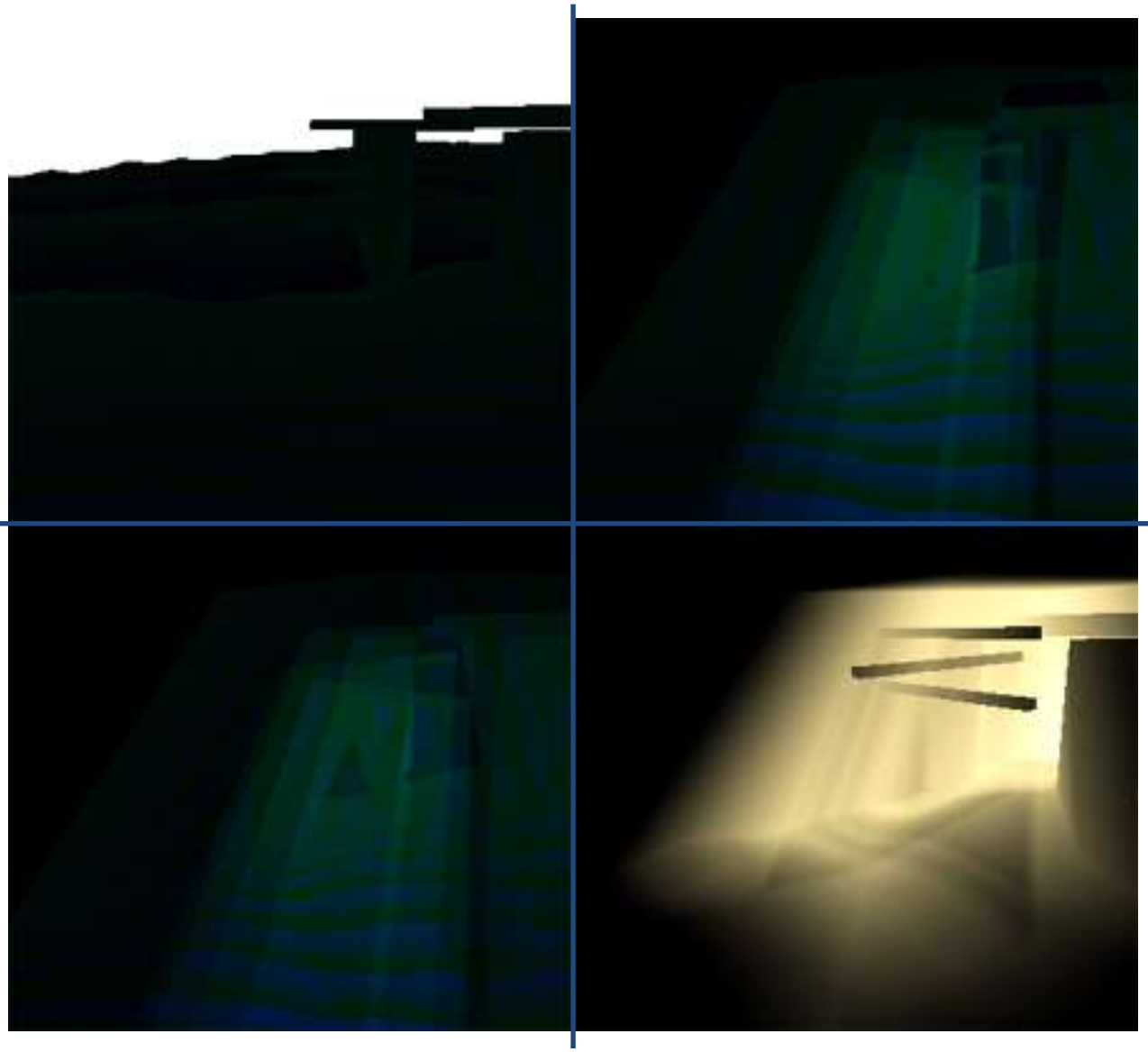
- Vertex šeideris gylį verčia tekstūros koordinatėmis
  - TexCoord.r = D * 1.0
  - TexCoord.g = $D * 2^{L}$ ie. G = D * 16
  - TexCoord.b = $D * 2^{2L}$ ie. B = D * 256

```
MUL  r0.xyz, r1.z, c[CV_DEPTH_TO_TEX_SCALE].xyz
```

RGB užkoduoti gyliai

MULTIMEDIJOS INŽINERIJOS KATEDRA

Objekto gylis



Galinių pusių suma

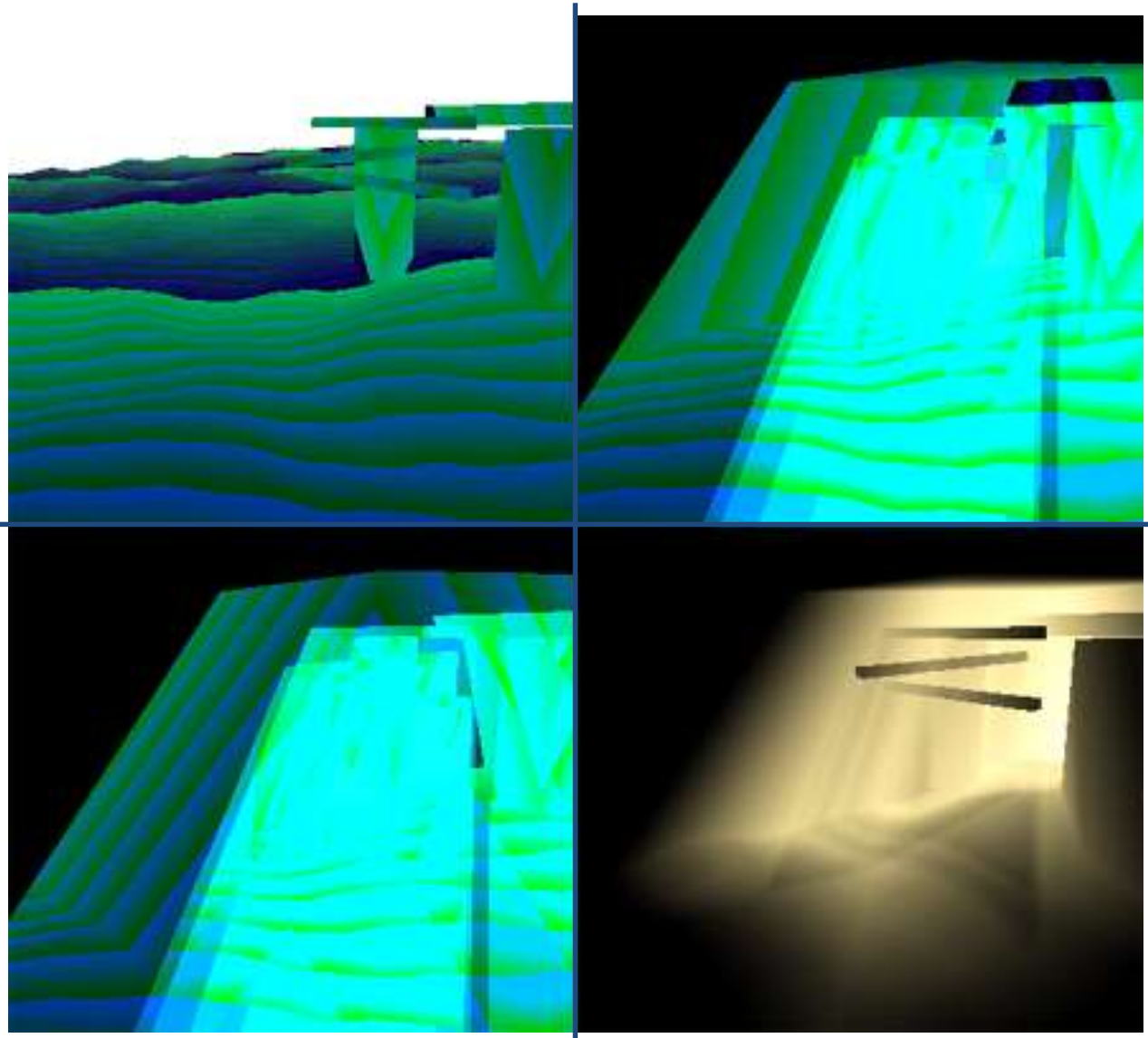
Priekinių pusių suma

Surenderiuotas vaizdas

Paryškintai

MULTIMEDIJOS INŽINERIJOS KATEDRA

Objekto gylis



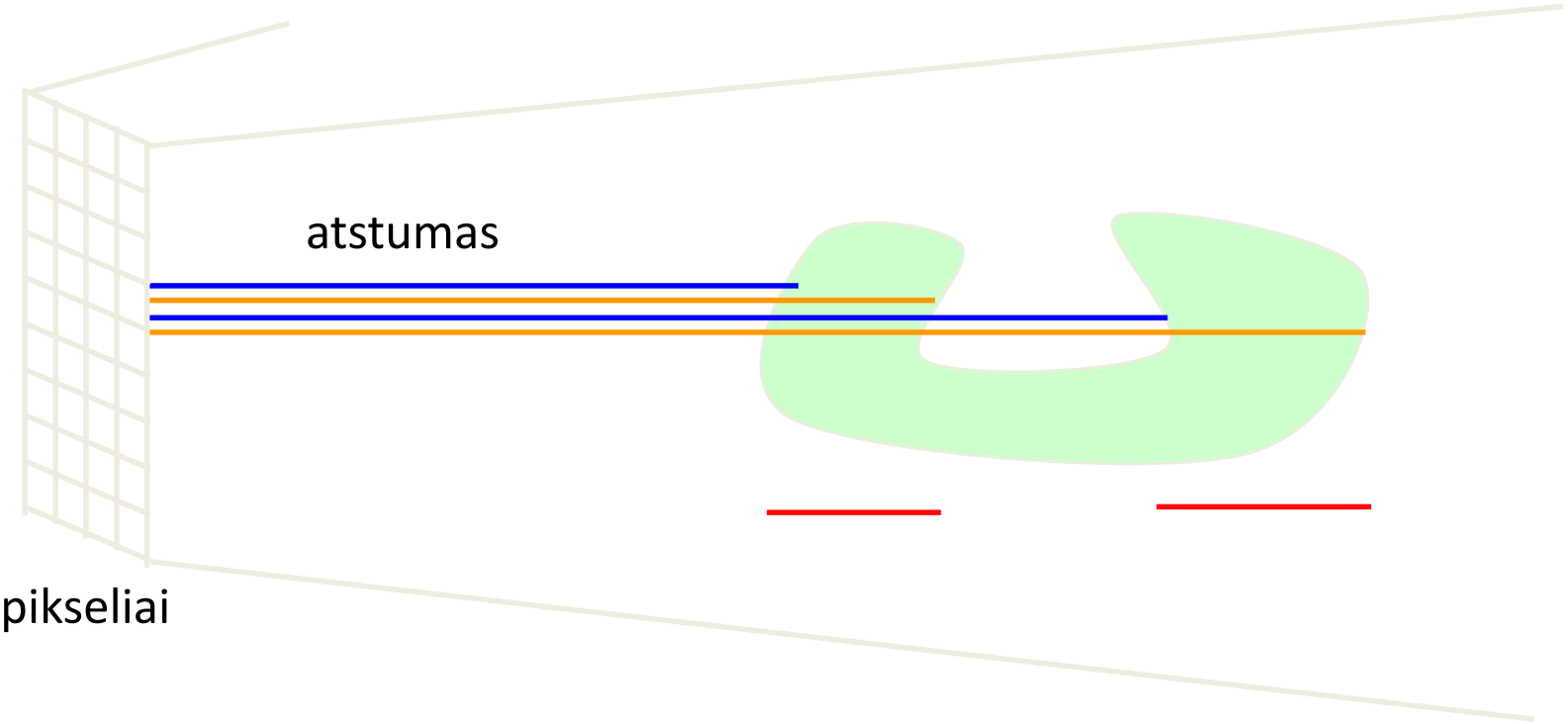
Galinių pusių suma

Priekinių pusių suma

Surenderiuo tas vaizdas

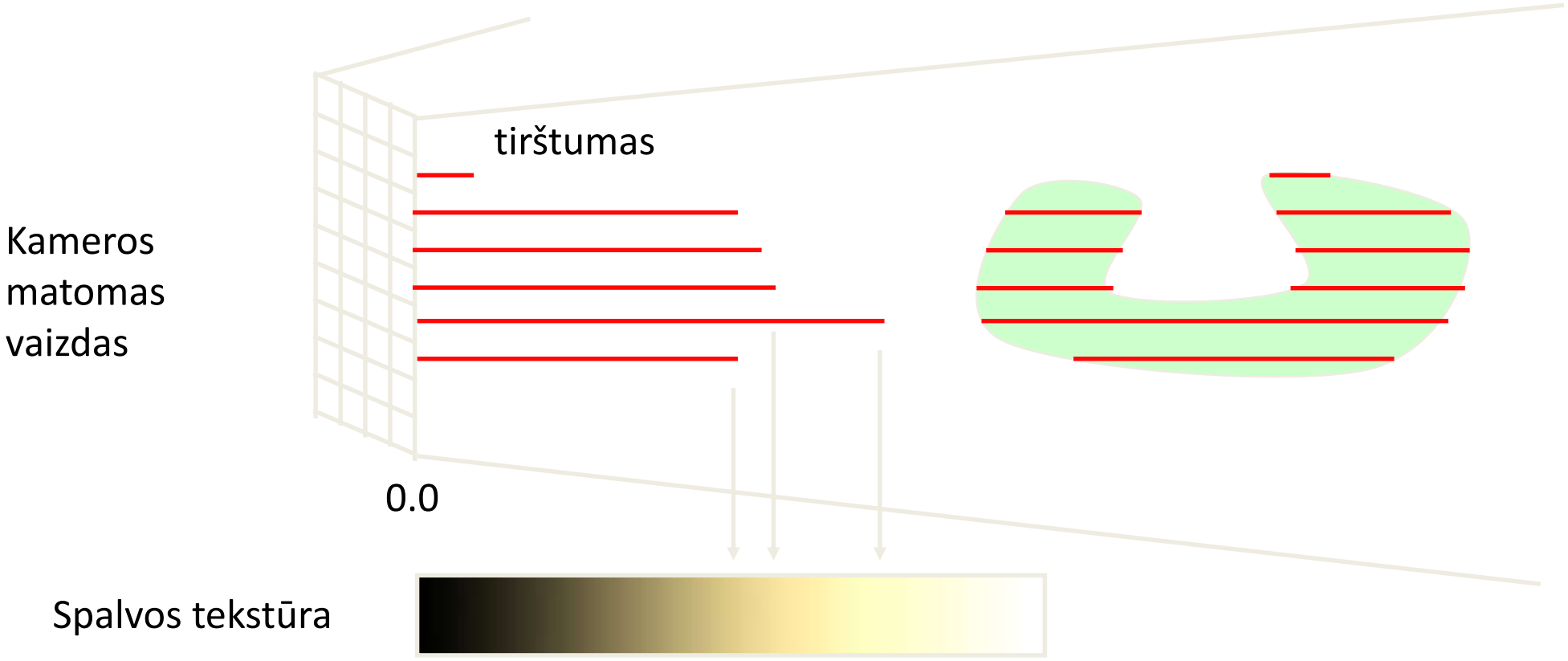
Tirštumo renderiavimas

Kameros matomas vaizdas



$$Thickness = \sum Back - \sum Front$$

Tirštumo renderiavimas



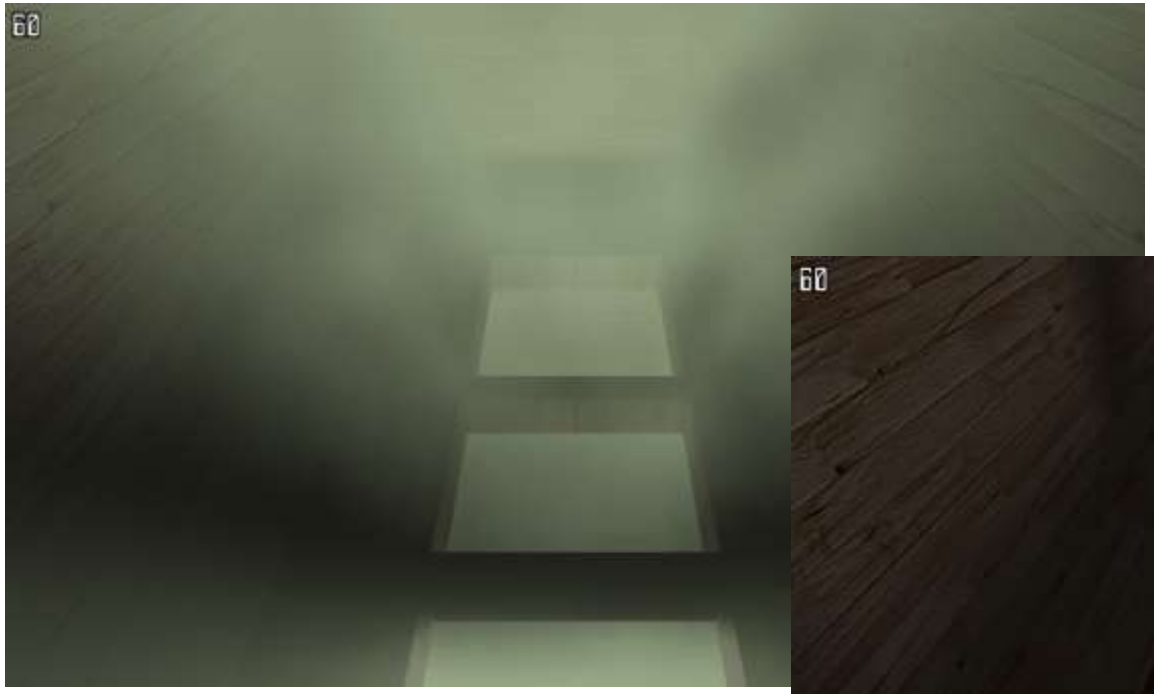
- Tirštumas * skalė ➔ TexCoord.x
- Spalvos rampa matematiškai arba „kūrybiškai“
- Lengvai valdoma

RGB verčių dekodavimas

- Viena DOT funkcija !
- **Decoded value = (D.r, D.g, D.b) DOT (1.0, 2^{-L}, 2^{-2L})**
- Kintamo kablelio (Floating point) tikslumas
- PS 2.0

Pavyzdžiai

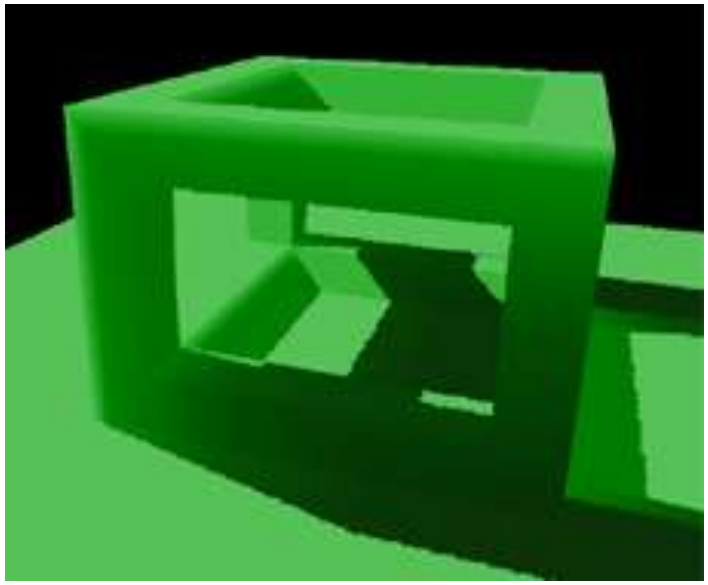
MULTIMEDIJOS INŽINERIJOS KATEDRA



Bandome (žr. moodle)

Persišviečiamumas (Translucency)

- „Stiklas“
- Spalvų rampa paremta atstumu kurį šviesa „nukeliauja“ per objektą
- Svarbu su shadow maps

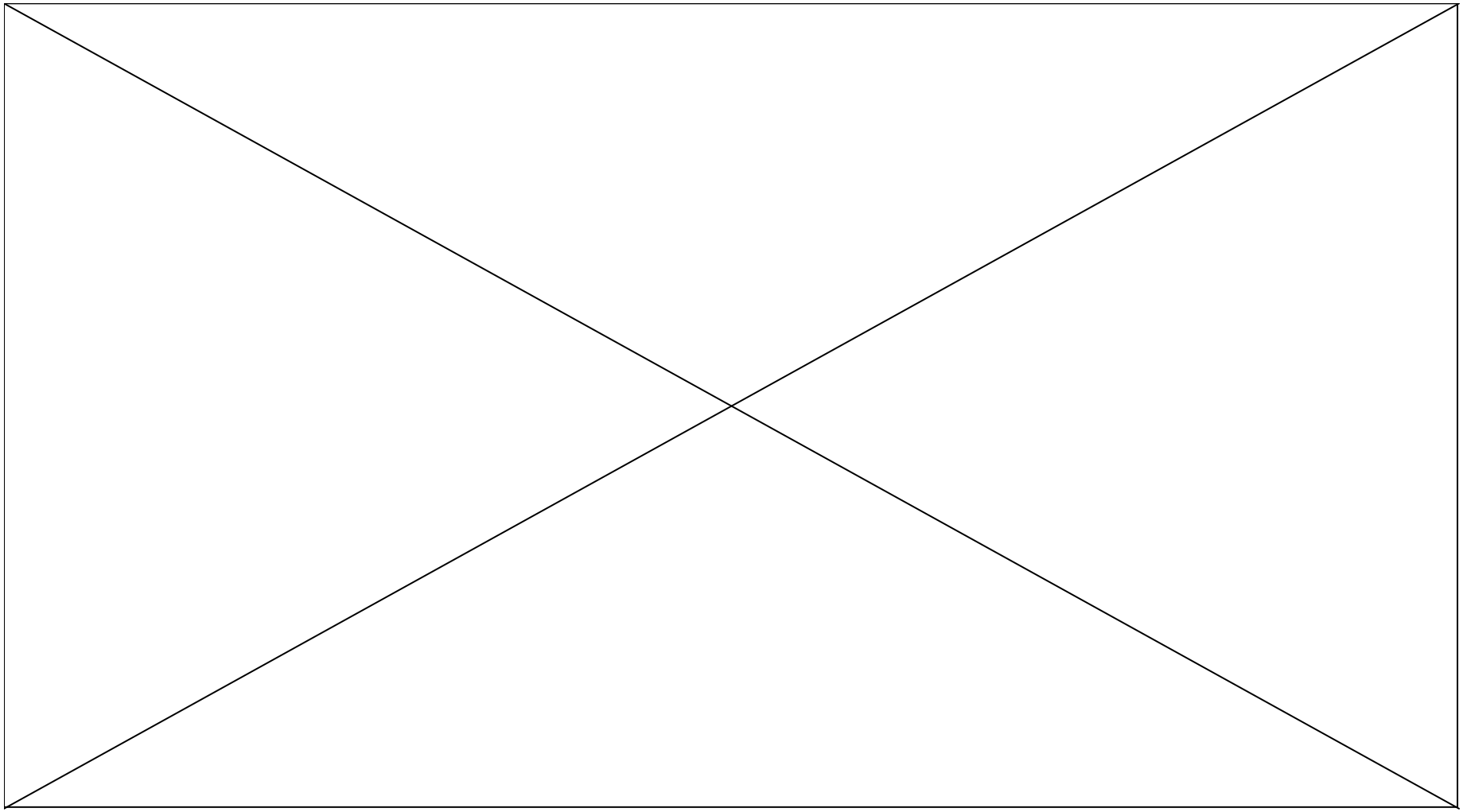


Greg James, NVIDIA

Simon Green, NVIDIA

Gyvai atrodo taip

MULTIMEDIJOS INŽINERIJOS KATEDRA



http://www.youtube.com/watch?v=vYqqFhPJ58E

Bandome (žr. moodle)

Paprastos efektų operacijos per HLSL

- Standartiškai turime:
- sampler2D input : register(s0);
  float4 main(float2 uv : TEXCOORD) : COLOR
  {
  float4 Color;
  Color = tex2D(input , uv.xy);
  return Color;
  }

Paryškinti?

- Float2 galime suprasti kaip koordinatę ir išskaidyti į uv.x ir uv.y
- Spalva yra float4 (argb), vadinasi galima pakeisti spalvą

- Color = tex2D(input , uv.xy)*3;
- Padauginus iš trijų vaizdas bus ryškesnis ☺
- Arba taip
- Color = tex2D(input , uv.xy)*uv.x;

Spalva?

- Color = tex2D(input , uv.xy);
  Color.b = Color.b*2;

- Arba

- Color = tex2D(input , uv.xy);
  Color.r = Color.r*sin(uv.x*100)*2;
  Color.g = Color.g*cos(uv.x*150)*2;
  Color.b = Color.b*sin(uv.x*50)*2;

Tampom?

- uv.x = uv.x * 0.5;
  Color = tex2D(input , uv.xy);

- Arba

- uv.x = uv.x / 0.5;
  Color = tex2D(input , uv.xy);

Dar?

- Color -= tex2D(input , uv.xy-0.003)*2.7f;
  Color += tex2D(input , uv.xy+0.003)*2.7f;
  Color.rgb = (Color.r+Color.g+Color.b)/3.0f;

- Color.rgb = (Color.r+Color.g+Color.b)/3.0f;
  if (Color.r<0.2 || Color.r>0.9) Color.r = 0;
  else Color.r = 1.0f;
  if (Color.g<0.2 || Color.g>0.9) Color.g = 0;
  else Color.g = 1.0f;
  if (Color.b<0.2 || Color.b>0.9) Color.b = 0;
  else Color.b = 1.0f;