Topic IV: Texturing & Effects

Part I: Texture Mapping, Shader, Transparency

Sarah Böning

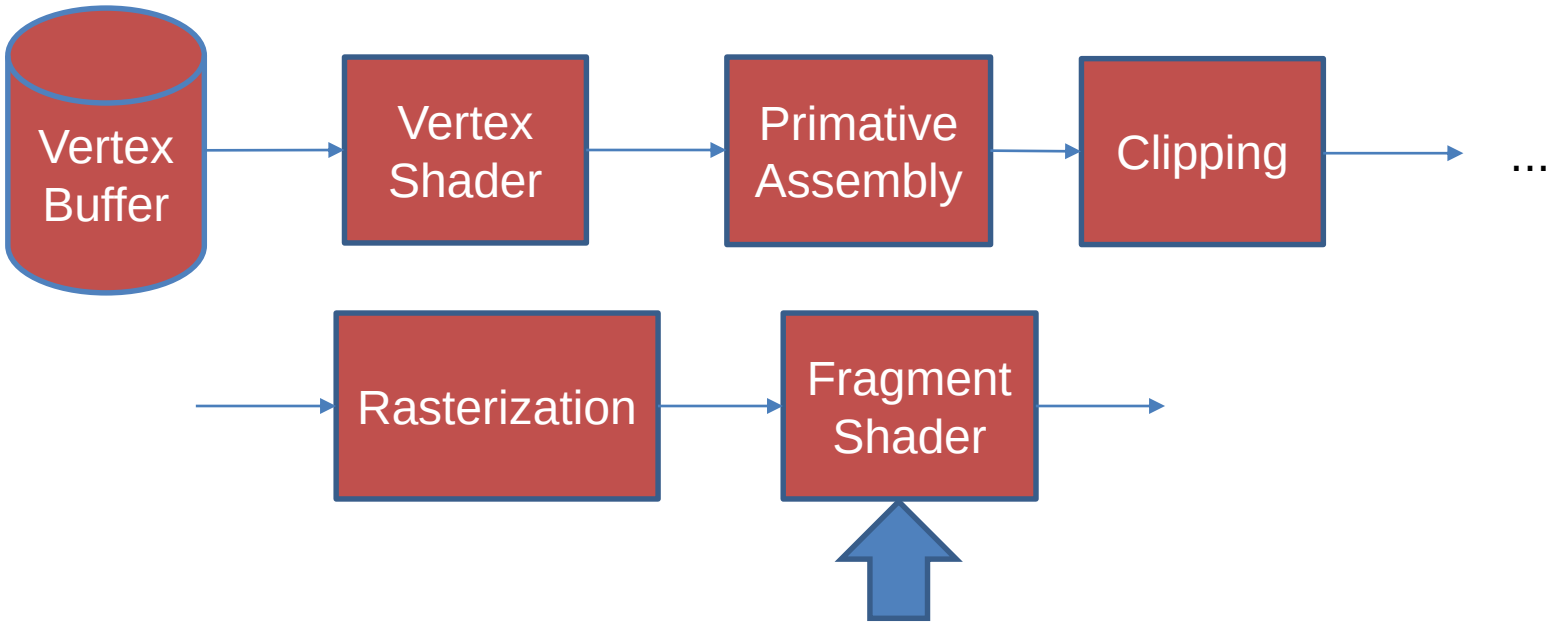
Bauhaus-Universität Weimar

Contents

- Texture Mapping
- Wrapping & Filteration
- Shaders
- Textures in OpenGL
- Transperancy

Bauhaus-Universität Weimar

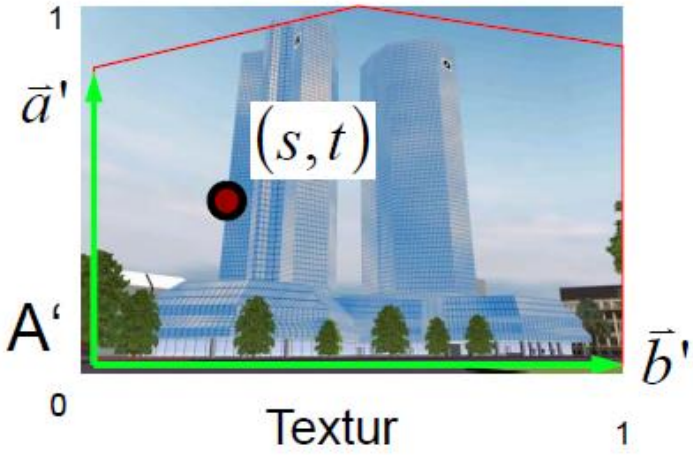
Textures in OpenGL Rendering Pipeline



Bauhaus-Universität Weimar

Texture Mapping

- Definition: texel = pixel of texture
- Map colour of texture on pixel from corresponding texture coordinate (s,t)
- $(s,t) \in [0,1] \times [0,1]$
- Image: $2^n \times 2^n$ texels



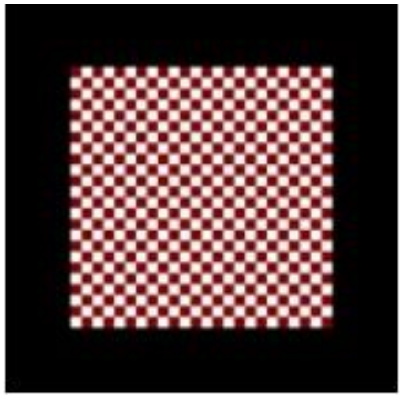
Bauhaus-Universität Weimar

Image source: Prof. Thorsten Grosch, Vorlesung Computergrafik I, WS 16/17, TU Clausthal

Wrapping

- What happens if texture coordinate is out of bound?
- GL_CLAMP: x<0 => x=0, x>1 => x=1
- GL_REPEAT: use only decimal places

Clamp



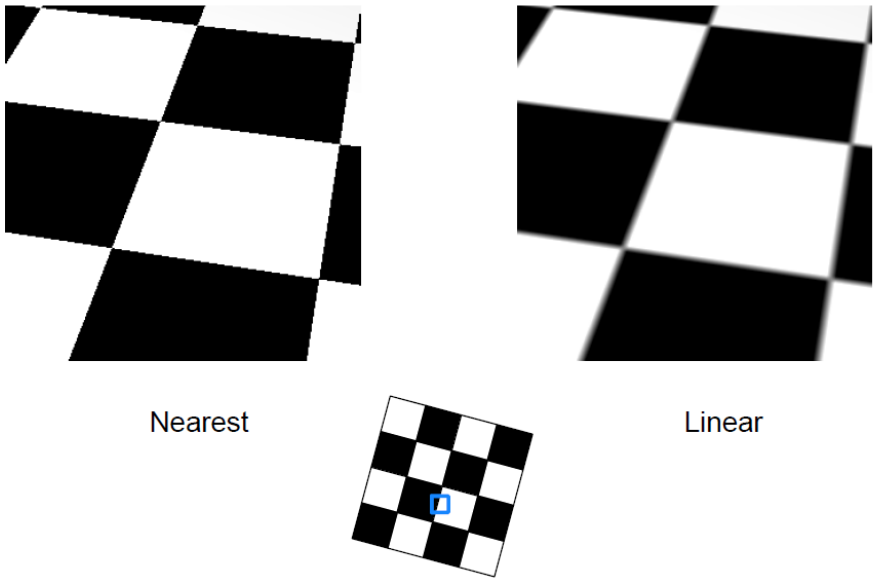
Repeat

Bauhaus-Universität Weimar

Image source: http://slideplayer.com/slide/7698973/

Texture filtering

- Magnify: 1 texel on several pixels
  - GL_NEAREST: take 1 texel
  - GL_LINEAR: bilinear interpolation of 4 near texels

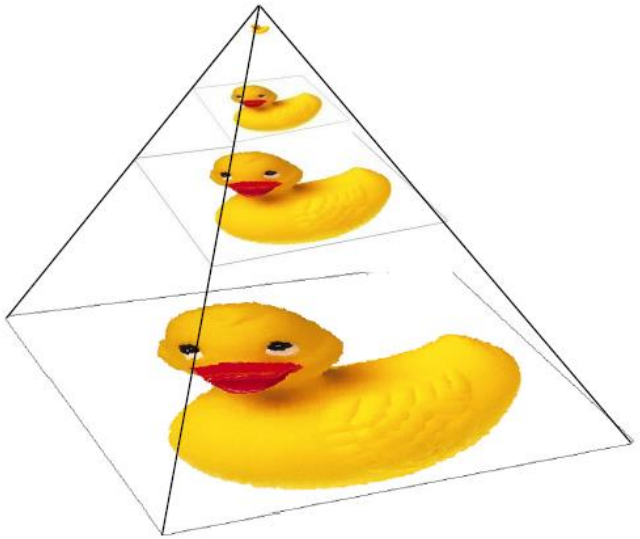


Bauhaus-Universität Weimar

Image source: Prof. Thorsten Grosch, Vorlesung Computergrafik I, WS 16/17, TU Clausthal

Magnify and MipMap

- Minify: several texels on one pixel
  - MipMap: texture in several sizes, level chosen by OpenGL



Bauhaus-Universität Weimar

Image source: Prof. Thorsten Grosch, Vorlesung Computergrafik I, WS 16/17, TU Clausthal

Vertex & Fragment (Pixel) Shader

- **Vertex Shader**: for every vertex, passing values along (e.g. color, texture coordinates)
- **Fragment Shader**: for every pixel, access texture at coordinates, set the color

Bauhaus-Universität Weimar

Textures in OpenGL

```
//Before: defining geometry/scene and buffers etc.
//Before: Image-Loader e.g. own or DevIL
glGenTextures(1,&ID);
glBindTexture(GL_TEXTURE_2D,ID);
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT );
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT );
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,
GL_NEAREST);
glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,width,height,0,GL_RGB,
GL_FLOAT,data);
glGenerateMipmap(GL_TEXTURE_2D);
```

Bauhaus-Universität Weimar

Textures in OpenGL: Shader

Vertex

```
#version 330 core

layout(location=0) in vec4 vPos;
layout (location=3) in vec4
vTexCoord;
out vec3 fTexCoord;

void main(){
        gl_position = vPos;
        fTexCord = vTexCoord.rgb;
}
```

Fragment

```
#version 330 core
uniform sampler2D image;
in vec4 fTexCoord;
out vec3 pixelColor;

Void main(){
        pixelColor =
texture(image,fTexCoord.st).rgb;
}
```

Bauhaus-Universität Weimar

Transparency

- Changing opacity: RGB → RGBα
- 4D vector for color in shaders:

```
//Vertex & Fragment
out vec4 color;

//Fragment
color.a = 0.5;
```

- Can be used for multi-textures, glass, sheer cloth, ...

Bauhaus-Universität Weimar

Transparency

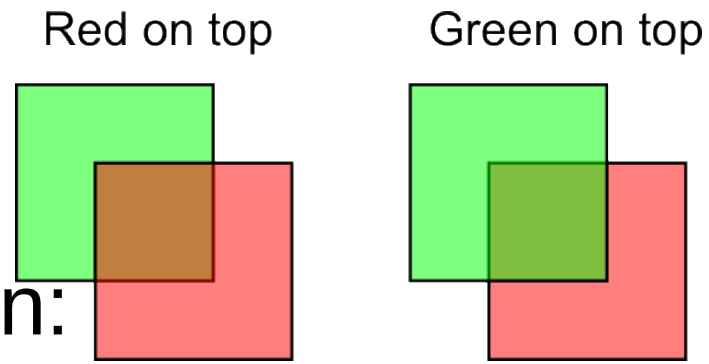


Image source: http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-10-transparency/

- Important: Order of colors
- Need to use Blend function:

```
glEnable (GL_BLEND);
glBlendFunc (sBlend, dBlend);
```

- *sBlend*: computation of source RGBα
- *dBlend*: computation of destination RGBα

Bauhaus-Universität Weimar

Literature

- Opengl-tutorial.org, accessed 17.04.18
- Prof. Thorsten Grosch, Vorlesung Computergrafik I, WS 16/17, TU Clausthal
- https://www.opengl.org/archives/resources/faq/technical/transparency.htm, accessed 16.04.18
- 3dm_ch10, Rtr_ch3/6

Bauhaus-Universität Weimar

Topic IV: Textures and Effects

Part II: Texture effects, procedural textures, billboards

Sarah Böning

Bauhaus-Universität Weimar

Contents

- Texture effects
  - Bump/normal/displacement mapping
- Procedural textures
- Billboarding

Bauhaus-Universität Weimar

Texture Effects

Bauhaus-Universität Weimar

Texture effects

- Bump mapping
  - Input is greyscale image → up or down
  - Object „has“ more smaller details → looks more realistic
  - Geometry can stay simple

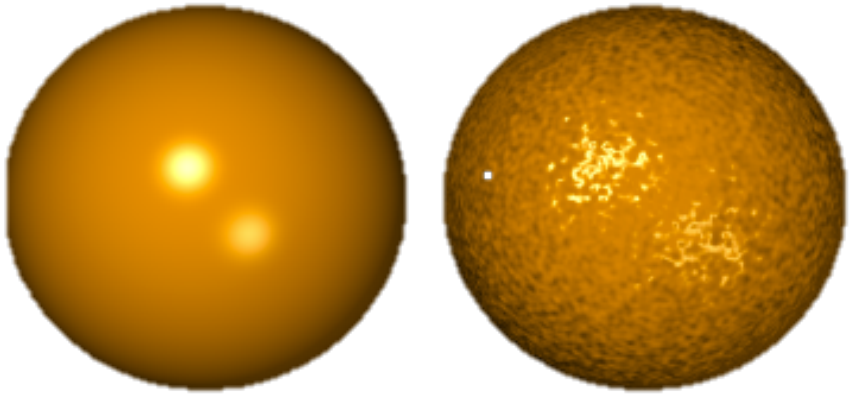
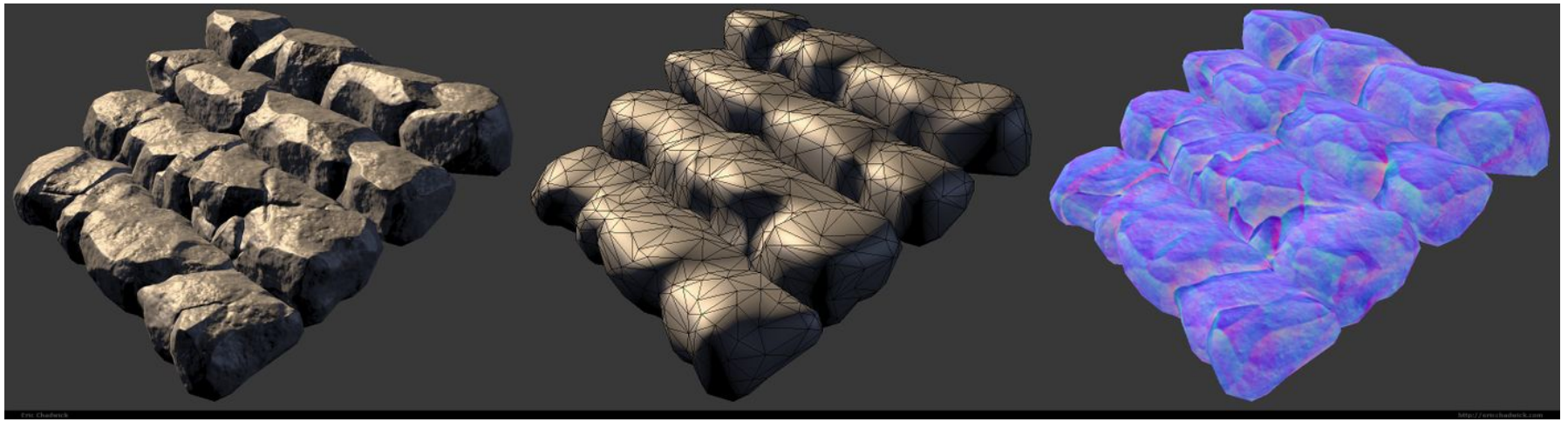


Image source: https://commons.wikimedia.org/wiki/File:Bump-mapping_example.png

Bauhaus-Universität Weimar

- Normal mapping
  - Similar to bump mapping
  - Input is used for orientation of surface normals: RBG used for XYZ
  - Object „has“ more depth details without adding complex geometry



Bauhaus-Universität Weimar

Image source: http://wiki.polycount.com/wiki/Normal_map

- Displacement mapping
  - Input is interpretated as shift in direction of surface normal
  - Meshes need to be subdivided in small triangles

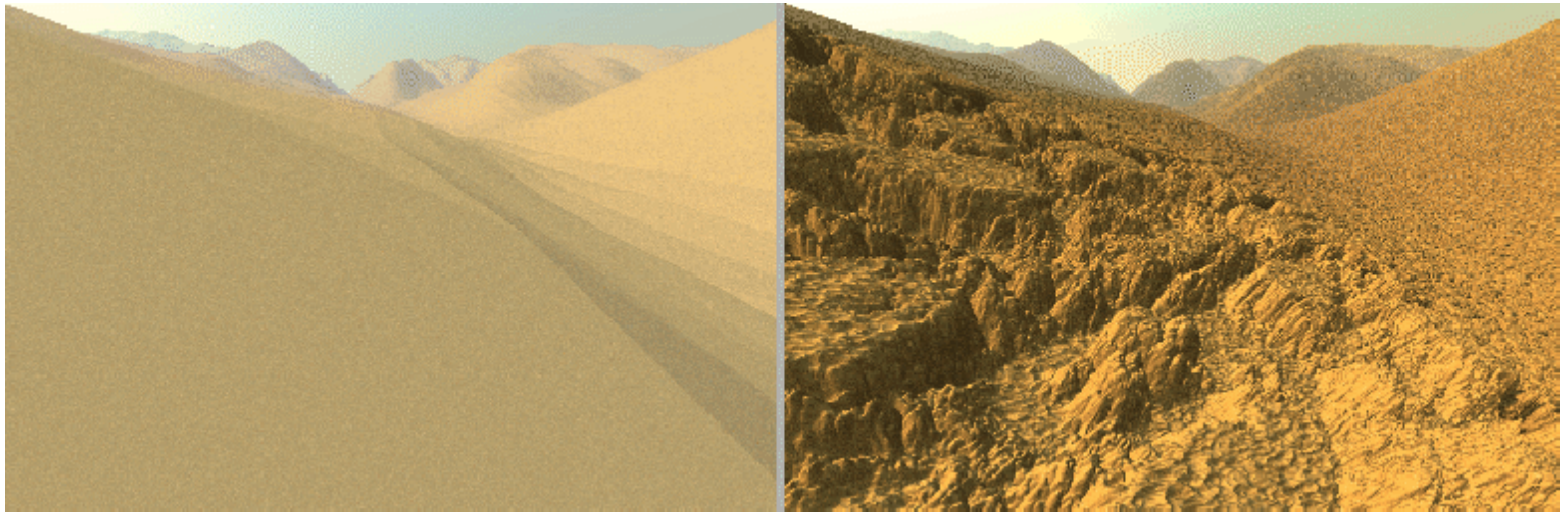


Image source: Prof. Thorsten Grosch, Vorlesung Computergrafik 1, WS 16/17

Procedural Textures

Bauhaus-Universität Weimar

Procedural textures

- Create texture from procedure function: coordinates → color
- Possible to combine textures/effects
- No limited resolution & size, high detail, „infinite“ patterns
  - Can be used in whole scene without repetition
  - Often used for wood, marble, stone, grass, water, clouds, …

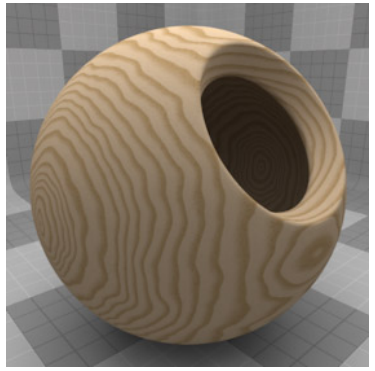


Image source: http://modo.docs.thefoundry. co.uk /modo/701/help/pages/shaderendering/shaderitems/wood.html

Image source: https://en.wikipedia.org/wiki/Procedural_texture

Image source: http://www.upvector.com/pages/Tutorials/Intro%20to%20Procedural%20Textures/image05/example1.jpg

Bauhaus-Universität Weimar

Example of procedural textures

$$f(x,y) = \frac{1 + \sin\left(x + \frac{noise(x*5, y*5)}{2}\right) * 50}{2}$$

Bauhaus-Universität Weimar

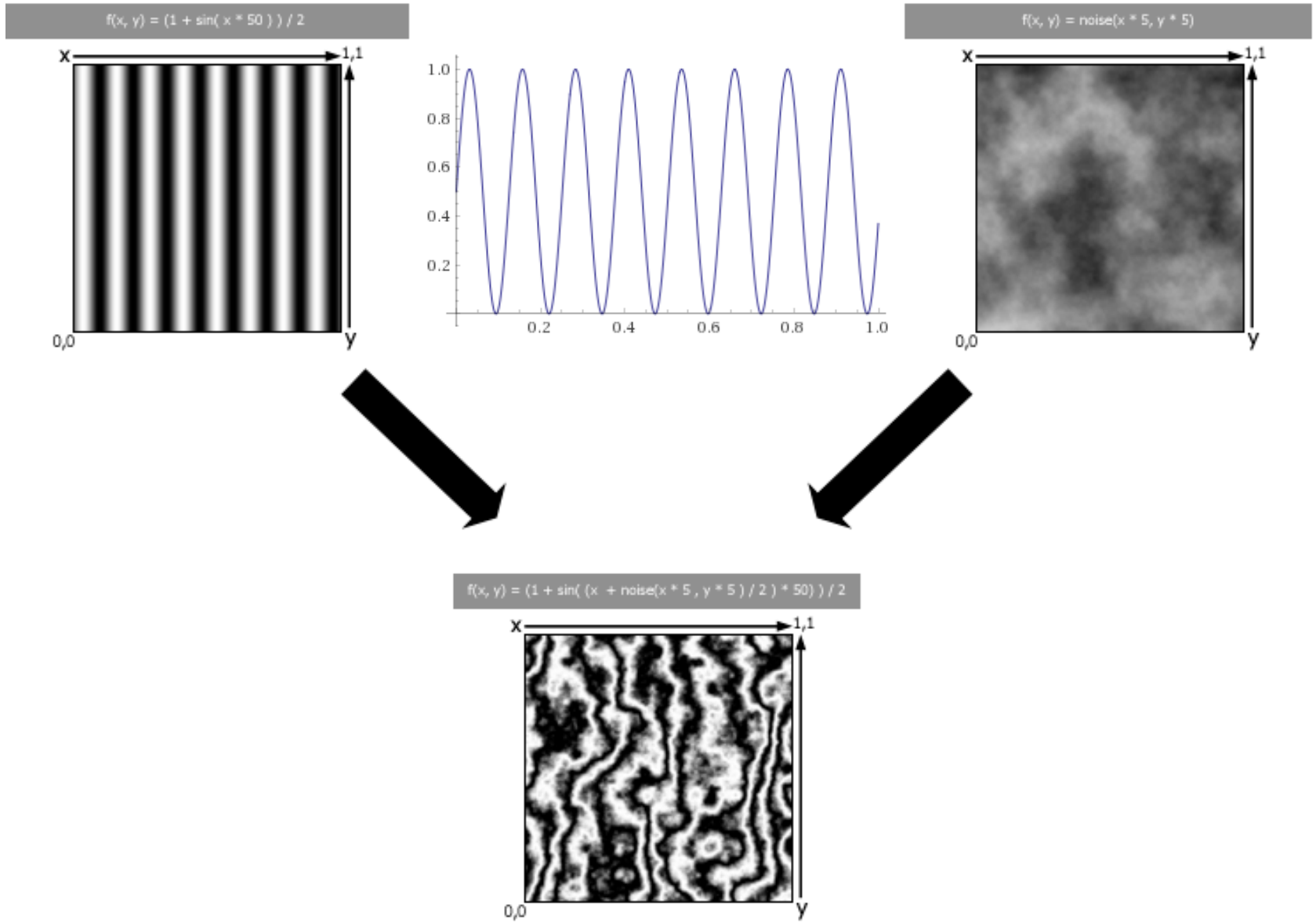
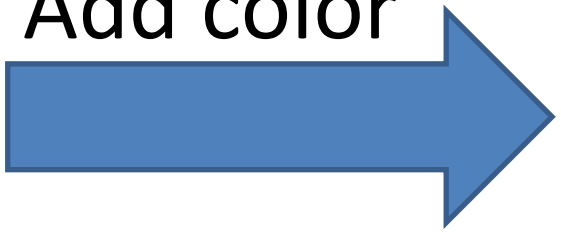
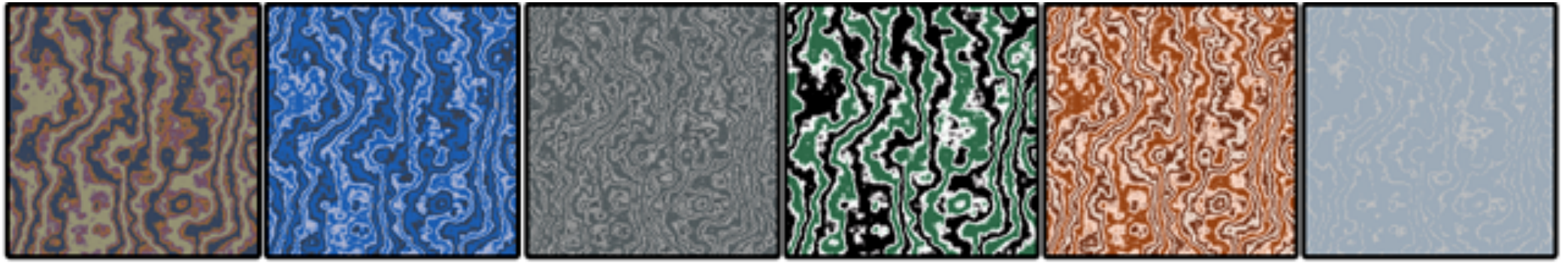


Image source: http://www.upvector.com/?section=Tutorials&subsection=Intro%20to%20Procedural%20Textures

Add color



Bauhaus-Universität Weimar

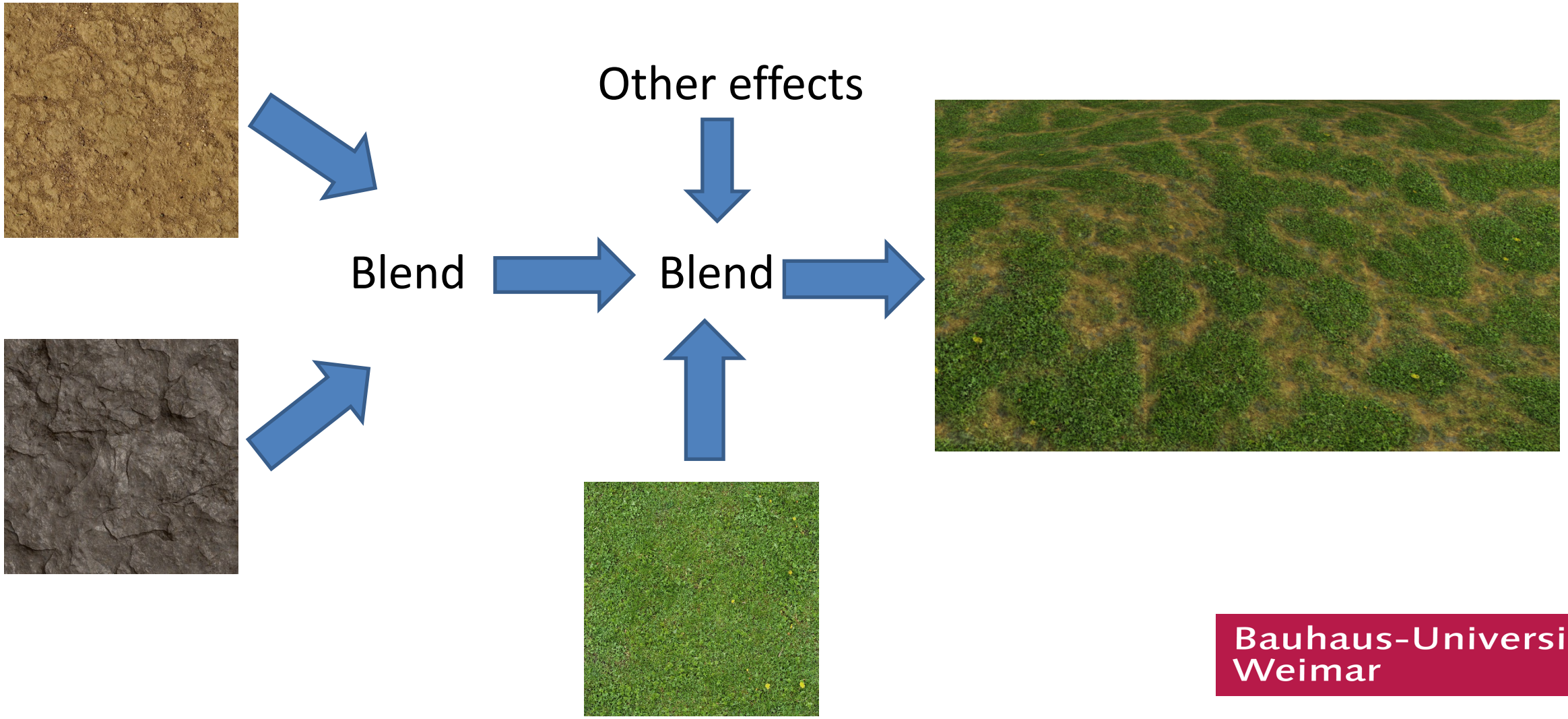


Some possible colorations

Bauhaus-Universität Weimar

Image source: http://www.upvector.com/?section=Tutorials&subsection=Intro%20to%20Procedural%20Textures

Procedural texture from images



Bauhaus-Universität Weimar

Source: http://www.proceduraltextures.com/2013/11/21/creating-a-procedural-land-texture-with-images/

Billboarding

Bauhaus-Universität Weimar

Billboards

- Object fixed in screen or world orientation → needs to be rotated every frame
- Has anchor point and own rotation matrix
  - Objects of same billboard type can use the same matrix
- Examples:

  Screen: GUIs, text, health bars, … **world**: smoke, fire, clouds, …

Bauhaus-Universität Weimar

World vs. Screen orientated Billboarding

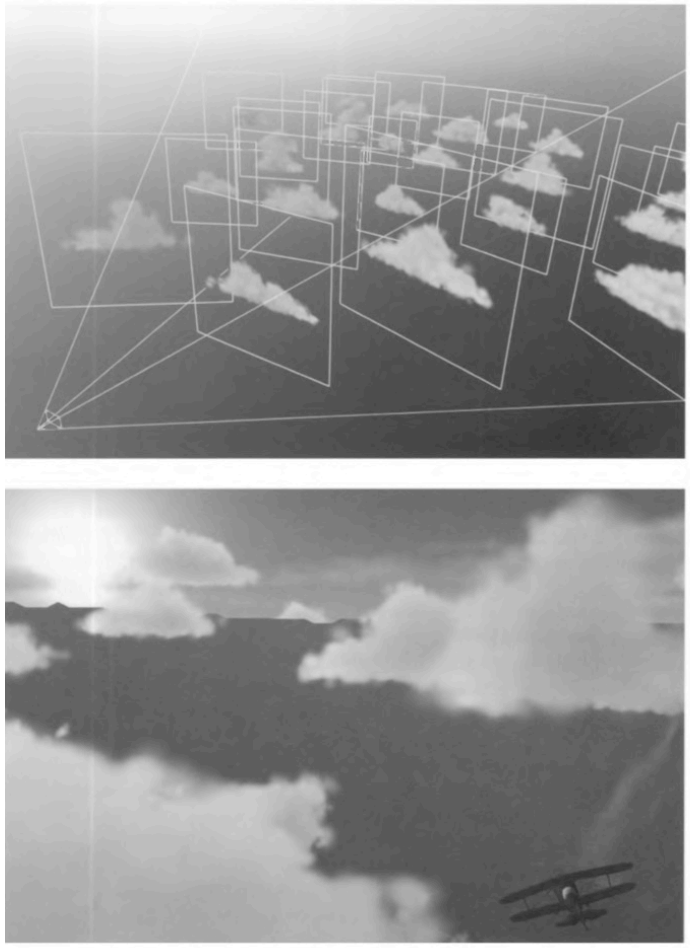


Image source: rtr_ch10, p. 451

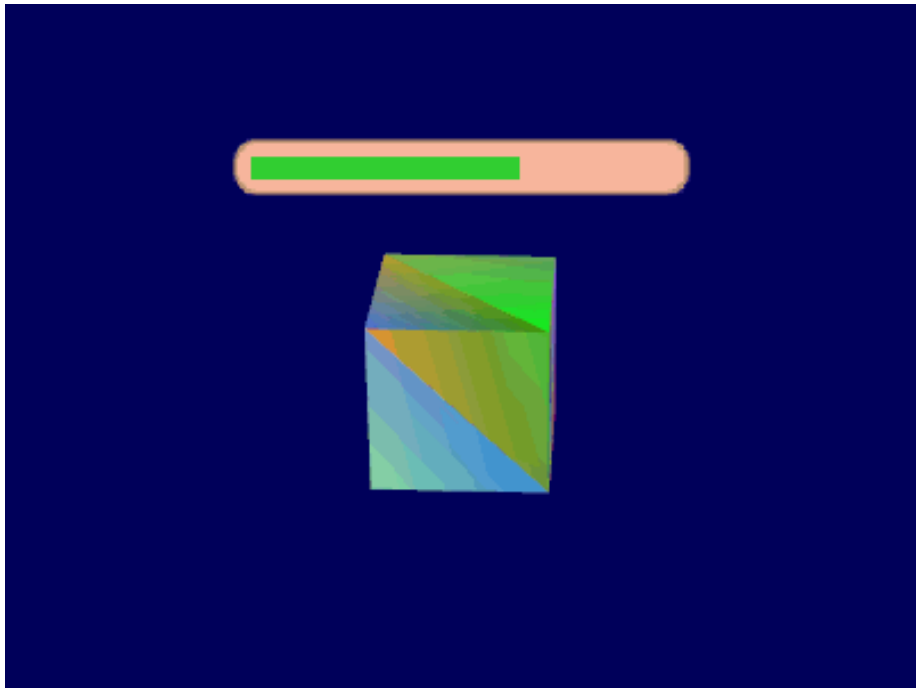


Image source: http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/billboards/

Bauhaus-Universität Weimar

Literature

- Prof. Thorsten Grosch, Vorlesung Computergrafik I, WS 16/17, TU Clausthal
- https://www.pluralsight.com/blog/film-games/bump-normal-and-displacement-maps
- http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/billboards/
- http://www.upvector.com/?section=Tutorials&subsection=Intro%20to%20Procedural%20Textures
- rtr_ch10

Bauhaus-Universität Weimar

Topic IV: Textures & Effects

Part III: Render-to-texture, Screen Space Effects

Sarah Böning

Bauhaus-Universität Weimar

Contents

- Render-to-Texture
- Screen Space Effects
  - Tone Mapping, God Rays, Depth of Field, Lense Flare, Motion Blur
- Text rendering

Bauhaus-Universität Weimar

Render-to-texture

- Map object‘s appearence of a rendered scene into texture
- Textured object is displayed quickly
- Used for a variety of effects (texture effects)
- Create empty texture → Render into it → (re)use it
- OpenGL:
  - write color in Framebuffer → texture in a shader or bind on different geometry

Bauhaus-Universität Weimar

Render-to texture Example

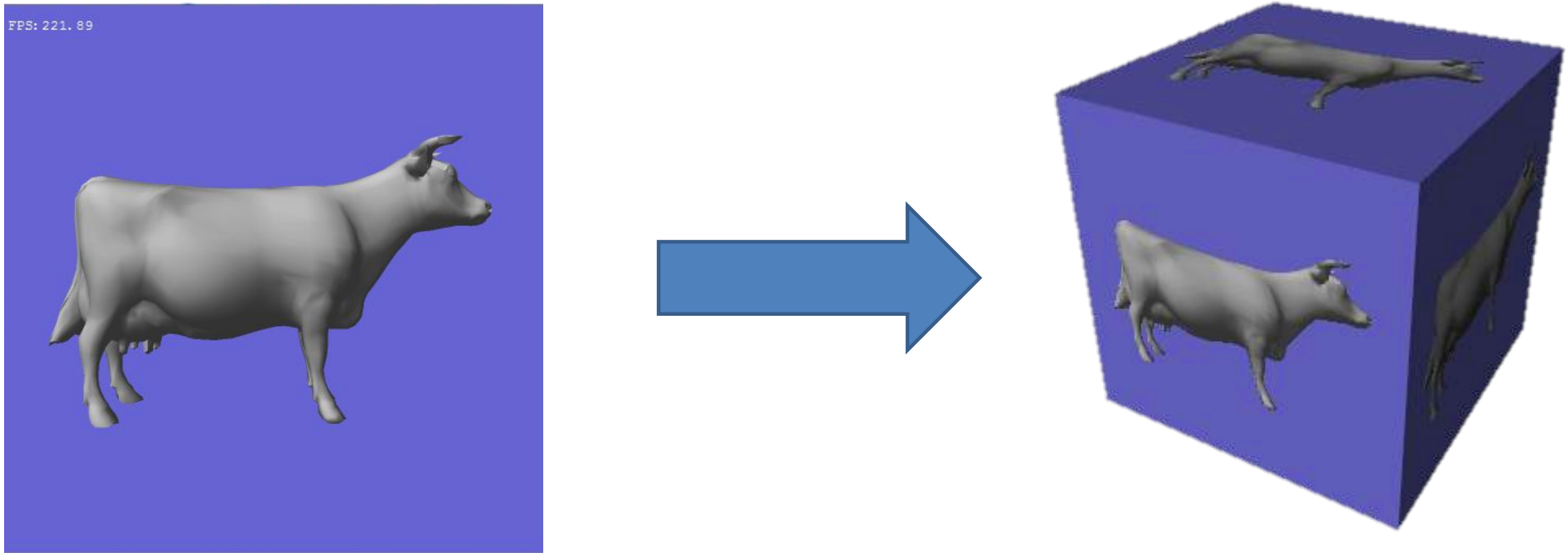


Image source: http://www.lighthouse3d.com/tutorials/opengl_framebuffer_objects/

Bauhaus-Universität Weimar

Screen Space Effects

Bauhaus-Universität Weimar

Tone Mapping

- Monitors can‘t display a wide range of light intensity
- Improves visuals, add effects
- Sample computed image to determine luminance levels

  → remap image based on results

- Example: Step out from tunnel →very bright at first, eyes need to adjust, going back to normal

Bauhaus-Universität Weimar

Image source: http://www.adriancourreges.com/blog/2015/11/02/gta-v-graphics-study/

Image source: https://i.stack.imgur.com/uCvQb.jpg

Bauhaus-Universität Weimar

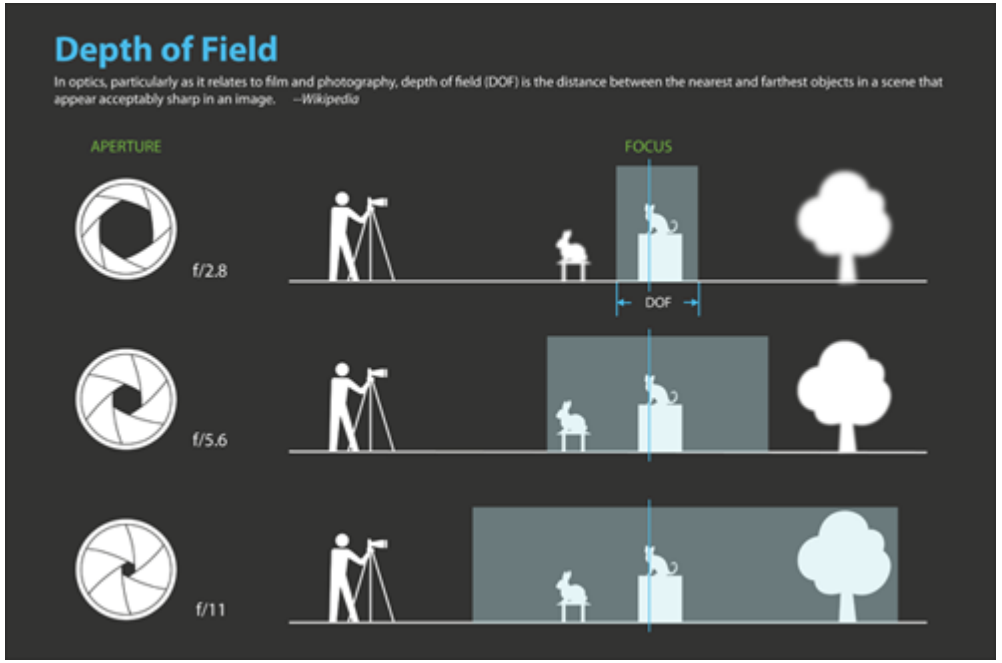
Depth of Field

Image source: https://gurushots.com/article/11-dynamic-images-that-demonstrate-the-power-of-shallow-depth-of-field

Bauhaus-Universität Weimar

Depth of Field in theory

- Real cameras/eyes have finite area of focus



- Used as effect in photography & video games, …

Bauhaus-Universität Weimar

Image source: http://www.thehurlblog.com/depth-field-character-story/

Depth of Field in pratice

- Render scene multiple times with slightly different MVP matrix
- Blend results together

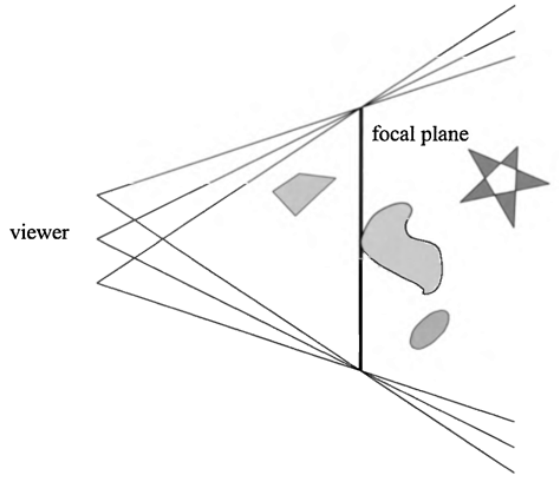


Image source: rtr_ch10

- Trade-off: effect vs rendering time

- OpenGL: Accumulation Buffer

Bauhaus-Universität Weimar

Other Screen Space Effects

- God Rays (crespular rays)
  - Physics: *Scattering process* - Rays get directed to viewer by particles (dust, fog, etc.) or water in the atmosphere
  - Calculated in Fragment Shader

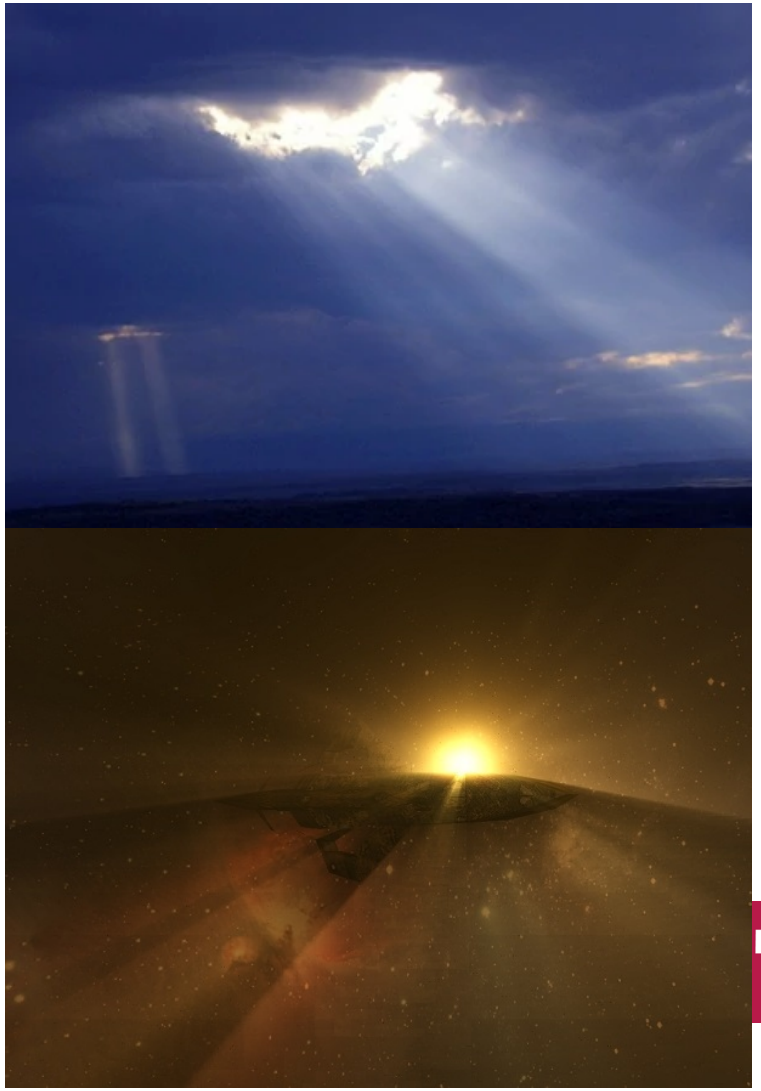


Image sources:
http://uncyclopedia.wikia.com/wiki/File:God_Rays.jpg
https://entwickler.de/online/opengl-weltraumeffekte-mit-motion-blur-lens-flare-co-how-to-114796.html

Other Effects

- Lense Flare
  - Very bright light reflection in lense
  - Simple implementation: billboards along reflection axis
  - Newer: calculated in fragment shader

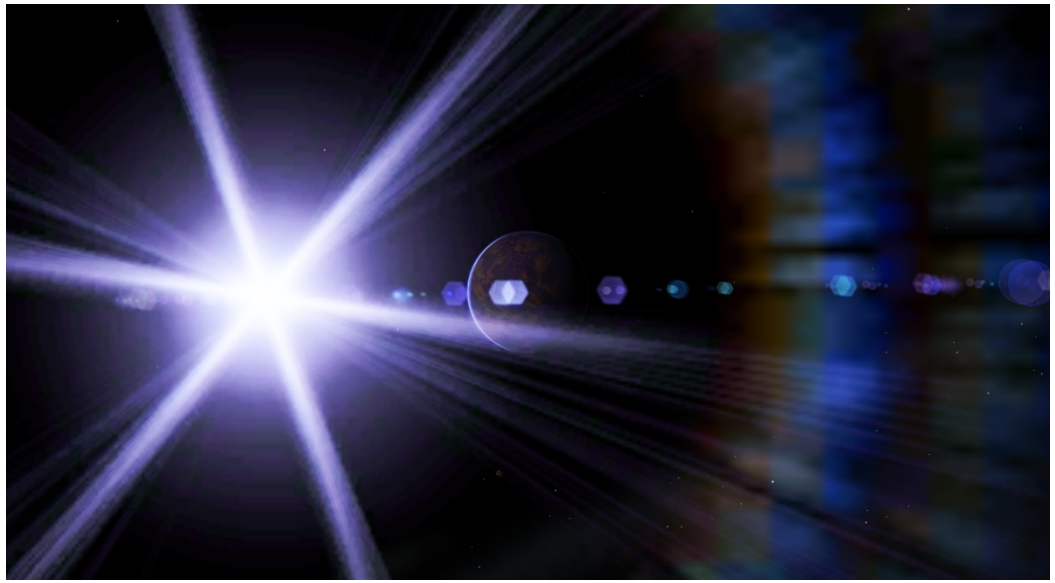


Image source: http://i.imgur.com/7pDBQ.jpg

Bauhaus-Universität Weimar

Other Effects

- Motion blur
  - Smoother motions (but: motion sickness!)
  - Add scene from previous frame to current scene
  - Set intensity

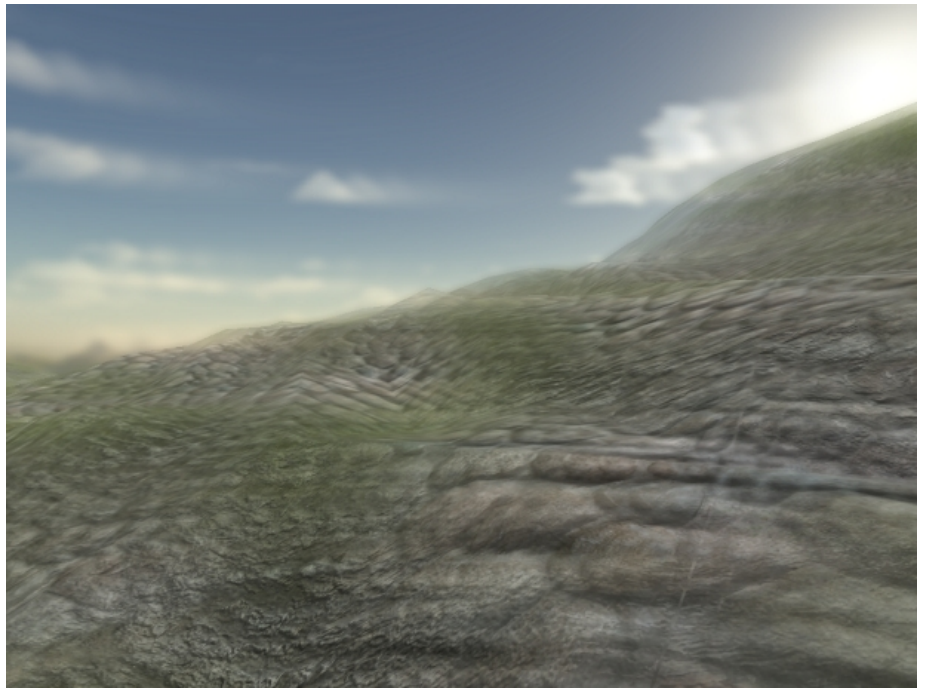


Image source: https://entwickler.de/online/opengl-weltraumeffekte-mit-motion-blur-lens-flare-co-how-to-114796.html

Bauhaus-Universität Weimar

Text Rendering

- Bitmap fonts
  - Extract all needed letters of a font into one texture
  - Every glyph has texture coordinates
  - Select corresponding glyph to render
  - Rendered on quad without background
  - Simple & easy, but not flexible

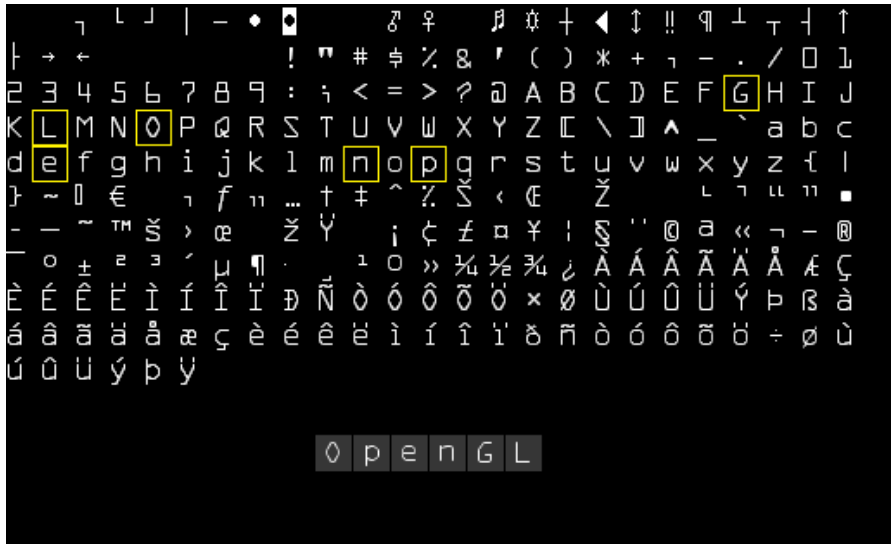
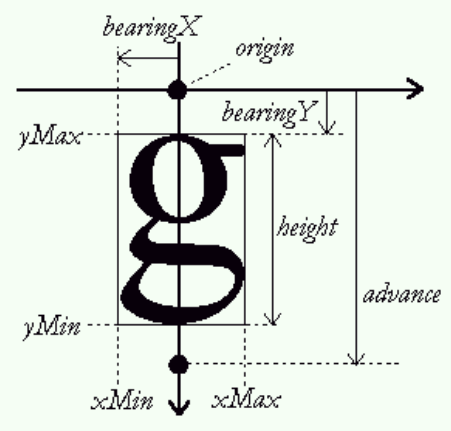


Image source: https://learnopengl.com/In-Practice/Text-Rendering

Bauhaus-Universität Weimar

Text Rendering

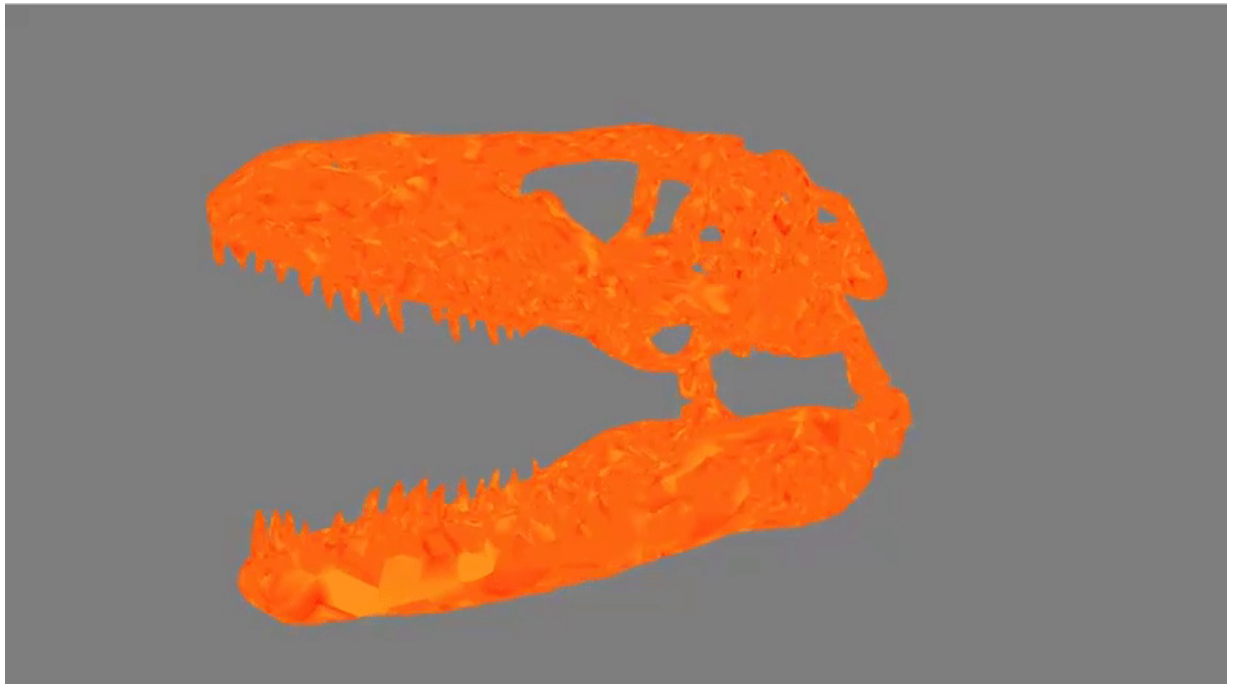
- FreeType Rendering
  - FreeType library
  - Loads TrueType fonts & creates bitmap textures for each glyph
  - Set different parameters like font size & height
  - Render in shader



Bauhaus-Universität Weimar

Image source: https://www.freetype.org/freetype2/docs/tutorial/step2.html

That‘s it for textures and effects!



Next week: slide-in presentation on Shading/Lighting

Bauhaus-Universität Weimar

Literature

- rtr_ch10
- http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-14-render-to-texture/
- http://www.lighthouse3d.com/tutorials/opengl_framebuffer_objects/
- https://entwickler.de/online/opengl-weltraumeffekte-mit-motion-blur-lens-flare-co-how-to-114796.html
- https://learnopengl.com/In-Practice/Text-Rendering

Bauhaus-Universität Weimar

Topic III: Shading and Lighting

Sarah Böning

Bauhaus-Universität Weimar

Contents

- Small revision on lighting
- Lighting in Shaders
- Light sources
- Environment- & Shadow maps

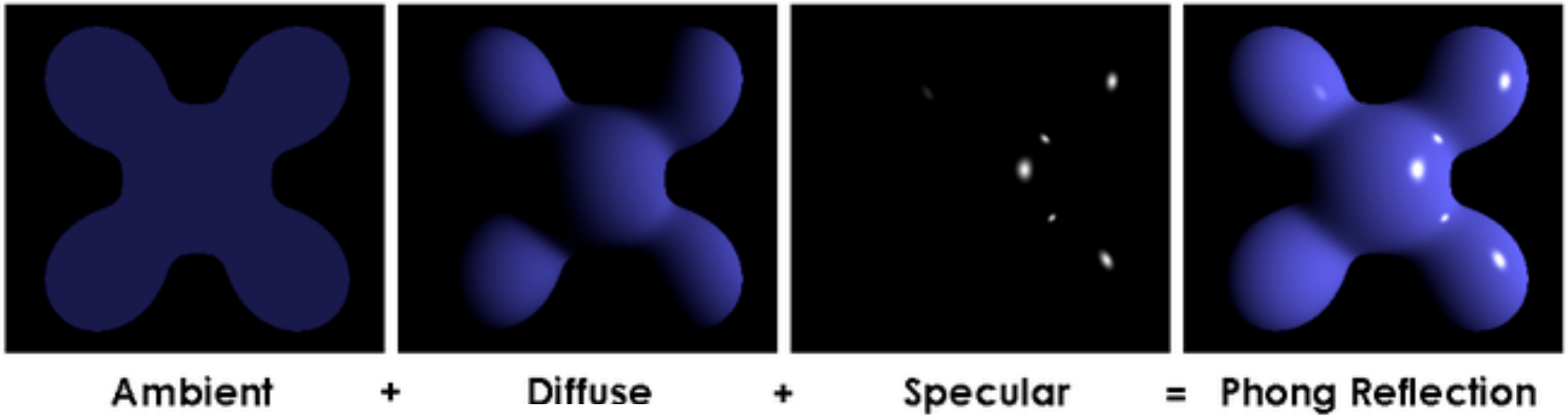
Bauhaus-Universität Weimar

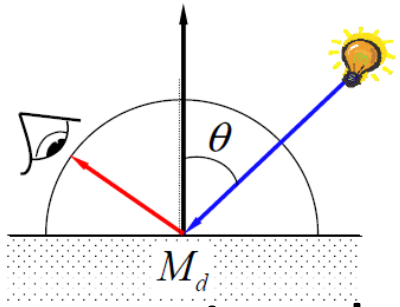
Shading

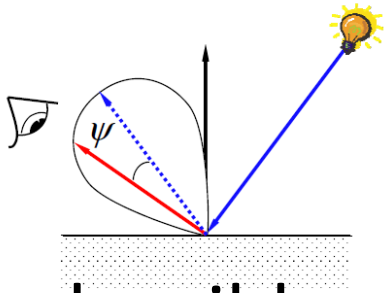
Bauhaus-Universität Weimar

Phong Shading

Image sources: https://en.wikipedia.org/wiki/Phong_shading, Prof. T. Grosch







- No physical correctness, just looks plausible

Bauhaus-Universität Weimar

Phong Reflection Model

$$L = M_d * L_a + \sum_{\#\ Lightsources} (M_d\ * \cos\theta_i + M_{sp} * cos^n\psi_i) * L_{rgb}$$

- L_{rgb} often white
- Light is additive: If L > 1 => clamp to 1
- Need (normalized) normal for every surface point (Vertex → Fragment Shader)

Bauhaus-Universität Weimar

Shading in Fragment/Vertex Shader

- Vertex: Light for every vertex, light gets interpolated
- Fragment: Normals get interpolated, light for every pixel

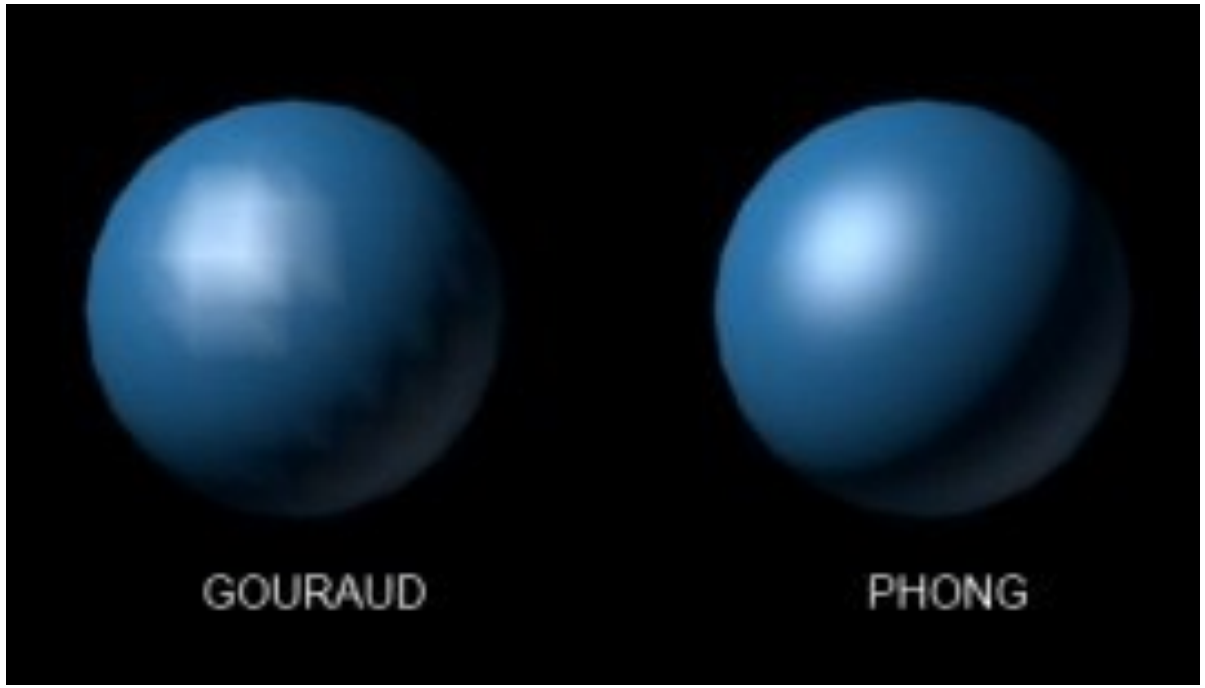


Image Source: https://www.quora.com/What-is-an-explanation-of-the-Gouraud-shading-and-phong-shading-models-in-simple-form

Bauhaus-Universität Weimar

Light Sources

- Directional light
  - Light source in infinte distance
  - Rays strike objects from parallel direction
  - Equal intensity
  - Example: sun

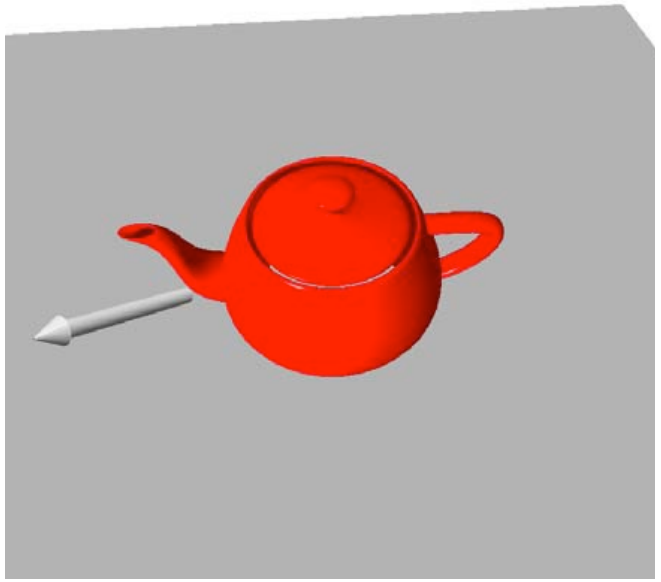


Image Source: Prof. T. Grosch

Bauhaus-Universität Weimar

Light sources

- Point light
  - Gives off light in equal amounts in all directions
  - Objects closer appear brighter
  - Example: Light bulb hanging from a cord

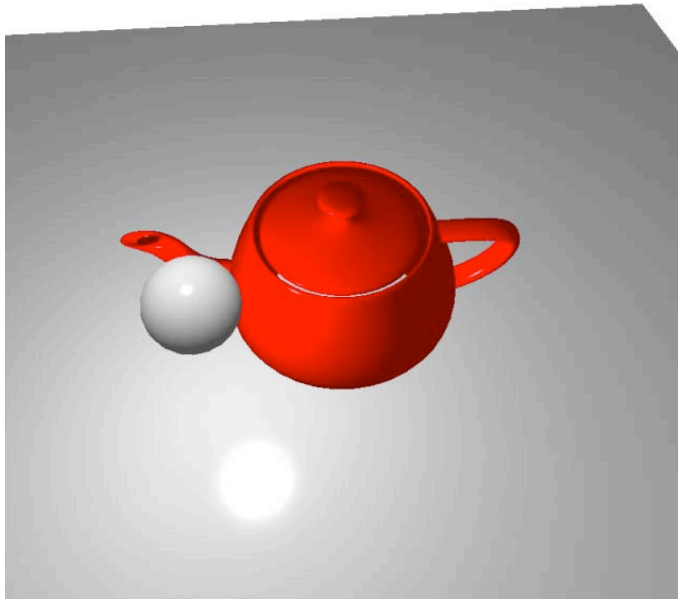


Image Source: Prof. T. Grosch

Bauhaus-Universität Weimar

Light sources

- Spotlight
  - Radiates light in a cone / only in specified angle
  - More light in the center
  - Falloff function: bright in center, soft edges
  - Example: Flashlight, car headlight

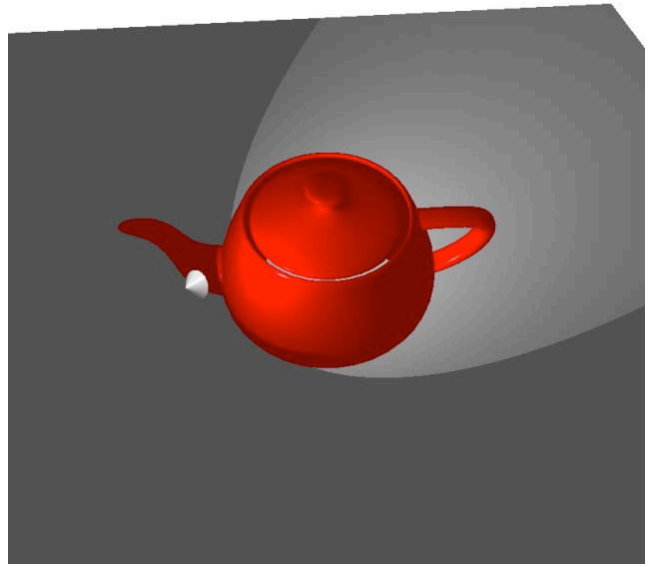


Image Source: Prof. T. Grosch

Bauhaus-Universität Weimar

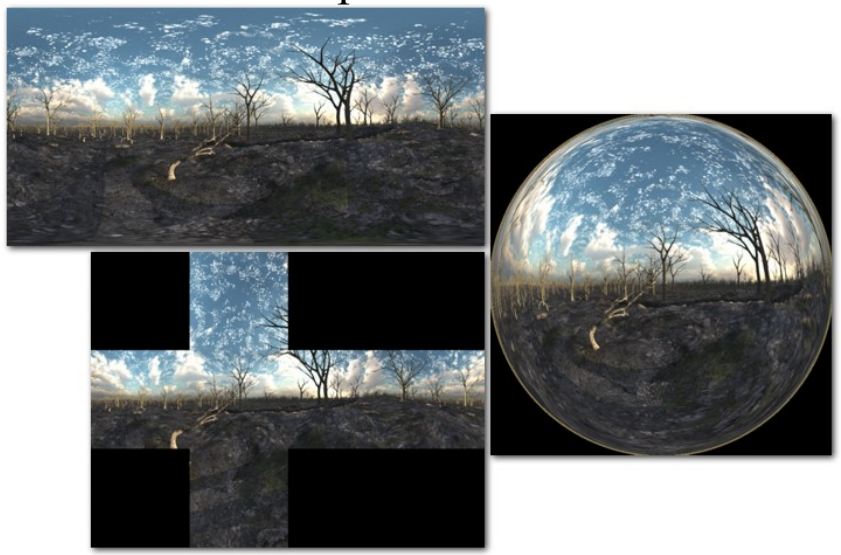
More Maps

Bauhaus-Universität Weimar

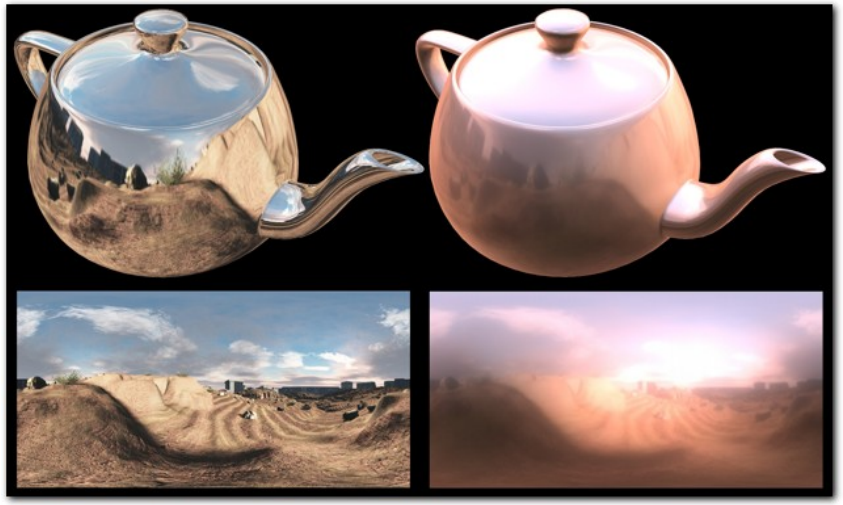
Enviroment Maps

- Reflection of enviroment on an object
- Enviroment as a texture → dynamic calculation of reflection with map

Cube Map



Sphere Map



Bauhaus-Universität Weimar

Image Source: http://spiralgraphics.biz/genetica/help/index.htm?environment_maps_explained.htm

Shadow Maps

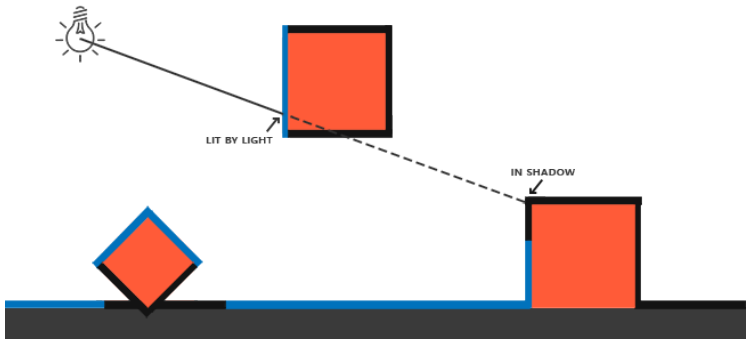


Image Source: https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

1. Generate Shadow Map
   - Render scene from light‘s point of view
   - Store depth values of visible objects in texture
   - Calculate map every time the postion of light/object changes
2. Draw shadow
   - Render scene from camera‘s POV
   - For every pixel: transform in light‘s coordinate space
   - Compare pixel‘s depth with depth from map
   - Pixel‘s depth >/< map depth →shadow /light

Bauhaus-Universität Weimar

Sources

- Prof. Thorsten Grosch, Vorlesung Computergrafik I, WS 16/17, TU Clausthal
- https://learnopengl.com/Advanced-Lightin/Shadows/Shadow-Mapping
- https://www.cs.uic.edu/~jbell/CourseNotes/ComputerGraphics/LightingAndShading.html

Bauhaus-Universität Weimar