# Textures and Shadows

Let's separate light from the darkness!

Elmar Eisemann
Delft University of Technology (TU Delft)

## Shading

How to transform

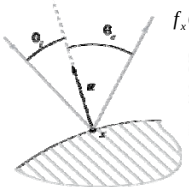
in

?

## Material and light transmission

**BRDF** (*Bidirectional Reflectance Function*)

Function from $IR^4$ in IR indicating how incident light is reflected in a point (ratio between in- and outgoing)



$$f_x(\theta_i, \theta_o, \lambda)$$

$\theta_i$ : incoming light
$\theta_o$ : outgoing light
$\lambda$ : wavelength

## Different Shading Models

- Standard Shading Models:
  - Ambiant
  - Diffuse (Lambertian)
  - Specular (Phong, Blinn-Phong)

## ...but something still looks odd...

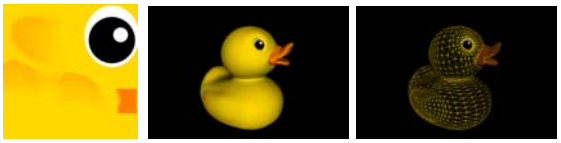
**Misses quick material changes**

## Example

- Skyrim, Bethesda

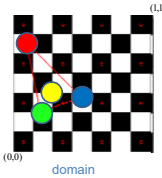
## Textures

- Old description:
  - An image drawn on top of your surface



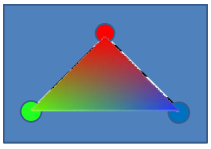
## Textures

- A discrete scalar field on a surface
  - Mapped on the surface via texture coordinates
    - Specified at each vertex `glTexCoord{123}{fi}`
    - Interpolated over triangles




domain

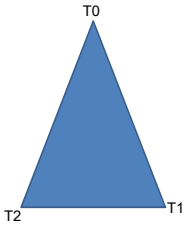
```
glBegin(GL_TRIANGLES);
 glTexCoord2f(0.2f,0.3f);
 glVertex2f(-1.0f,0.0f);
 glTexCoord2f(0.5f,0.4f);
 glVertex2f(+1.0f,0.0f);
 glTexCoord2f(0.1f,0.8f);
 glVertex2f( 0.0f,1.0f);
glEnd();
```

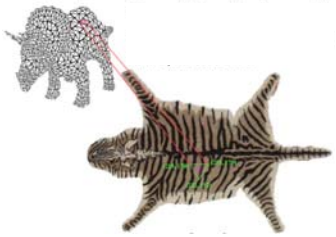


3D View

## How to define Texture Coordinates?

- We will provide tex coords by hand!



Texture coordinates are handled like attributes and interpolated over triangle!!!

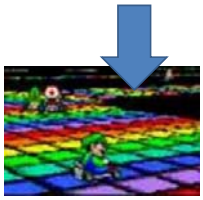
## Questions?


## Textures

- Extremely useful and efficient!
- What are the problems?

## Texture issues…

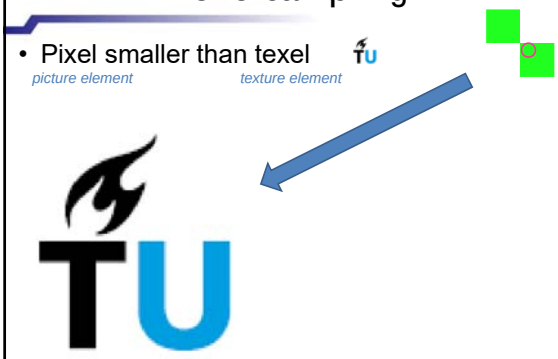
- Not always completely beautiful…

## Oversampling

- Pixel smaller than texel

*picture element* *texture element*

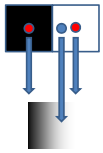


Nearest Neighbor

## Oversampling

- Idea: Linear interpolation between texels

Center position: A B and call the associated texel colors col(A) and col(B)



For a position P between A and B, we define the color at P as:

alpha := (P-A)/(B-A)

Then col(P) = (1-alpha) * col(A) + alpha * col(B)

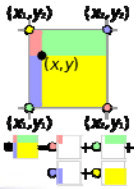
Try out:
A=254, B=255, Col (A) = 0.5 , Col (B)=0.9, P= 254.5
P (corresponding to a screen pixel center) needs the color exactly between texel A and B.

## Oversampling

- Bilinear interpolation
  - Linear interpolation: alpha * col1 + (1-alpha) * col2

  - Bilinear interpolation:

Linear in x direction  Linear in y direction




## Oversampling

- Pixel smaller than texel

*picture element* *texture element*

Bilinear interpolation

Nearest Neighbor  Bilinear Interpolation

## Oversampling

- Example:

Call of Duty - Modern Warfare 2

- For nearby elements: more resolution = better

## Texture issues… part 2

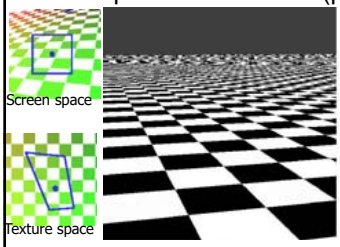
- But far away, you get undersampling

?

Why are the lines not continuous?

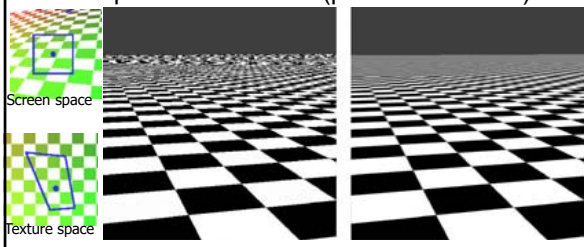
## 2. Undersampling

- One screen pixel does not necessarily correspond to one texel (pixel in a texture).



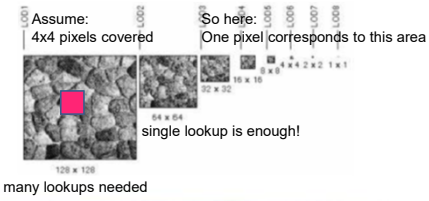

## 2. Undersampling

- One screen pixel does not necessarily correspond to one texel (pixel in a texture).



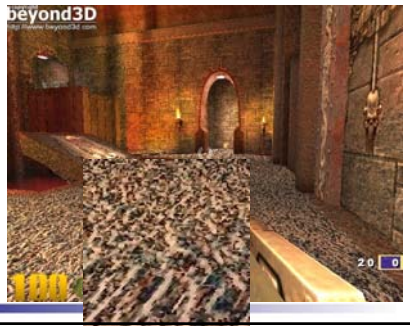

## MipMapping: Approximate Filtering

- Precompute *Mipmap pyramid:*
  - *Hierarchical texture*
  - *Reduce resolution by 2x2 on each level*
  - *Filter « appropriately » (usually average of 4 pixels)*

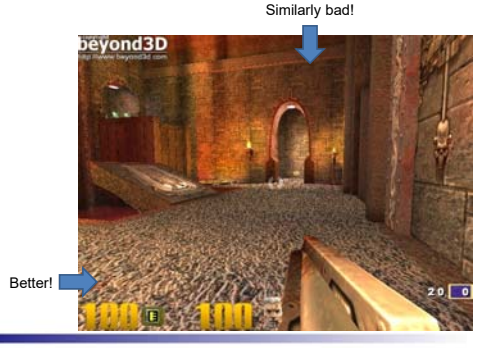




## MipMapping: OFF

- Just Nearest Neighbor looks noisy

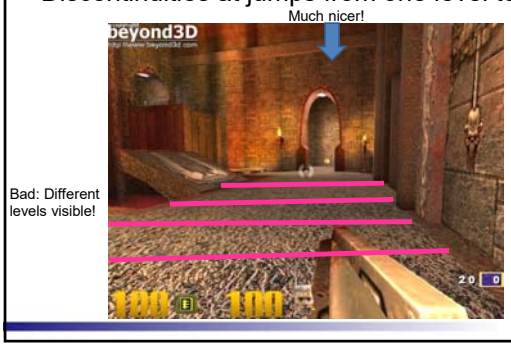



## MipMapping: Off + Linear Filtering




## MipMapping On (use nearest level)

- Discontinuities at jumps from one level to next



## MipMapping: TriLinear Filtering

- Blend (mix) between different mipmap levels

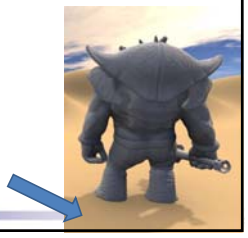
Beautiful!


## Texture Summary

- Attach a detailed image to a surface via texture coordinates
- Efficient filtering solutions: near use linear, far use MipMaps

## Textures

- Can be used for many things… including shadows.

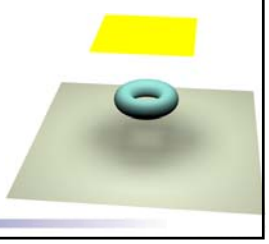


## Questions?


## What is a Shadow?

- WordNet:

*Shade within clear boundaries*
*or*
*An unilluminated area.*

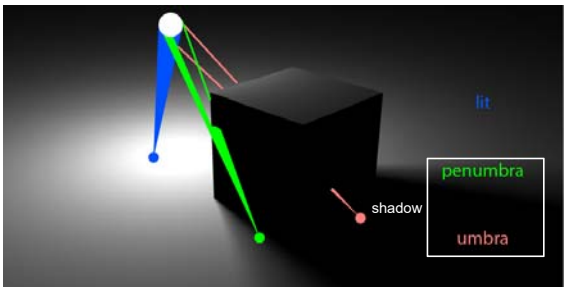


## What is a Shadow?

- Hasenfratz et al. [2003]:

*Shadow [is] the region of space for which at least one point of the light source is occluded.*

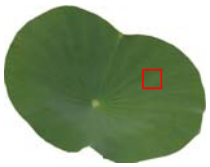
## What is a Shadow?




## What is a Shadow?

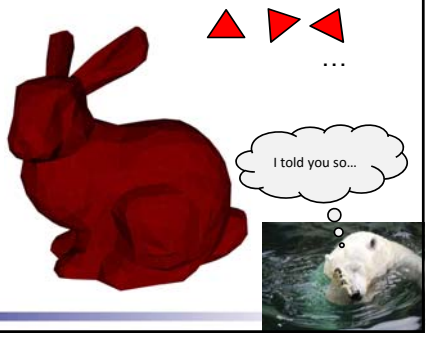
- Hasenfratz et al. [2003]:

*Shadow [is] the region of space for which at least one point of the light source is occluded.*




## Our models are simple…

- Typically a list of triangles




## What is a Shadow?

- Hasenfratz et al. [2003]:

*Shadow [is] the region of space for which at least one point of the light source is occluded.*

## How to draw shadows?

- Artists know how to draw shadows!
  Or not?

Fra Carnevale (1467)

## How to draw shadows?

- Artists know how to draw shadows!
  Or not?

Signorelli (1488)

## How to draw shadows?

- Artists know how to draw shadows!
  Or not?

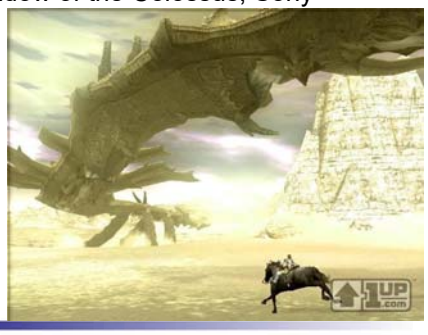
Moore
(1893)

## How to draw shadows?

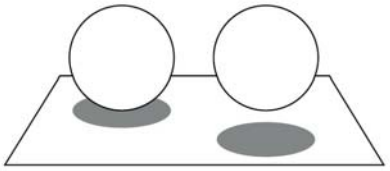
- Drawing shadows is apparently difficult…

## So why not just ignore shadows?

- Shadow of the Colossus, Sony

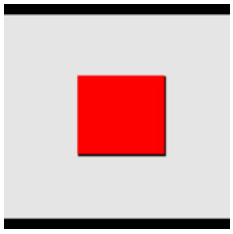


## So why not just ignore shadows?



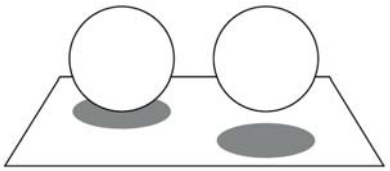
## Psychophysical-Experiments

[Kersten et al. 96]

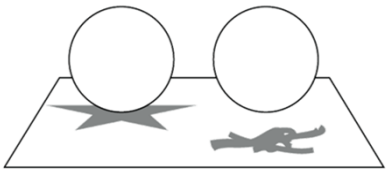


## So why not just ignore shadows?

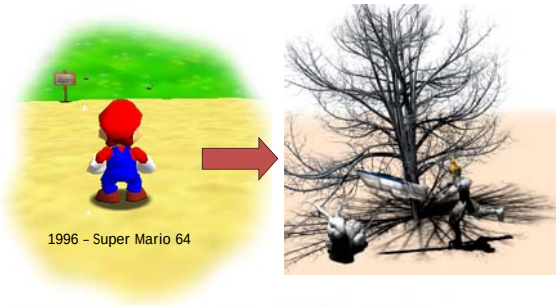
- But this is not a good argument for realistic shadows…



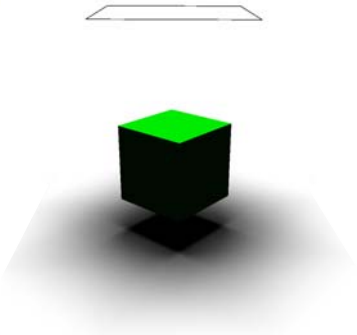
## So why not just ignore shadows?



## Simple shadows can be sufficient

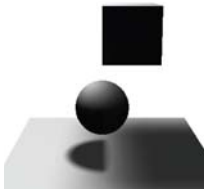



## Plausible shadows



## Plausible shadows

- Attention: “plausible” can often fail



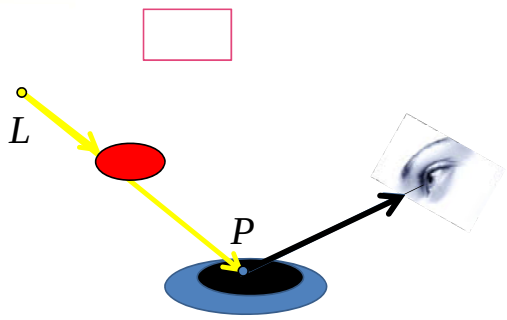
## Realistic shadows are important

## Realistic shadows are important

- Many contexts in which accuracy is needed:
  - Architecture
  - Simulation
  - Movies
  - …

Desert villa by Studio Aiko

## How to compute shadows?




## Hard Shadows

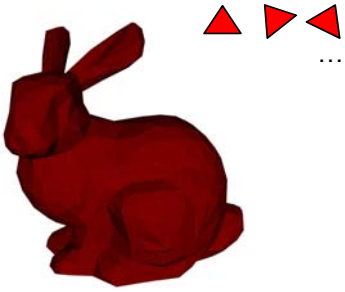
Point lights do not create penumbrae

One way to calculate: shoot rays to the light…

## How to accelerate the process?

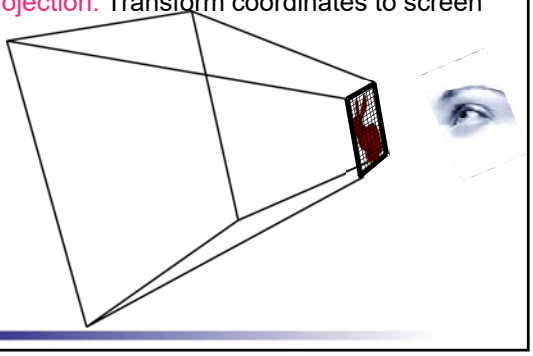

## Simplified Graphics Pipeline

- Models are typically lists of triangles



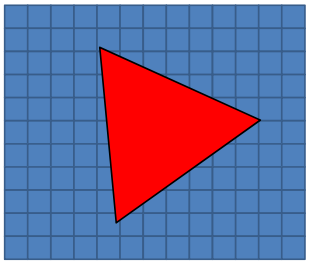

## Simplified Graphics Pipeline

- Projection: Transform coordinates to screen



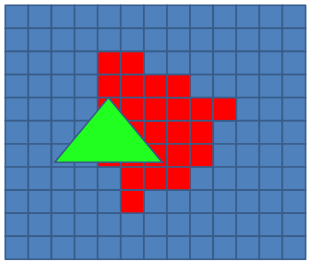
## Simplified Graphics Pipeline

- Rasterization: Fill screen pixels



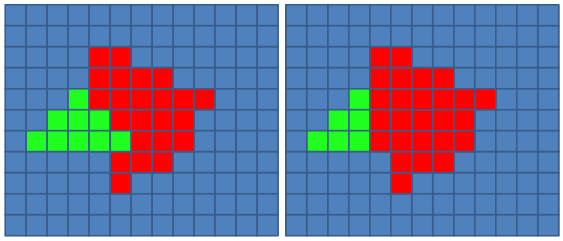
## Simplified Graphics Pipeline

- Catch: Let's look at a second triangle…



## Simplified Graphics Pipeline

- Catch: Drawing order changes result

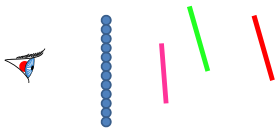


Need to keep nearest pixels

## Simplified Graphics Pipeline

[Catmull74] , [Strasser74]

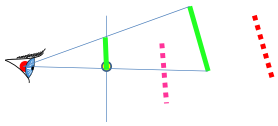
- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



## Simplified Graphics Pipeline

[Catmull74] , [Strasser74]

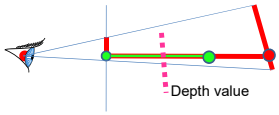
- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



## Simplified Graphics Pipeline

[Catmull74] , [Strasser74]

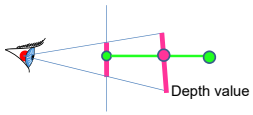
- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel




## Simplified Graphics Pipeline

[Catmull74] , [Strasser74]

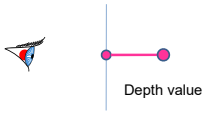
- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



## Simplified Graphics Pipeline

[Catmull74] , [Strasser74]

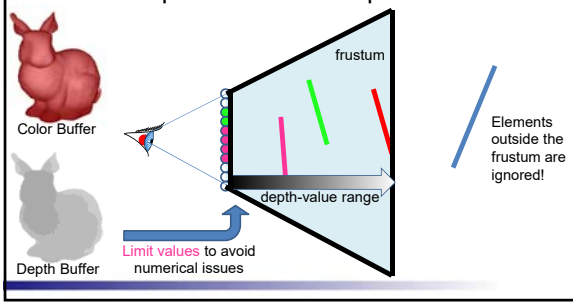
- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel




## Simplified Graphics Pipeline

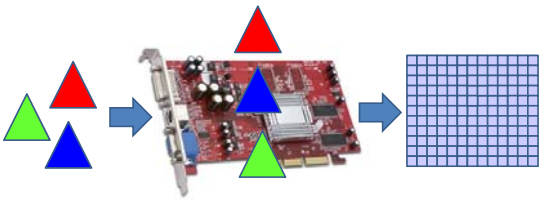
[Catmull74] , [Strasser74]

- Depth Buffer: Avoid sorting triangles!
- Store a depth value in each pixel




## Simplified Graphics Pipeline

- Highly parallelizable
  ➡ Graphics Processing Units (GPUs)



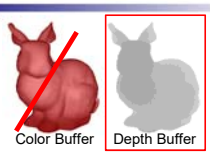
## Simplified Graphics Pipeline

- Local computations:
- Processor only knows its current triangle

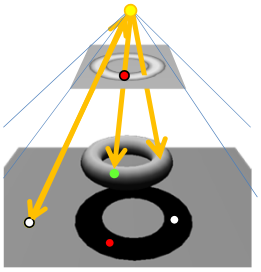
  this is NOT enough for shadows

- Naïve solution:
  Loop over all triangles
  for each pixel is too expensive

## Shadow Mapping [Williams78]




1. render view from light
**use depth buffer**



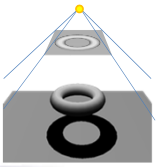
2.render from viewpoint

## Shadow Mapping [Williams78]

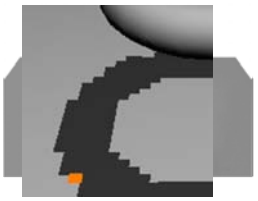
1. Render depth buffer from light (**Shadow Map**)
   – Store result in a texture

2. Render from viewpoint – activate a fragment shader
   – For each drawn pixel:
     Compare its distance
     to corresponding distance
     in the Shadow Map
     - Equal: pixel is lit
     - Farther: pixel is in shadow

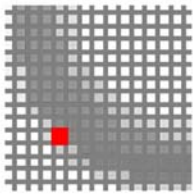
Resolution mismatch



## Problem 1: Discretization



from viewpoint

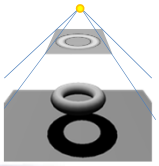


from light

Lord of the Rings
Electronic Arts

## Shadow Mapping [Williams78]

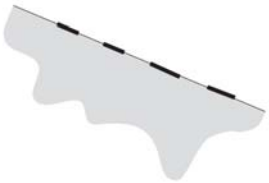
1. Render depth buffer from light (**Shadow Map**)
   - Store result in a texture
2. Render from viewpoint – activate a fragment shader
   - For each drawn pixel:
     Compare its distance
     to corresponding distance
     in the Shadow Map
     - Equal: pixel is lit
     - Farther: pixel is in shadow



## Problem 2: Depth Bias

Too small Too big

- Self-shadowing
  - Discretisation
  - Limited precision
- Solution:
  add an offset,
  but not too much!



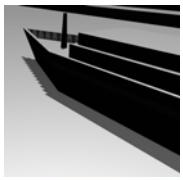
## Problem 2: Depth Bias

- "Real-world" example in Crysis by Crytek

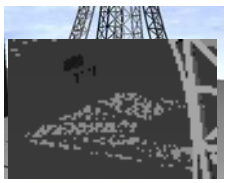
## Problem 3:

- Does this ring a bell?

Reconstruction
- Staircase artifacts

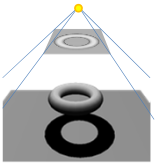


Oversampling
- No bandlimiting



## Shadow Mapping [Williams78]

- Simple, efficient, and easy to implement
- Compatible to most object representations
- Additional hardware support
- Variants are common
  - (games, movies…)



## Conclusion

- Textures and filtering methods
- Introduction to shadows
- Shadow Mapping (texture-based shadows)
  - Discussion of shortcomings

- QUESTIONS???


## A few sample exam questions

## Mid-Term

Example Questions:

1) Open: Discuss briefly the downsides of the Graphics Pipeline (Rasterization)

2) Closed: Explain why points at infinity are not influenced by translations

3) Closed: Given point P=… and Matrix M=… compute MP. What kind of transformation is this?

## Mid-Term

4) Thinking: Shortened question – compare practical

A mesh is represented via 3 arrays: 3Dvertices, 2D texcoords, and an index array, in which 3 consecutive indices define a triangle.

Why is such a representation usually more efficient in memory than storing each triangle with 3 vertices and 3 2D texcoords?

For what models would it not matter?

## Mid-Term

6) Transfer: We want to achieve a comic look for our character, i.e., black when faces are opposing the light, white when they don't.

Define a function (pseudo code) that returns a color (RGB float triplet) and which takes normal, light position, and vertex position as an input.

```
f(normal, vertexpos, lightpos){
        …
        return (r,g,b);
}
```

## Thank you very much for your attention!