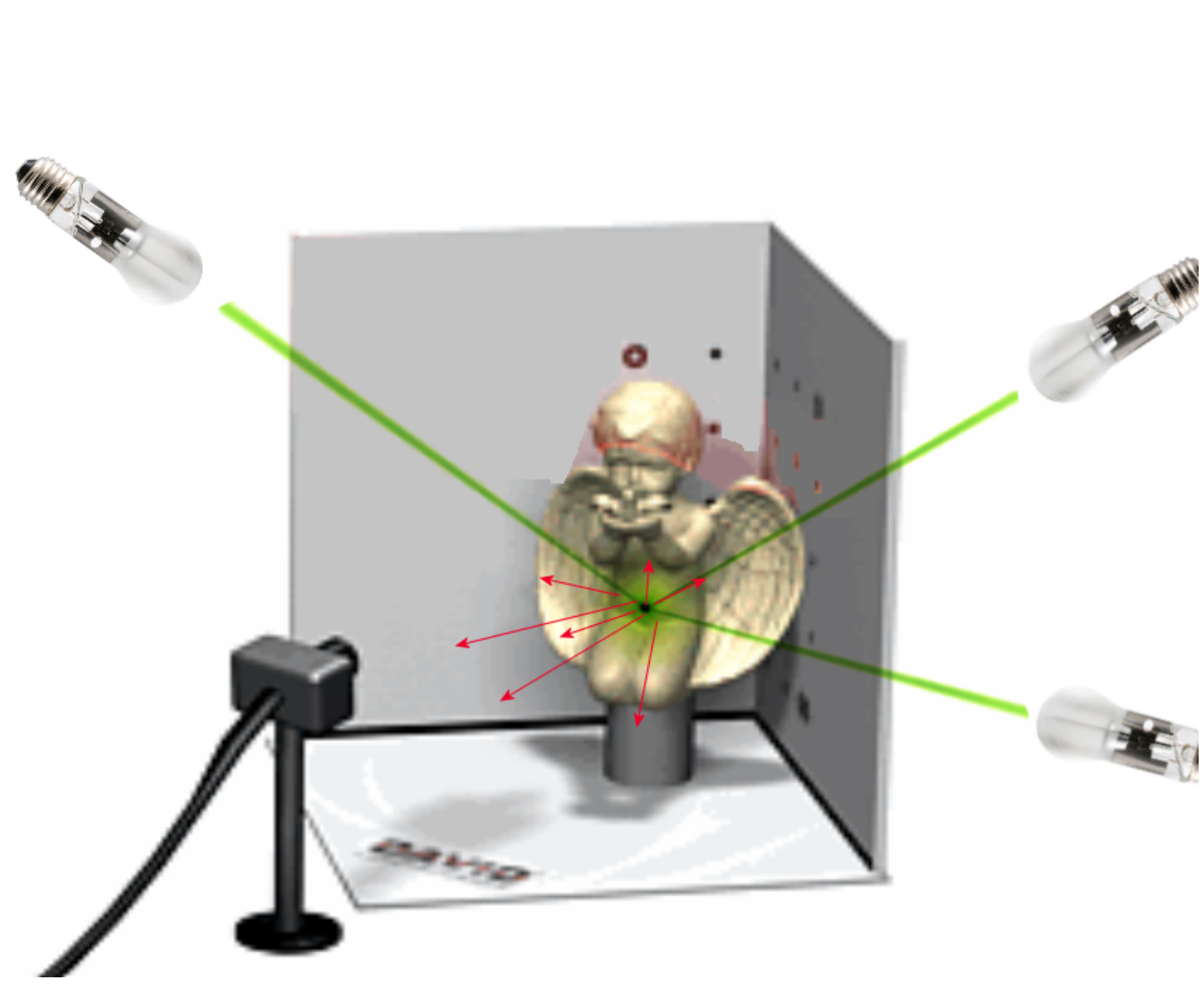
Project: BRDF Estimation on Objects

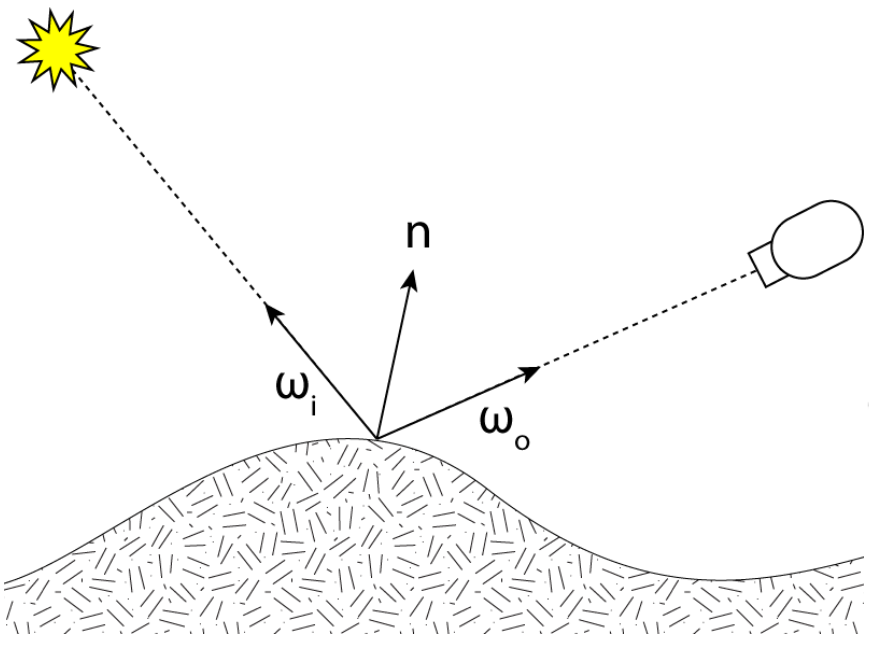
February 17th, 2012

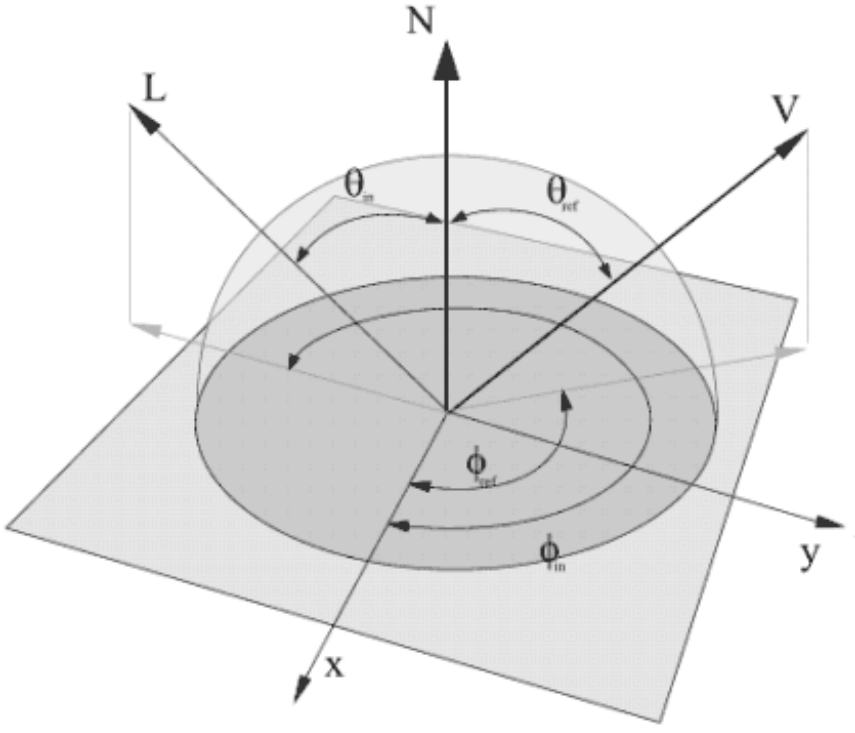
Rapid Prototyping
TU-Berlin
Christian Thurow & Christopher Leiste

Idea: 3D model + BRDF of material + texture = realistic representation



Bidirectional-Reflectance-Distribution





BRDF Models

Phong-Model

$$I = kd \cdot cos(\phi) + ks \cdot \frac{n+2}{2\pi} \cdot cos^n(\theta)$$

Blinn-Phong-Model

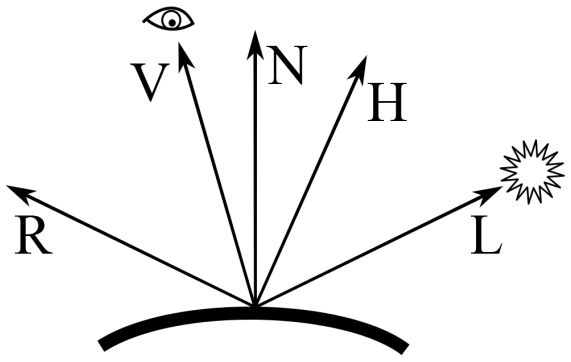
$$I = kd \cdot cos(\phi) + ks \cdot cos^n(\theta')$$

"Our Model"

$$I = kd \cdot cos(\theta') + ks \cdot cos^n(\theta')$$

Angular Relations

$$\phi = \vec{L} \cdot \vec{N}$$
$$\theta = \vec{R} \cdot \vec{V}$$
$$\theta' = \vec{N} \cdot \vec{H}$$



Review

What we have:

- Laserscanner, provides: 3D Model of Object
- Webcam, provides: Images of Object

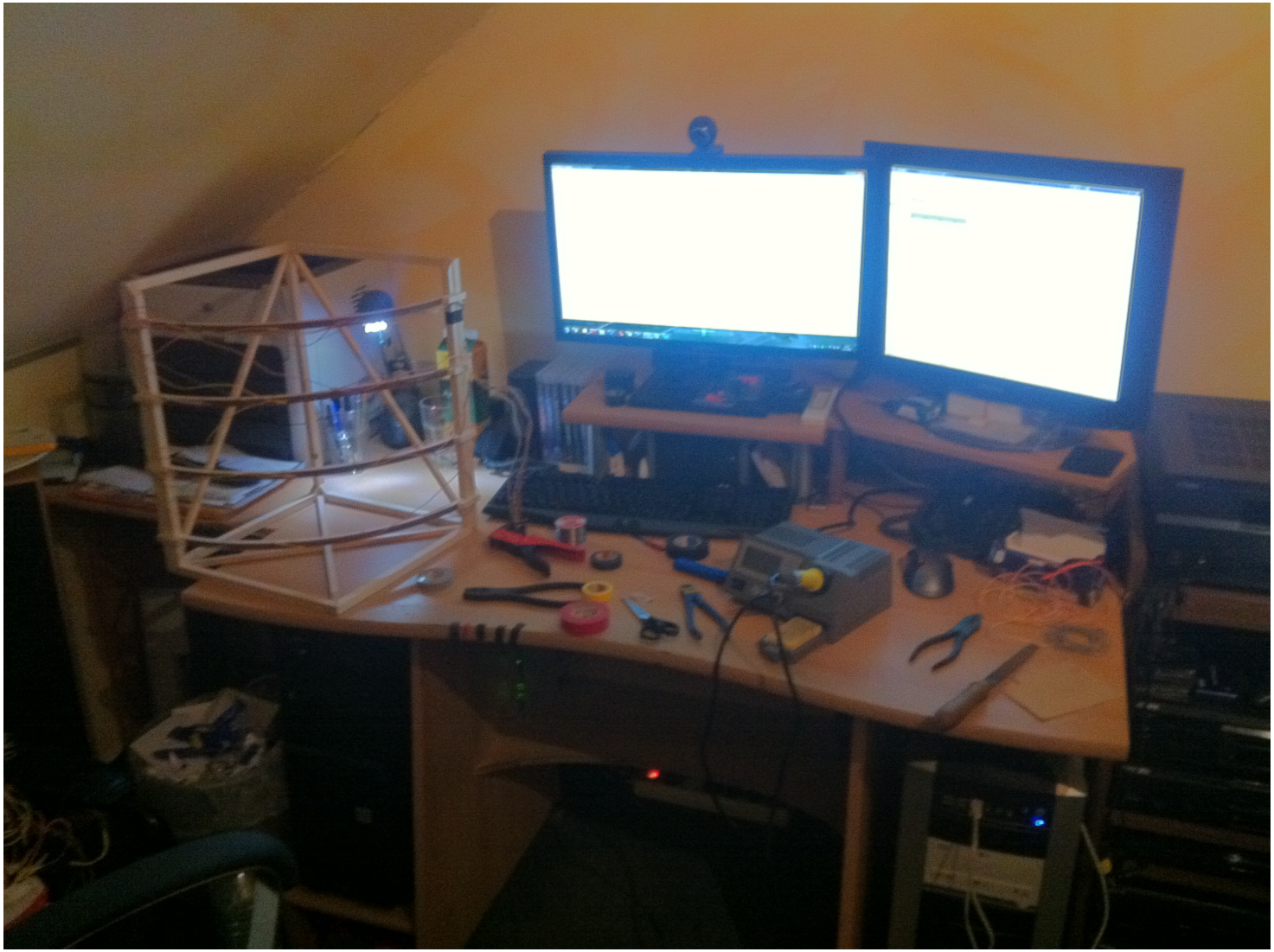
What we need:

- movable light source or distributed light sources
- program that calculates BRDF function on each point on Mesh

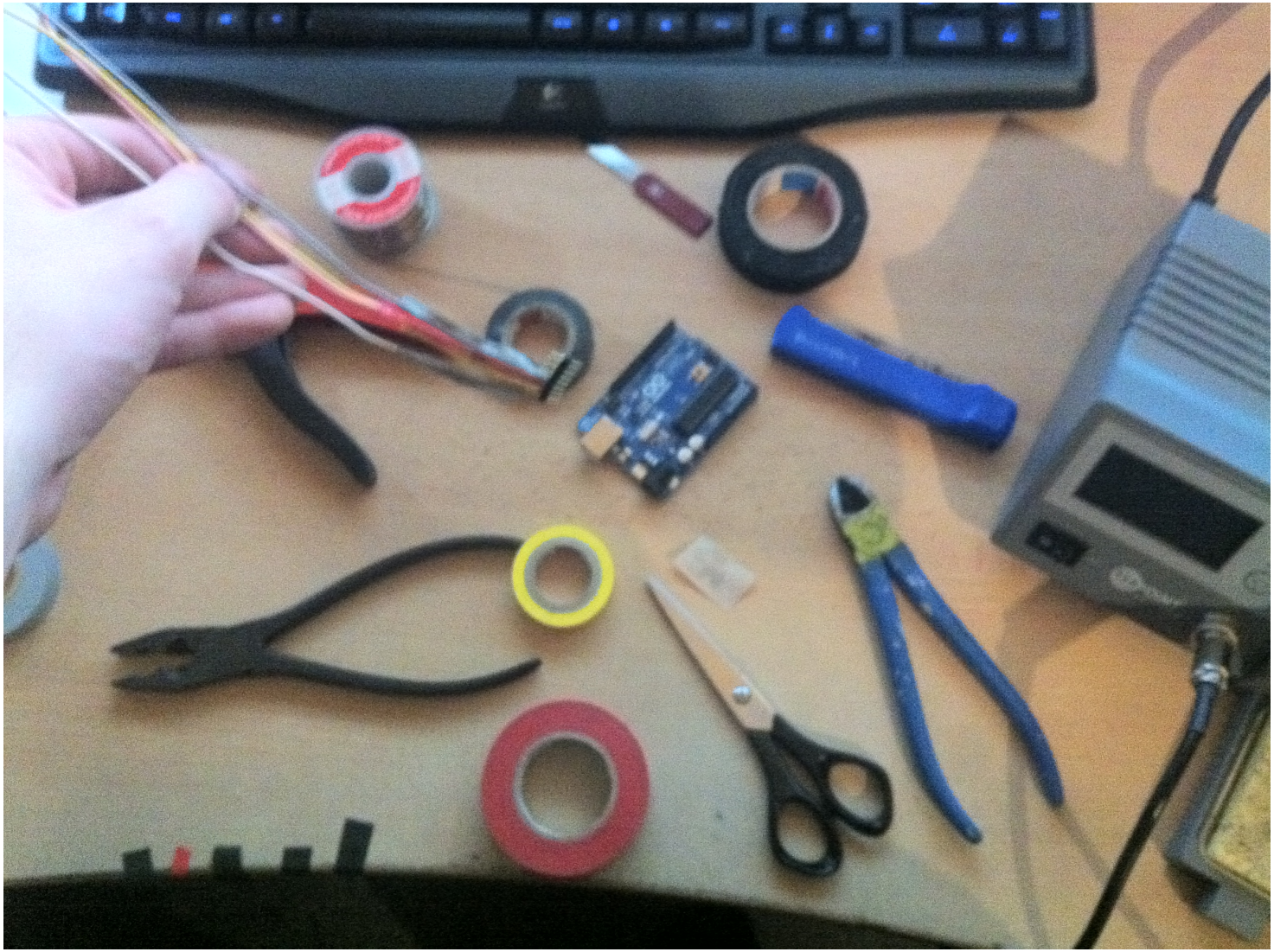
Building the Scanner - Hardware Skeleton 1

Building the Scanner - Hardware Skeleton 2

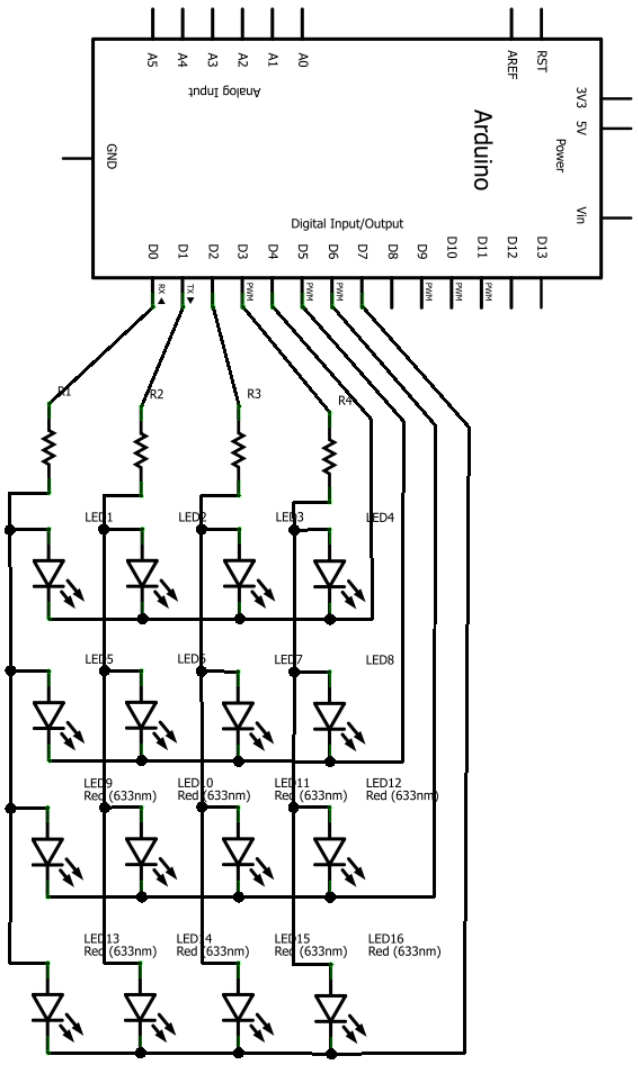
Building the Scanner - Wiring the LEDs 1



Building the Scanner - Wiring the LEDs 2



Building the Scanner - Wiring the LEDs 3

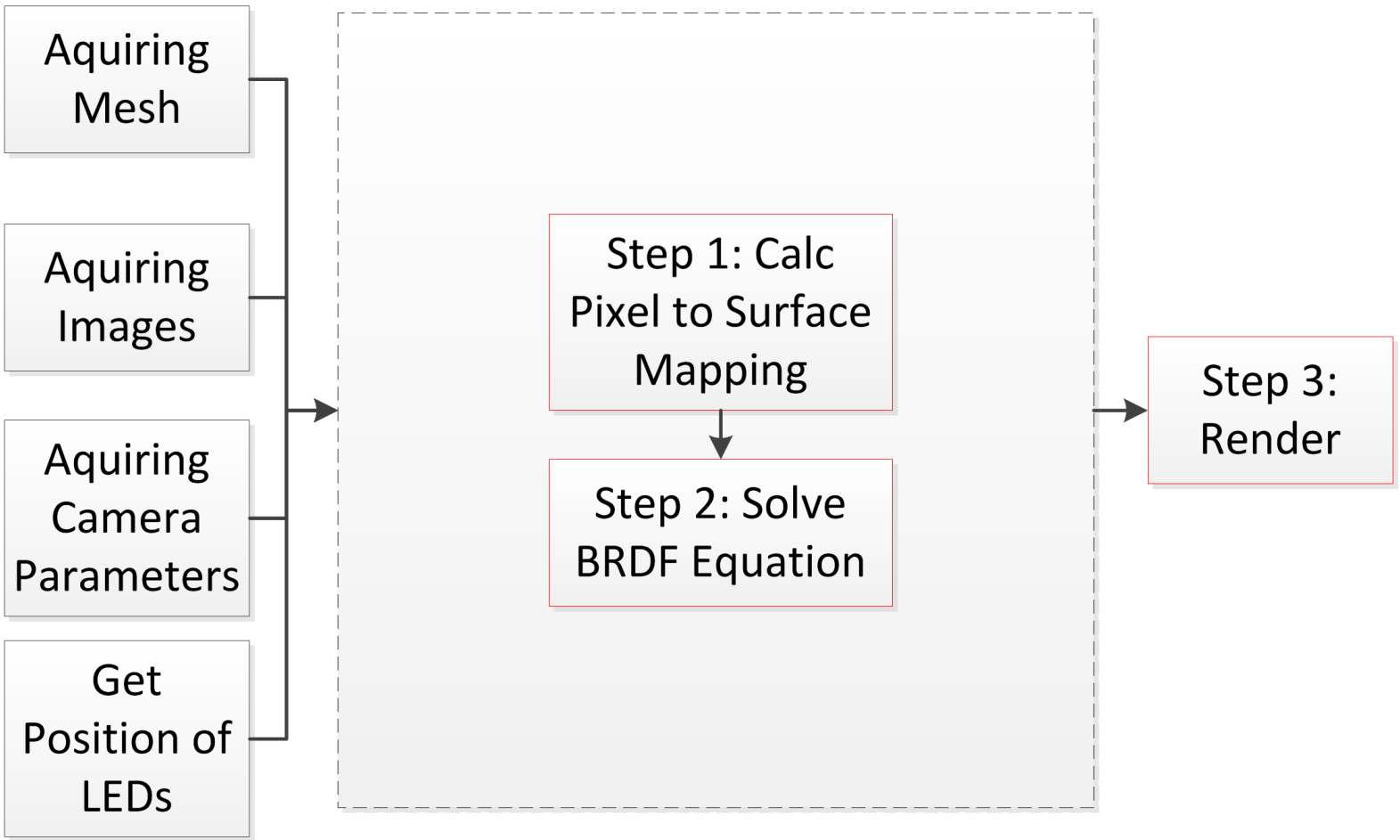


Building the Scanner - Programming the Arduino

```
void AllOFF(int offTime)
{
   //disable all!
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(0, HIGH);
  digitalWrite(1, HIGH);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);

  delay(offTime);
}

void OneON(int numLED, int onTime)
{
  //=================================================================
  //first ROW
  //=================================================================
  if(numLED == 1)
  {
    //enable led 1
    digitalWrite(0, HIGH);   // set the LED on
    digitalWrite(1, LOW);    // set the LED off
    digitalWrite(2, LOW);    // set the LED off
    digitalWrite(3, LOW);    // set the LED off

    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);

    delay(onTime);
    return;
  }

  if(numLED == 2)
  {
    //enable led 1
    digitalWrite(0, LOW);   // set the LED on
    digitalWrite(1, HIGH);    // set the LED off
    digitalWrite(2, LOW);    // set the LED off
    digitalWrite(3, LOW);    // set the LED off

    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
```

cable color:	white	yellow	pink	grey	red	green/red	brown/white	green
	Columns:				Rows:			
LED #:	D0:	D1:	D2:	D3:	D4:	D5:	D6:	D7:
NONE	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH
1	HIGH	low	low	low	low	HIGH	HIGH	HIGH
2	low	HIGH	low	low	low	HIGH	HIGH	HIGH
3	low	low	HIGH	low	low	HIGH	HIGH	HIGH
4	low	low	low	HIGH	low	HIGH	HIGH	HIGH
5	HIGH	low	low	low	HIGH	low	HIGH	HIGH
6	low	HIGH	low	low	HIGH	low	HIGH	HIGH
7	low	low	HIGH	low	HIGH	low	HIGH	HIGH
8	low	low	low	HIGH	HIGH	low	HIGH	HIGH
9	HIGH	low	low	low	HIGH	HIGH	low	HIGH
10	low	HIGH	low	low	HIGH	HIGH	low	HIGH
11	low	low	HIGH	low	HIGH	HIGH	low	HIGH
12	low	low	low	HIGH	HIGH	HIGH	low	HIGH
13	HIGH	low	low	low	HIGH	HIGH	HIGH	low
14	low	HIGH	low	low	HIGH	HIGH	HIGH	low
15	low	low	HIGH	low	HIGH	HIGH	HIGH	low
16	low	low	low	HIGH	HIGH	HIGH	HIGH	low

Coding the BRDF-Estimator



Step 1: Calc Pixel to Surface Mapping

- aligning the mesh in 3D world to Images
- use gluUnproject to find mapping between surface points and pixels
- shoots rays from pixel to 3D world, returns the coordinate of the triangle if hit

Step 2: Solve BRDF Equation

```
for (each pixel in scene) do:
{
   if (pixel maps to surface on mesh)
   {
       - Get Angles Phi, Theta and Theta'
       for (each color channel) do:
       {
          - Get Intensities at current pixel of all 16 Images
          - Solve Equation(Input: angles, Intensities)
          - Save Values to Surface Data Structure for Step3
       }
   }
}
```

Step 2: Solve Equation in Detail

- Levenberg-Marquart Solver
- dlevmar_dif(function, max iterations, guess, extraData, settings, etc.)
- initial guess required
- linear equation:
- function:

```
x[i] = p[0]*cosPhi + p[1]*pow(cosThetaDash,p[2])
```

- vector x:16 entries, contains pixel intensities
- vector p:3 entries, contains resulting parameters
- vector for angles: need to get passed separately

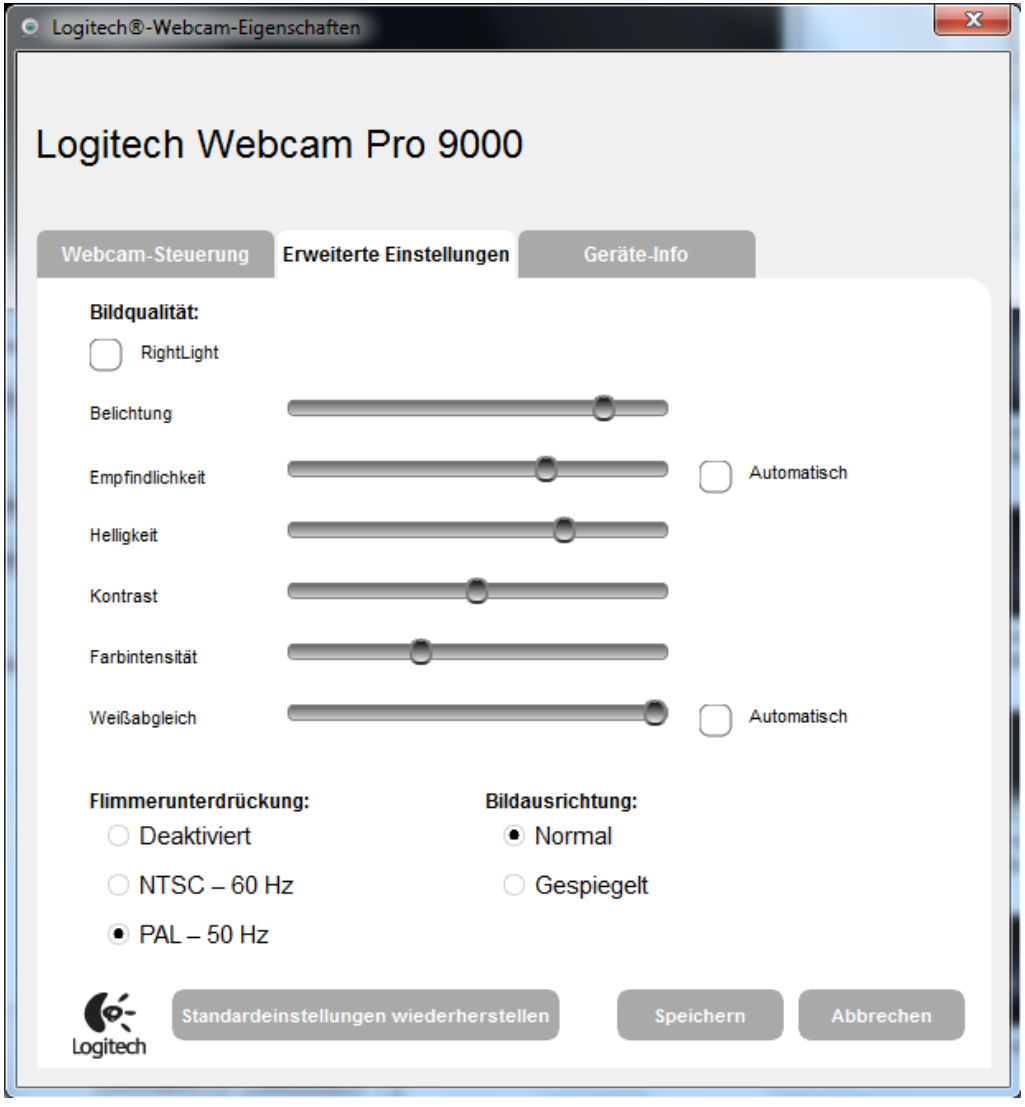
Step 3: Render

- on each frame, calc angles for every visible surface
- recalc the pixel intensity for R,G and B according to reflectance model
- ->very calculation intensive!
- therefore: implemented partly as shader on graphics card
- about 20fps for a 25k triangle mesh on a modern(2011) cpu+gpu

What didn't go so well..

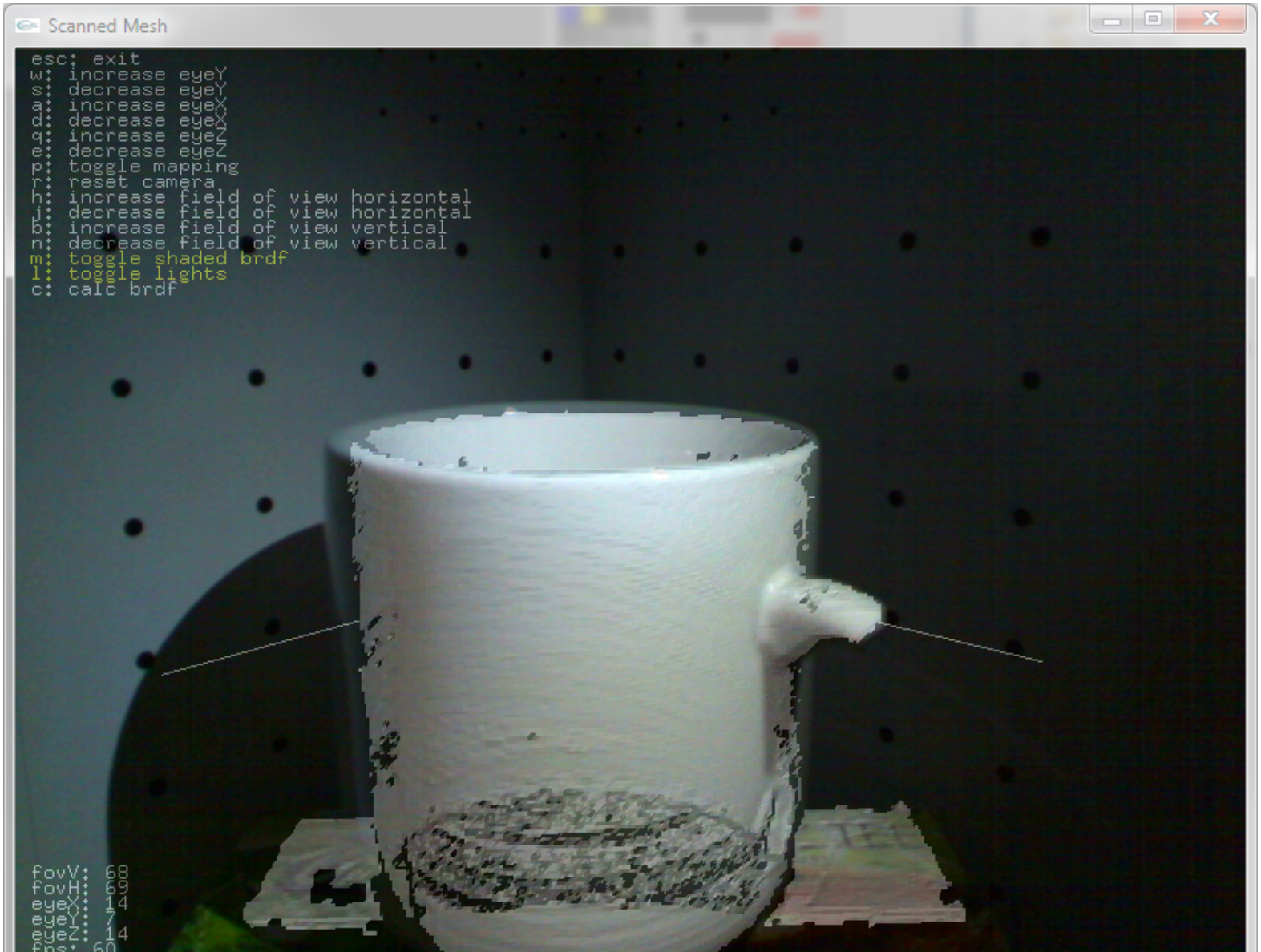
- hard to find good information
- reverse engineering of intrinsic camera parameters
- initial guess for equatation solver
- camera parameters unprecise from DAVID software
- correct camera settings?
- camera heavily distorted
- alignment OpenGL model with camera image
- not all pixels map to one triangle - why?
- takes long to take measurements -> only 3 samples

Camera Settings



Alignment Example 1

Alignment Example 2



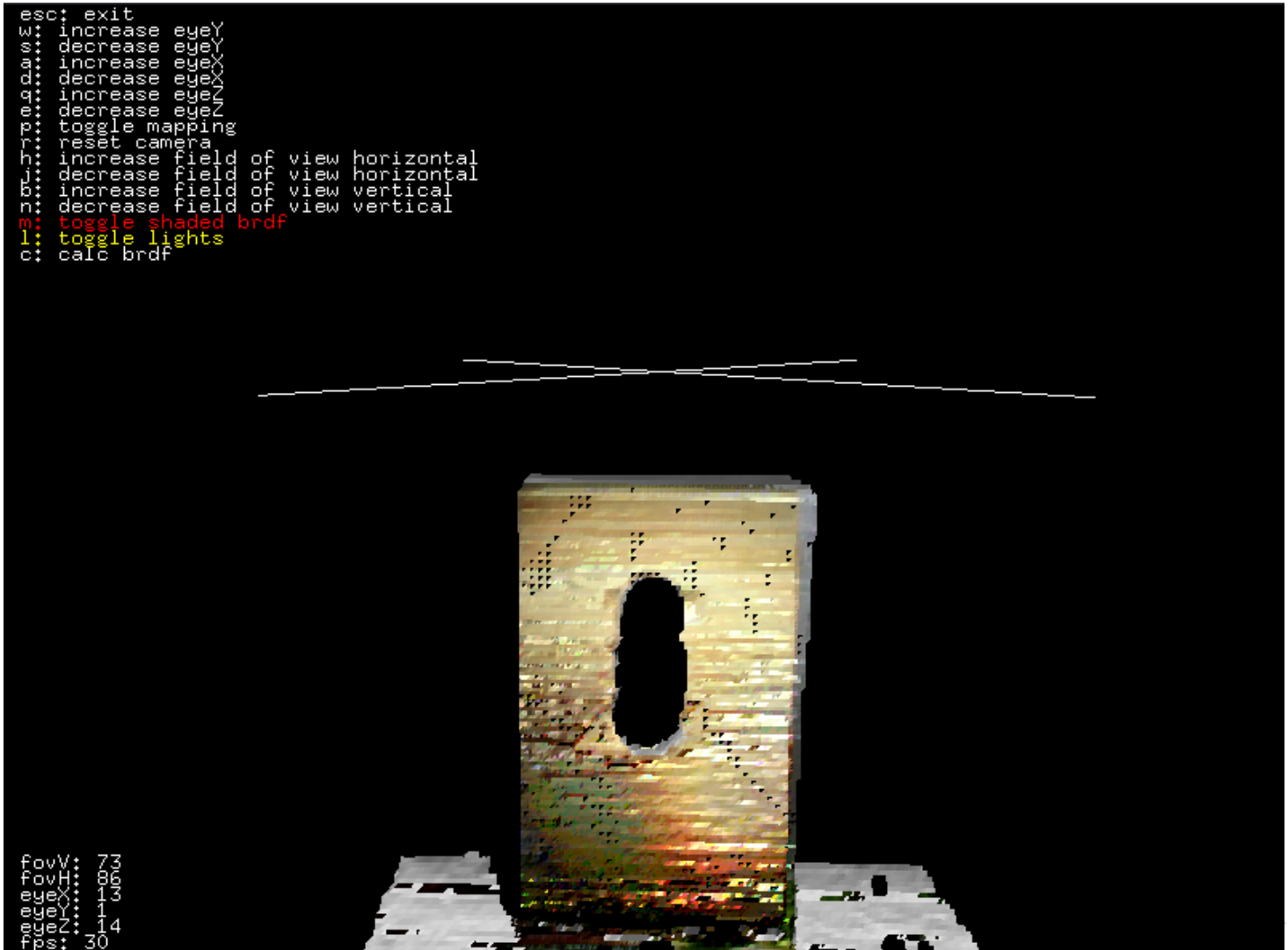
Alignment Example 3

Complexity

- Levenberg-Marquardt algorithm highest complexity
- complexity linear with number of pixels in images
- number of images not carrying much weight

Example 1: Timber, Blinn-Phong

- average values: $k_{diff} = 2.91, k_{spec} = 1.67, n = 3.71$

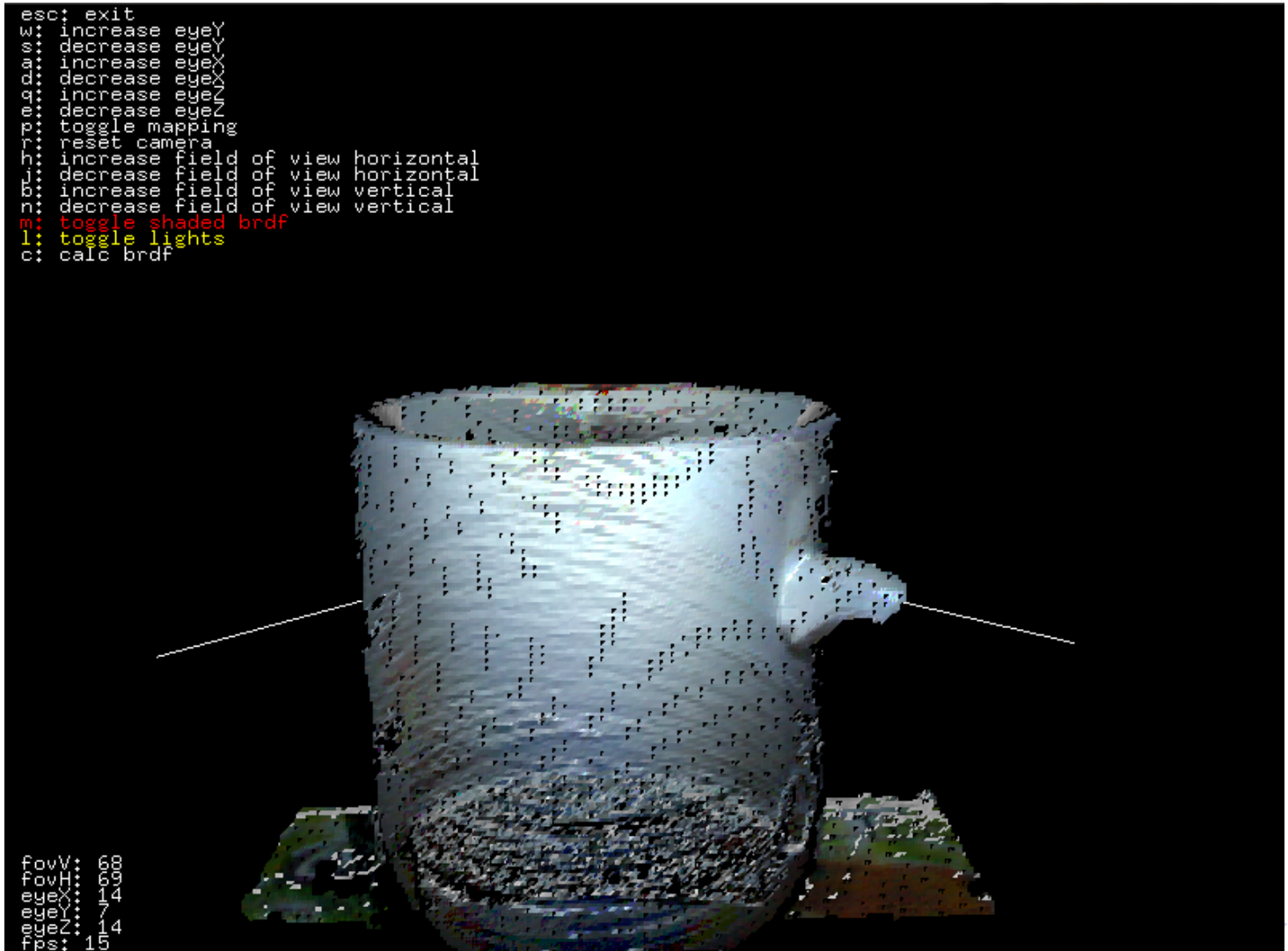


Example 2: Clay, Blinn-Phong

- average values: $k_{diff} = 0.91, k_{spec} = 0.82, n = 3.09$

Example 3: Porcelain, Blinn-Phong

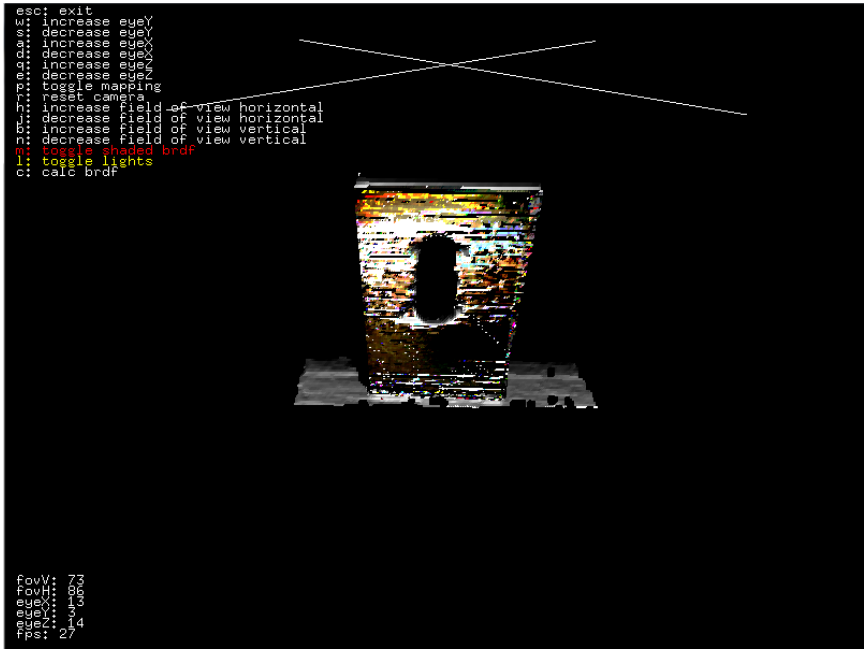
- average values: $k_{diff} = 2.49, k_{spec} = 1.16, n = 2.3$



Blinn-Phong vs. Phong

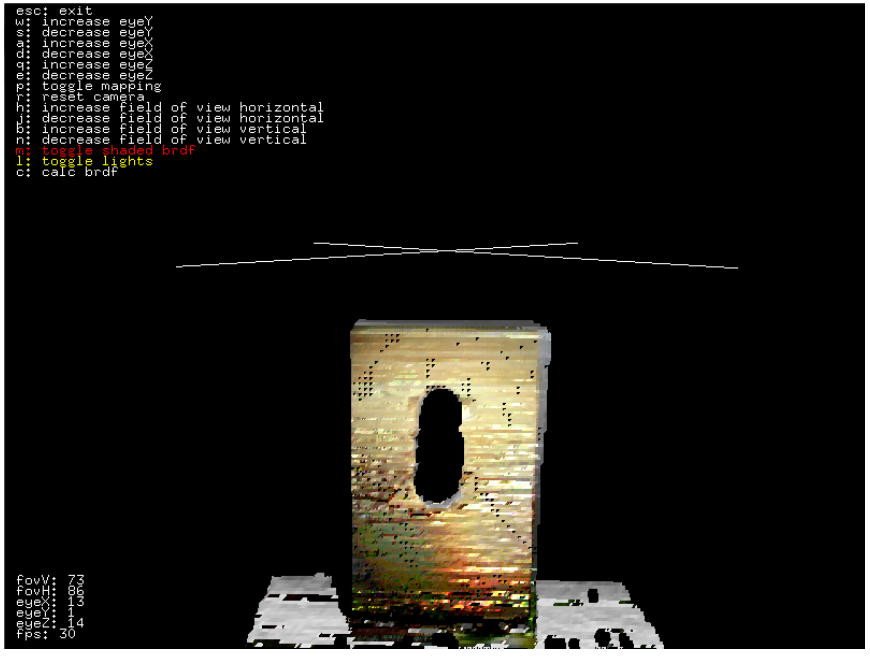
average values Phong:

$k_{diff} = 1.97, k_{spec} = 1.14, n = 2.91$



average values Blinn-Phong:

$k_{diff} = 2.91, k_{spec} = 1.67, n = 3.71$



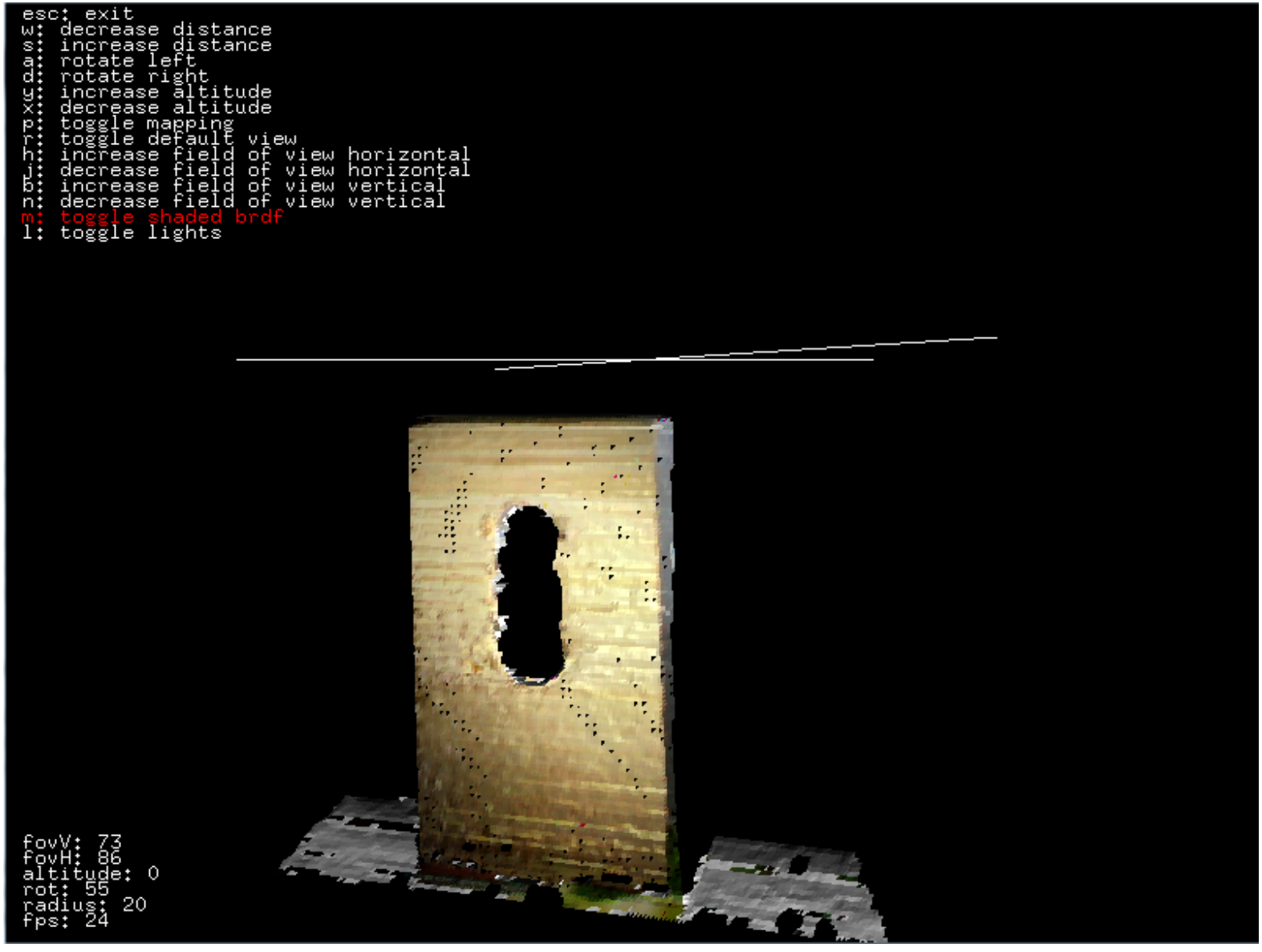
Blinn-Phong, Timber Demo

Live Demo!

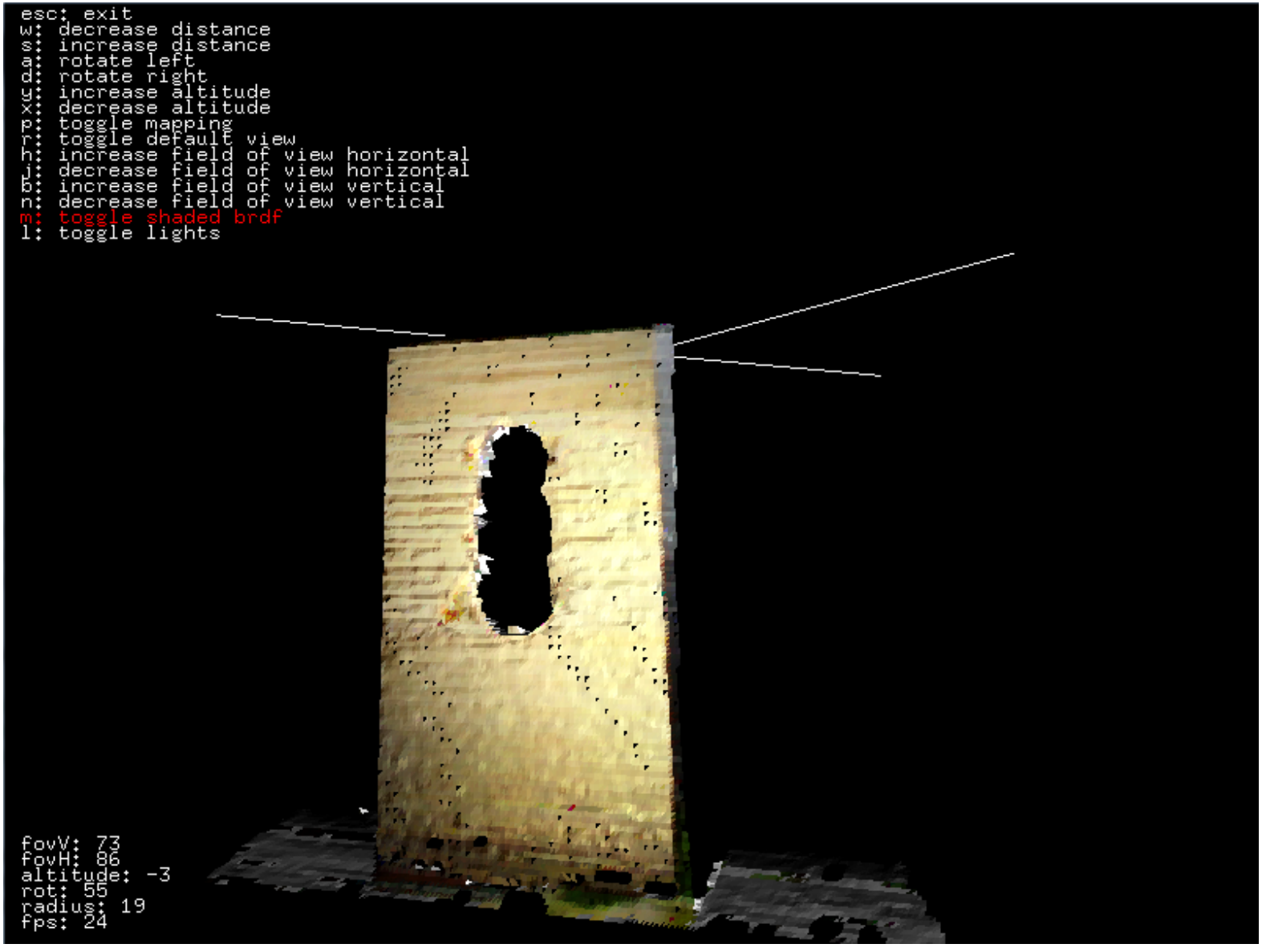
Related Works

- Gerhard Meister, Uni Hamburg, 1995 - Messung der spektralen Reflexionsfunktion (BRDF) ausgewählter Oberflächen bei natürlicher Beleuchtung
- Wojciech Matusik, MIT, 2005 - Experimental Analysis of BRDF Models

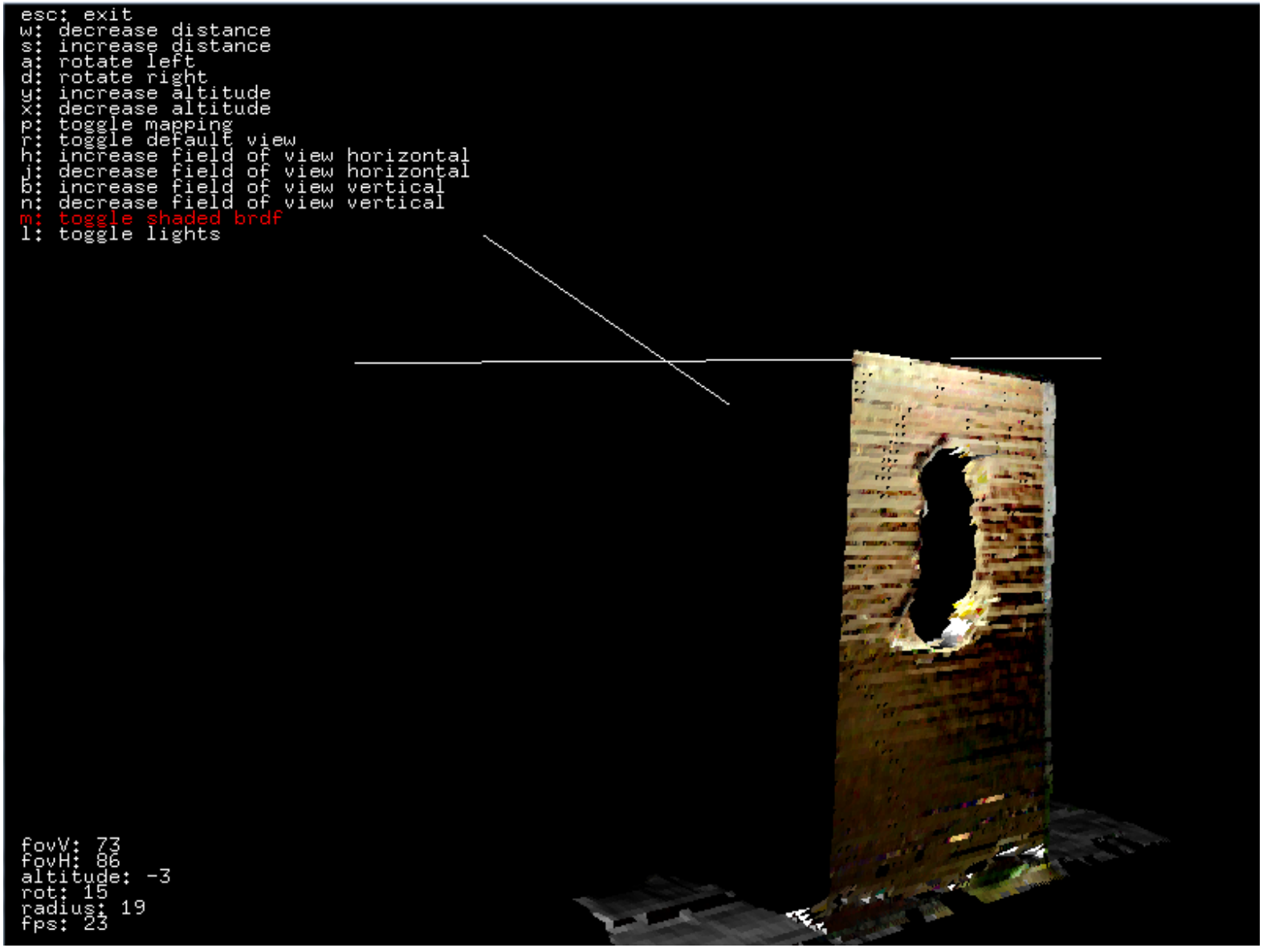
Our Model, Timber Ex.1



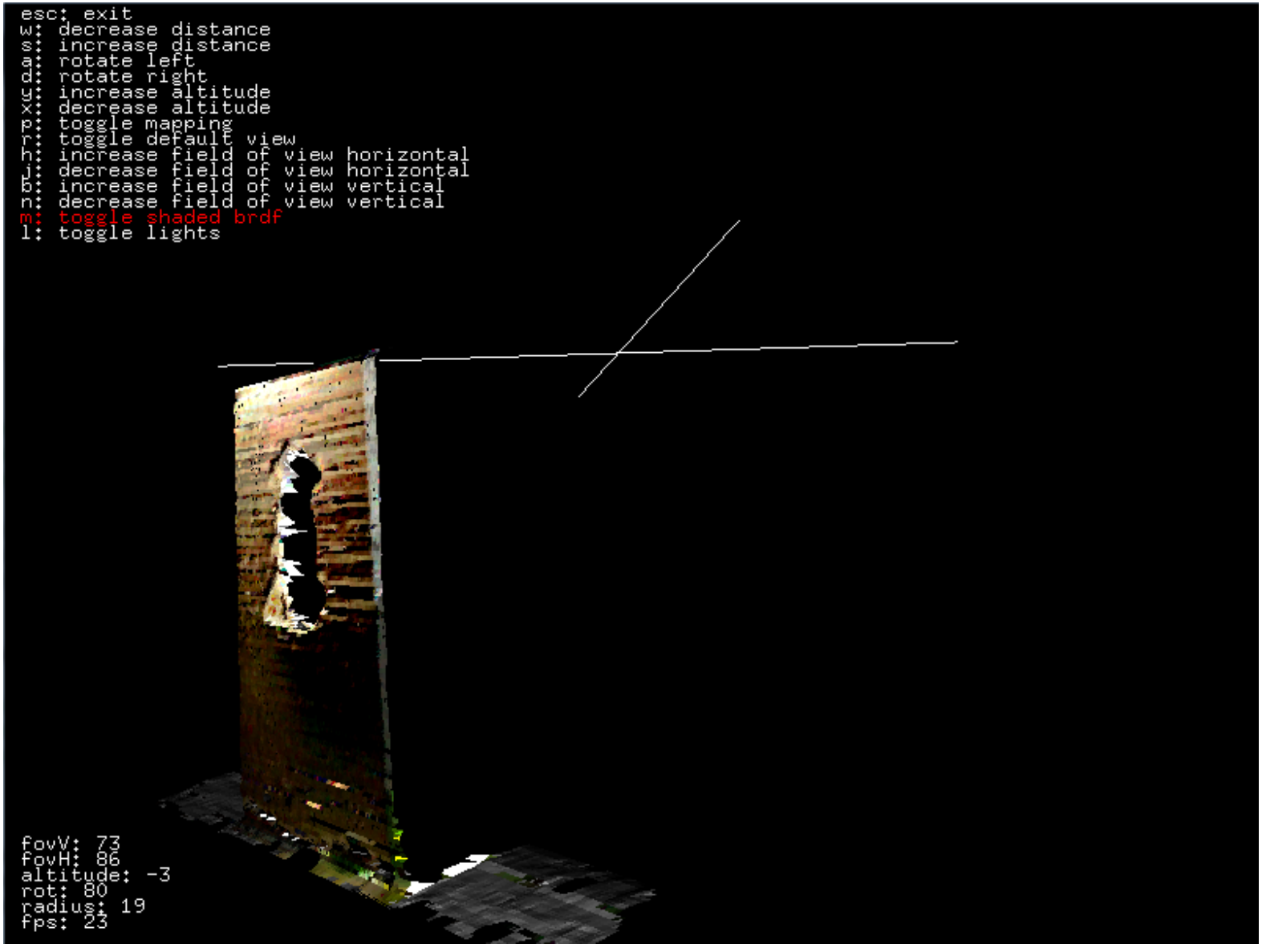
Our Model, Timber Ex.2



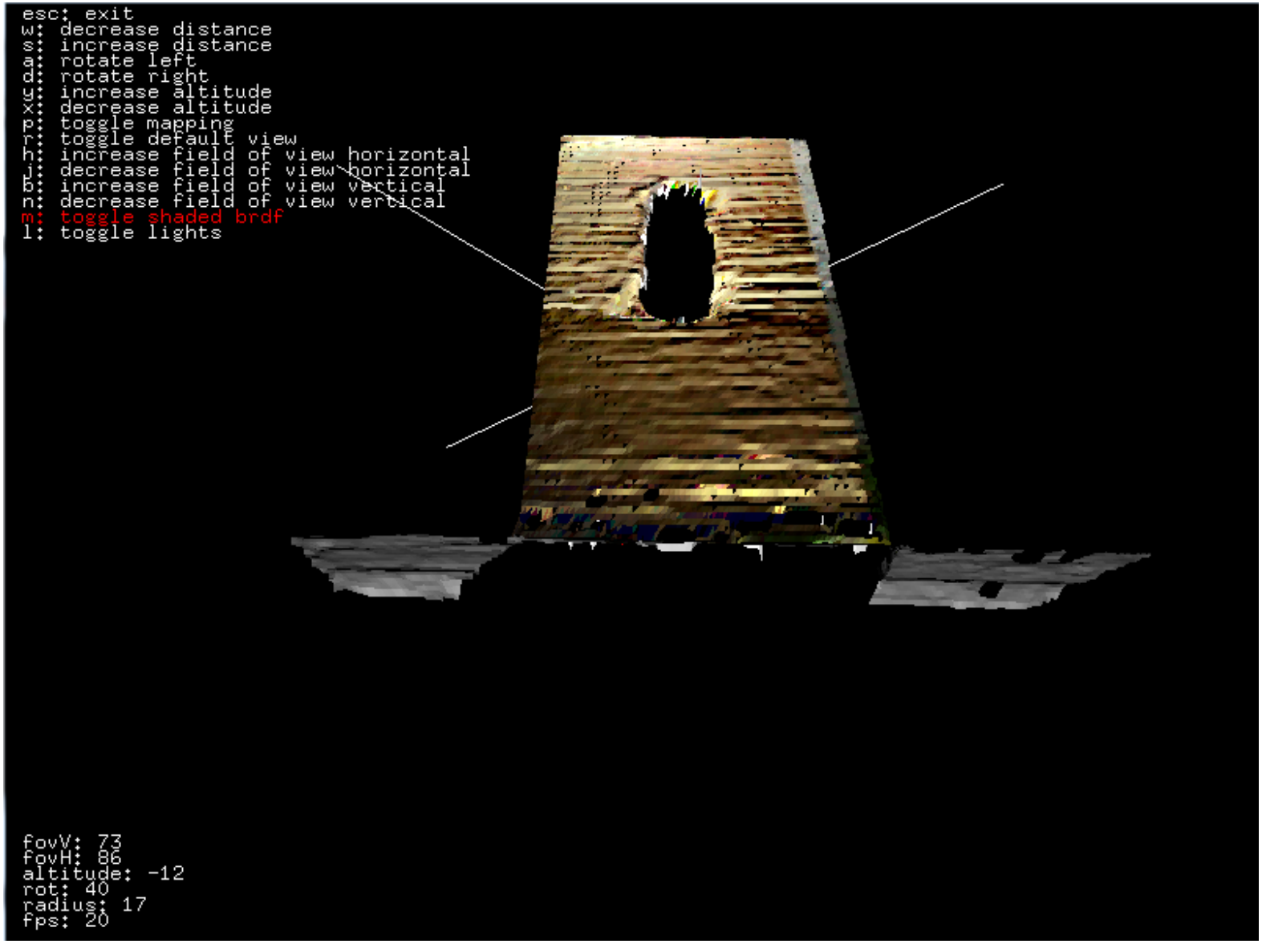
Our Model, Timber Ex.3



Our Model, Timber Ex.4



Our Model, Timber Ex.5



Limitations

- can't scan too reflectant surfaces
- resolution is limited
- camera is prone to noise

Improvements

- smooth over small neighbourhood of vertices(reducing noise)
- load old results if needed, instead recalculating each time
- run entire calculation on graphicscard(CUDA, not just a shader)
- use different solver
- measure texture + brdf separate

Questions?

References

image http://en.wikipedia.org/wiki/File:BRDF_Diagram.svg

paper "Einige BRDF Modelle", Nikolaus Gebhardt,
http://www.irrlicht3d.org/papers/BrdfModelle.pdf

paper "MODELS OF LIGHT REFLECTION FOR COMPUTER SYNTHESIZED PICTURES"
James F. Blinn, SIGGRAPH 1977, pp 192–198.

diplomarbeit
"Messung der spektralen Reflexionsfunktion (BRDF) ausgewählter Oberflächen bei natürlicher Beleuchtung
Gerhard Meister, Uni Hamburg, 1995
http://censis.informatik.uni-hamburg.de/publications/diplom.pdf

paper "Experimental Analysis of BRDF Models" Wojciech Matusik, MIT, 2005
http://people.csail.mit.edu/addy/research/ngan05_brdf_eval.pdf

library "OpenCV.org"