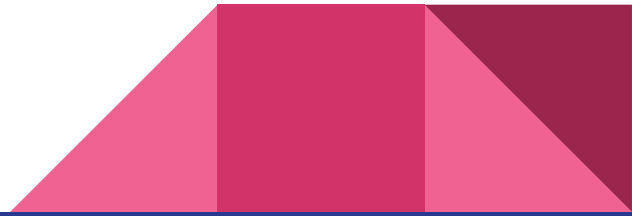
Ray tracing UV light disinfection of hospital rooms

By Oscar Fickel

Under supervision of Frank van der Stappen & Robbert Bentvelsen

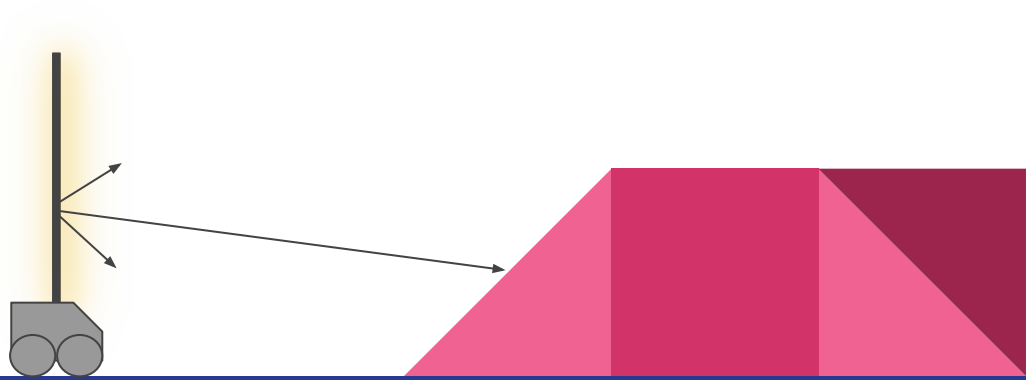
Content

- Concept
- Approach
- Results
- Implementation
- Limitations / future work



Concept

- UV lamp robot moves to points in a room
- Visualising UV radiation on the surfaces of this room
- Using ray tracing, an increasingly popular computer graphics technique



Approach

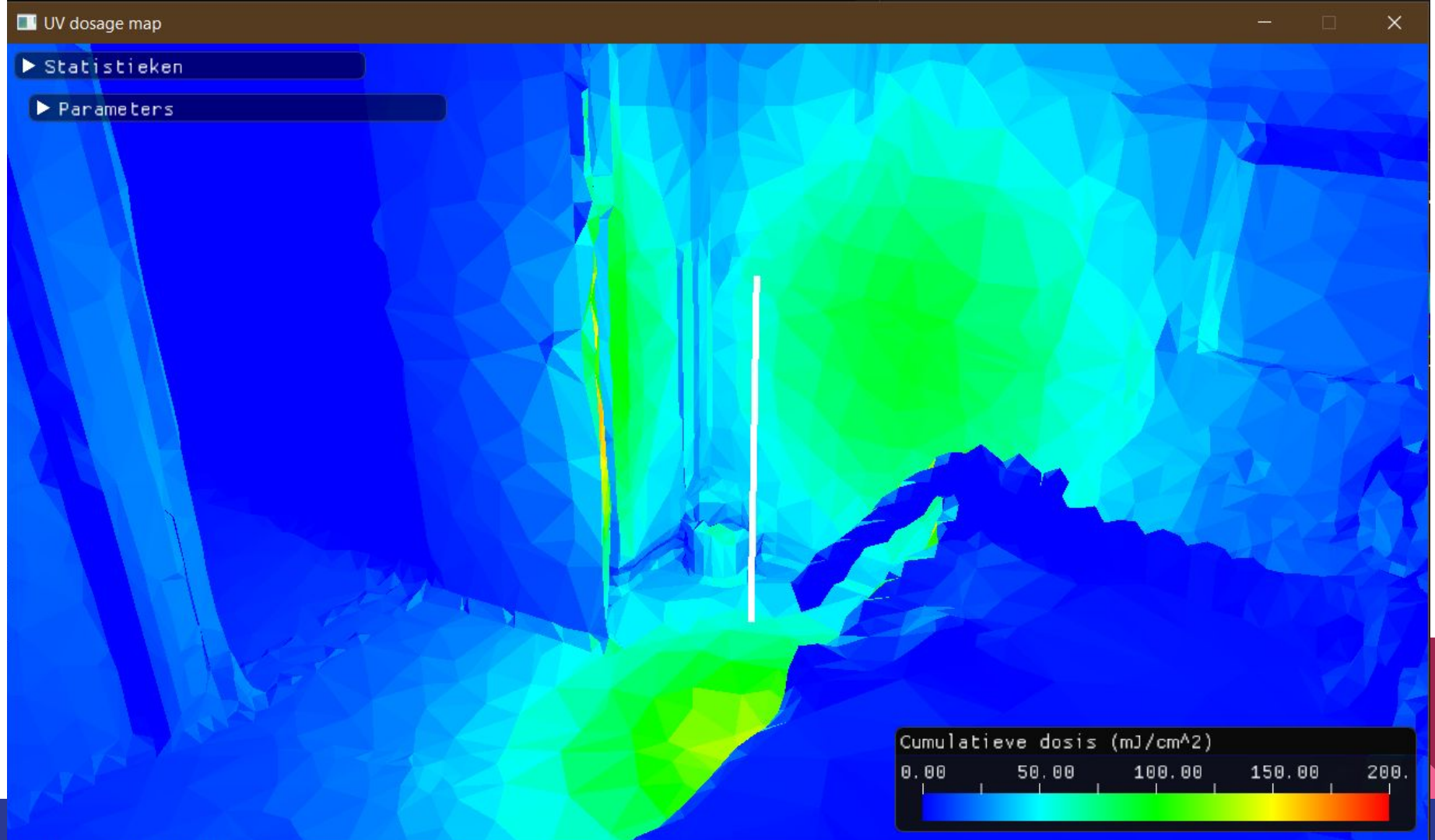
- Produce a heatmap
- Compute once a dose/color per triangle
- Cumulative dosage & maximum irradiance

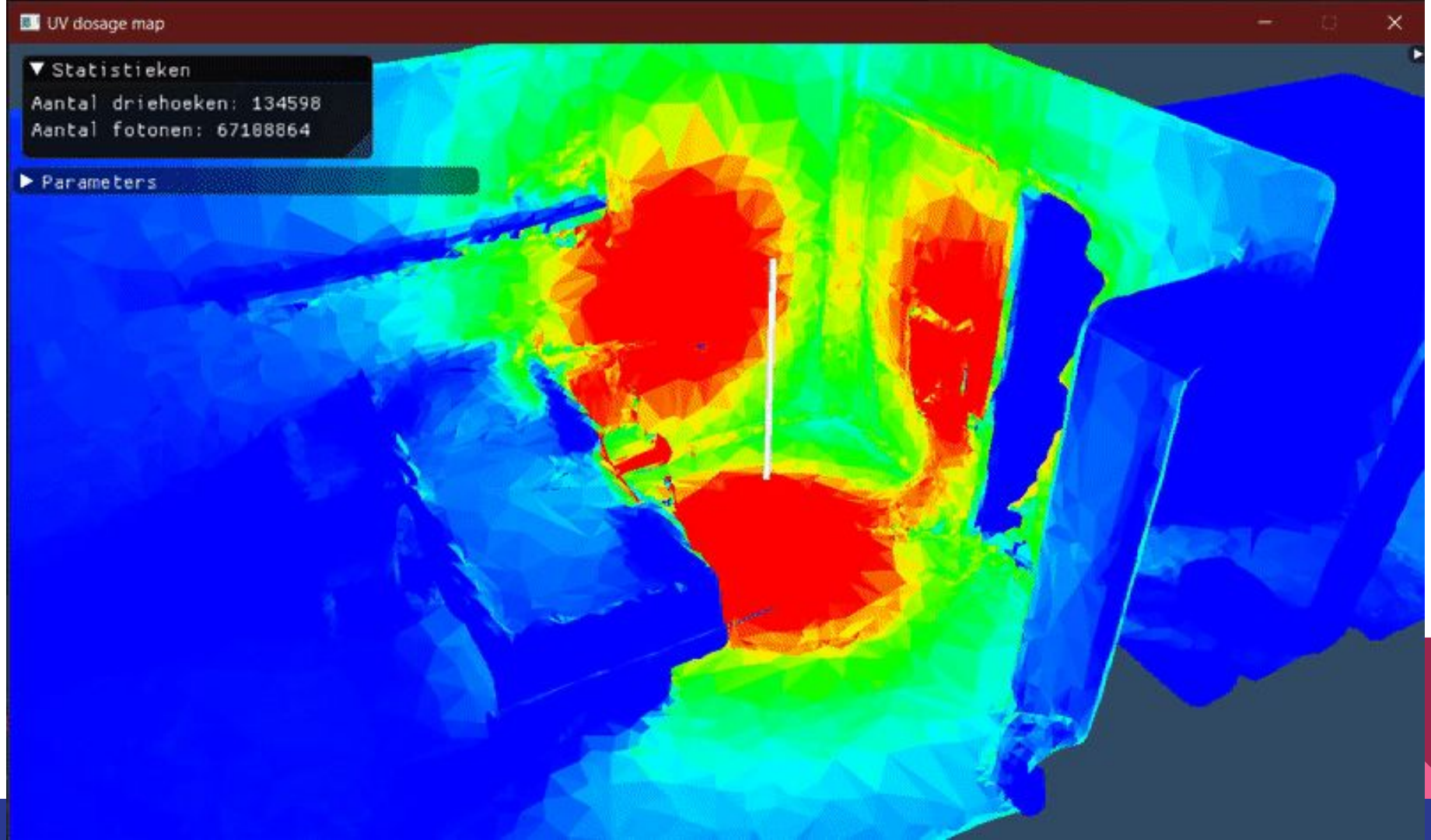
3D model

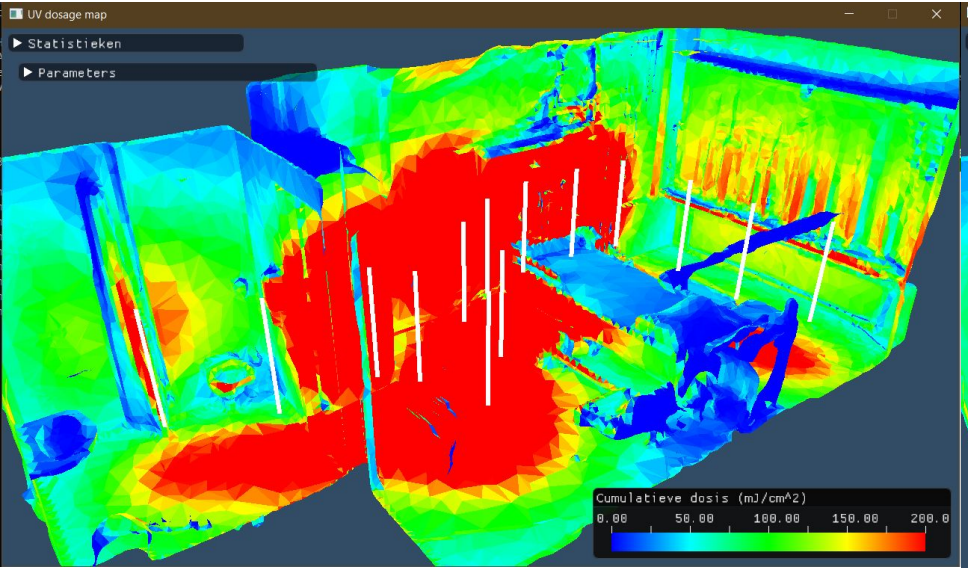
- Obtained by using LiDAR technology in iPhones, via Polycam
- Slightly variable level of detail
- Not super robust, but decent, and is both cheaper and more efficient compared to the alternatives

Results

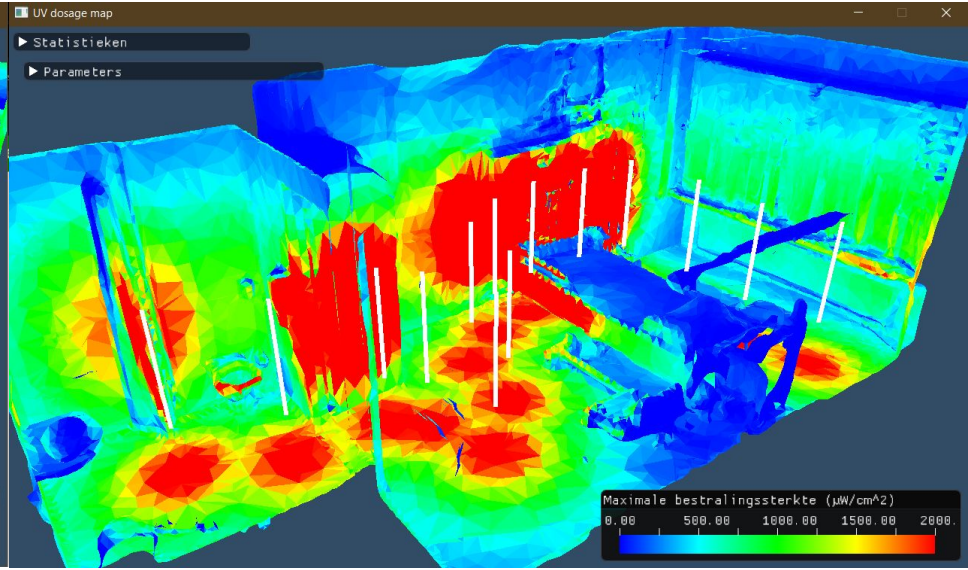

UV dosage map
Statistieken
Parameters
Cumulatieve dosis (mJ/cm^2)
0.00
50.00
100.00
150.00
200.


UV dosage map
Statistieken
Parameters
Cumulatieve dosis (mJ/cm^2)
0.00
50.00
100.00
150.00
200.


UV dosage map
Statistieken
Aantal driehoeken: 134598
Aantal fotonen: 67108864
Parameters



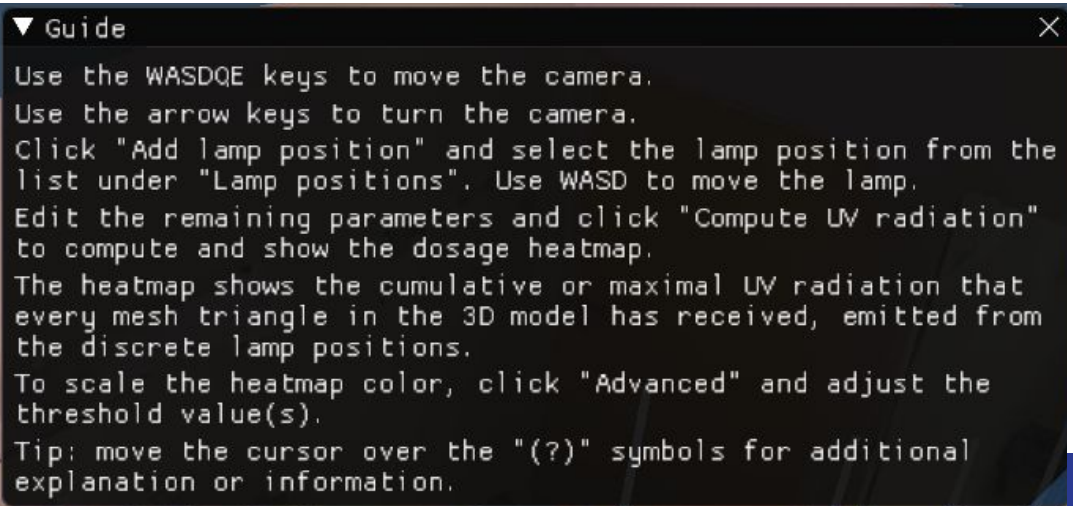
Cumulative dosage

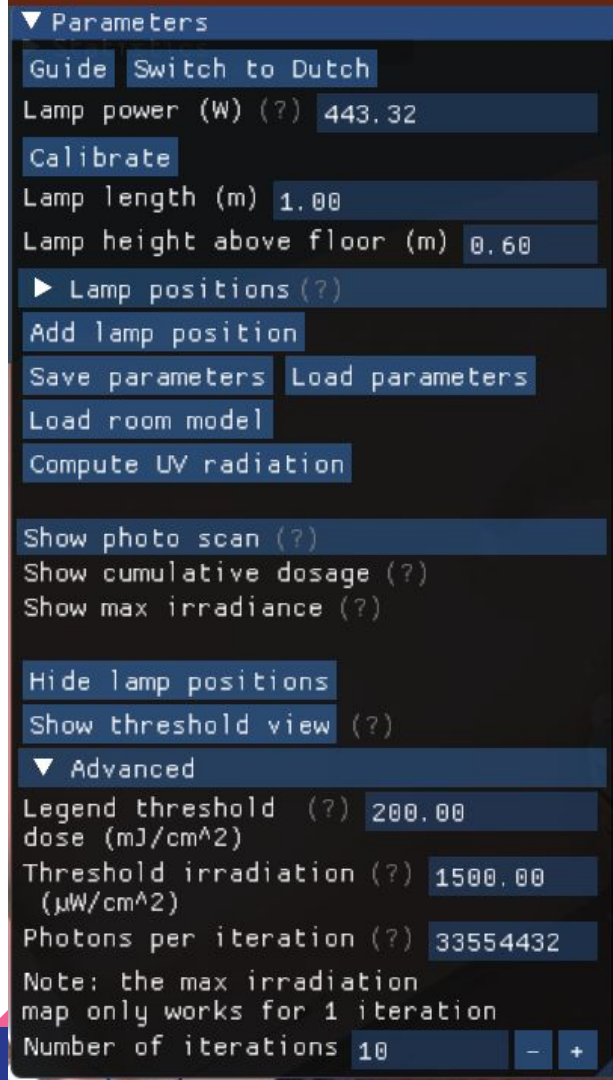


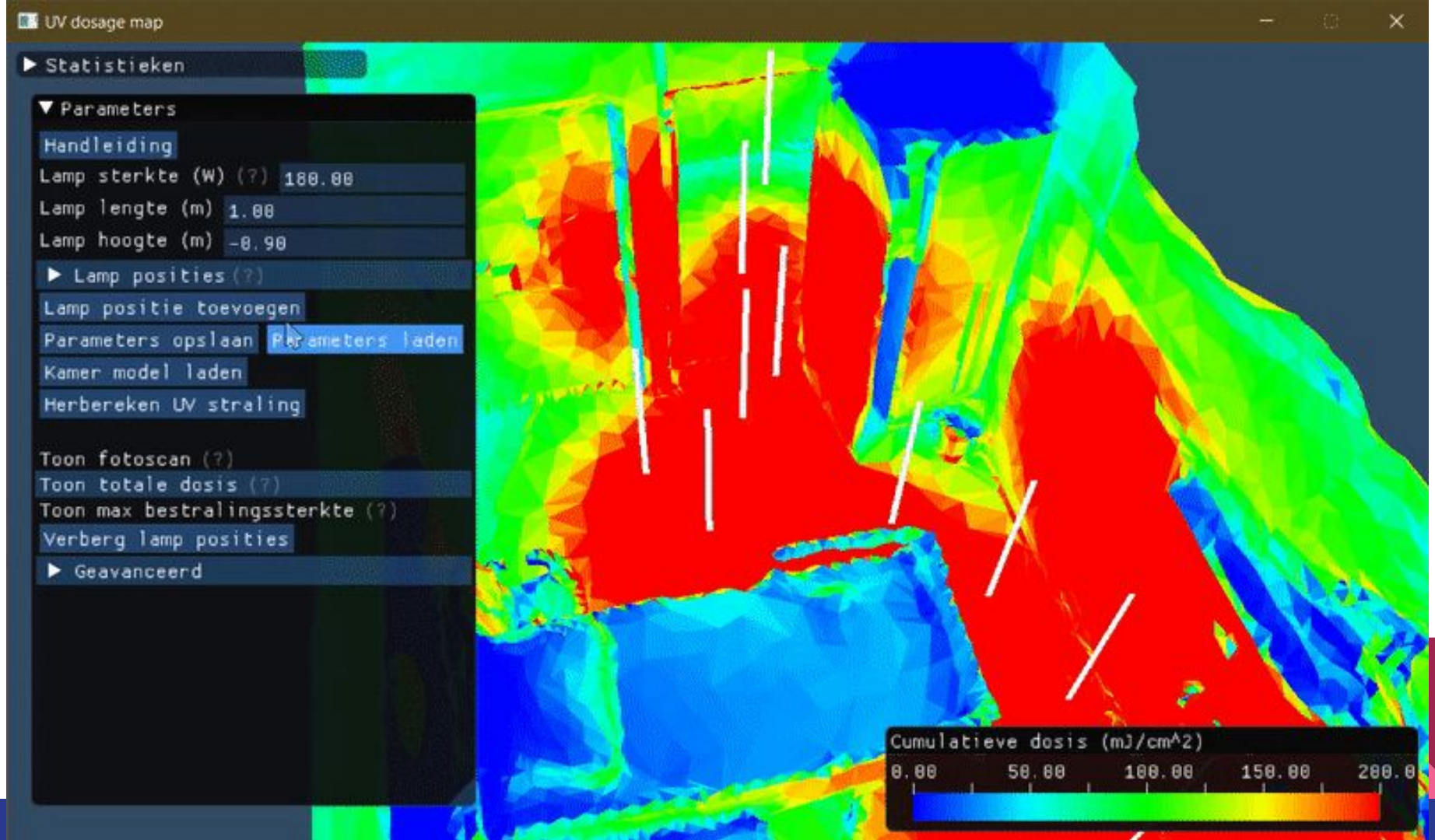
Maximum irradiance

User Interface

- Lamp parameters
- Editable light positions



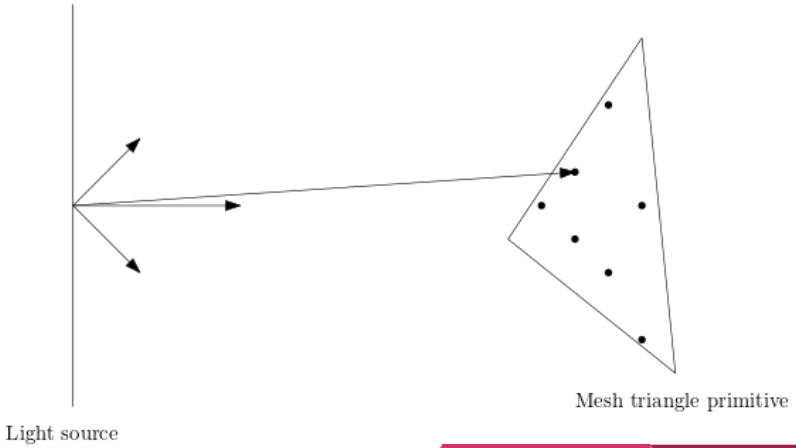


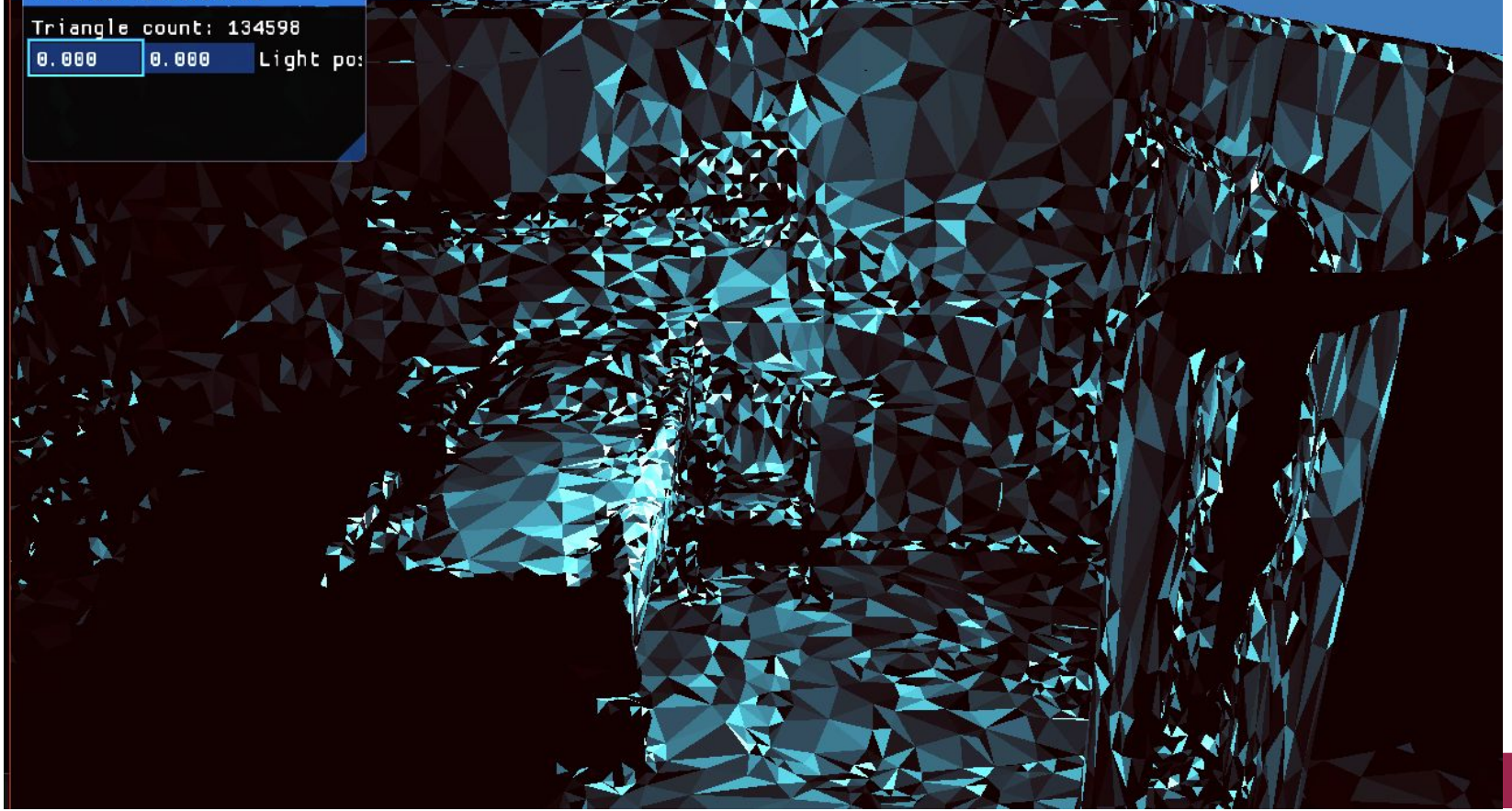

UV dosage map
Statistieken
Parameters
Handleiding
Lamp sterkte (W) (?) 180.00
Lamp lengte (m) 1.00
Lamp hoogte (m) -0.90
Lamp posities (?)
Lamp positie toevoegen
Parameters opslaan
Parameters laden
Kamer model laden
Herbereken UV straling
Toon fotoscan (?)
Toon totale dosis (?)
Toon max bestralingssterkte (?)
Verberg lamp posities
Geavanceerd
Cumulatieve dosis (mJ/cm^2)
0.00
50.00
100.00
150.00
200.0

Implementation

Counting photons per triangle

- Rays/photons sent out from light source (vertical line) in all directions
- Count per triangle the number of hits
- Divide by area to get photon density
- Angle of light is automatically handled
- Same for inverse square law:
  intensity decreases in proportion
  to square distance
- Smaller triangles are more noisy





A low photon count results in inaccuracies at smaller triangles, luckily the use of a BVH acceleration structure allows for shooting a large enough number of photons

Implementation

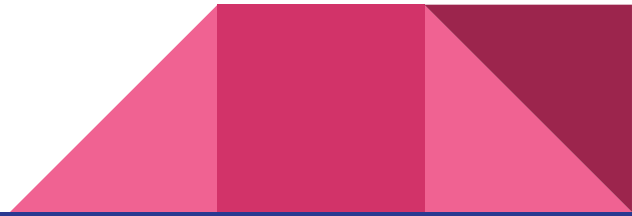
- Computing irradiance & dose
- Within a few seconds on the GPU:
  Generating rays, computing intersections,
  then applying formula & mapping to colors

$$D = \frac{I * \sum_{l=0}^{L} (\Delta t_l * n_l)}{A * \frac{N}{L}} * 0.1$$

where I $[W]$ is the intensity of the light source, L is the number of light positions, n_l is the number of photons received by the triangle from light position l, Δt_l is the time duration per light position, A $[m^2]$ is the surface area of the triangle, and N is the total number of photons. The fraction is multiplied by 0.1 to convert from Jm^{-2} to $mJcm^{-2}$.

Limitations / (Possible) Sources of Inaccuracy

- Mesh accuracy
- Number of photons
- Lamp is approximated as vertical line
- User entered parameters
- Lamp movement is ignored
- Air and surface conditions are (somewhat) ignored
- Reflections are ignored
- Lamp needs 2 minute warm up



Additional Future work

- More advanced calibration & validation
- Optimisation of disinfection strategy