Photo by afloresm, Flickr

Efficient Simulation of Small Mirrors in Central Receiver Systems

Andrii Podriez
RWTH Aachen University

Aachen, 15 October 2025

hybr2d Theory of Hybrid Systems Informatik 2 | RWTH AACHEN UNIVERSITY

Introduction

Central Receiver Systems

1. Demand for renewable energy keeps growing
2. Central receiver systems (CRS) provide storage-backed, dispatchable solar power.

Introduction

Central Receiver Systems

1. Demand for renewable energy keeps growing
2. Central receiver systems (CRS) provide storage-backed, dispatchable solar power.

SunFlower

1. Planning and optimizing CRS is not a simple task
2. SunFlower simulates CRS using ray tracing methods

Motivation

Problem 1

1. Other simulation tools provide means of simulating environmental obstructions.
2. There is no model for obstacles in SunFlower.
3. Taking obstructions into account during simulation and optimization of CRS can be crucial.

Motivation

Problem 1

1. Other simulation tools provide means of simulating environmental obstructions.
2. There is no model for obstacles in SunFlower.
3. Taking obstructions into account during simulation and optimization of CRS can be crucial.

Problem 2

1. A new parameter scheme for small mirrors (facets) was previously introduced in SunFlower
2. This scheme was not fully integrated, resulting in some services not functioning properly
3. Furthermore, new scheme can be labour intensive in some cases

Optical model

Ray tracing methods

1. Every ray tracing technique begins by tracing a ray in the Sun's direction
2. **Analytical** ray tracer calculates an ideal reflected ray
3. **Monte Carlo** ray tracer randomly perturbs the reflected ray
4. **Convolution** ray tracer uses probability distribution to estimate chances of reflected rays hitting the receiver

Optical model

Ray tracing methods

1. Every ray tracing technique begins by tracing a ray in the Sun's direction
2. **Analytical** ray tracer calculates an ideal reflected ray
3. **Monte Carlo** ray tracer randomly perturbs the reflected ray
4. **Convolution** ray tracer uses probability distribution to estimate chances of reflected rays hitting the receiver

Heliostats

1. Movable mirrors that track the sun
2. Each mirror is composed of facets
3. Facet is divided into the pieces (cells) during the simulation

Output power

1. Reflective surface of the heliostat is composed of facets
2. Facets are divided into pieces, each with an area of A_{cell}

Given the amount of solar radiation received per m^2, I_{DNI} we can compute ideal power contributed by every individual cell P_{in} reflecting solar radiation

$$P_{\text{in}} = I_{\text{DNI}} \cdot A_{\text{cell}}$$

Output power

1. During the simulation incoming solar radiation is represented by a ray cast from each facet cell in Sun's direction
2. Then we trace a reflected ray according to the chosen ray tracing method
3. While tracing those rays we calculate various **mean efficiencies**, which represent energy losses caused by real-world factors
4. By multiplying the ideal power with the efficiencies we calculate the actual approximated output power P_{out}

$$P_{\text{out}} = P_{\text{in}} \cdot \eta_{\text{cos}} \cdot \eta_{\text{ref}} \cdot \eta_{\text{aa}} \cdot \eta_{\text{sp}} \cdot \eta_{\text{ts}} \cdot \eta_{\text{shading}} \cdot \eta_{\text{blocking}}$$

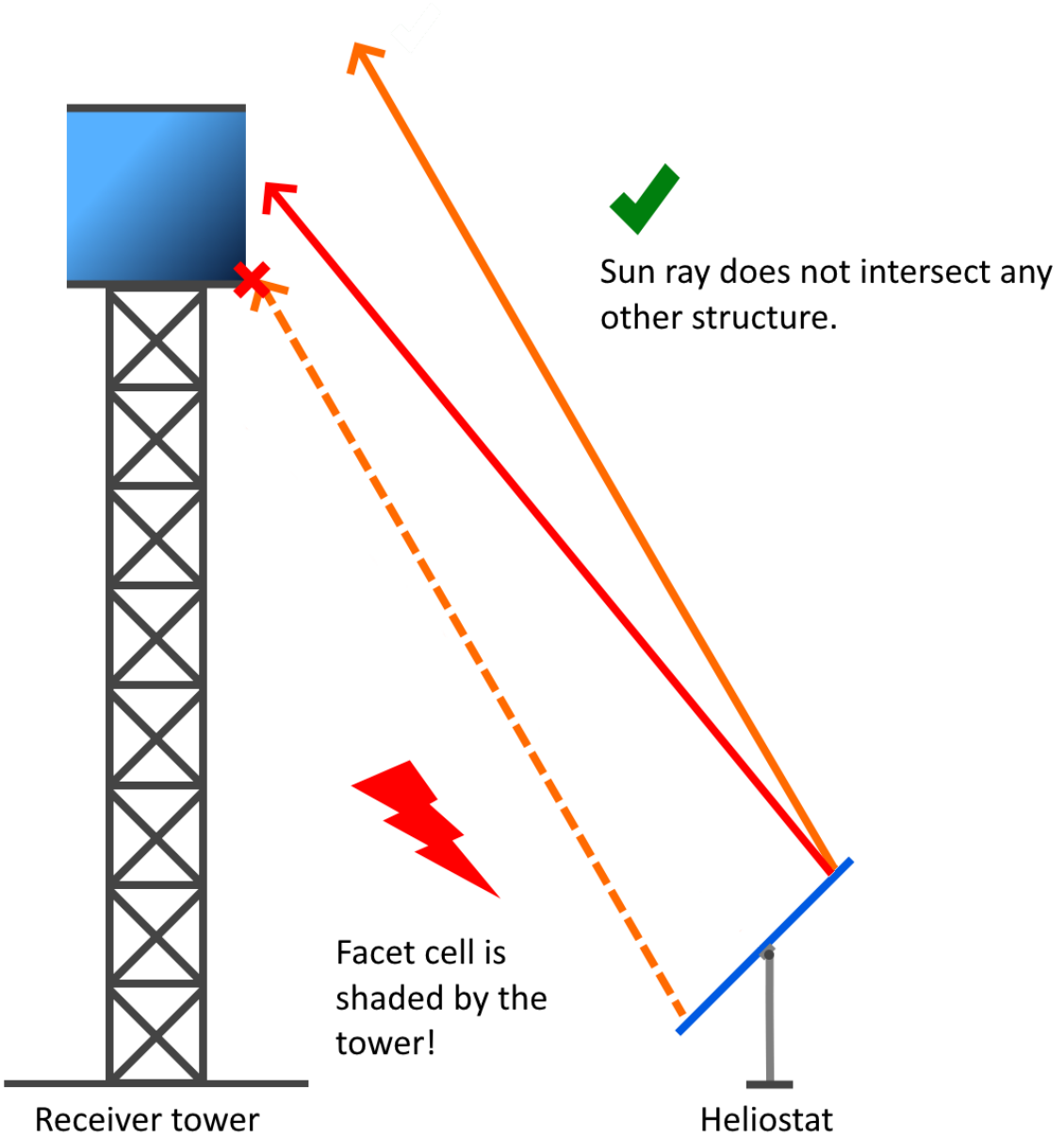
Shading and Blocking

Receiver tower and other heliostats on the field might obstruct the Sun rays, **shading** the mirrors, or the reflected rays, **blocking** the mirrors.
Blocking and shading is checked for every facet cell by tracing intersections of rays with other heliostats. Areas of affected cells are then summed as A_{shaded} and A_{blocked}. η_{shading} and η_{blocking} are then computed as ratios of shaded and blocked areas to the total mirror area of simulated heliostat field. A_{total}:

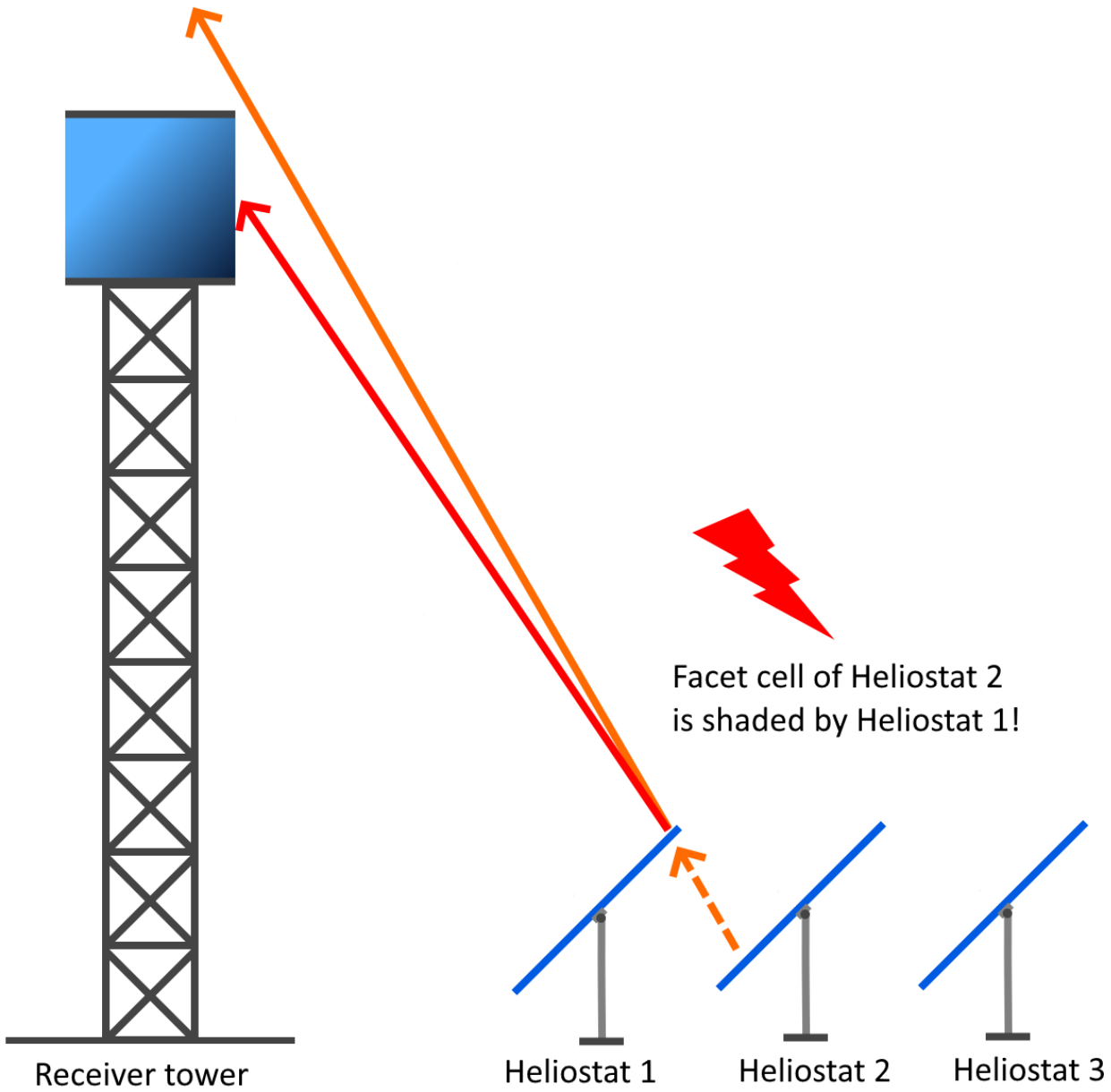
$$\eta_{\text{shading}} = 1 - \frac{A_{\text{shaded}}}{A_{\text{total}}} \qquad \eta_{\text{blocking}} = 1 - \frac{A_{\text{blocked}}}{A_{\text{total}}}$$

Tower shading efficiency η_{ts} is computed following similar approach with the receiver tower.

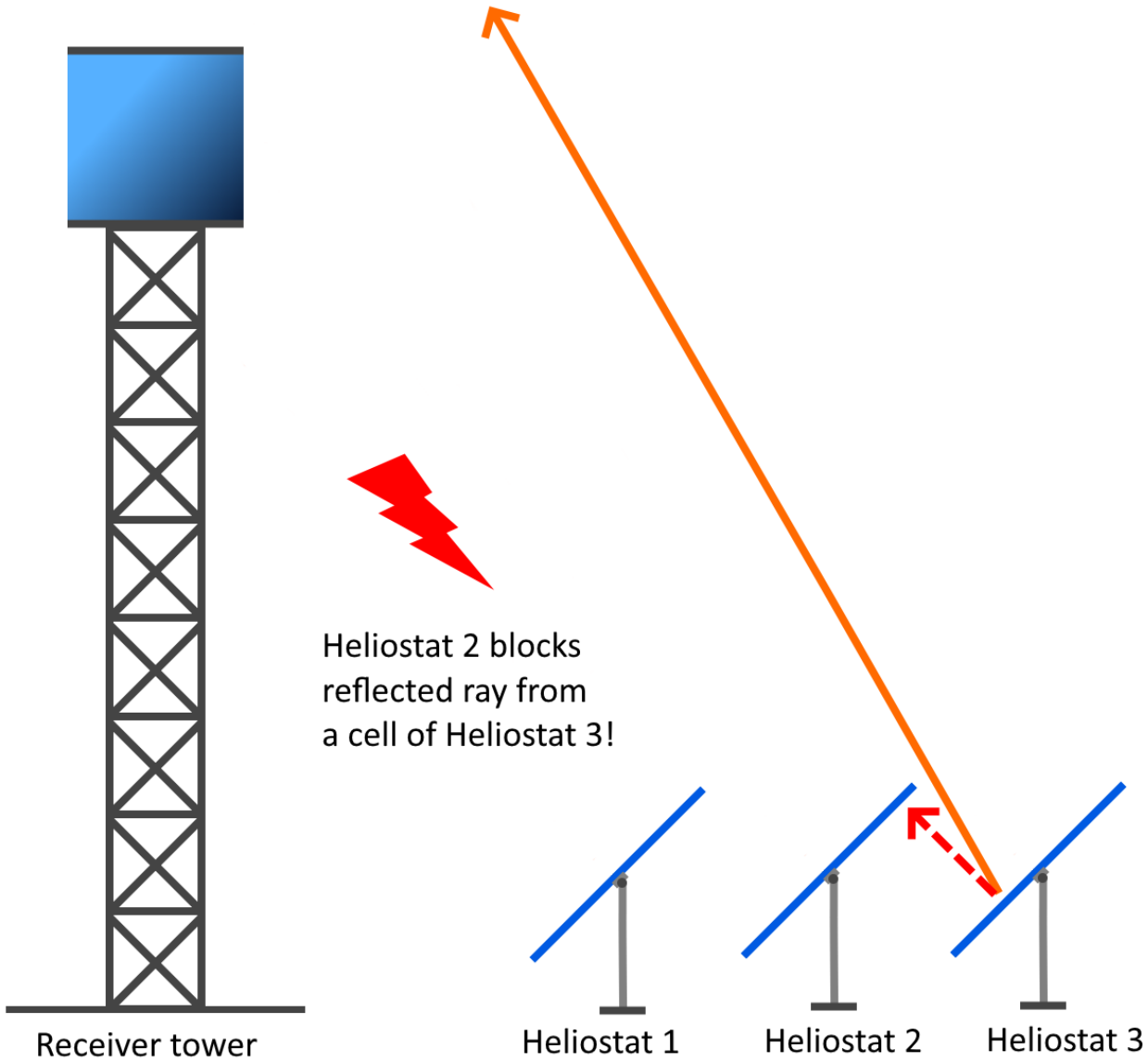
Tower Shading



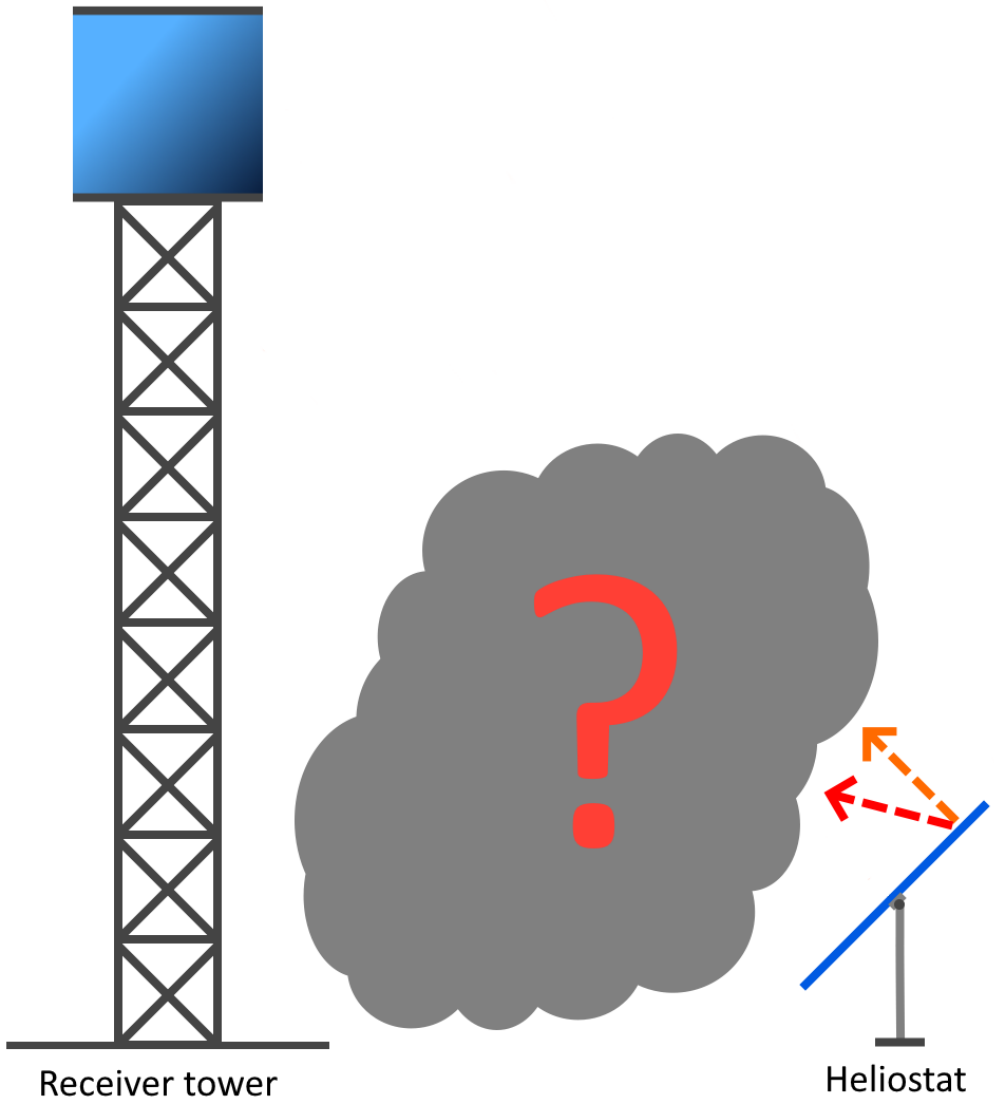
Heliostat Shading



Heliostat Blocking



What other structures could shade or block facet cells?



Obstacles Shading and Blocking

In order to address energy losses caused by obstructions on the field we introduce new mean efficiencies η_{os}, η_{ob}. We expand the shading and blocking checks executed during the simulation by additionally tracing for intersections with simulated obstacles. The new efficiencies are then calculated in a similar manner total from shaded or blocked mirror areas $A_{\text{obstacle shaded}}$, $A_{\text{obstacle blocked}}$.

$$\eta_{\text{os}} = 1 - \frac{A_{\text{obstacle shaded}}}{A_{\text{total}}} \qquad \eta_{\text{ob}} = 1 - \frac{A_{\text{obstacle blocked}}}{A_{\text{total}}}$$

Heliostat-related efficiencies were renamed to η_{ms} and η_{mb}.

Obstacle Model

Every obstacle in the simulation is introduced as a tuple

$$O = (p_0, P, \text{isShading}, \text{isBlocking})$$

- $p_0 \in \mathbb{R}^3$ is the position of obstacle within the simulation
- P is a list of different parameters that define the geometry of the object
- isShading is a tracing procedure that checks if obstacle located at p_0, defined by parameters from P intersects the Sun ray.
- isBlocking is a tracing procedure that checks if obstacle located at p_0, defined by parameters from P intersects the reflected ray **before** it reaches the receiver.

Polygonal obstacle

Two types of obstacles were introduced within this work: polygonal and cylindrical. Polygonal obstacle is defined by vertices from parameter V.

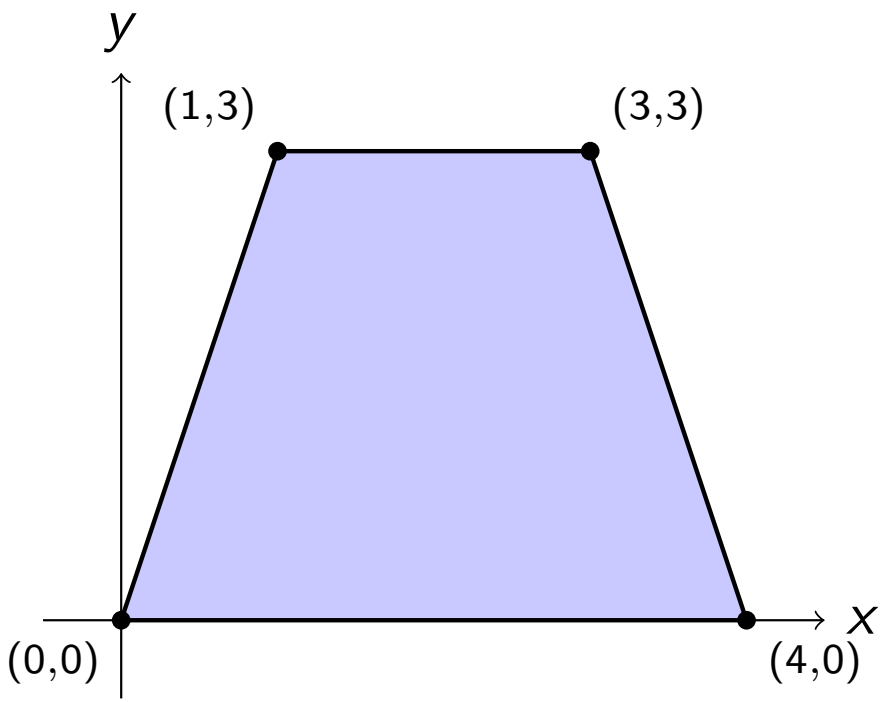


Figure: 2D polygon with vertices $V = \{(0,0),(1,3),(3,3),(4,0)\}$.

Polygonal obstacle

Given position p_0 and a surface normal vector parameter $\vec{v_n}$ we define the 3D plane of the obstacle. Then we construct an orthonormal basis on it and project 2D polygon onto it.

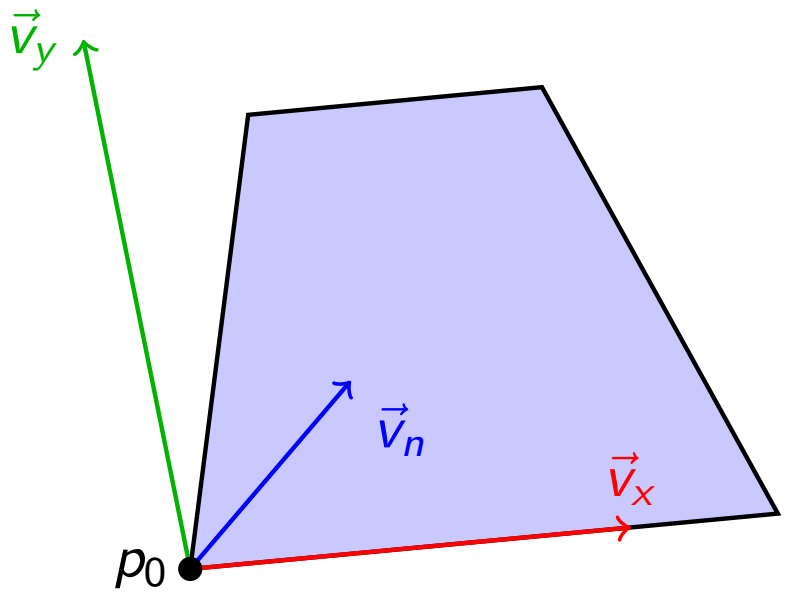


Figure: Polygonal obstacle with local coordinate axes $\vec{v}_x$ and $\vec{v}_y$.

Blocking and Shading Algorithms

We check for intersection between the ray and a minimal bounding sphere constructed around the polygon. This step used as a pre-filter, avoiding further more expensive computations.

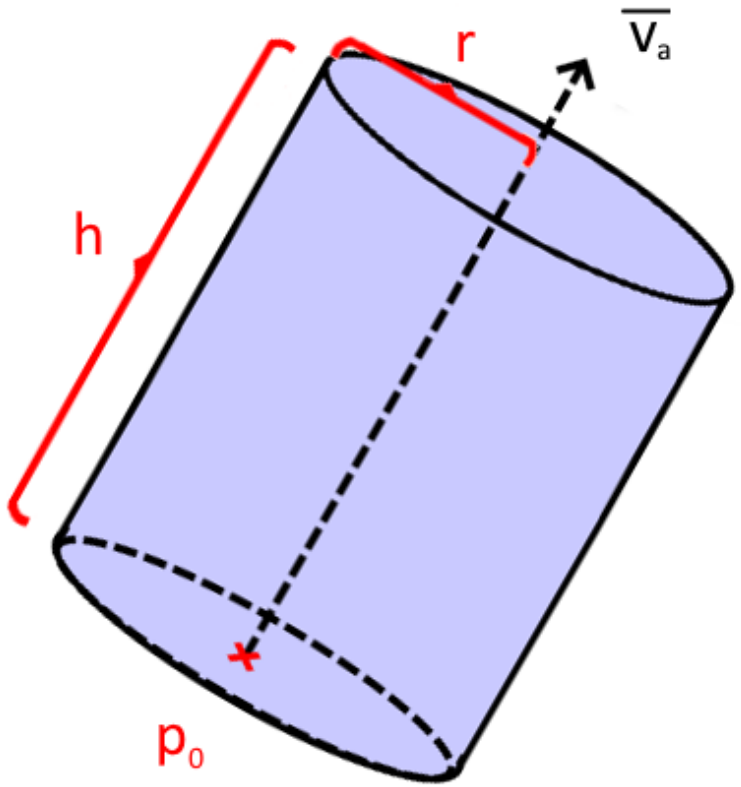
If ray did hit the sphere, we find an intersection point between the ray and the plane of the polygon

Intersection point is projected onto local 2D frame specified by axes $\vec{v_x}$, $\vec{v_y}$. Then, we perform a point-in-polygon (even-odd) test on a 2D intersection point, determining whether or not the polygon was intersected by the ray.

Additionally, in case of blocking test, we compare the ray-plane intersection point to the target point along the ray, ensuring that receiver is not "behind" the obstacle.

Cylindrical obstacle

Finite cylinder positioned at p_0 with radius r and height h. Its axis is oriented along vector $\vec{v_a}$.



Blocking and Shading Algorithms — Cylinder

We substitute equation of traced ray into an equation of a point within an *infinite* cylinder of radius r and get a quadratic equation.

If the equation has roots we check for intersection a *finite* cylinder by checking for hits with caps of the cylinder.

We still check for special cases: ray parallel to the cylinder and equation has no real roots.

In case of blocking checks we compare every intersection point computed during the execution against the target point.

Model Integration

- We now add input parameters for every obstacle into an ordered list $\mathbb{O}$.
- During the simulation, ray tracers can now iterate through $\mathbb{O}$ and call isShading, isBlocking procedures for every Sun ray or any reflected ray for each facet cell.
- Translator module exports obstacles in $\mathbb{O}$ to GPU device.
- Before the simulation every obstacle $O \in \mathbb{O}$ is additionally provided with a **name** parameter

Model Validation

Validity of the obstacle model was evaluated within a minimalistic power plant setup with a receiver tower and a single heliostat with one $1\,\mathrm{m}^2$ rectangular facet.

- Representative cylindrical and polygonal obstacles were introduced to the simulation scene. Their position p_0 was systematically changed, expecting full, partial and no shading effects. Similar placements were made to test blocking cases.
- For every obstacle-placement combination 3 simulations were run: using a convolution ray tracer, Monte Carlo with 100 rays per m^2 and Monte Carlo with 10000 rays per m^2

Results using convolution

Obstacle	$A_{\text{obstacle shaded}}$	$A_{\text{obstacle blocked}}$	A_{total}
Fully blocking polygon	–	$1\,\text{m}^2$	$1\,\text{m}^2$
Partially blocking polygon	–	$0.5\,\text{m}^2$	$1\,\text{m}^2$
Non-blocking polygon	–	0	$1\,\text{m}^2$
Fully shading polygon	$1\,\text{m}^2$	–	$1\,\text{m}^2$
Partially shading polygon	$0.5\,\text{m}^2$	–	$1\,\text{m}^2$
Non-shading polygon	0	–	$1\,\text{m}^2$
Fully blocking cylinder	–	$1\,\text{m}^2$	$1\,\text{m}^2$
Partially blocking cylinder	–	$0.5\,\text{m}^2$	$1\,\text{m}^2$
Non-blocking cylinder	–	0	$1\,\text{m}^2$
Fully shading cylinder	$1\,\text{m}^2$	–	$1\,\text{m}^2$
Partially shading cylinder	$0.498\,\text{m}^2$	–	$1\,\text{m}^2$
Non-shading cylinder	0	–	$1\,\text{m}^2$

Results using Monte Carlo, 100 rays per m^2

Obstacle	$A_{\text{obstacle shaded}}$	$A_{\text{obstacle blocked}}$	A_{total}
Fully blocking polygon	–	$1\,m^2$	$1\,m^2$
Partially blocking polygon	–	$0.492\,m^2$	$1\,m^2$
Non-blocking polygon	–	–	$1\,m^2$
Fully shading polygon	$1\,m^2$	–	$1\,m^2$
Partially shading polygon	$0.5\,m^2$	–	$1\,m^2$
Non-shading polygon	0	–	$1\,m^2$
Fully blocking cylinder	–	$1\,m^2$	$1\,m^2$
Partially blocking cylinder	–	$0.531\,m^2$	$1\,m^2$
Non-blocking cylinder	–	0	$1\,m^2$
Fully shading cylinder	$1\,m^2$	–	$1\,m^2$
Partially shading cylinder	$0.498\,m^2$	–	$1\,m^2$
Non-shading cylinder	0	–	$1\,m^2$

Results using Monte Carlo, 10000 rays per m^2

Obstacle	$A_{\text{obstacle shaded}}$	$A_{\text{obstacle blocked}}$	A_{total}
Fully blocking polygon	–	$1\,m^2$	$1\,m^2$
Partially blocking polygon	–	$0.503\,m^2$	$1\,m^2$
Non-blocking polygon	–	–	$1\,m^2$
Fully shading polygon	$1\,m^2$	–	$1\,m^2$
Partially shading polygon	$0.5\,m^2$	–	$1\,m^2$
Non-shading polygon	0	–	$1\,m^2$
Fully blocking cylinder	–	$1\,m^2$	$1\,m^2$
Partially blocking cylinder	–	$0.498\,m^2$	$1\,m^2$
Non-blocking cylinder	–	0	$1\,m^2$
Fully shading cylinder	$1\,m^2$	–	$1\,m^2$
Partially shading cylinder	$0.498\,m^2$	–	$1\,m^2$
Non-shading cylinder	0	–	$1\,m^2$

Mean Efficiencies Waterfall

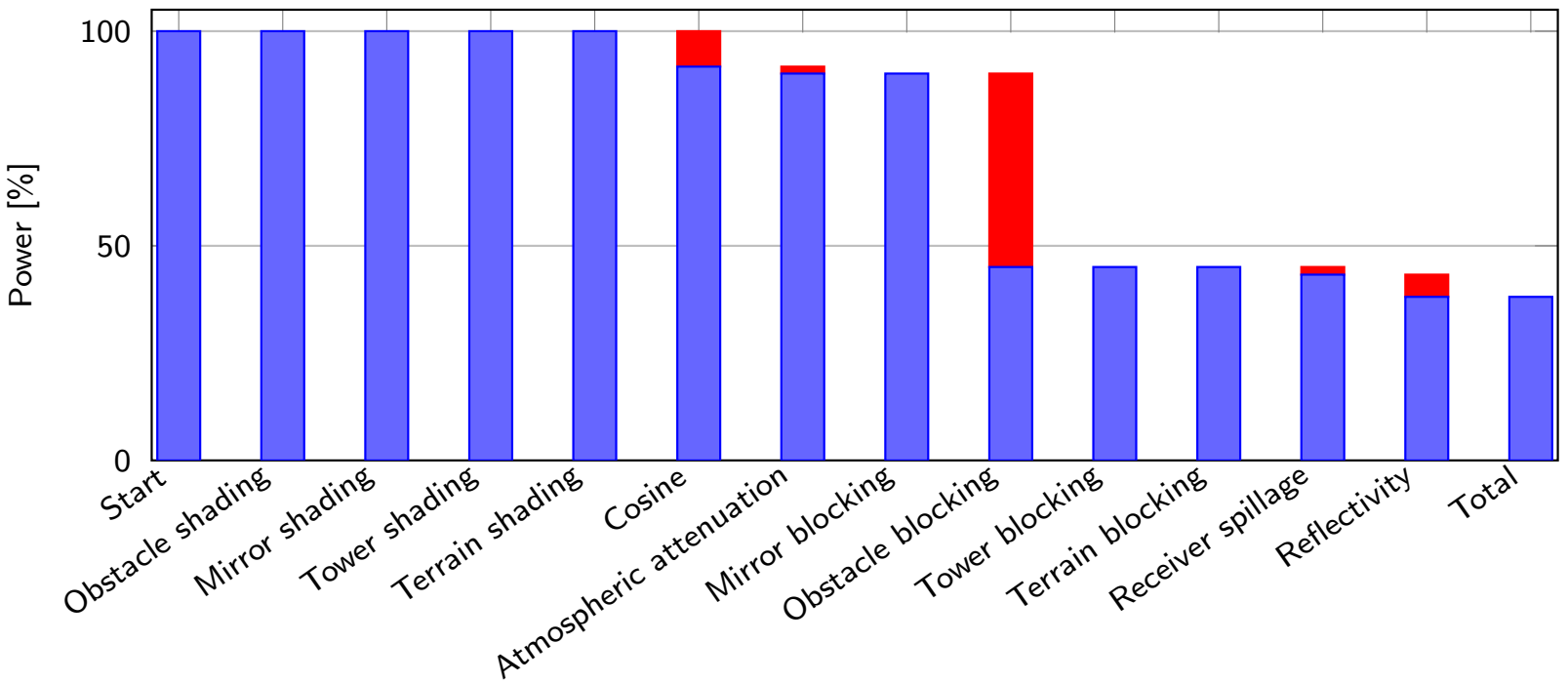
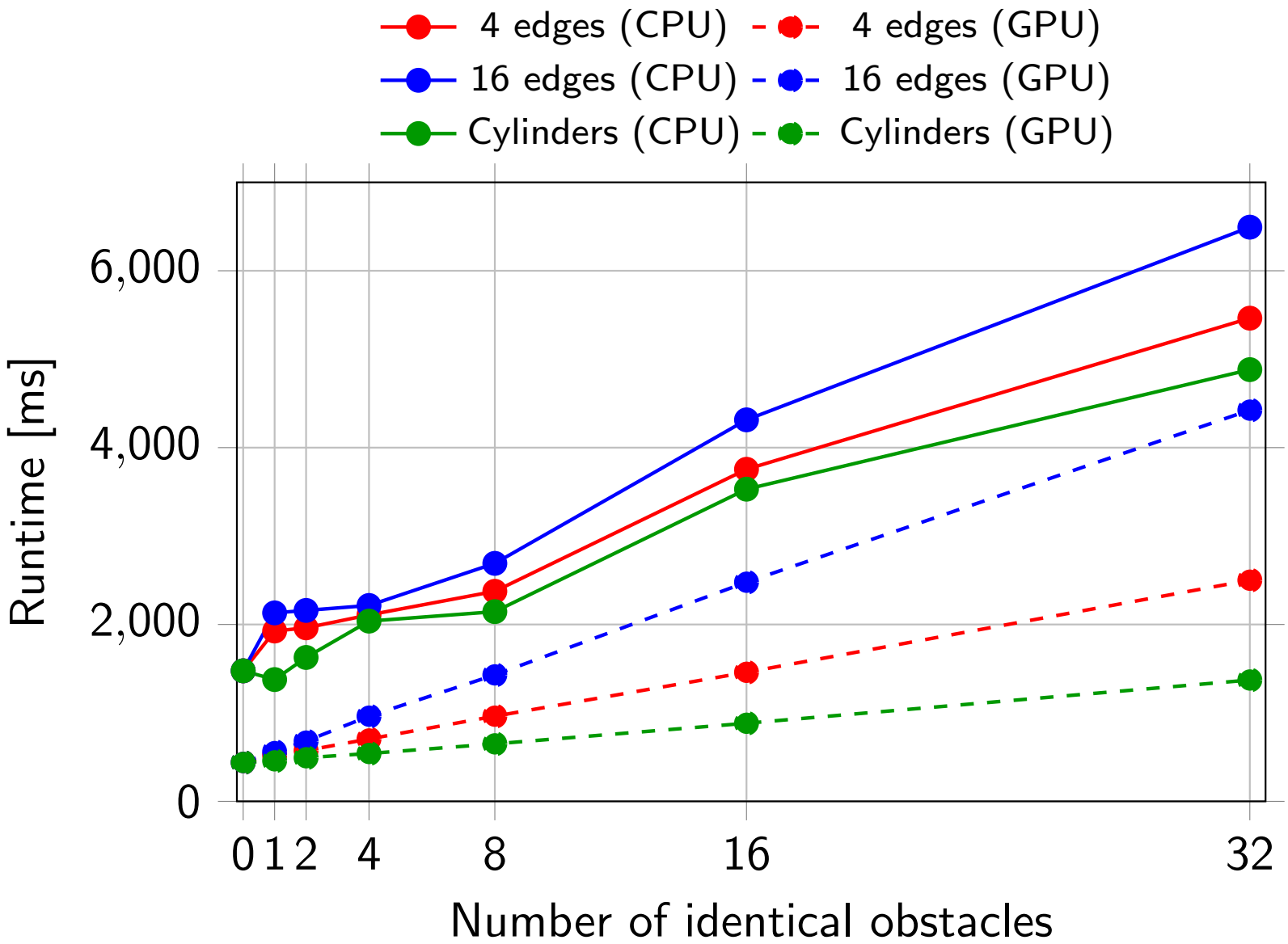


Figure: Optical losses from 100% incident power. Partially blocking polygon on Convolution

Runtime Evaluation

- In order to evaluate impact of obstacles on ray tracer runtime, series of simulations with obstacles added to the PS10 scene were run using Monte Carlo on CPU and GPU.
- These obstacles were designed to trigger worst-case blocking checks of all 624 heliostats. Up to 32 cylinders, 4-edge polygons and 16-edge polygons were placed between the heliostat field and the receiver tower

Results


4 edges (CPU)
4 edges (GPU)
16 edges (CPU)
16 edges (GPU)
Cylinders (CPU)
Cylinders (GPU)
Runtime [ms]
6,000
4,000
2,000
0
0 1 2 4 8 16 32
Number of identical obstacles

Further Improvements

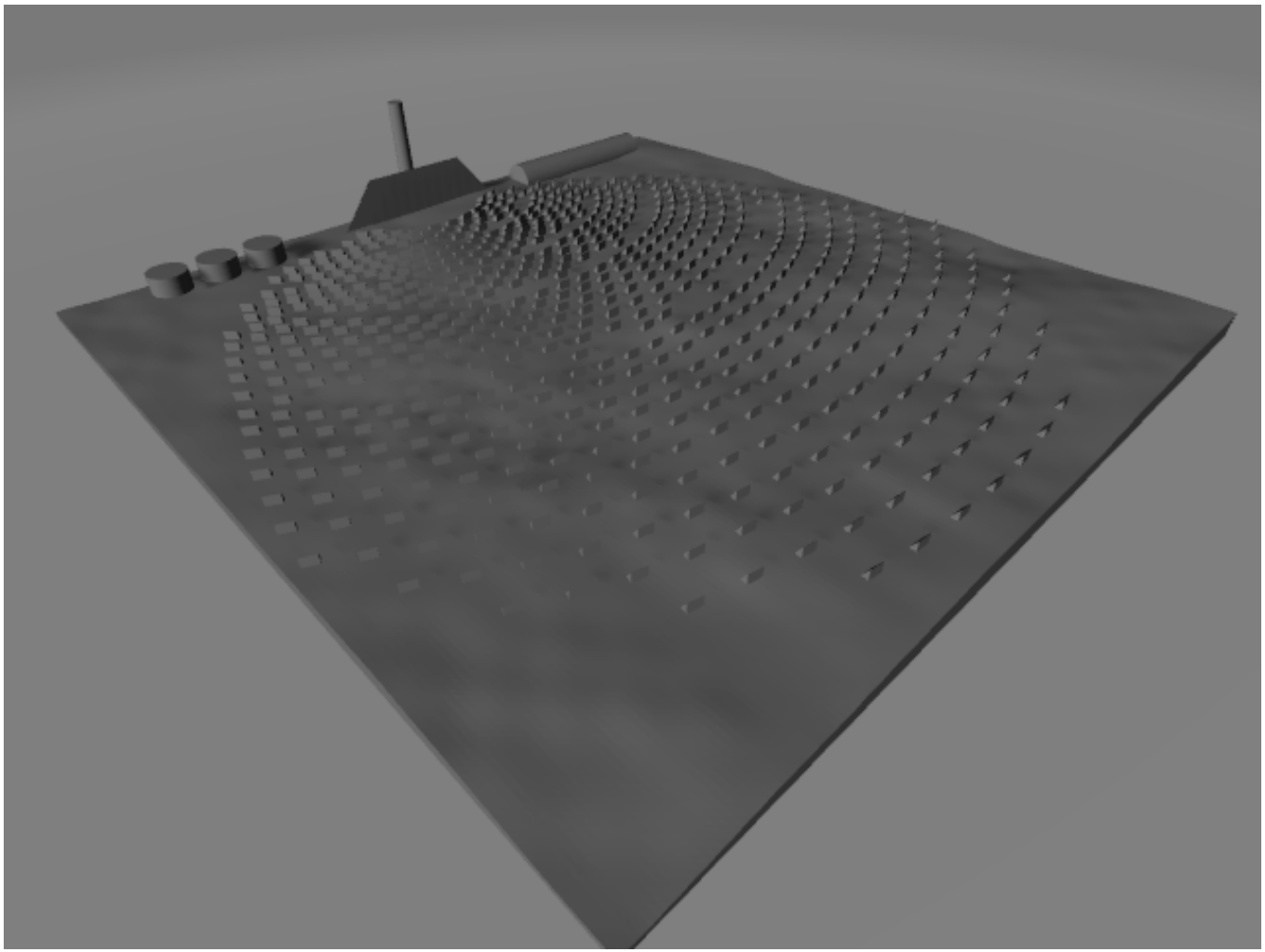
Apart from newly introduced obstacle model, several outstanding facet-related issues have been addressed. Main cause of this issues was unfinished integration of new parameter scheme, which describes each facet individually instead of relying on global flags.

- Some of SunFlower's services were relying on a deprecated parameter schema, when processing input data for facets
- 3D-Printing tool relied on legacy parameter schema, leading to output of STL files being not possible.
- Tonatiuh script generation tool used to rely on deprecated global parameters as well, making export to Tonatiuh also not possible

3D-Printing

- STL is a common 3D-Printing format
- An algorithm that iterates through every individual facet parameters list was implemented.
- It correctly calculates the height and width of a minimal bounding box that includes every facet, without relying on global parameters.
- As new algorithm solved the integration issue, obstacles were also introduced to STL scene as triangular fan meshes.

PS10 power plant scene with 5 obstacles

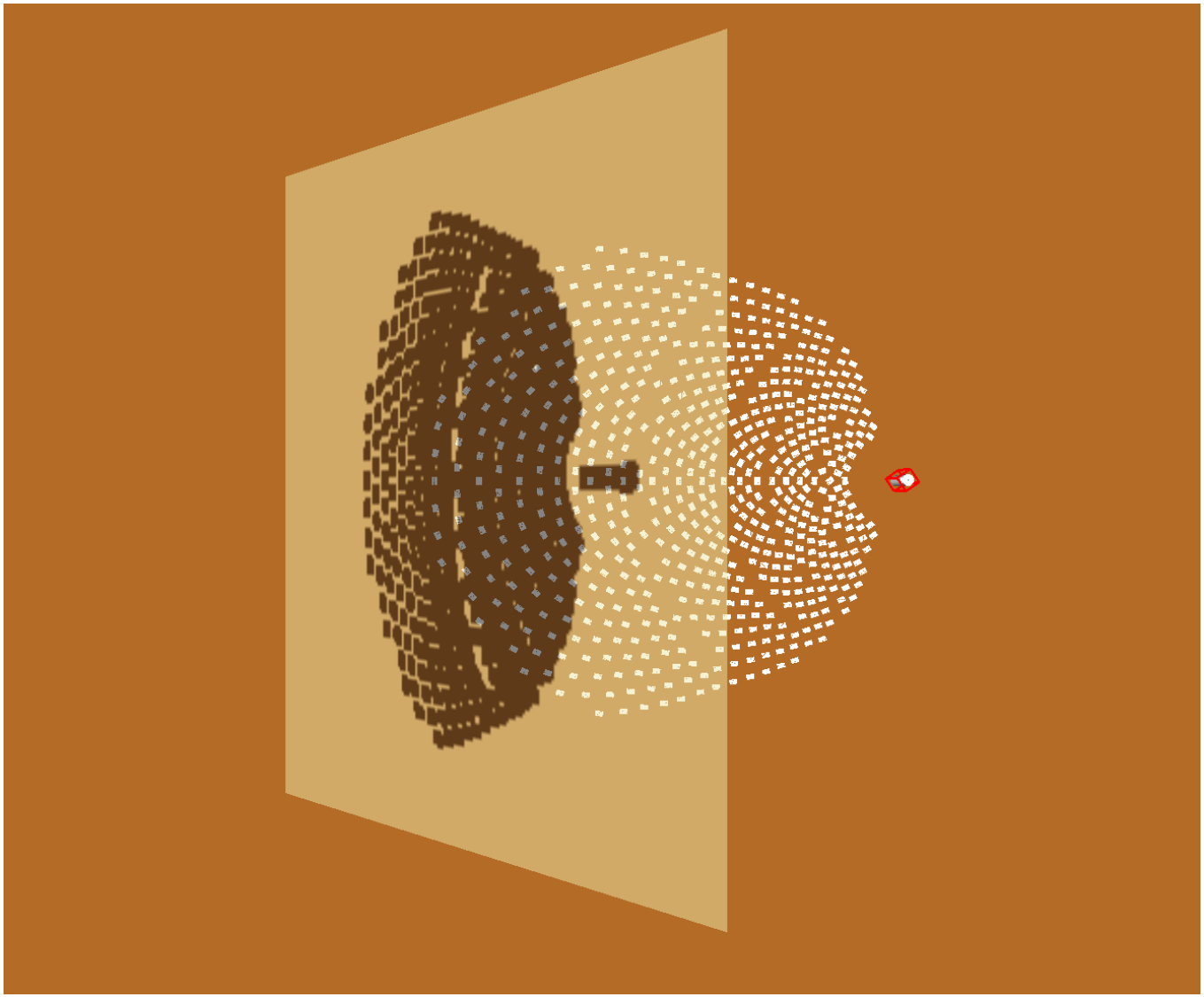


Screenshot made in Microsoft 3D Viewer

Tonatiuh

- Tonatiuh is an open-source ray tracing software designed for the simulation of solar concentrating systems.
- Scene from SunFlower can be exported as a script, that will create this in Tonatiuh.
- Update of facet parameters service was referring to, has resolved the issue.

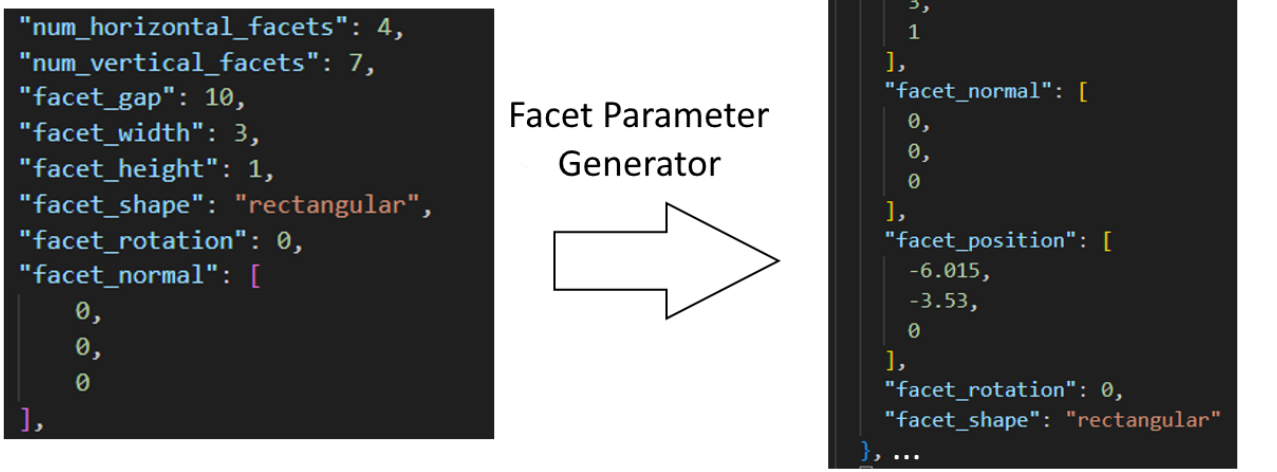
PS10 scene imported into Tonatiuh 2.2.4



Screenshot made in Tonatiuh 2.2.4

Automatic Generation of Facet Parameters

- Specifying every single facet, especially its position on the surface can be labour intensive
- We address this problem by introducing a new service that generates individual facet parameters from global values set once for every facet



Simplified input scheme

List "facets" includes further 27 individual facet fields

Summary

Summary

- Obstacle model enables simulation of obstructions within the CRS field
- Outstanding facet-related issues have been resolved

Outlook

- New shapes of obstacles
- New visualization methods for import/export of scenes

Summary

Summary

- Obstacle model enables simulation of obstructions within the CRS field
- Outstanding facet-related issues have been resolved

Thank you!

Outlook

- New shapes of obstacles
- New visualization methods for import/export of scenes

References I

[1] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering, 4th*. A K Peters/CRC Press, 2018, pp. 966–967. ISBN: 978-1138627000.

[2] Lukas Aldenhoff. "Raytracer for Central Receiver Systems using GPU". Master's Thesis. RWTH Aachen University, 2021.

[3] Manuel J. Blanco, Juana M. Amieva, and Azael Mancillas. "The Tonatiuh Software Development Project: An Open Source Approach to the Simulation of Solar Concentrating Systems". In: *ASME International Mechanical Engineering Congress and Exposition Proceedings Series*. 2005. DOI: 10.1115/IMECE2005-81859.

[4] Paul Bourke. *Calculating the Area and Centroid of a Polygon*. http://paulbourke.net/geometry/polygonmesh/. 1988.

References II

[5] João P. Cardoso et al. "New functionalities for the Tonatiuh ray-tracing software". In: *SolarPACES 2017: International Conference on Concentrating Solar Power and Chemical Energy Systems*. 2017. DOI: 10.1063/1.5067212.

[6] Christer Ericson. *Real-Time Collision Detection*. Morgan Kaufmann, 2005, pp. 194–198. ISBN: 978-1558607323.

[7] Florian Hövelmann. "Accelerated Raytracer for Solar Tower Power Plants". Bachelor's thesis. RWTH Aachen University, 2019.

[8] Florian Hövelmann, Lukas Aldenhoff, and Pascal Richter. "Analytic Ray Tracer on GPU for Central Receiver Systems". In: *Journal of Renewable Energy* (2023). DOI: 10.1016/j.solener.2023.112149.

References III

[9] Florian Hövelmann, Lukas Aldenhoff, and Pascal Richter. "Integrated Convolution Ray Tracer on GPU for Central Receiver Systems". In: *Journal of Renewable Energy* (2023). DOI: 10.2139/ssrn.4227667.

[10] International Energy Agency. *Global Energy Review 2025*. Tech. rep. Paris: International Energy Agency (IEA), 2025.

[11] Rafael Osuna et al. "PS10, Construction of a 11MW solar thermal tower plant in Seville, Spain". In: *Conference paper: SolarPACES*. 2006.

[12] REN21. *Renewables Global Status Report 2025: Global Overview*. Tech. rep. Paris: REN21 Secretariat, 2025. URL: https://www.ren21.net/gsr-2025/.

[13] Pascal Richter. "Simulation and Optimization of Solar Thermal Power Plants". Dissertation. RWTH Aachen University, 2017.

References IV

[14] David Elias Walter Schütte. "Modeling and Simulation of Tilt-and-Roll Heliostats in Central Receiver Systems". Bachelor's Thesis. RWTH Aachen University, 2024.

[15] Tim Wendelin. "SolTRACE: A New Optical Modeling Tool for Concentrating Solar Optics". In: *ASME 2003 International Solar Energy Conference*. 2003. DOI: 10.1115/ISEC2003-44090.

[16] Haitao Zhang et al. "Tracking mechanism and cosine effect study of Module-Heliostat Solar Collector". In: *4th International Conference on Machinery, Materials and Information Technology Applications (ICMMITA 2016)*. Atlantis Press. 2017, pp. 469–474. DOI: 10.2991/icmmita-16.2016.87.