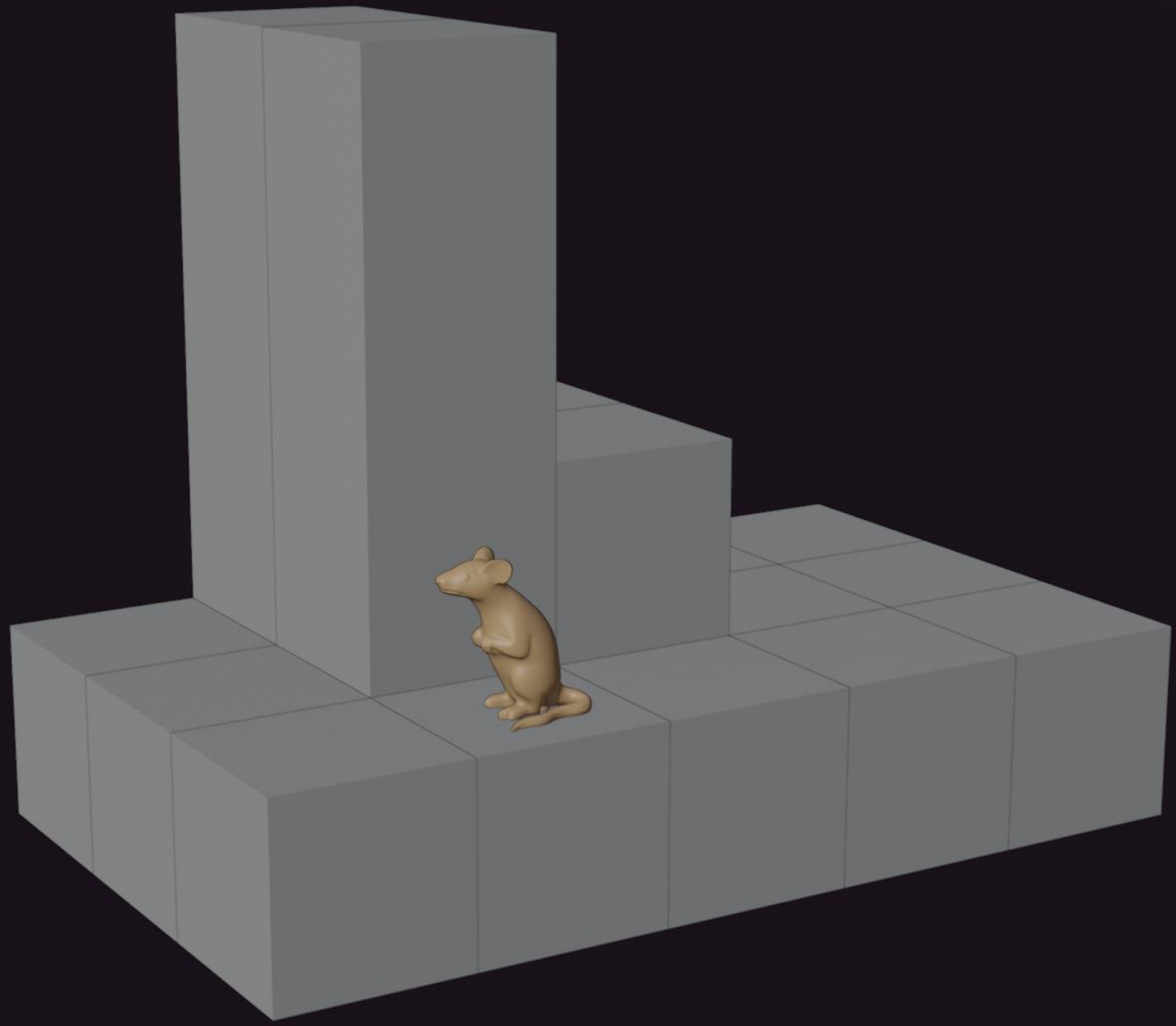
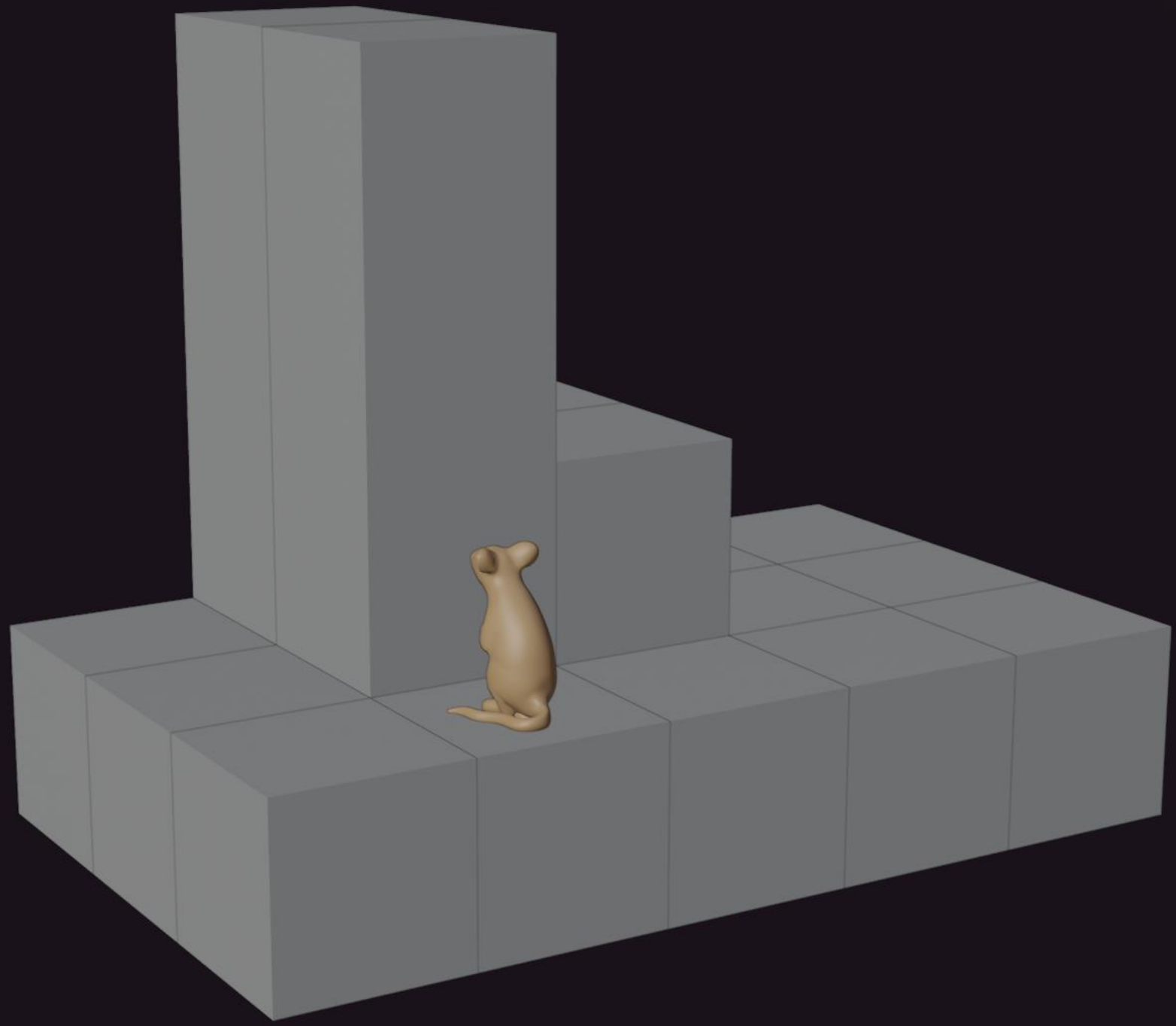
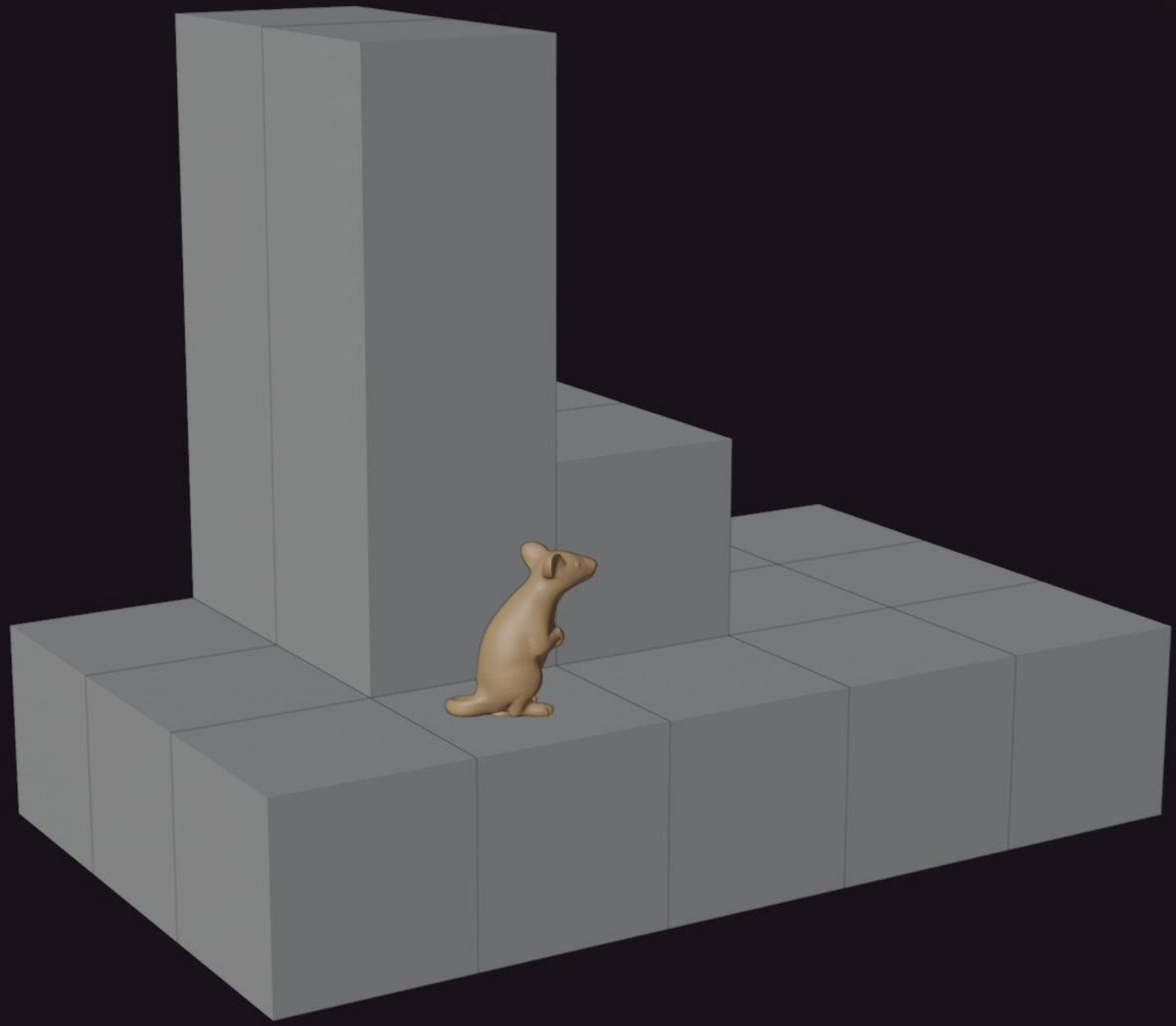
SQL is a Programming Language

Visibility in 3D Terrain

Patrizia Lenhart



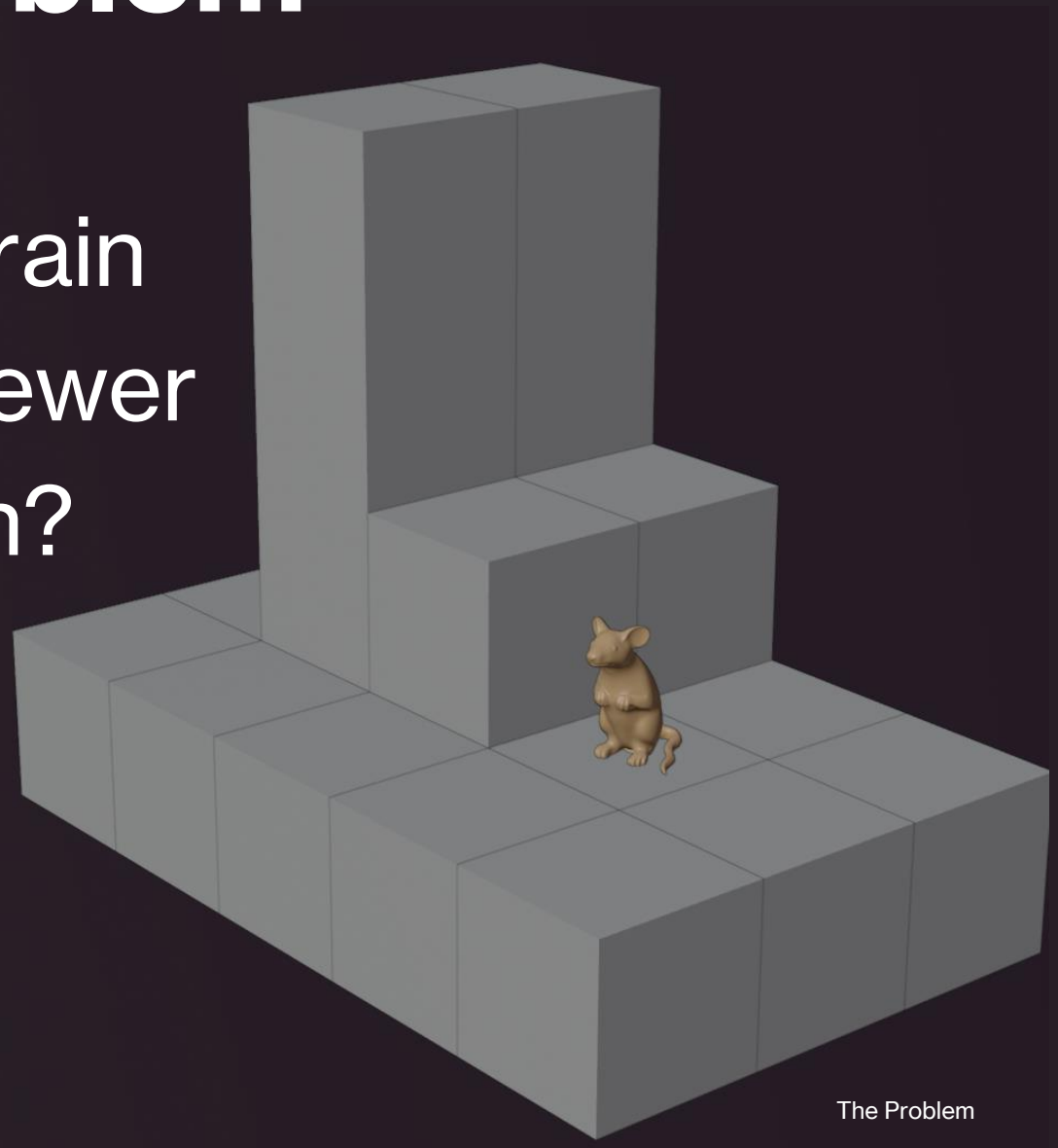




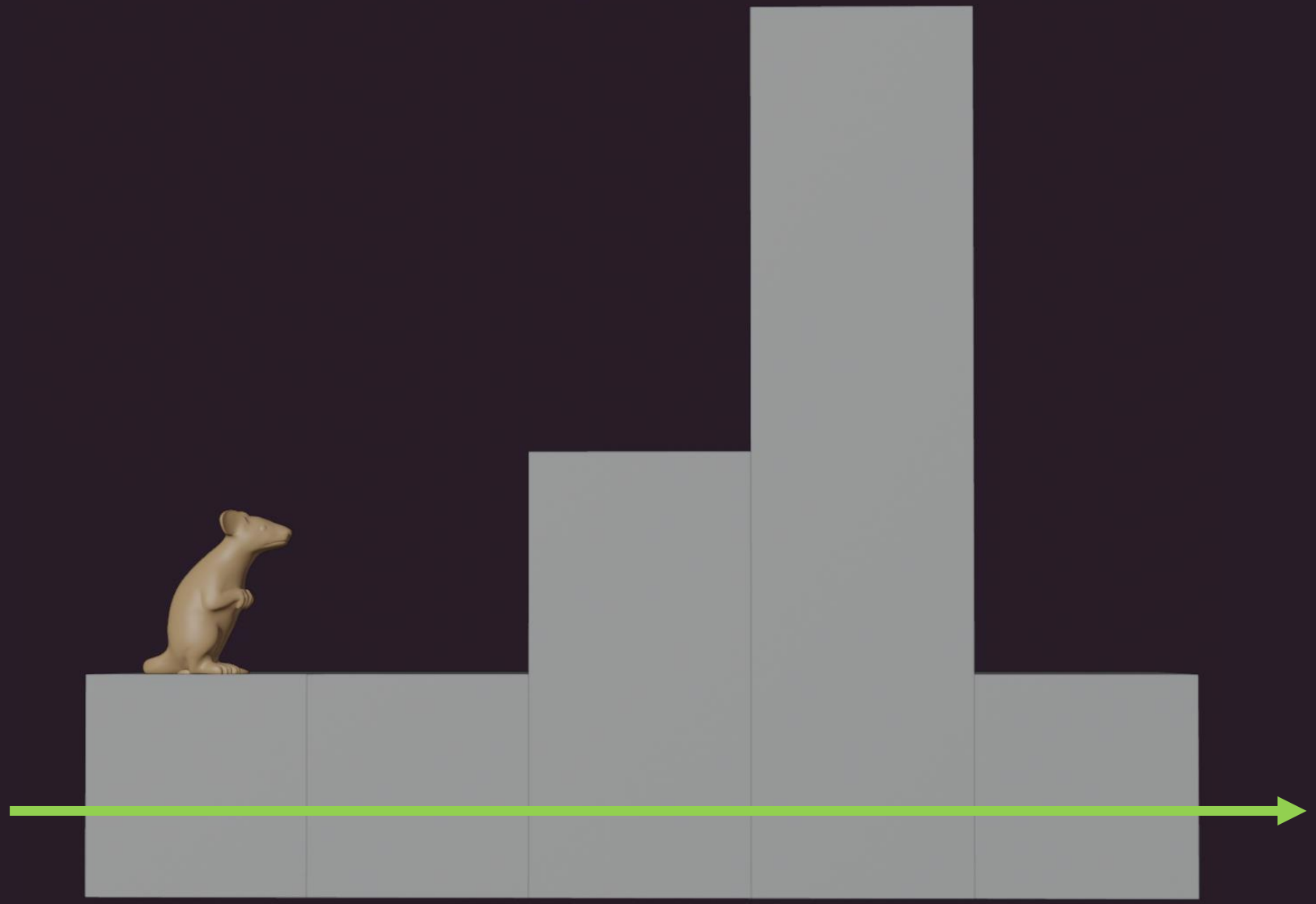

Which parts
of the map
can I see?

The Problem

Which points of a hilly terrain are visible from a given viewer position using a max-scan?



Visibility in 2D Terrain

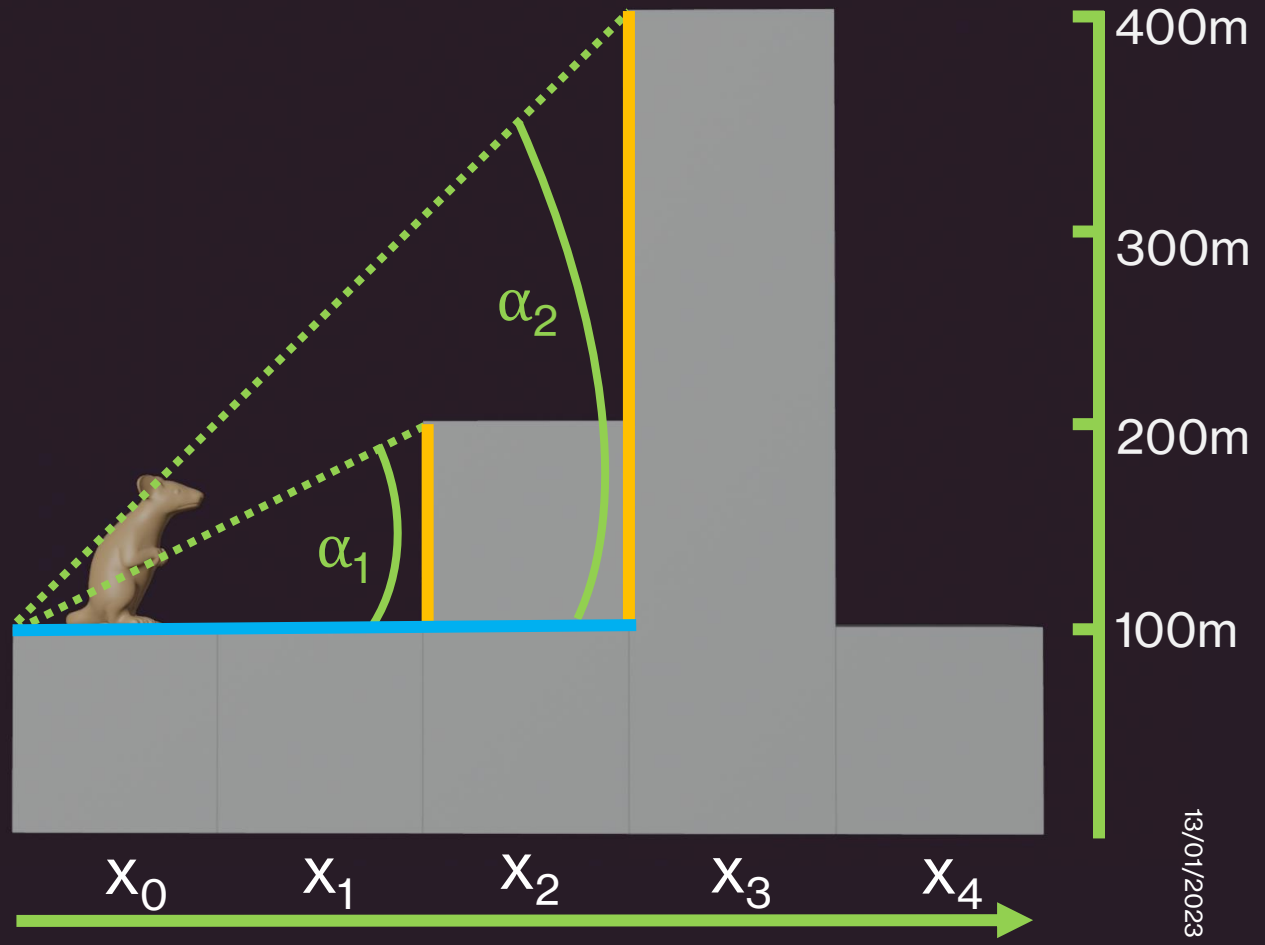


Visibility in 2D Terrain

For every point x :

$$\tan(\alpha) = \frac{Opposite}{Adjacent}$$

$$\alpha = \arctan\left(\frac{Opposite}{Adjacent}\right)$$



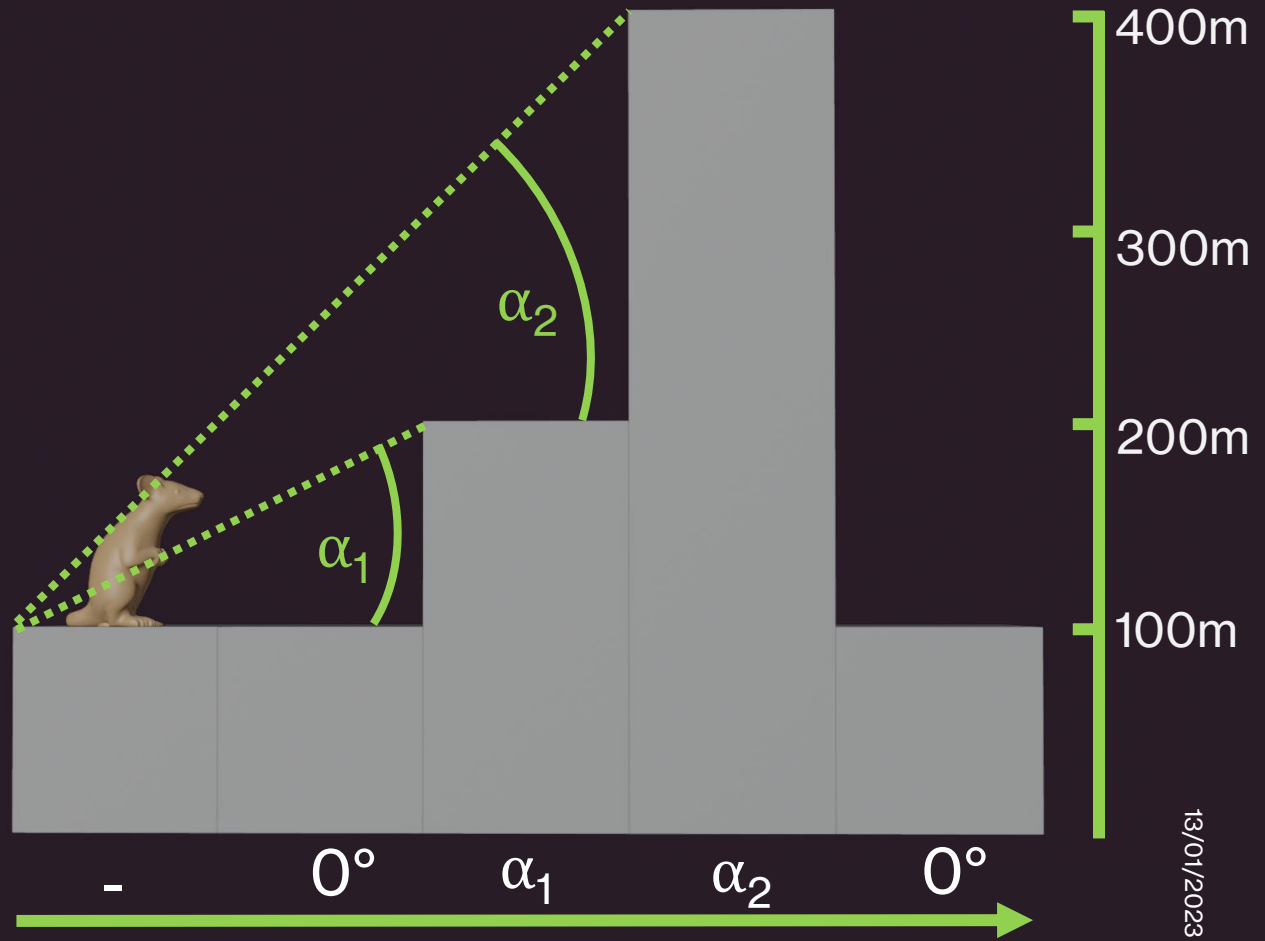
13/01/2023

Visibility in 2D Terrain

For every point x :

$$\tan(\alpha) = \frac{Opposite}{Adjacent}$$

$$\boldsymbol{\alpha = \arctan\left(\frac{Opposite}{Adjacent}\right)}$$

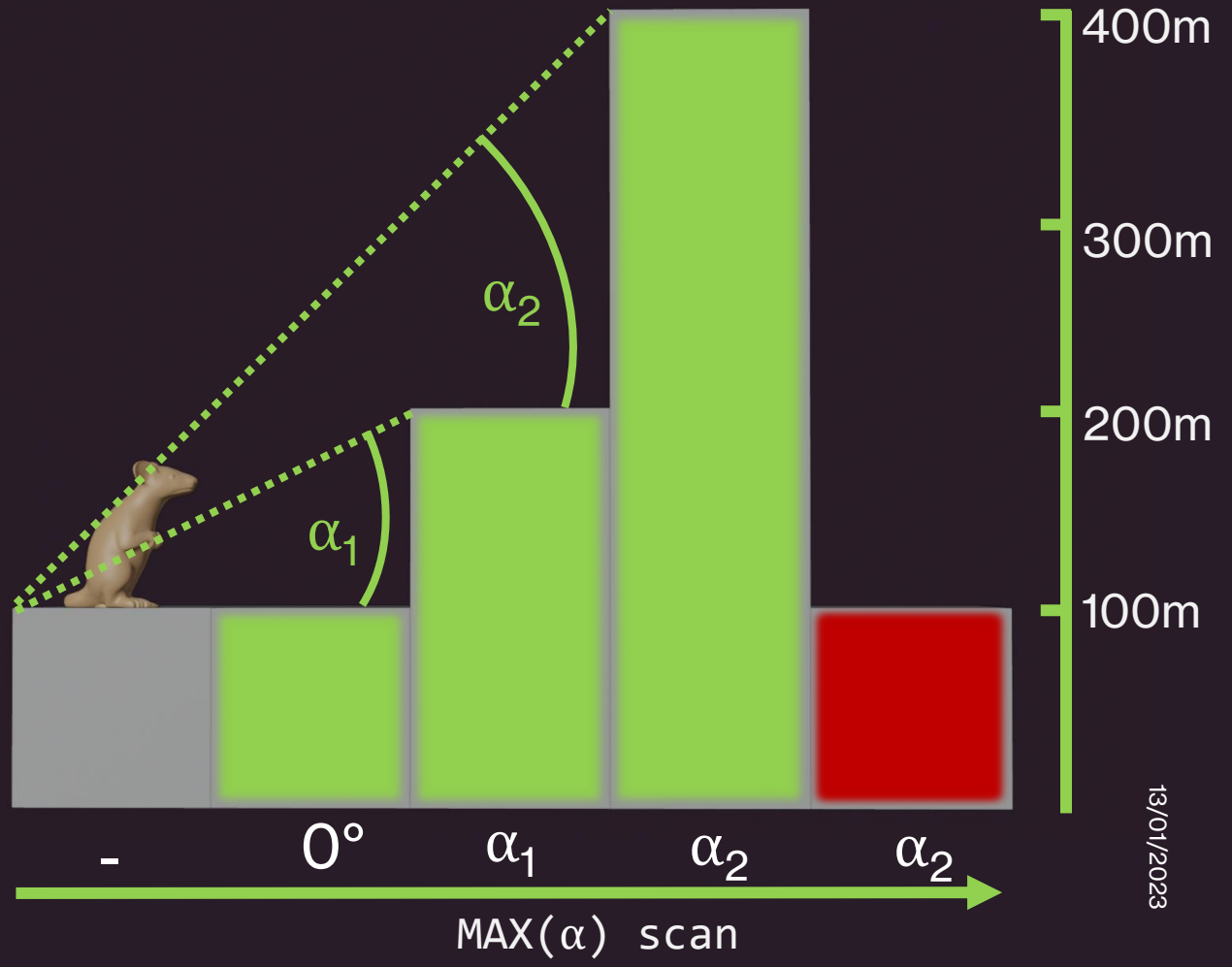


Visibility in 2D Terrain

For every point from viewing point :

If the current α <= `MAX(previous α)`

→ **Point is not visible**

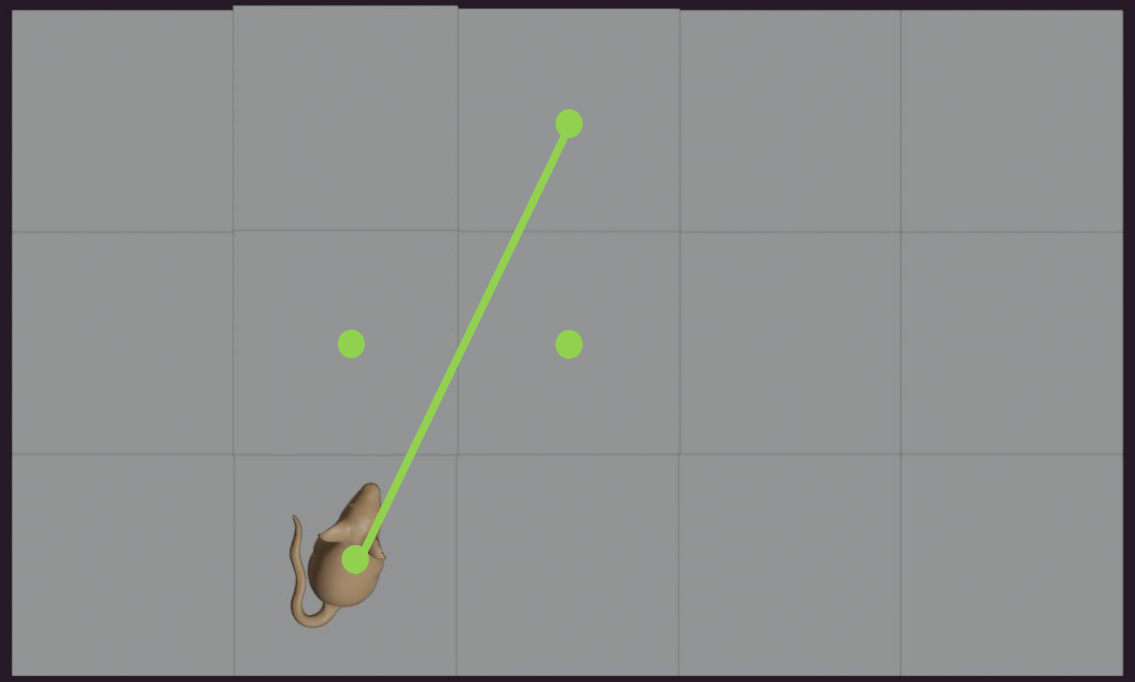


13/01/2023

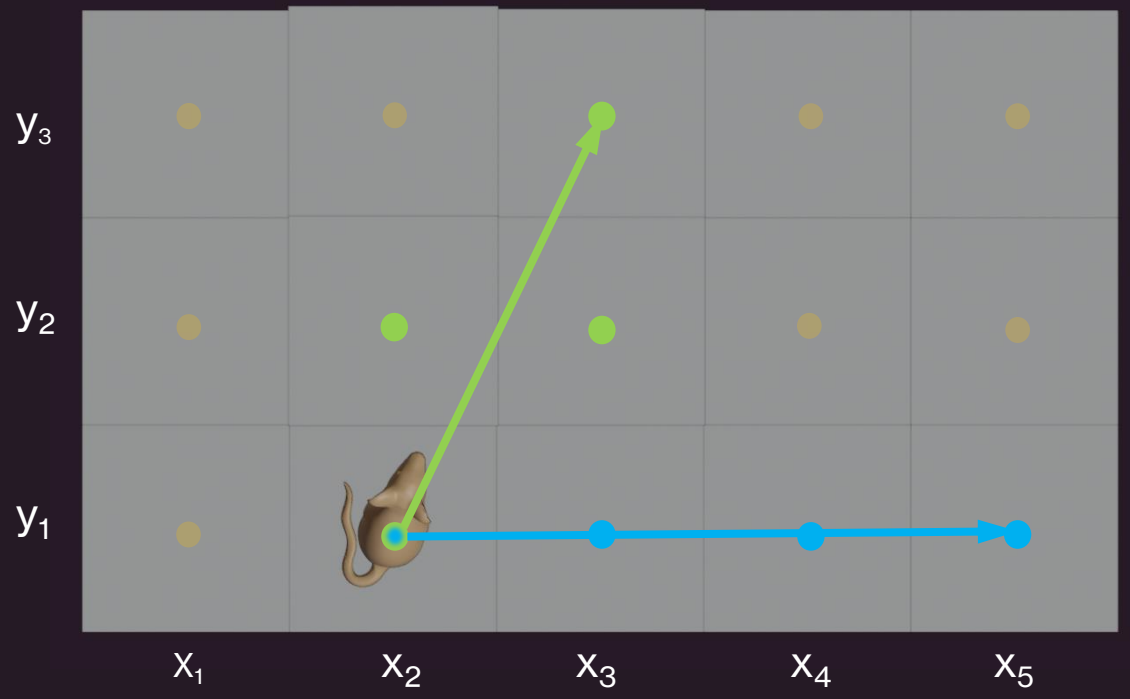
Visibility in 3D Terrain

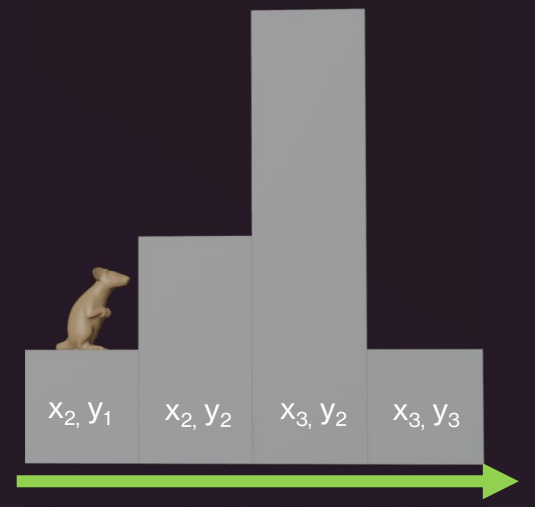
Just like 2D, but:

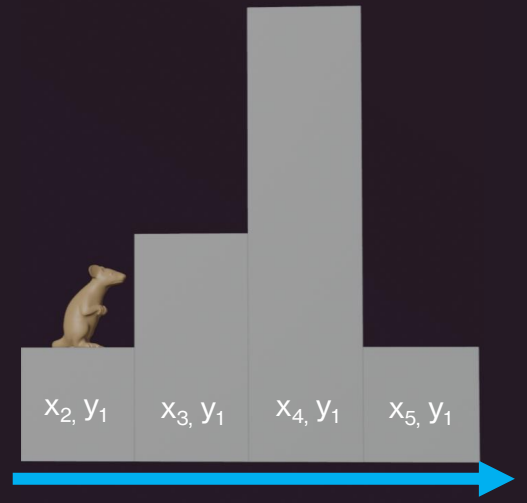
We must find all the boxes that our ‘line of sight’ hits



Visibility in 3D Terrain



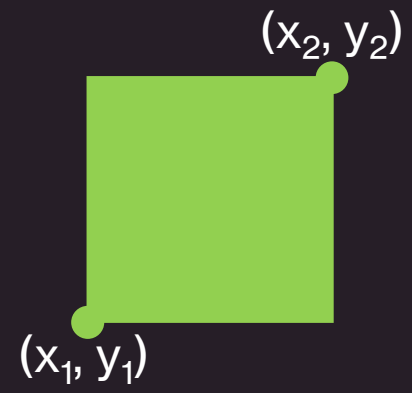




Visibility in 3D Terrain

Big thanks to Geometric Data-Types!
They include:

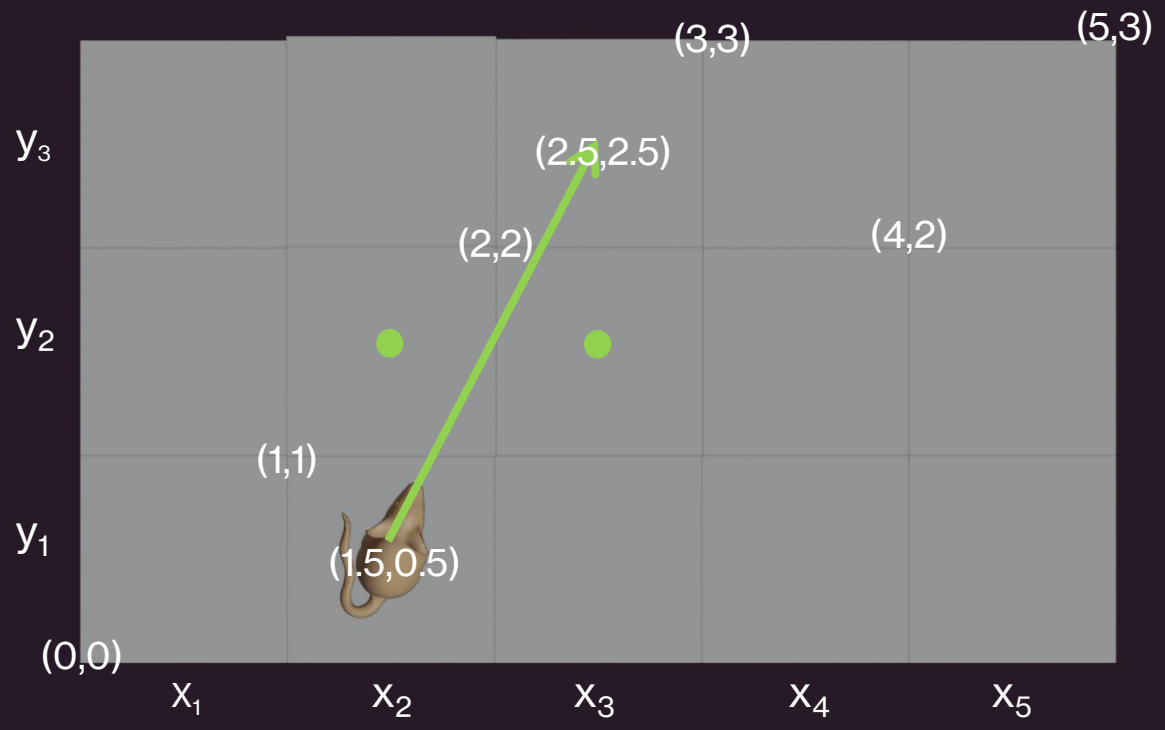
- Boxes = ($(x_1, y_1),(x_2, y_2)$)



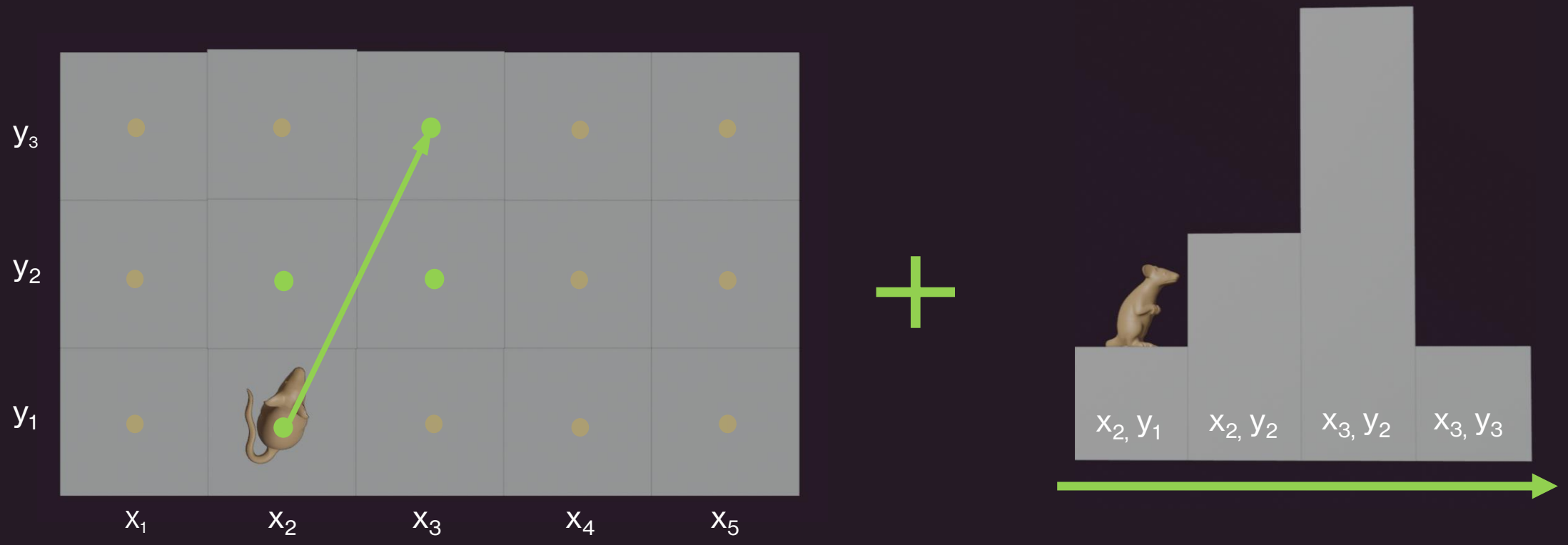
- Line Segments ($(x_1, y_1),(x_2, y_2)$)

Visibility in 3D Terrain

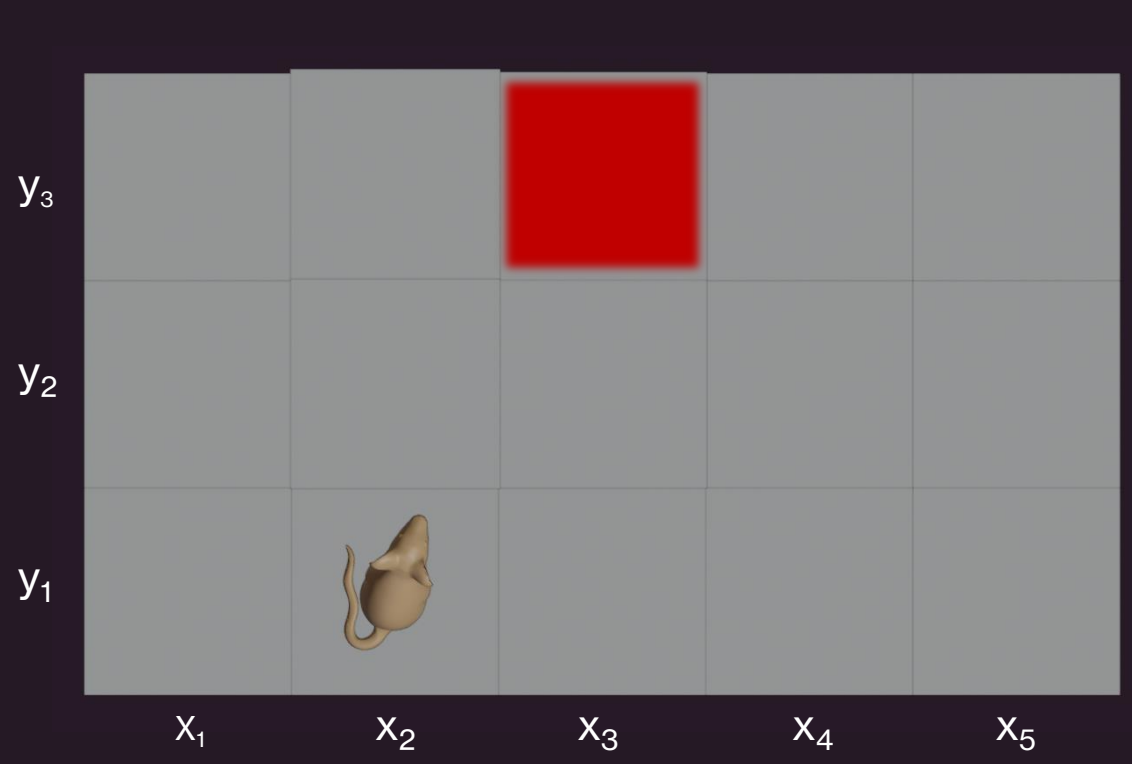
- `geometric_type ?# geometric_type → Boolean`
  Do these objects intersect?

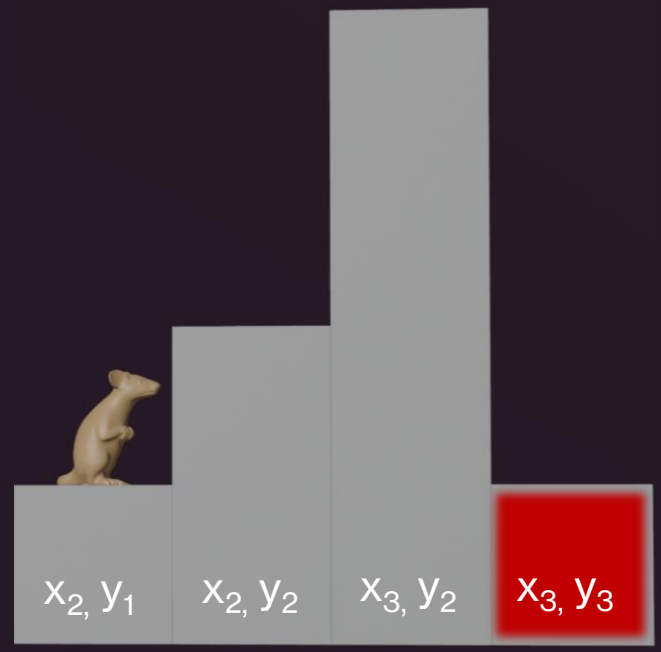


Visibility in 3D Terrain



Visibility in 3D Terrain





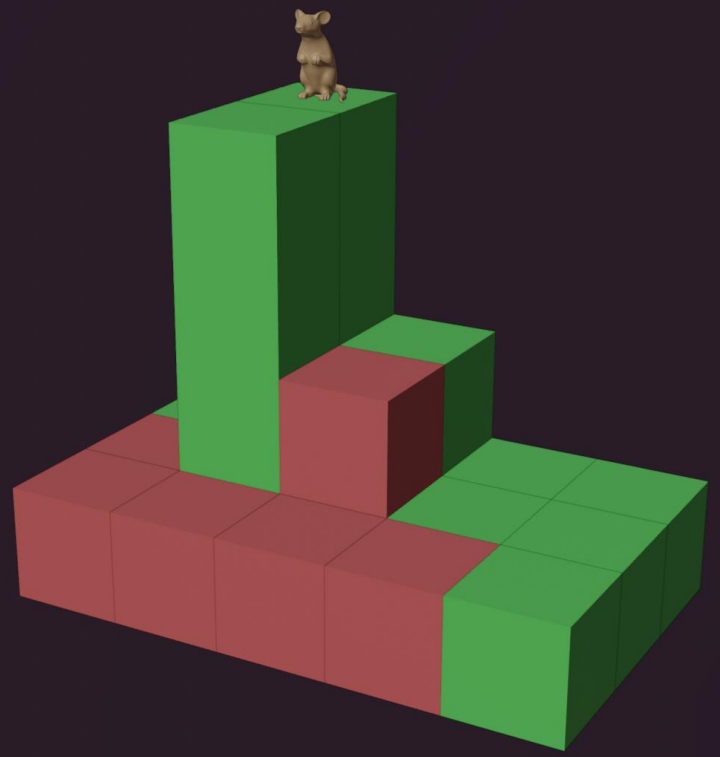
→ Point not visible

Visual- and Code Presentation

Code implemented using PostgreSQL

```
    ORDER BY x, y
),
make_ray(x1, y1, alt, box1, x2, y2, box2, angle, line, inter) AS (
    SELECT m1.x,
           m1.y,
           m1.alt,
           m1.box AS box1,
           m2.x AS x2,
           m2.y AS y2,
           m2.box AS box2,
           CASE
               WHEN m1.x = :posx AND m1.y = :posy THEN null        -- for negative angles to be seen and also covered t
               ELSE atan((alt-(SELECT alt FROM final_map AS m WHERE m.x = :posx AND m.y = :posy)) /(sqrt((:posx - m1.x)
               END AS angle,                                                                                    -- ca
           lseg(point((:posx - 0.5), (:posy - 0.5)), point(m2.x-0.5, m2.y-0.5)) AS line,
           f_intersect( b m1.box,  t lseg(point((:posx - 0.5), (:posy - 0.5)), point(m2.x-0.5, m2.y-0.5))) AS inter -
    FROM make_boxes AS m1, (SELECT x, y, box FROM make_boxes ) AS m2
),
max_scan(x, y, arr_angle, max) AS (
    SELECT r.x2, r.y2, array_agg(r.angle), MAX (r.angle) -- max-Scaaaaaan
    FROM make_ray AS r
    WHERE r.inter = true
    GROUP BY r.x2, r.y2
```

Visuals made in Blender



[Insert live demo here]