Internship Defense

Collaborative and immersive scatterplot rendering

Sandy Le Pape

28th of August 2019

Introduction

Presentation of the Organization

Synergy building hosts CSIRO's Data61 team, and the laboratory.

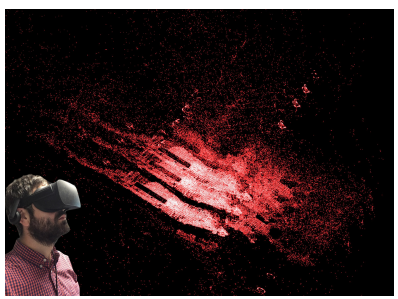
Figure: Synergy building

Figure: Laboratory logo

Context of the project

Main axis : improve interactivity with mass data through :

Immersive Analytics = Data Analytics + Virtual 3D Worlds



(a) *Immersive analytics* = (b) Data Analytics + (c) Immersive world

Figure: *Immersive Analytics* signification

Context : delivering a data asset for energy forecasting

↪ Focused on Australian consumption.

Limitations and objective

Resources available :

- datasets related to energy,
- data visualization projects.

Limitations of the existing projects :

- immersive visualisation cannot be shared,
- rendering for analysis can be improved.

Challenges

- Implement a collaborative solution for data exploration,
- Find tools for a better multiuser data analysis.

Unity, game engine used

Unity : allows to implement applications, incuding for VR.

Figure: *Unity* game engine

Table of contents

Overview of the existing basis

Datasets used

NEAR project

NEAR : platform gathering energy data, research and reports
↪ Deduce energy behaviour / consumer profiles in Australia.

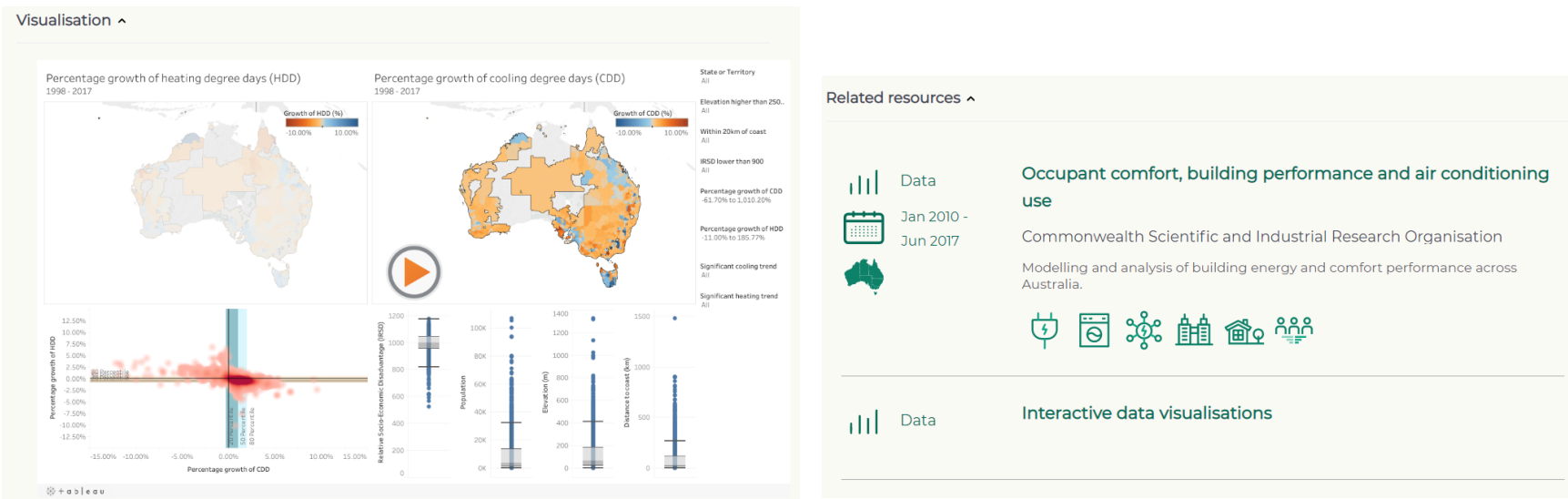


Figure: NEAR platform

AREMI

AREMI : platform with large amounts of spatial energy data

↪ Study on map renewable energy repartition in Australia.

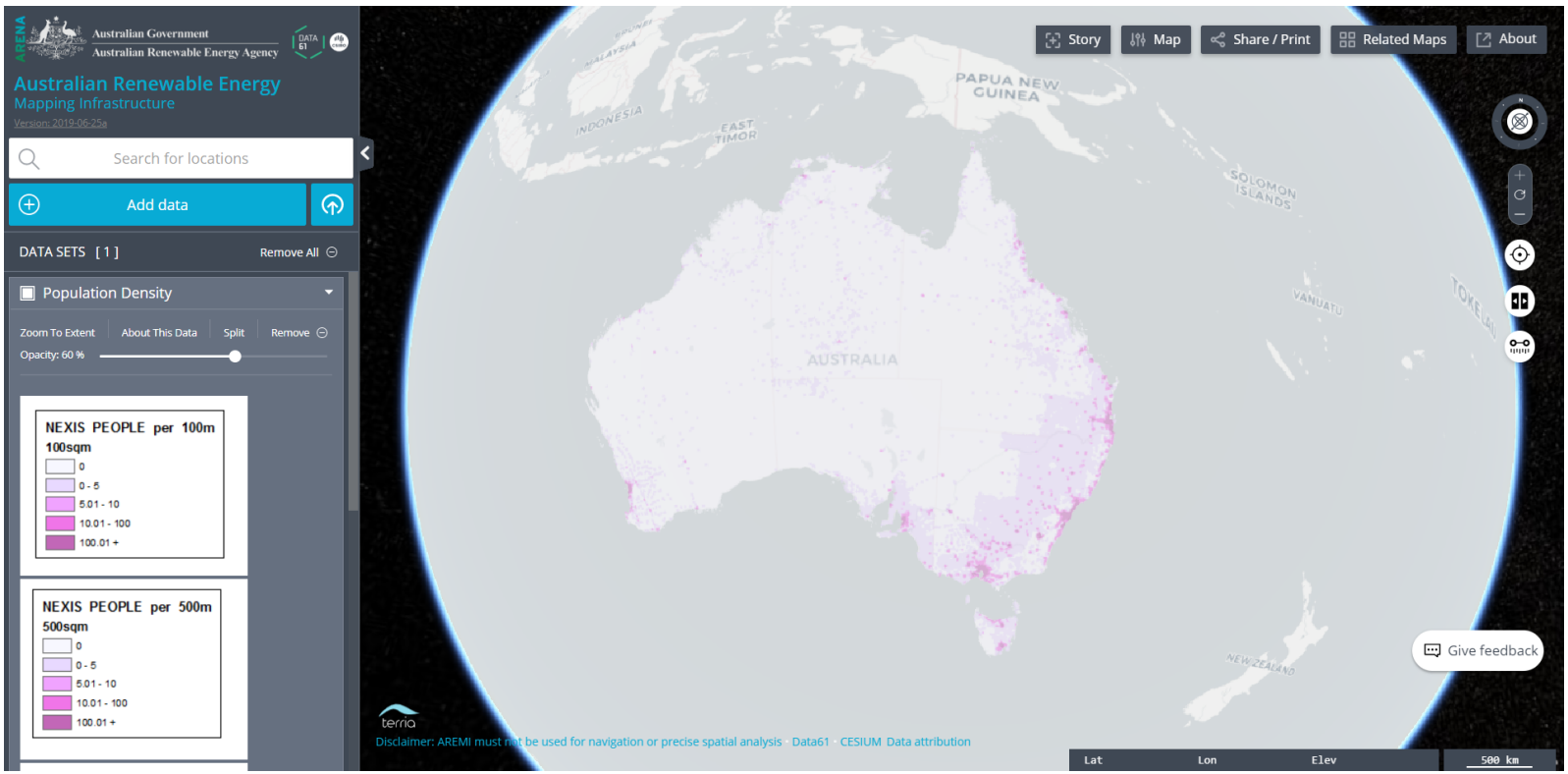


Figure: AREMI platform

Data visualization projects

MapsSDK-Unity

Principle : tool for visualization of data on 2D Bing
↪ Input : a location provided in longitude / latitude.

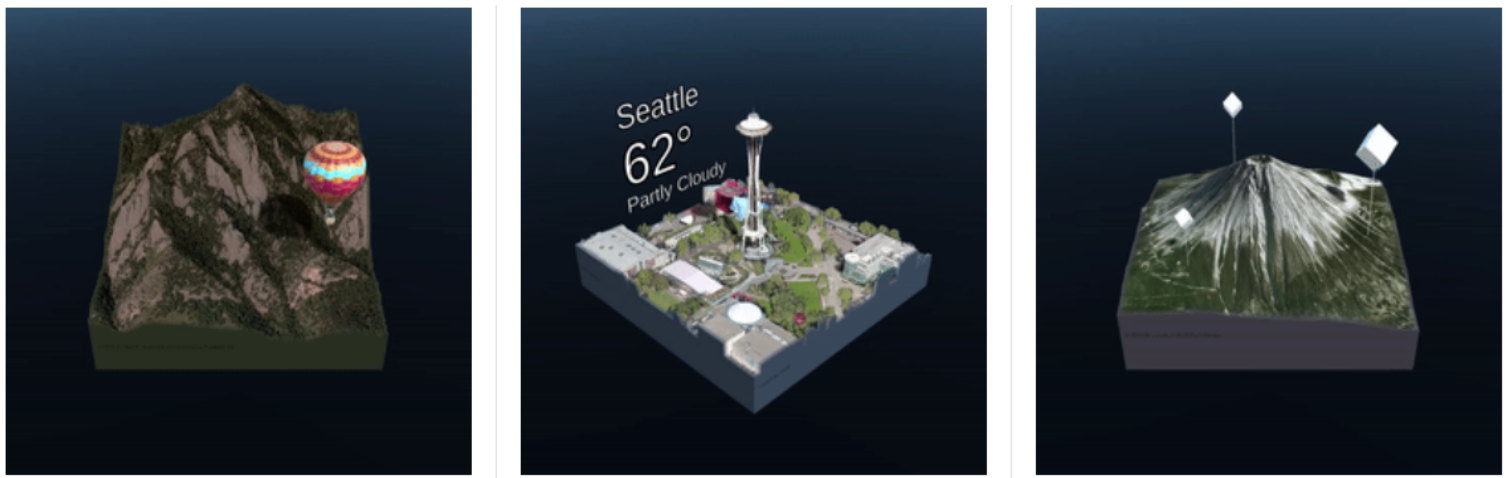


Figure: MapsSDK-Unity

ImAxes

Principle : tool for visualization / exploration of multivariate data
↪ Input : an Excel database with many dimensions.

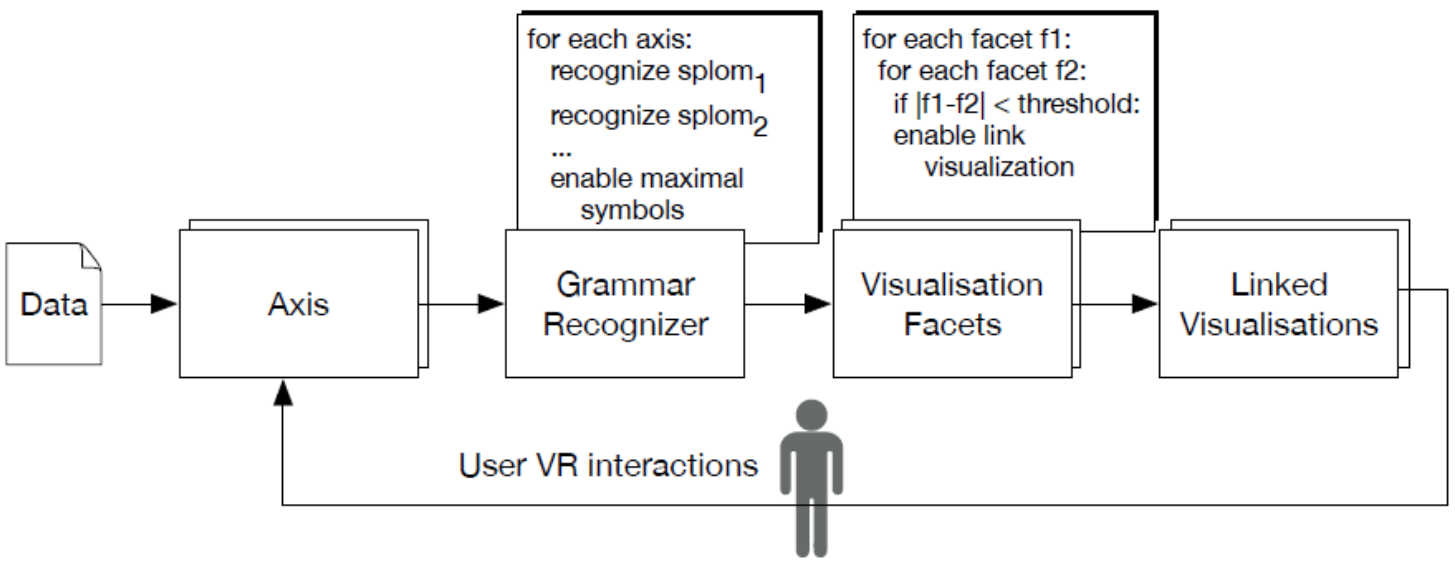


Figure: ImAxes diagram

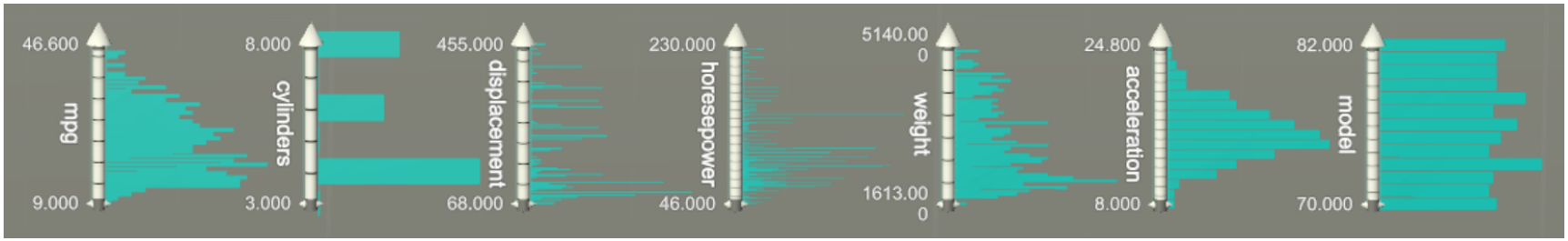
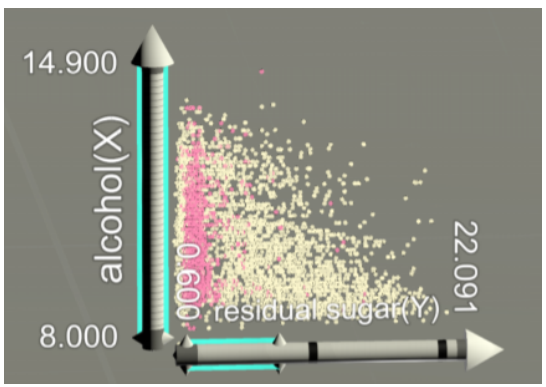
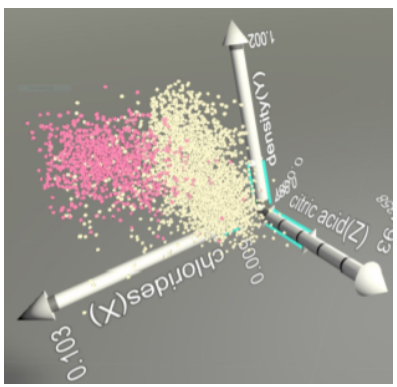


Figure: Set of histograms

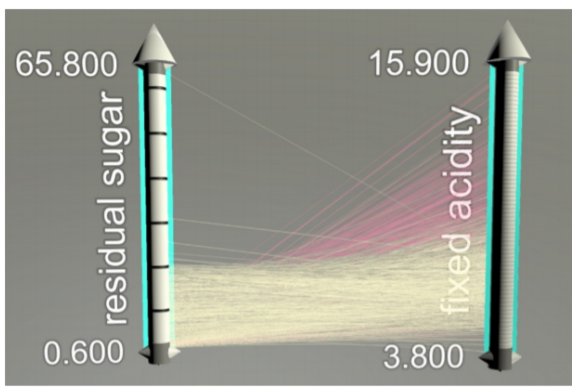
- Basic representations built :



(a) 2D scatterplot



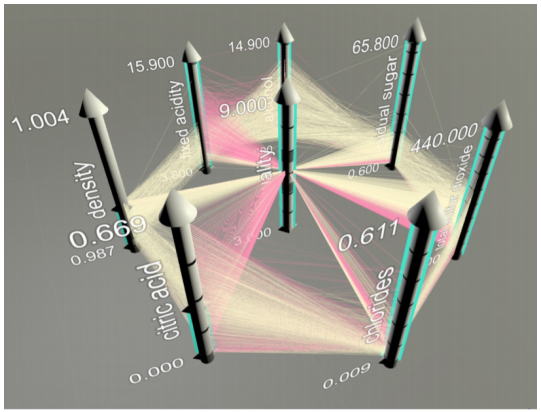
(b) 3D scatterplot



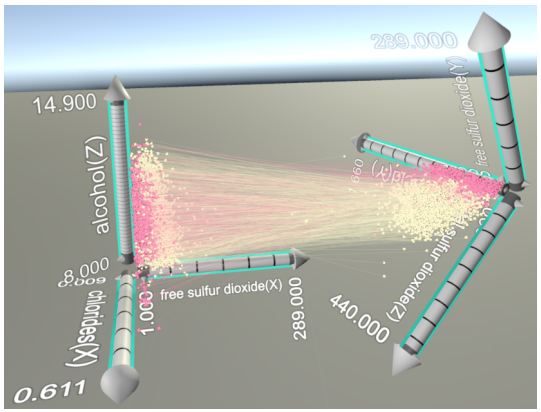
(c) Parallel coordinates

Figure: Main visualizations

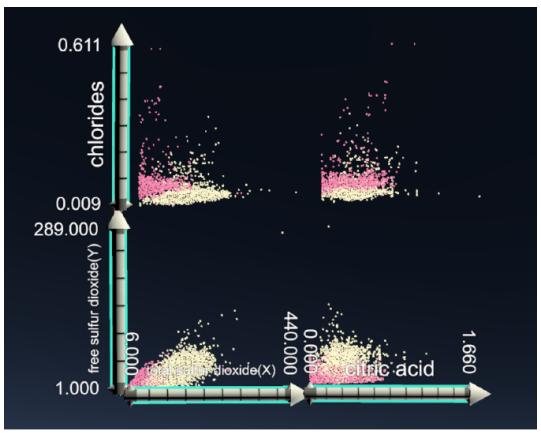
- More evolved representations :



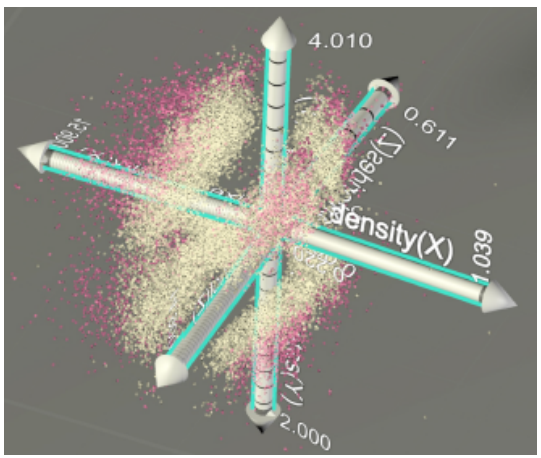
(a) Circular representation



(b) Mixed representation



(c) SPLOM



(d) 3DSPLOM

Figure: More complex visualizations

Collaborative environment

Photon PUN, networking development API

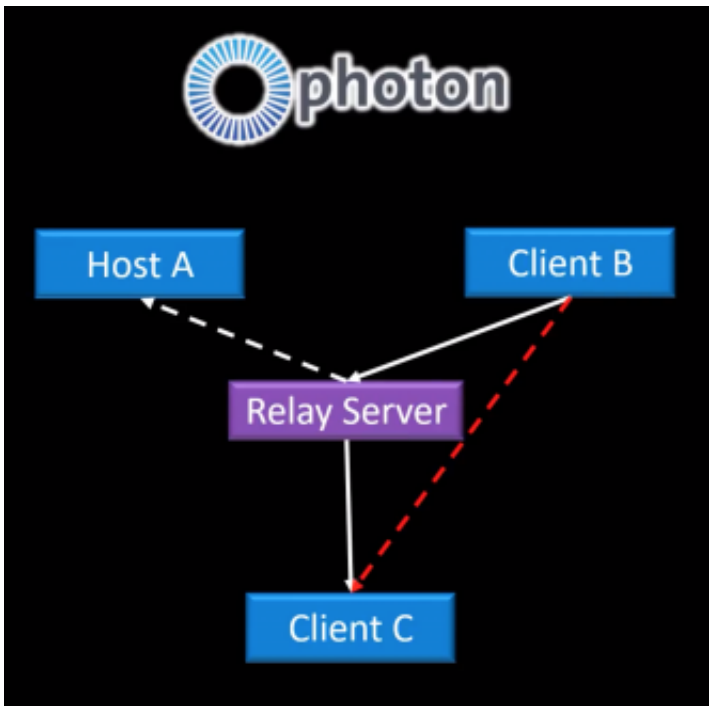


Figure: PUN architecture

- Network game engine : **Photon**;
- Version 1 used;
- *peer-to-peer* sending of messages.

Sequencing of windows

Sequencing of scenes

- One scene for the connection,
- One scene for the *immersive analytics experience*

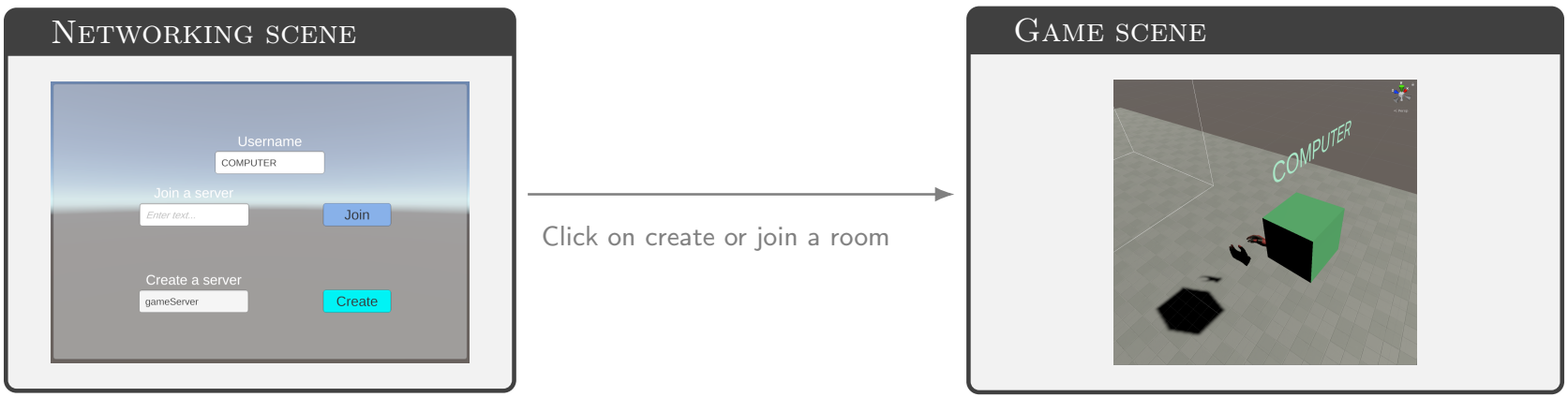


Figure: Sequence of windows

Sequencing of panels

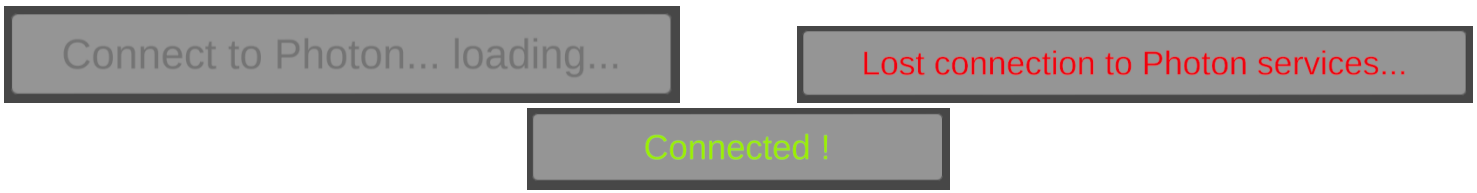
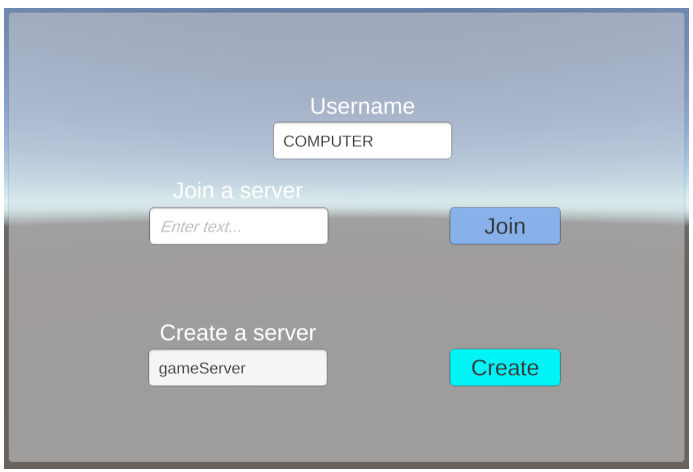
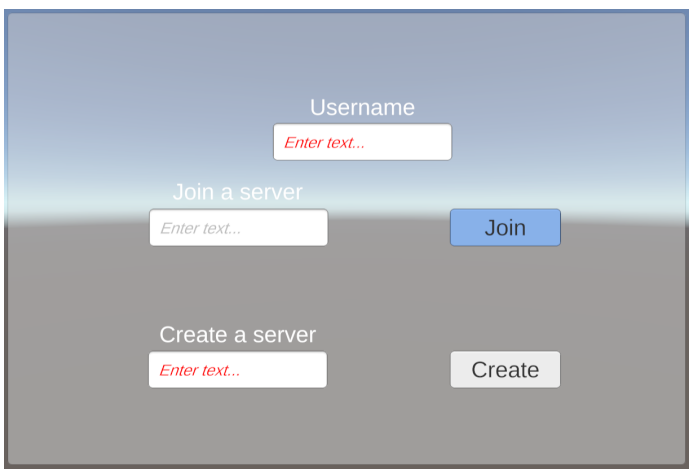


Figure: Sequencing of panels



(a) Main menu



(b) Red text

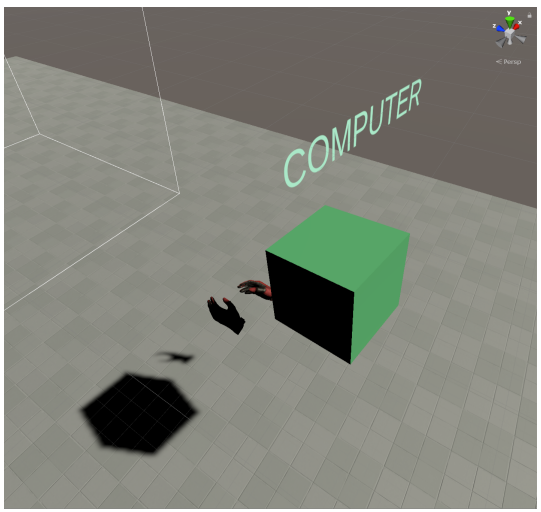
Figure: The different panels appearing

Vision of objects

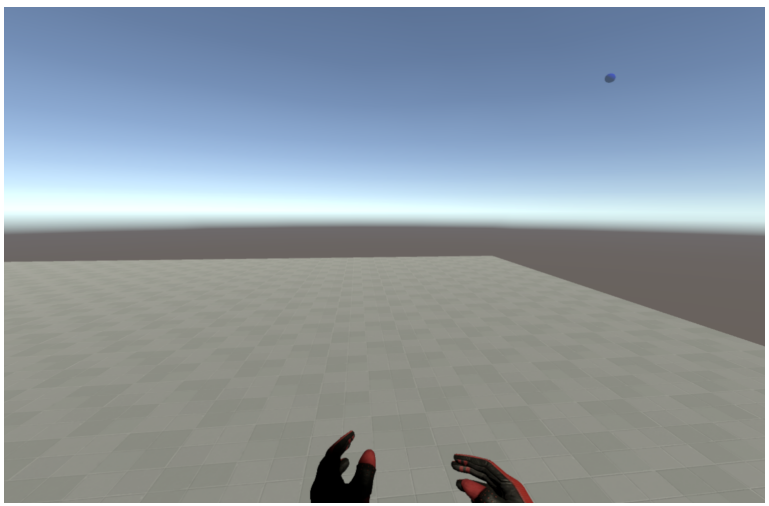
Modelisation of players

Instantiation of body parts :

- headset $\leftrightarrow$ colored cube with player username above
- *motion controllers* $\leftrightarrow$ glove prefabs



(a) Visualized in Unity



(b) Visualized in headset

Figure: Different views of the user

Offset between users' 3D world

Each player linked to VR headset running on a dedicated computer
$\Rightarrow$ loaded 3D scene identical BUT its delimitation.

Strategy :

① Draw very similar game zones for each VR headset.
$\hookrightarrow$ **Observation :** still a little offset $\Rightarrow$ not very precise.

② World mapping to finetune correspondence to remove offset.

Focus on step ② : *Master Client*'s world taken by default.

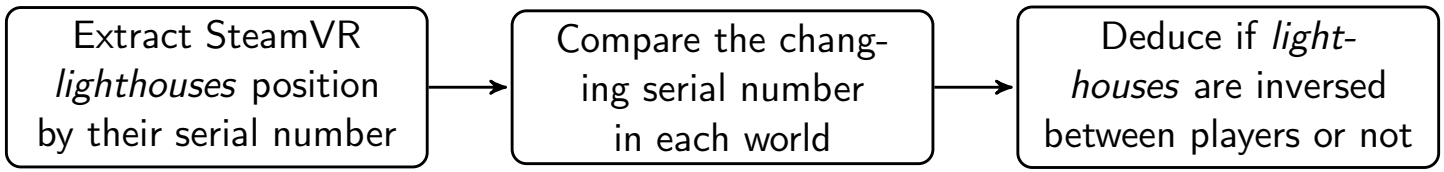
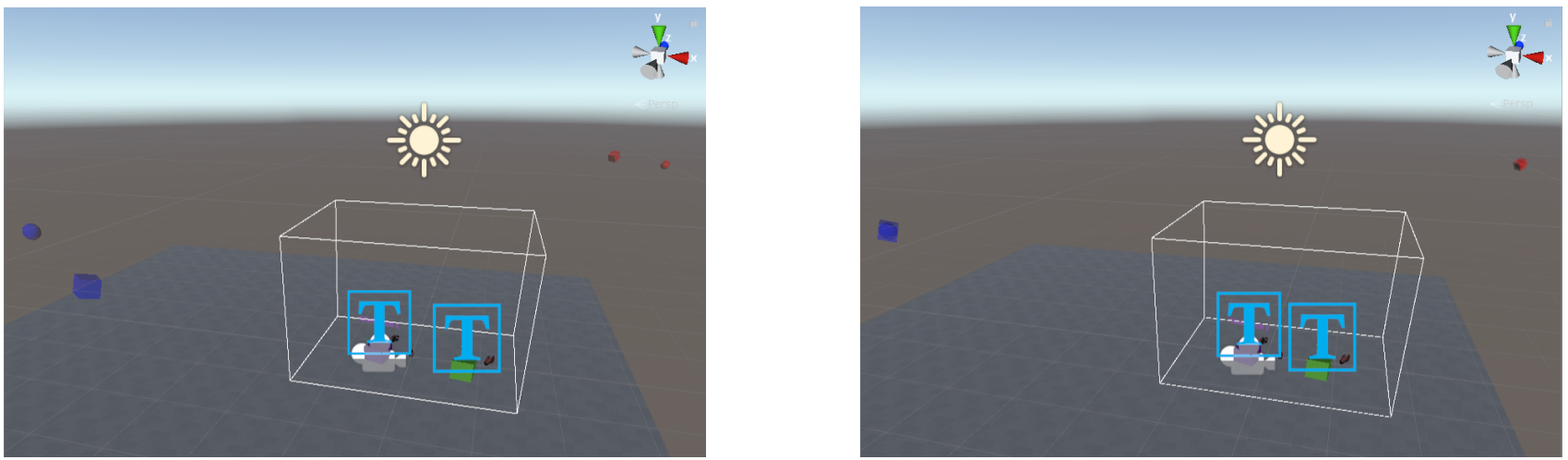


Figure: World Mapping pipeline

Obtained results :



(a) Before world mapping

(b) After world mapping

Figure: World mapping feature

Interaction with objects

Sending of messages architecture

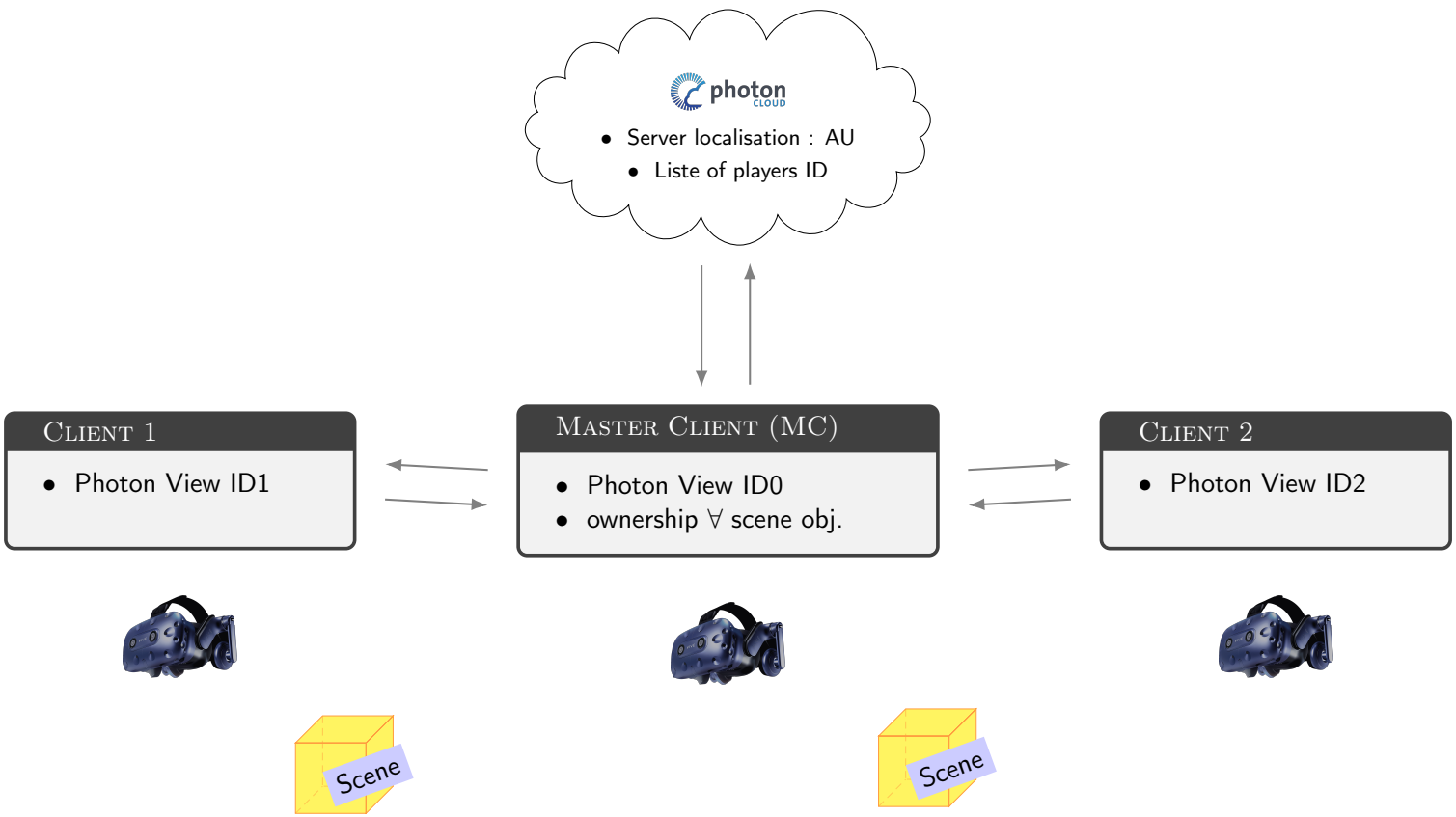
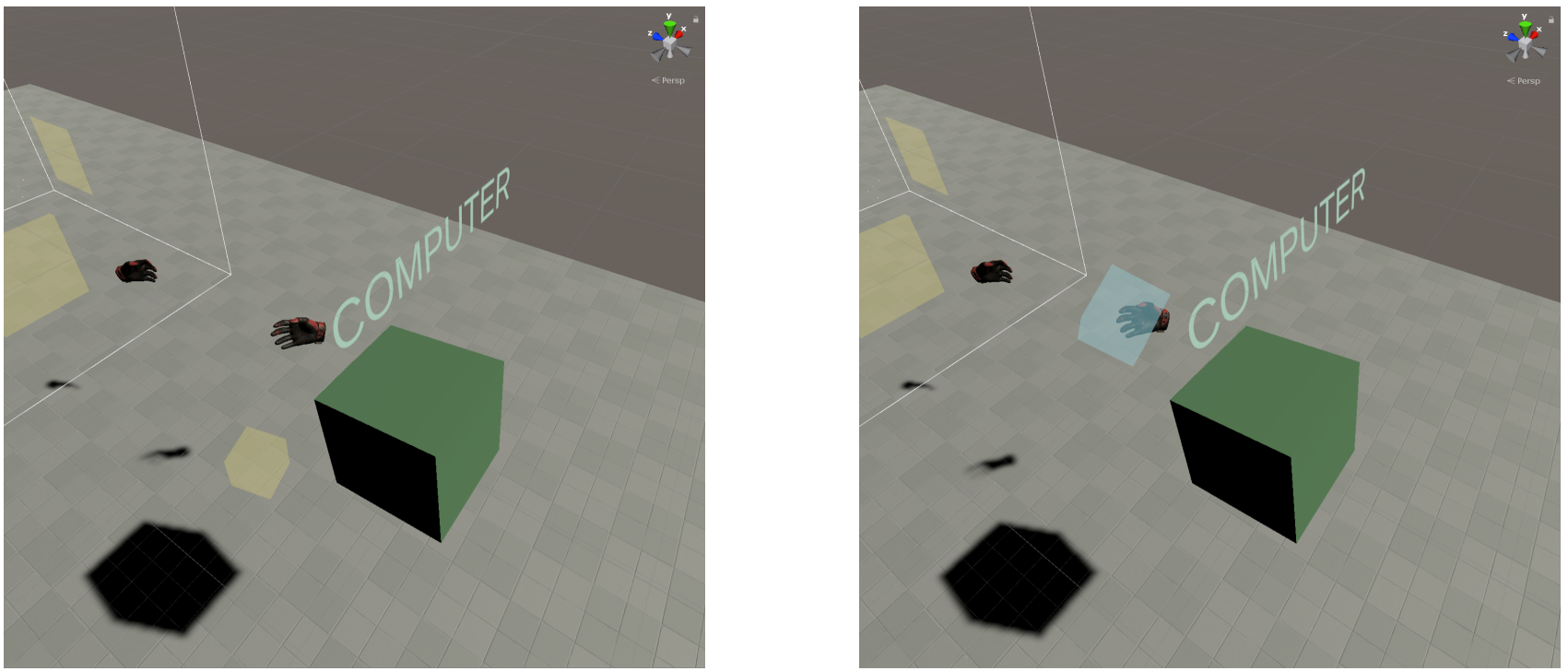


Figure: Ownership and Cloud updates in Photon PUN

Major concerns :

- Manage ownership of each scene object,
- Update movements of players / scene objects.



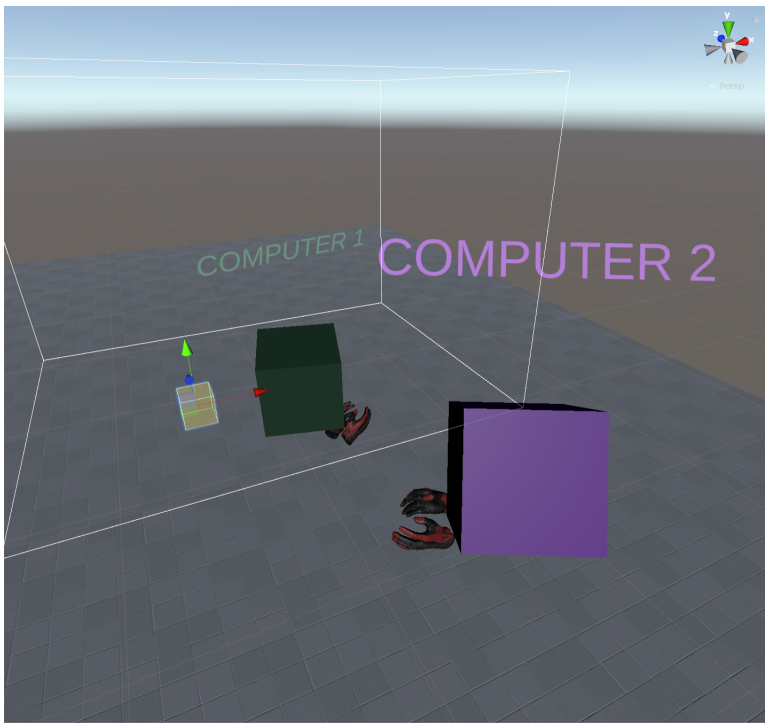
(a) No intersection

(b) Intersection

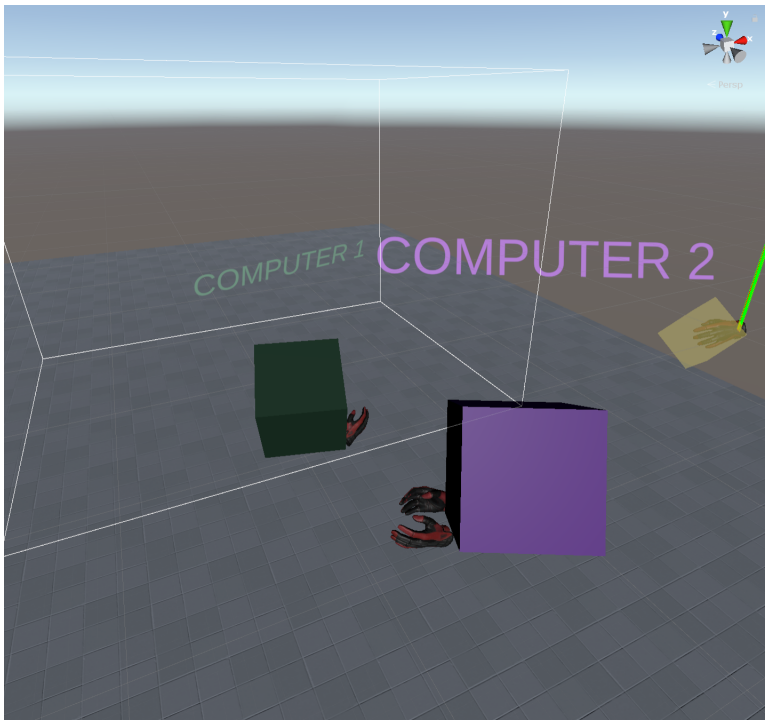
Figure: Cube color switching according to *colliders*

Physics simulation issues

Photon limitation : does not manage *physics* on the server itself
$\Rightarrow$ Use Unity *physics* and manage updates ourselves.



(a) Cube not grabbed



(b) Cube grabbed by computer 1

Figure: Cube visualized on the computer 2 scene view screens

Rendering of the visualization

Unity3D graphics pipeline

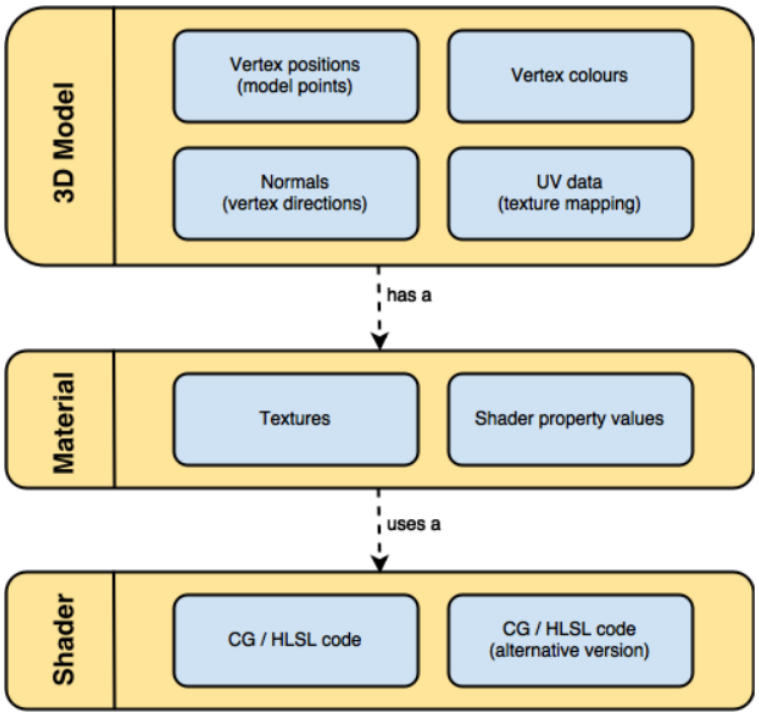


Figure: Rendering workflow

- *Shaders* run on GPU;
- Unity Shader language : **ShaderLab**;
- HLSL between snippets.

Data points representation

Targeted parts of Direct3D graphics pipeline

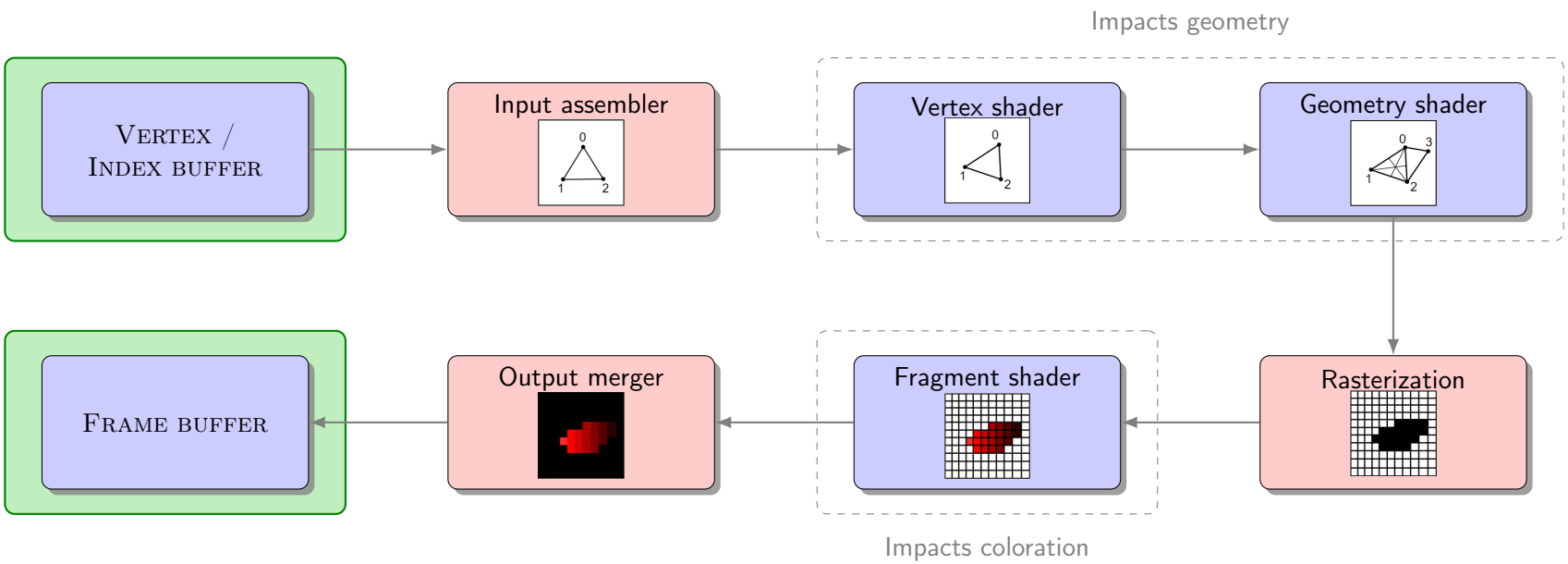


Figure: Direct3D Graphics pipeline

Construction of polygons

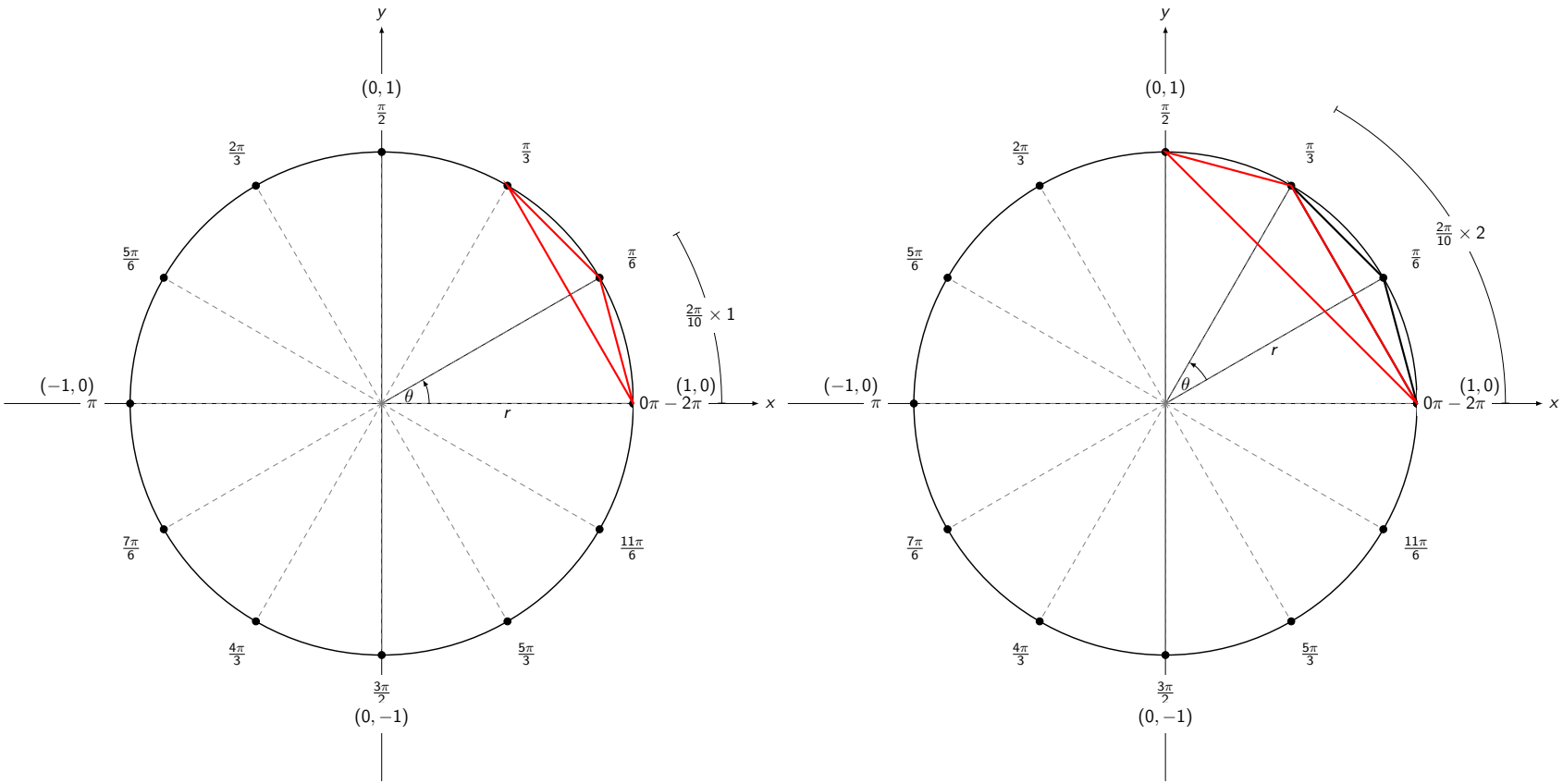
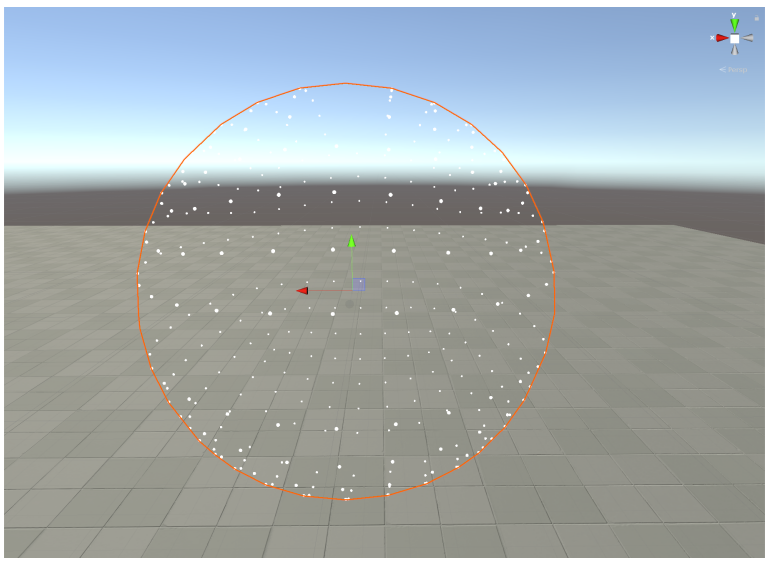


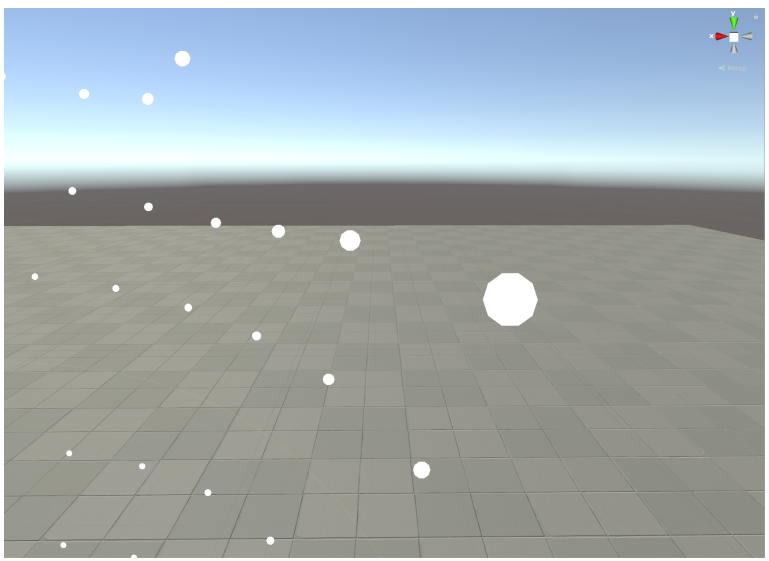
Figure: Computation of the two first polygon vertices

Formula to compute the position of polygon vertices :

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} + \begin{pmatrix} \cos(\theta) \times xAspect \\ \sin(\theta) \times yAspect \\ 0 \\ 0 \end{pmatrix} \times polygonSize$$



(a) Spherical point cloud



(b) Points drawn as polygons

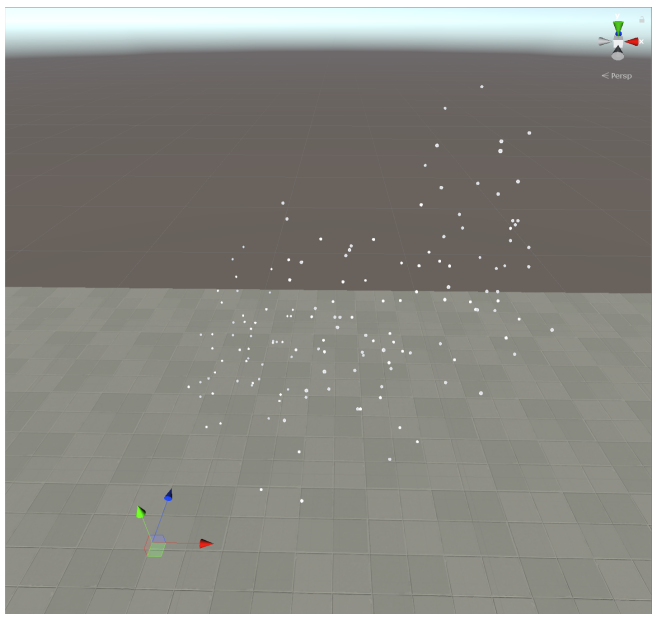
Figure: Spherical point cloud and its characteristics

Data send to shaders

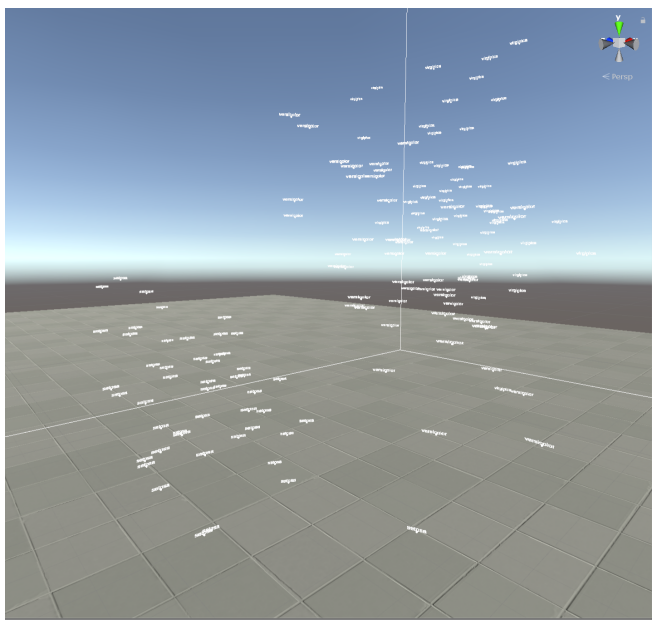
2 scripts needed :

→ for reading from Excel file spatial coord. and data dimensions;

→ for importing this information in the Unity 3D scene.



(a) Cloud built from Excel database



(b) Legend displayed next to points

Figure: Cloud and its caracteristics

Data analysis tools implemented

Infinite clipping planes

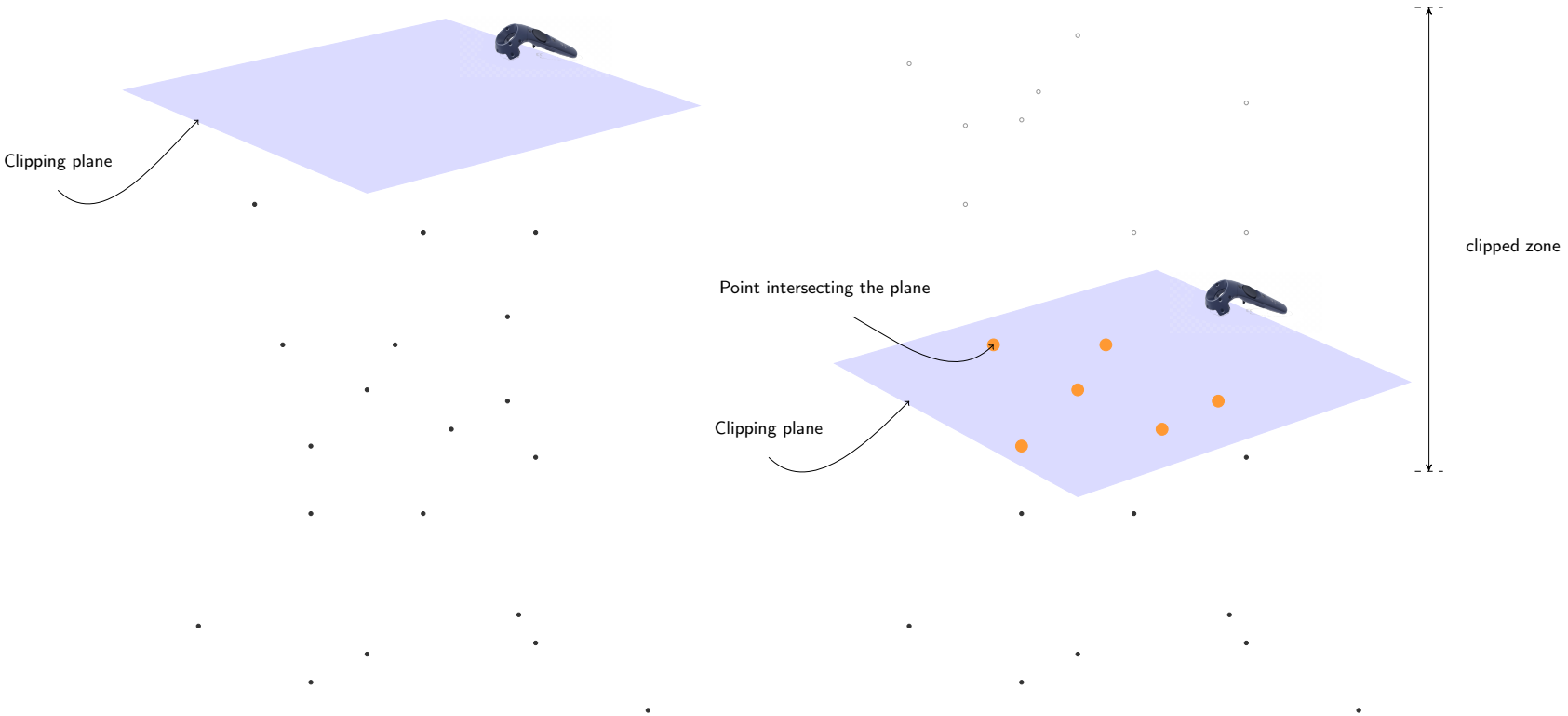
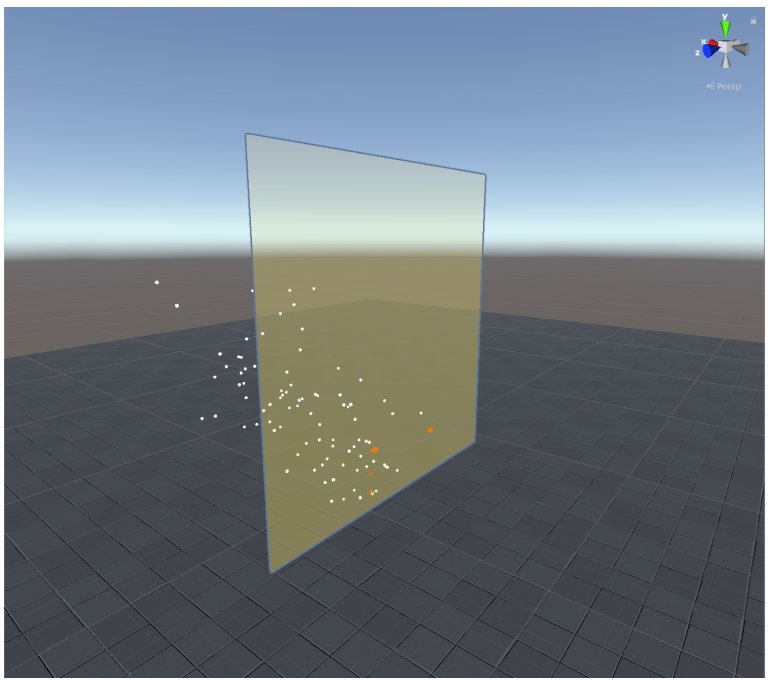


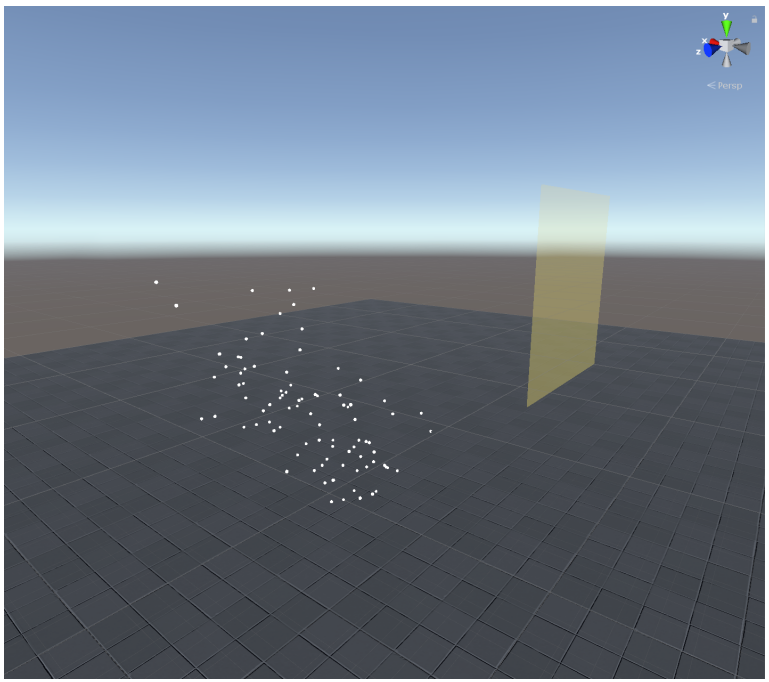
Figure: Illustration of clipping principle

Distance from point to plane : $\forall \vec{p} \in cloud, d = (\vec{p} - \vec{c}, \vec{n})_{\mathbb{R}^3}$

Problem observed : size of the clipping plane is ignored
$\Rightarrow$ Clipping feature applied too often.



(a) Plane intersecting points

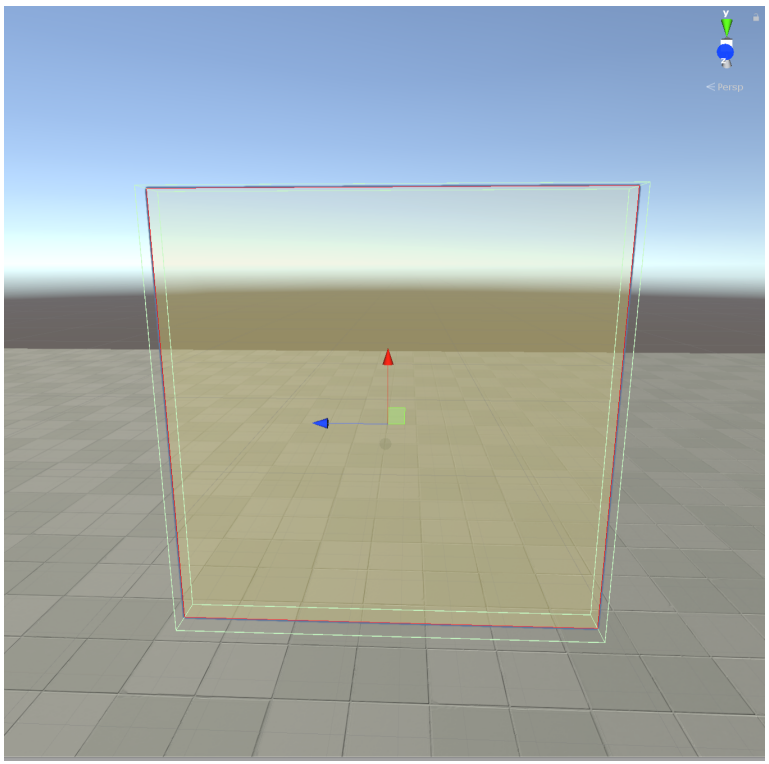


(b) Plane not intersecting points

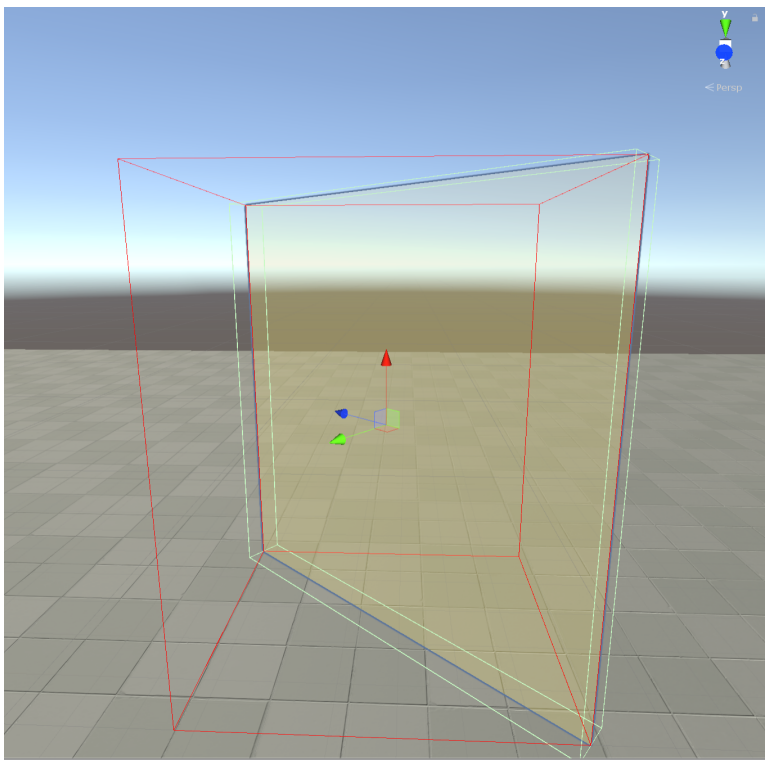
Figure: Issue raised with infinite clipping planes

Finite clipping planes

Attempt ① : compute plane bounds via Unity components.
Obtained results : not working if rotation applied on plane.



(a) Plane modified by translation



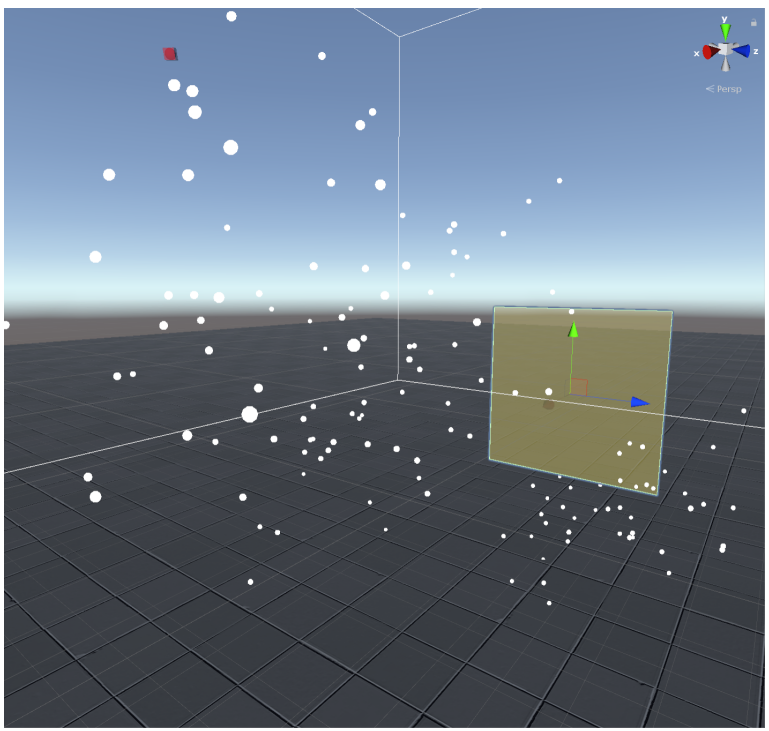
(b) Plane modified by rotation

Figure: Illustration of AABB collision

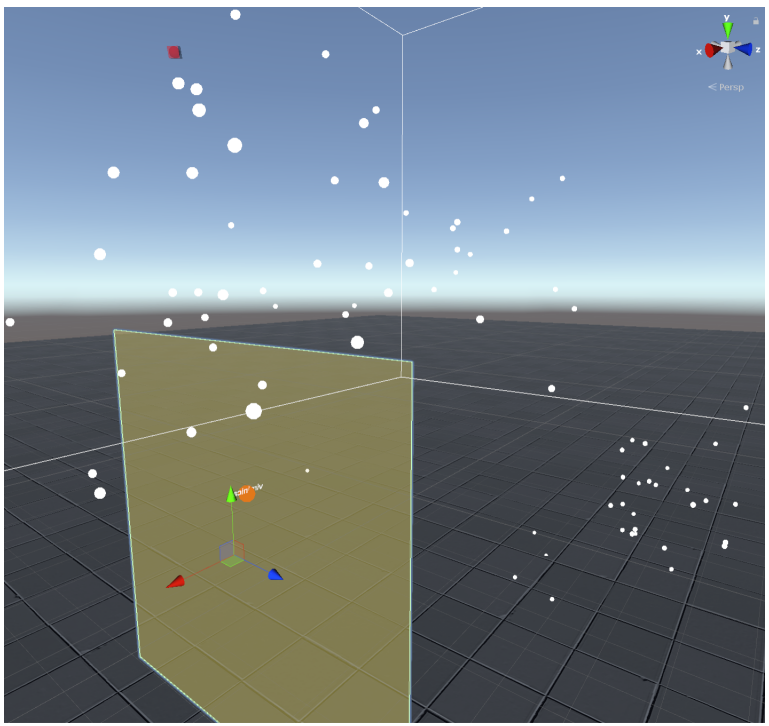
Attempt ② : compute thin cube bounds via its 8 corners.

$$corner = R \times \begin{pmatrix} sgn(width/2) \\ sgn(height/2) \\ sgn(depth/2) \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Obtained results : just working for collinear axes.



(a) Plane behind point cloud



(b) Plane clipping points

Clipping boxes

Attempt ③ : see clipping plane as a clipping box.

Strategy :

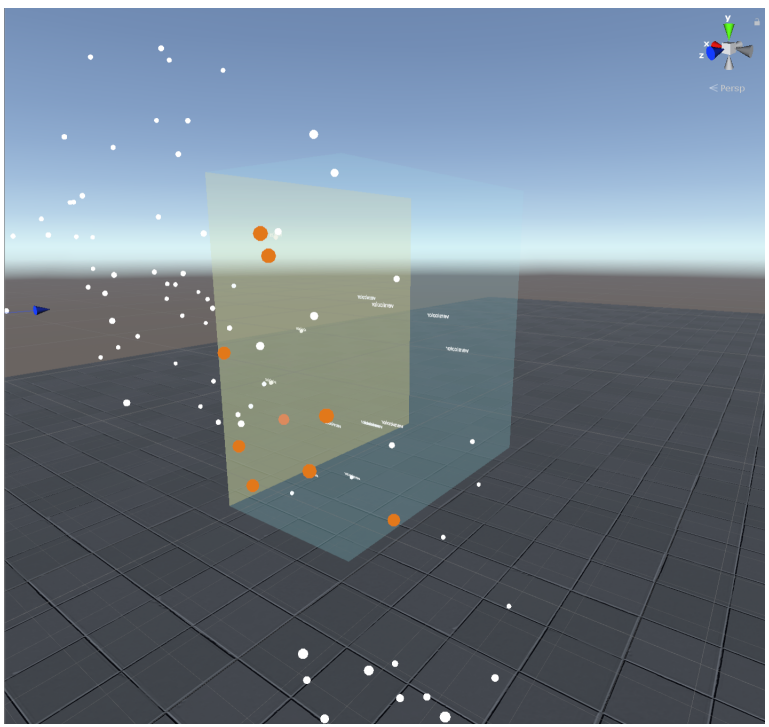
① Implement a clipping box feature.
↪ Depth of the box along normal plane axis important enough

② Attach a clipping plane to one of its face.

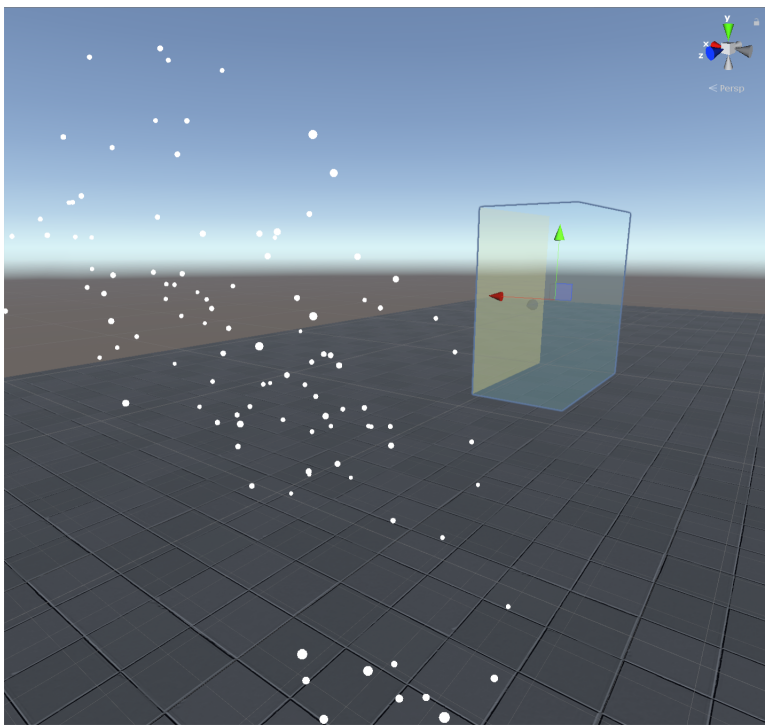
Distance from point to box :

$$d = \left\| \max\left(\vec{0}, \vec{d}\right) \right\|_{\mathbb{R}^2} + \min\left(\max\left(d_i\right)_{i \in [|1,3|]}\right)$$

Obtained results :



(a) Points are inside the box



(b) Points are outside the box

Figure: Results of clipping plane with box attached

Highlight effect and reflection models

Highlight planes prototype

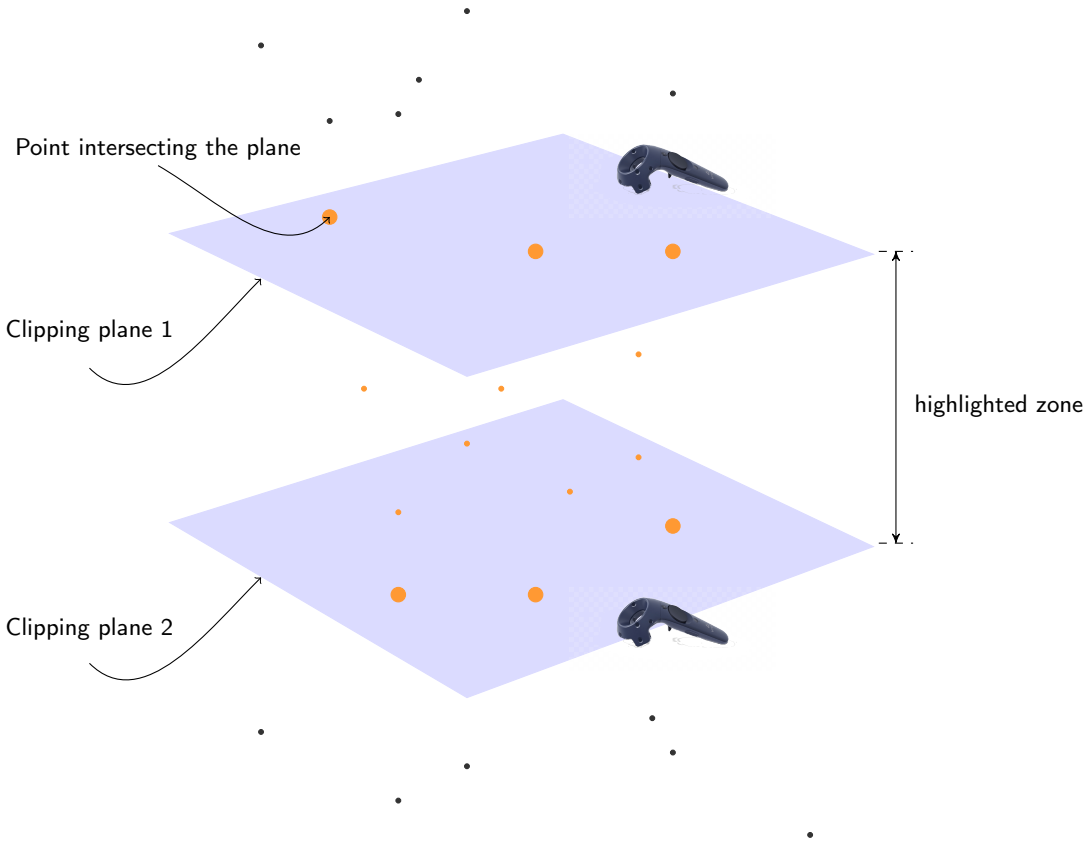
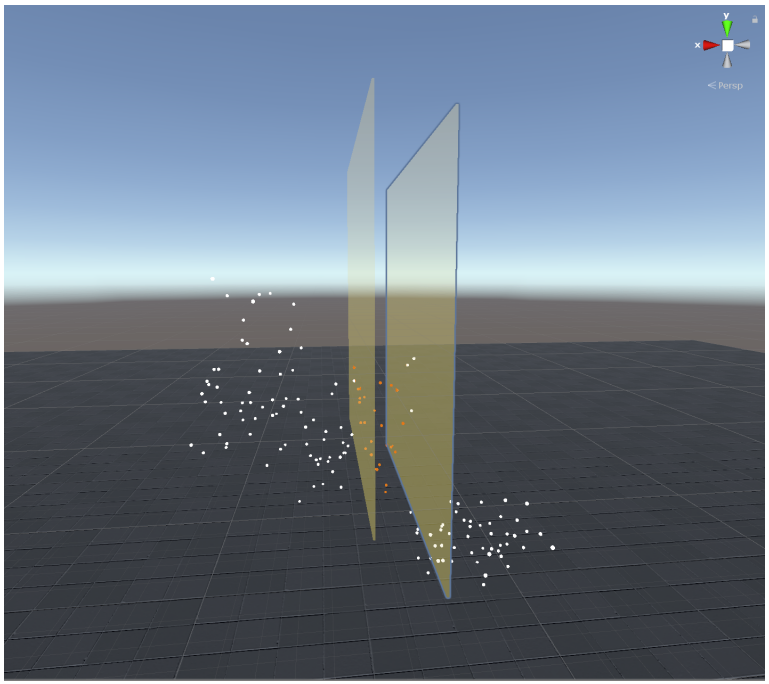
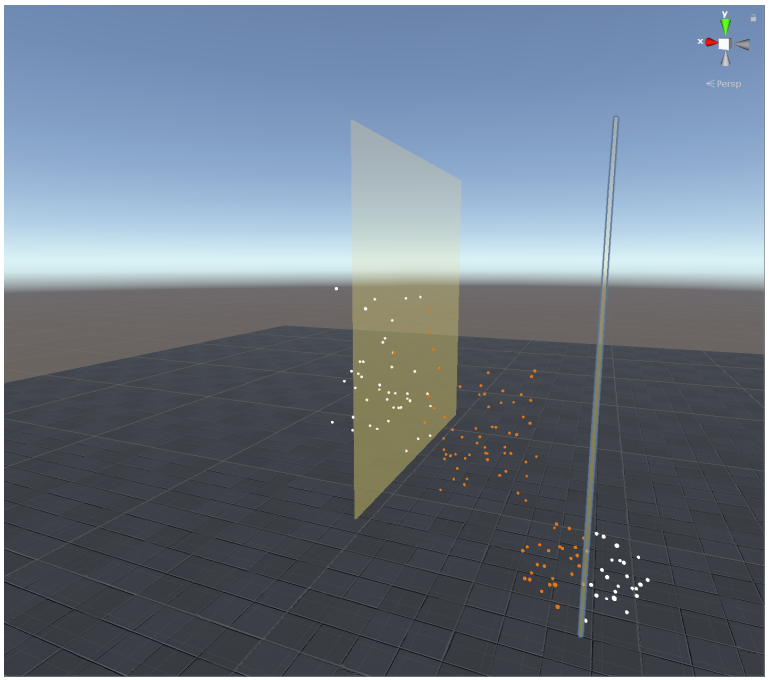


Figure: Illustration of highlight feature

Obtained results :



(a) Highlighting thin portion



(b) Highlighting thick portion

Figure: First prototype of highlight planes

Observation : highlight effect is a bit basic.

Lambertian reflection model

Principle : diffuse lighting

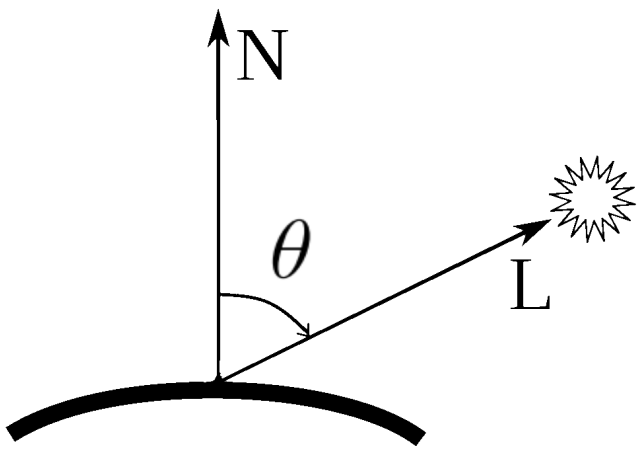
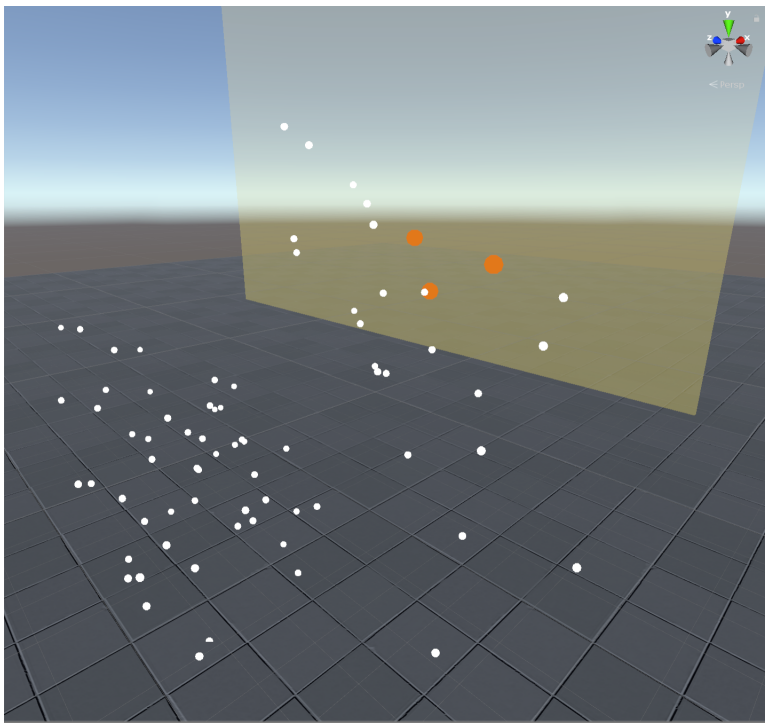


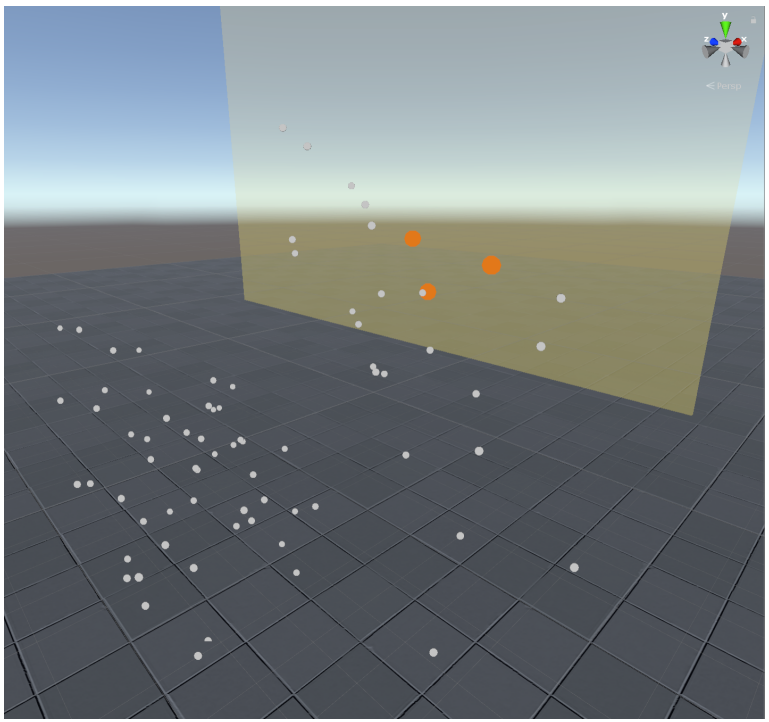
Figure: Lambertian reflection model and useful vectors

$$I_{refl} = \max\left\{0, \left(\vec{L}, \vec{N}\right)_{\mathbb{R}^3} \times c \times I_{incid}\right\}$$

Obtained results : for a datapoint cloud



(a) Diffuse light disabled



(b) Diffuse light enabled

Figure: Results for Lambertian reflection model

Observation : it's more of a trick than a real solution.

Blinn-Phong reflection model

Principle : specular lighting

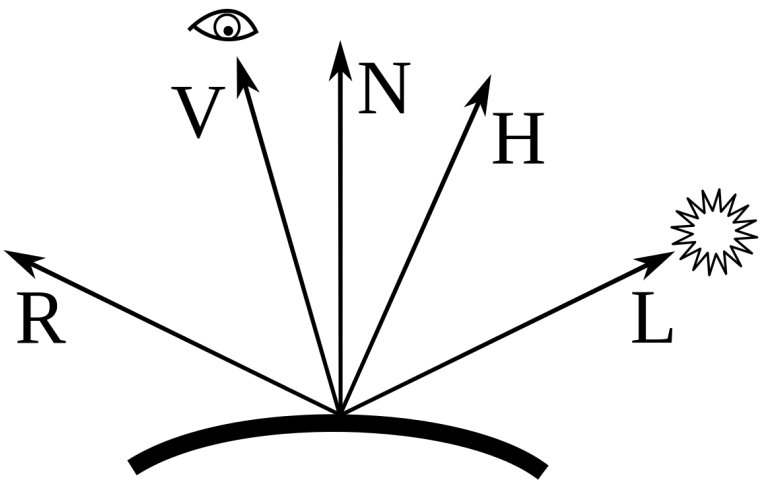


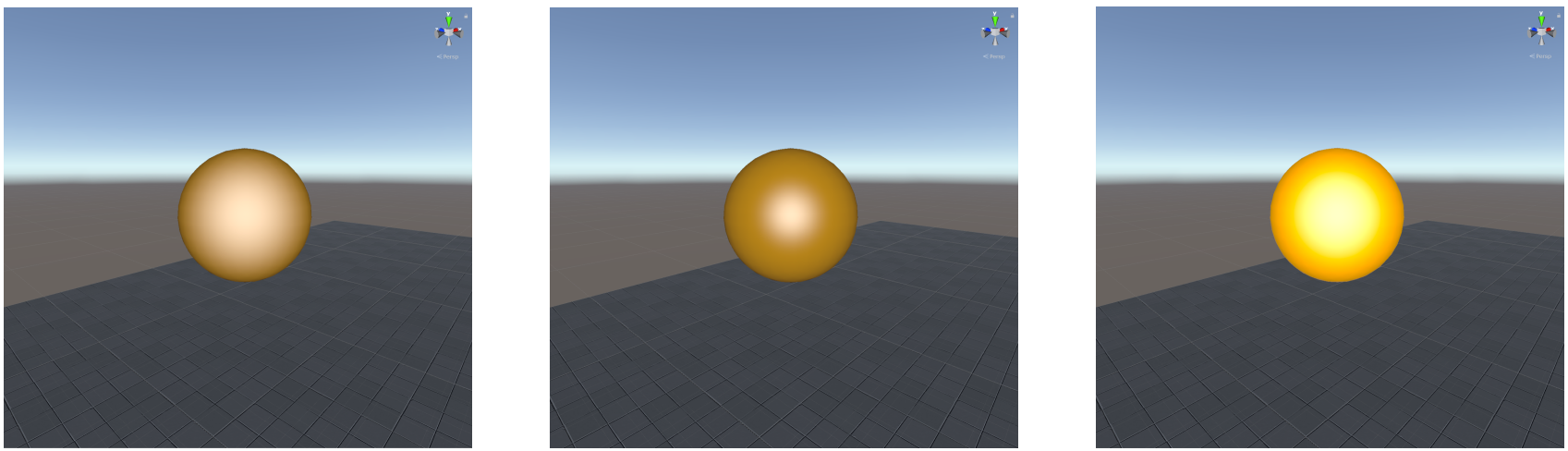
Figure: Blinn-Phong reflection model and useful vectors

$$I_p = k_a i_a + \sum_{m \in \mathit{lights}} \left\{ k_d \left(\vec{L}_m, \vec{N} \right)_{\mathbb{R}^3} i_{m,d} + k_s \left(\vec{R}_m, \vec{V} \right)_{\mathbb{R}^3}^{\alpha} i_{m,s} \right\}$$

Attempt ① : RGB system ⇒ gave a dull material.

Attempt ② : HSB system ⇒ material with configurable variables
↪ brightness, contrast, saturation.

Obtained results : for a datapoint



(a) Basic Blinn-Phong (b) Blinn-Phong (c) HSB Blinn-Phong

Figure: Evolution of Blinn-Phong lighting results

Conclusion

Collaborative and immersive scatterplot rendering

✔ **Balance of contributions :**

- improvement of my mastery in Unity,
- work with Virtual and Mixed Reality technologies,
- implement classical computer graphics algorithms and tools.

✔ **Main perspectives :**

- import the project into the ImAxes tool,
- implement a solution to know another player's view,
- refine communication between different users.

Thanks for your attention