SEME – OPTIS

Semaine d'Étude Maths-Entreprises *avec* OPTIS

Visibility optimizations

- OPTIS
  - Current context
  - Optimizations needs
- SEME
  - State of the art
  - 2D experimentations / software resolutions
  - 3D prototypes / mixed hardware solutions

Current context

- OPTIS activities and products
  - Created in 1989
  - Aim to minimized the physical prototyping by providing virtual simulations for product validation, ergonomie, measures, experiences…
  - Products : from optical conception to realistic virtual reality experiments via material & light measures
  - Applications : Auto, Areo, Lighting, Electronics, energie, research, architecture, luxury, medical, automation…

  - Optis in 2015 :

200 experts	2400 clients	14 partnerships
3 R&D center	8100 licences	50 countries

Current context

- Aim to mix technos :

  Our physically-based lighting engine (not real time)

  with our 3D Virtual Reality engine (real time)

  - *SceneGraph based*
  - *OpenGL and DirectX supported*
  - *Forward and Deffered rendering techniques supported*
  - *Physic, Network, Haptic & Sound engines also supported for VR deployements*

- **Rendering performances is our current bottleneck**

Culling optimizations needs

Typical use cases :

- Single product review under different physical conditions (*ligthing simulations*)
- Interactive simulation into Virtual Room (*3D physical interactions with tracking actors and multimodal feedbacks*), mainly for maintenance and/or design ergonomie
- Hybrid semi-real time distributed interactive and collaborative distant review from different platforms (*desktop, CAVE…)*

Physics-based simulations for reliable decision-making, using OPTIS software

Culling optimizations needs

Typical scenes :

- Big 3D CAD objects well detailed from outside but also inside, with many small subparts
  (*cars, planes, complex factories…)*
- Environements could be light (*only skybox with the CAD model*)
  up to very detailed landscape (*atmospheric sky with long racetrack including animated manikins*)

Need CULLING optimizations

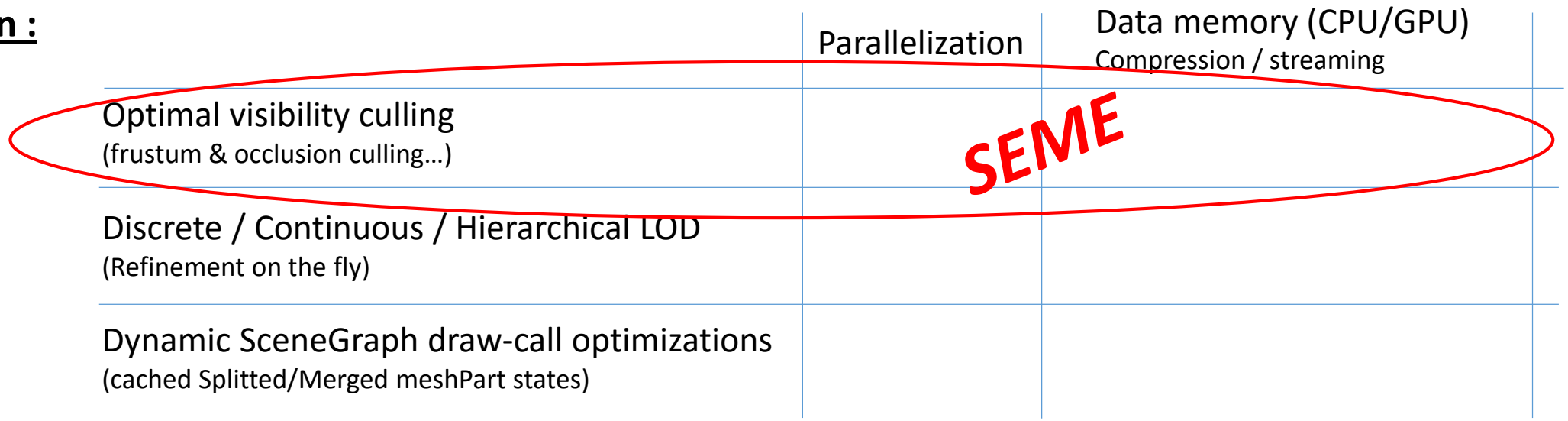
Culling optimizations needs

Culling : *describes the early rejection of objects of any kind that don't contribute to the final image*

Benefit *:*

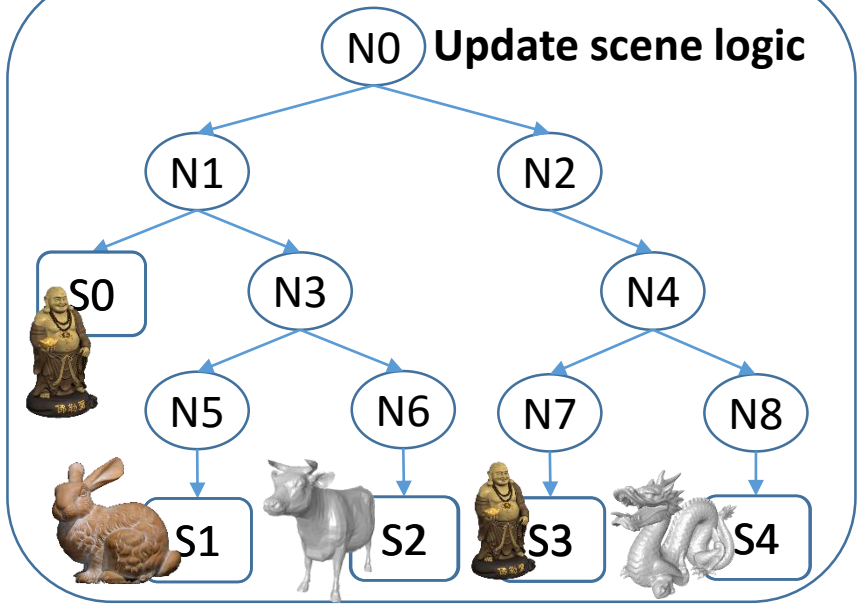
- Freeing our artists from the burden of manual markups and additional work
- Improving realtime rendering performances (*& runtime memory…*)
- Keeping scale flexibility in any complexe scenarios (*indoor/outdoor…*)
- Keeping Virtual Reality platform deployement flexibility (*CAVE, Stereo…*)

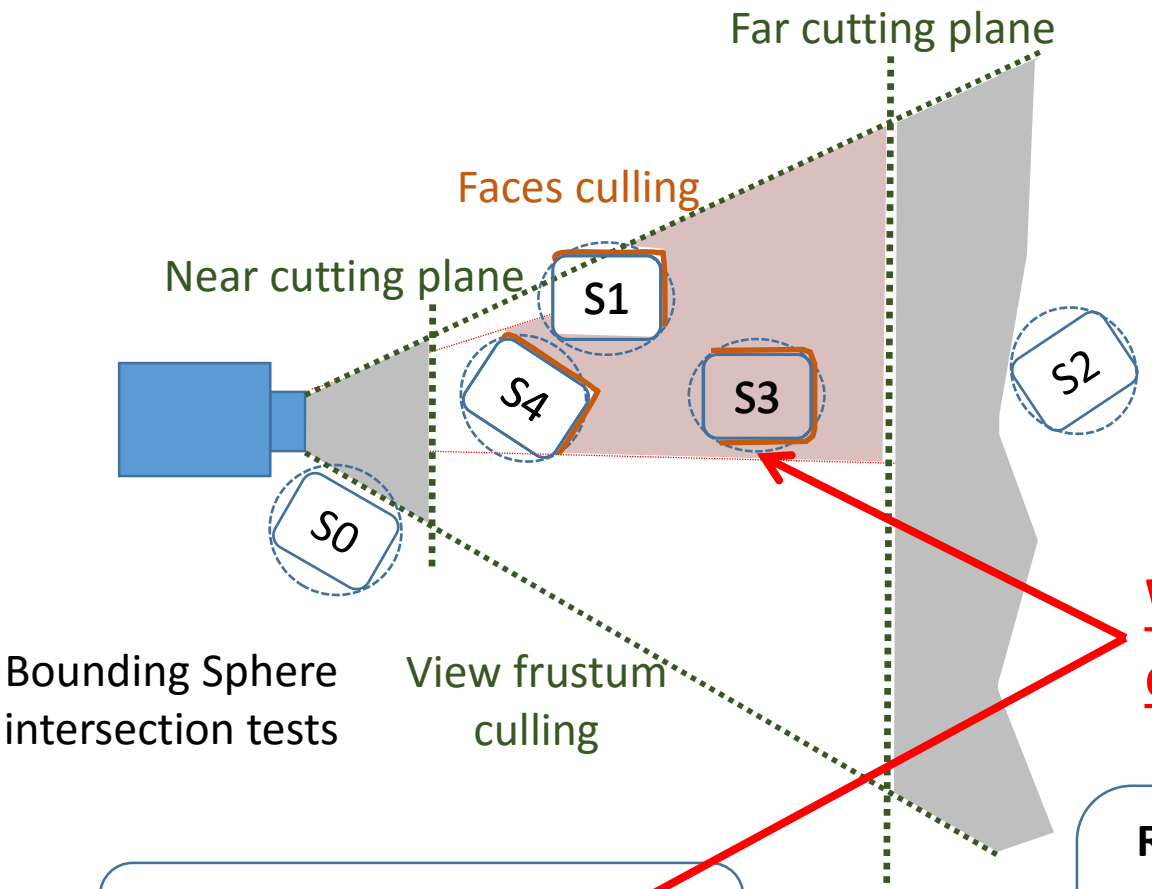
Plan :



Culling optimizations needs

INPUTS





Register all shapes

S0 S1 S2 S3 S4

Culling

Render all shapes

S1 ~~S3~~ S4

Rendering

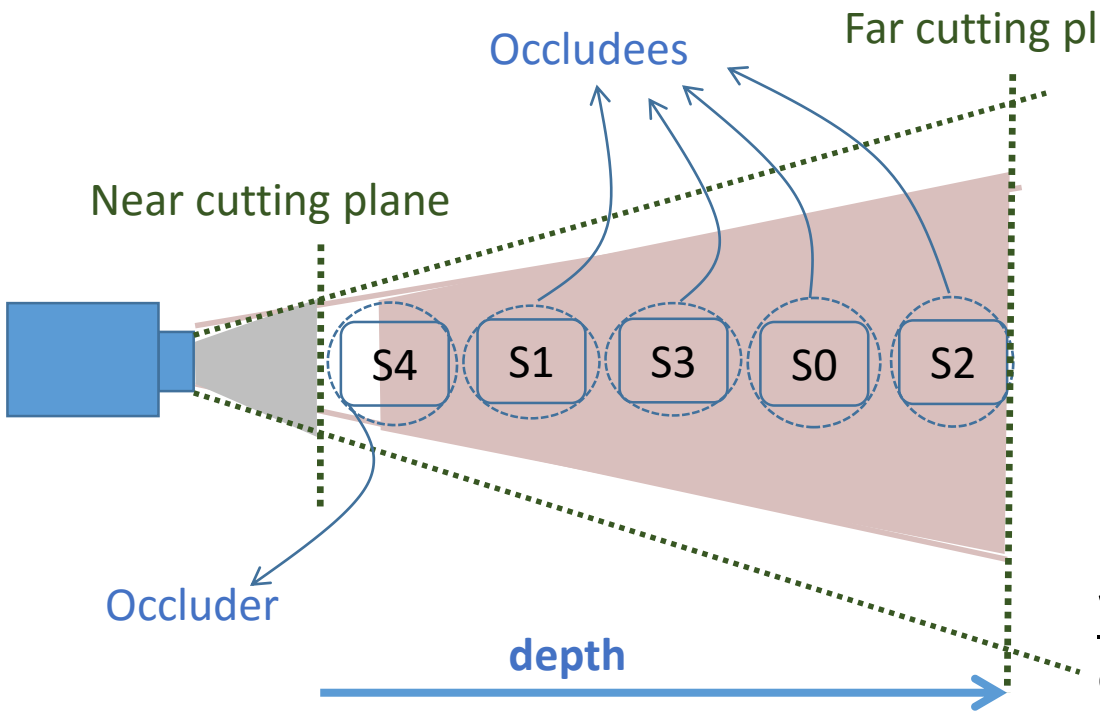
Rendering screen

Image

N = Node = rotation/scale/translation

S = Shape = vertices/textures

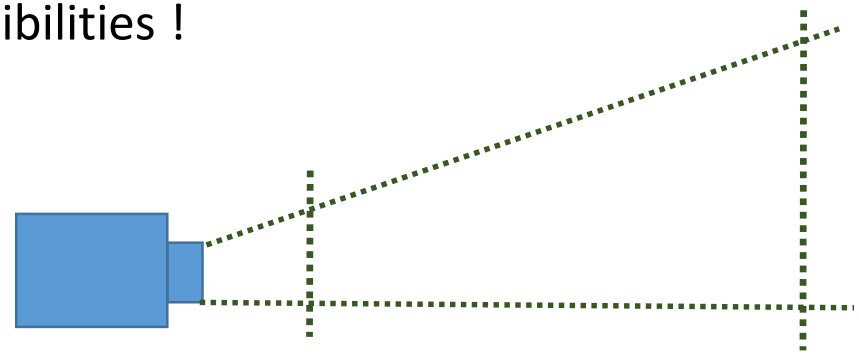
Culling optimizations needs



In many cases, we want the far cutting plane to be **as far as real visibility** for our simulations accuracy !

4/5 shapes will be processed to be drawn unnecessarily
Imagine extra the work/memory to render a densely populated model (city…)

How to prevent those shapes to be process by GPU ?
Warning: We want to keep **asymmetric frustum** compatibilities !



Freedom to proceed

- With your knowledges
- With your experiences
- With your methods
- With your technologies
- …

- Any support available : jesnault@optis-world.com

Some guidelines if needed 1/2

- Literature survey about existing algorithmes
  - Avantages / Drawbacks
  - Limits

- Starting references
  - http://orion.lcg.ufrj.br/cg2/downloads/sombras/Visibility%20Survey.pdf
  - http://de.slideshare.net/Umbra3/

- The Visibility Problem
  - From-point vs. from-region
  - Precomputed vs. runtime
  - Object space vs. image space
  - Conservative vs. Approximate
  - Software vs. Hardware queries

- Going further, try to also adress :
  - Static vs. dynamic **o**bjects
  - Transparent vs. mirror objectcs
  - Deformable (cloth, fluids) vs. Objects with holes

Some guidelines if needed 2/2

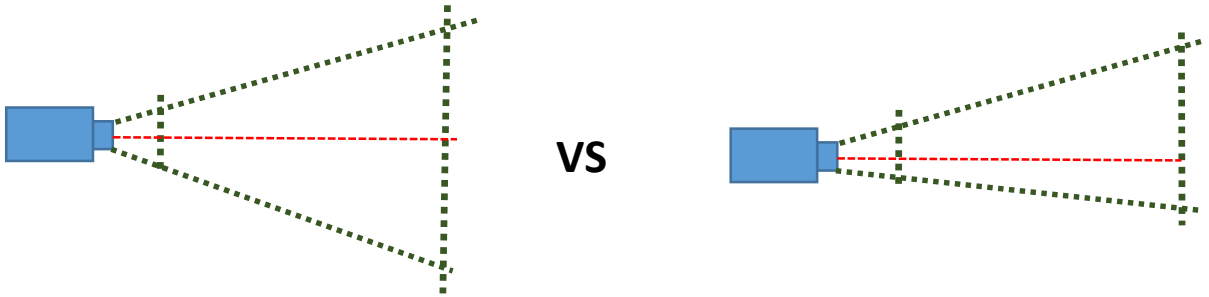
Going from 2D software (CPU) experimentations until 3D mixed hardware (CPU/GPU) solutions

- Test a few solutions in 2D, keeping in mind that it should generalize to 3D
  - Examples (not yet explored) : http://www.dca.fee.unicamp.br/projects/mtk/batageloM/
- Explore any possibilities of culling optimizations…
  - Other bounding shapes test intersection ? Dynamically shifting the far plane Depth?
  - Potentially Visible Set lookup, Contribution culling, Hierarchical (surface/volume) culling, Spacial Partitioning (Quadtree / Octree) … ?
- …Using all possible technics and technologies :
  - Precomputed data structure (for CPU/GPU) to be loaded before runtime ?
  - Prediction from previous frames and/or multi-threading runtime queries ?
  - GL/DX (even from Vulkan, Dx12…), Unity3D, UnrealEngine, Ogre3D, OpenSceneGraph…
    - But precise the version used

Glossary

- Rendering engine : Program which process some data (from CPU) to output a final image (from GPU)

- View frustum : Region of space in the modeled world that may appear on the screen. It's a truncation with parallel near & far planes which defined a Field Of View (also called pyramid of vision).
- Asymmetric frustum : By default, the view frustum is arranged symmetrically around the camera's centre line but it doesn't necessarily need to be, one side could be smaller angle to the centre line than the opposite side.



- Single sided & BackFace culling : Optimizations processes to determines whether a polygon face (defined by the normal computed from vertices indices) is visible from camera direction or not in order to discard them from the rendering process.

Thanks

Questions ?

More investigation suggestions

- https://www.cg.tuwien.ac.at/~msh/viscull.pdf
- http://fr.slideshare.net/jasinb/embarrassingly-parallel-computation-for-occlusion-culling
- http://www.cse.chalmers.se/~uffe/vfc.pdf
- http://www.bogotobogo.com/Games/spatialdatastructure.php