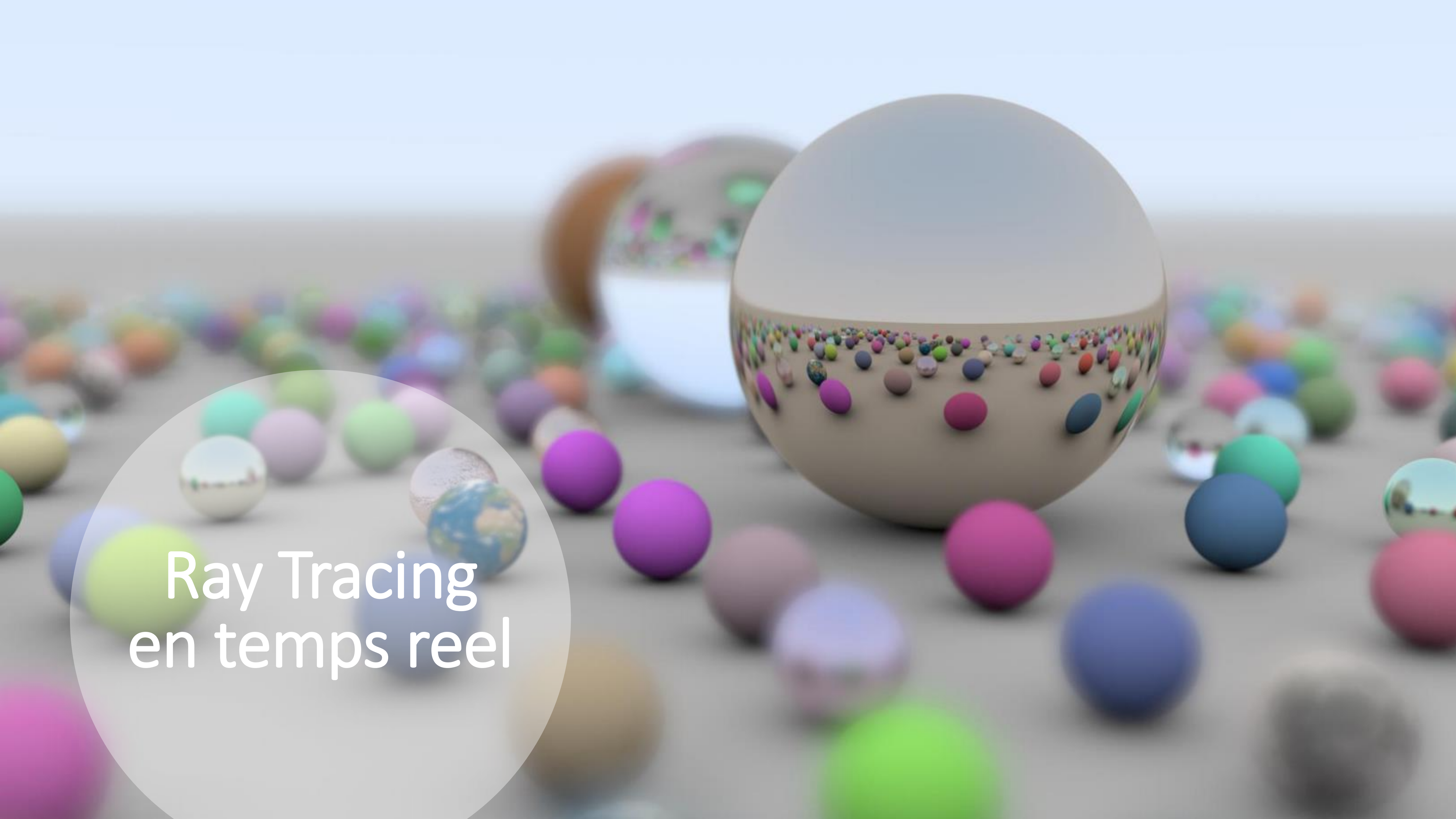
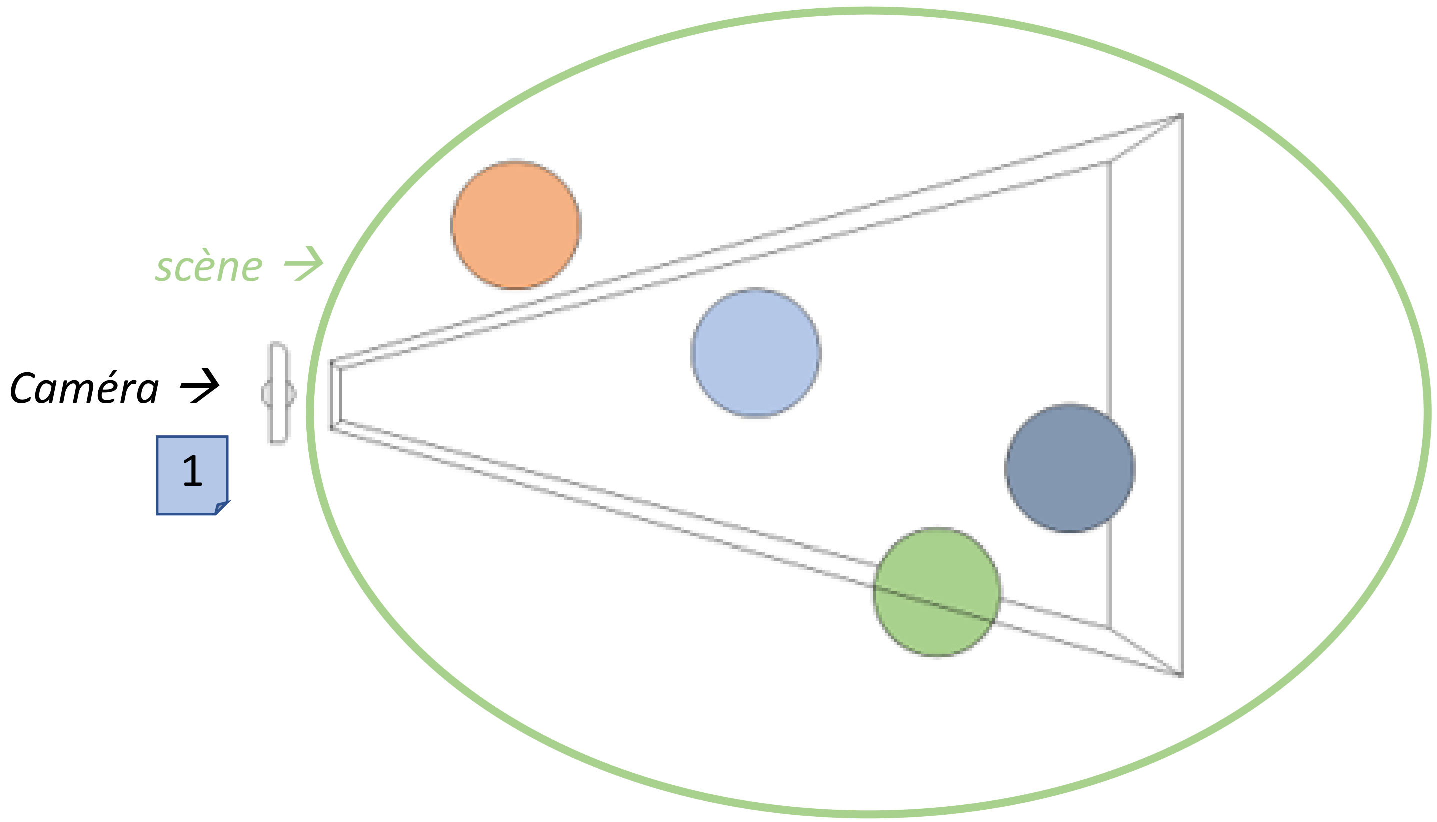


Ray Tracing en temps reel

Exposé

- Méthode de rendu actuelle
- Evolution et principe du Ray Tracing
- Denoising / Filtering
- Technologie RTX
- Limites, Futurs

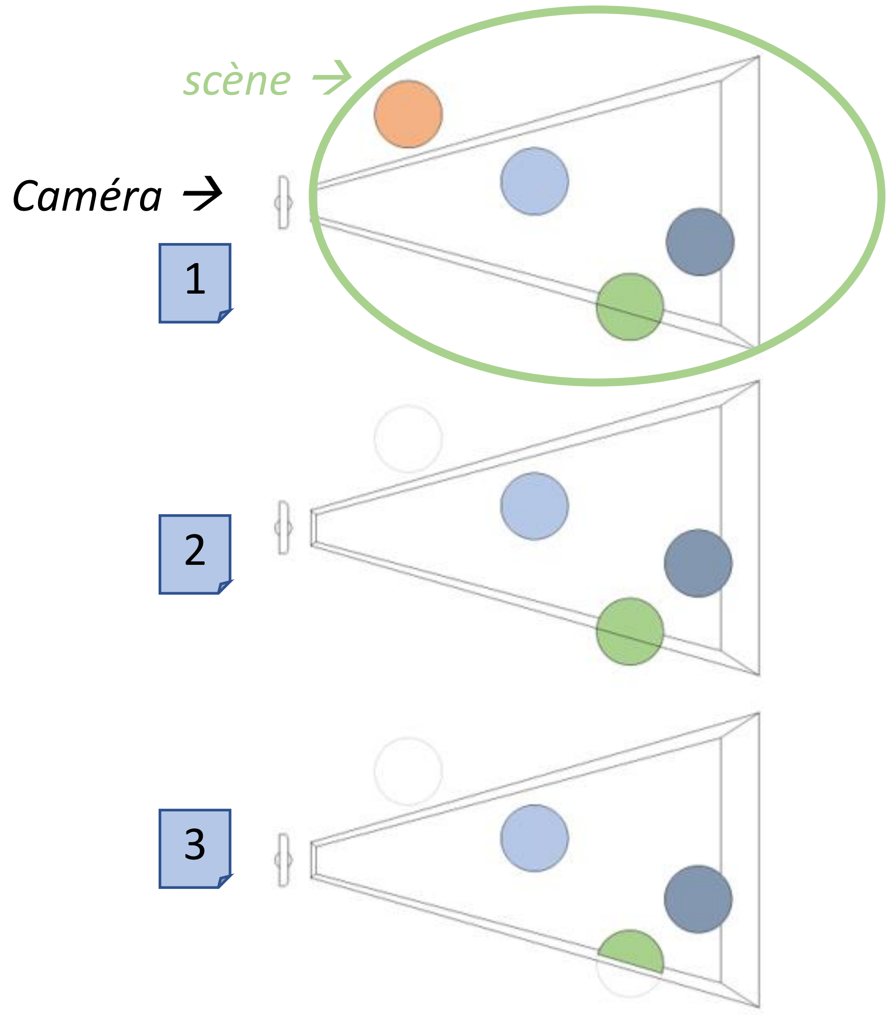
Méthode actuelle : Rastérisation



-> pixel reduction techniques, back-face culling, z-buffering…

https://devblogs.microsoft.com/directx/announcing-microsoft-directx-raytracing/

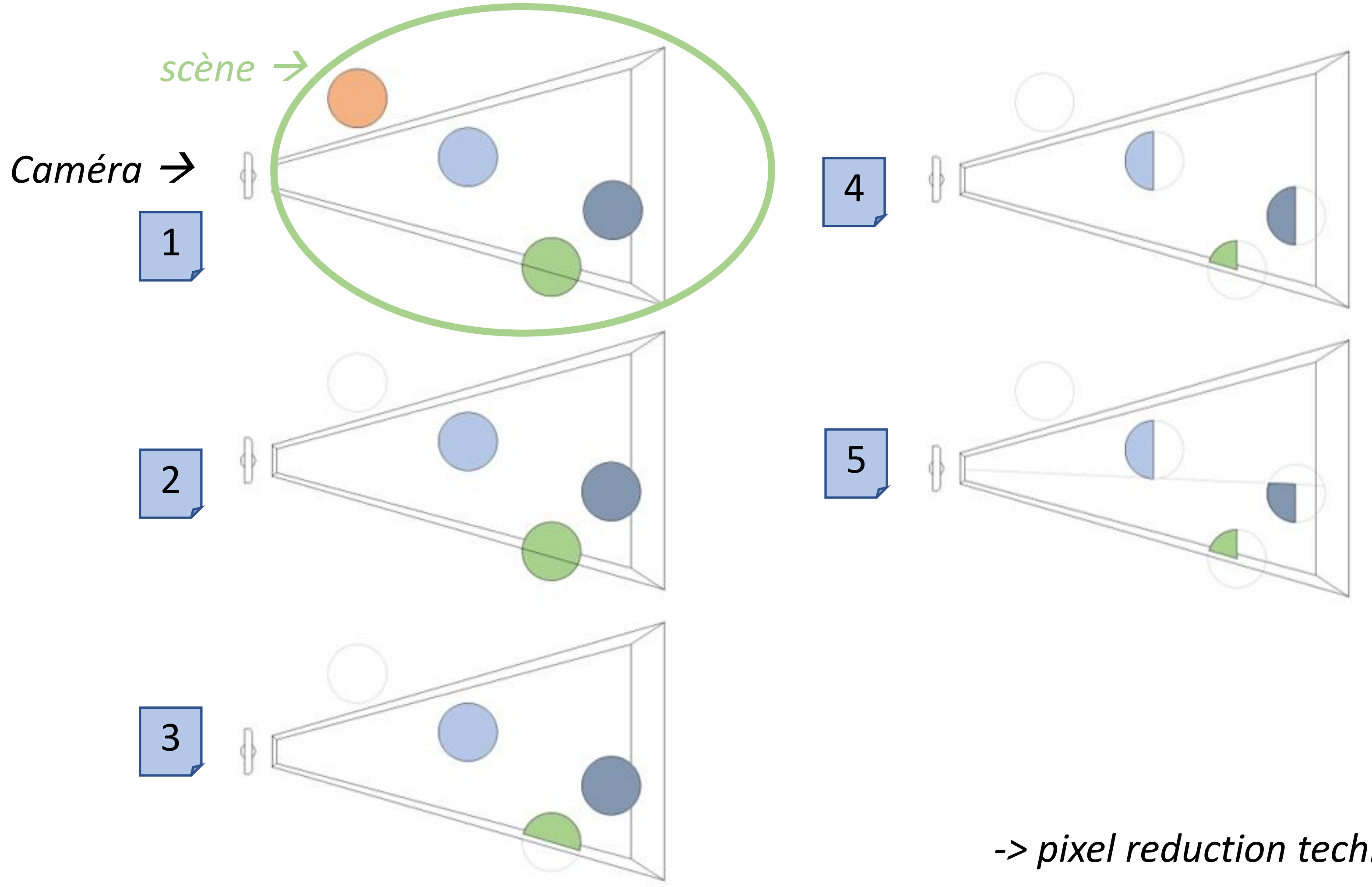
Méthode actuelle : Rastérisation



-> pixel reduction techniques, back-face culling, z-buffering…

https://devblogs.microsoft.com/directx/announcing-microsoft-directx-raytracing/

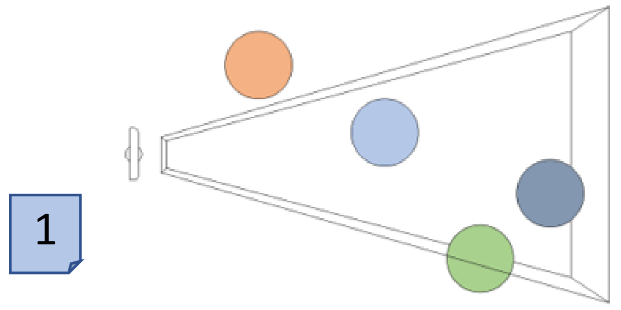
Méthode actuelle : Rastérisation

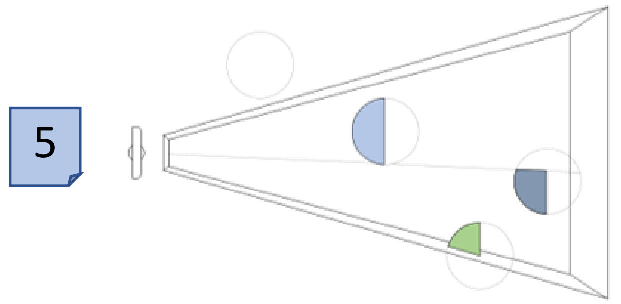


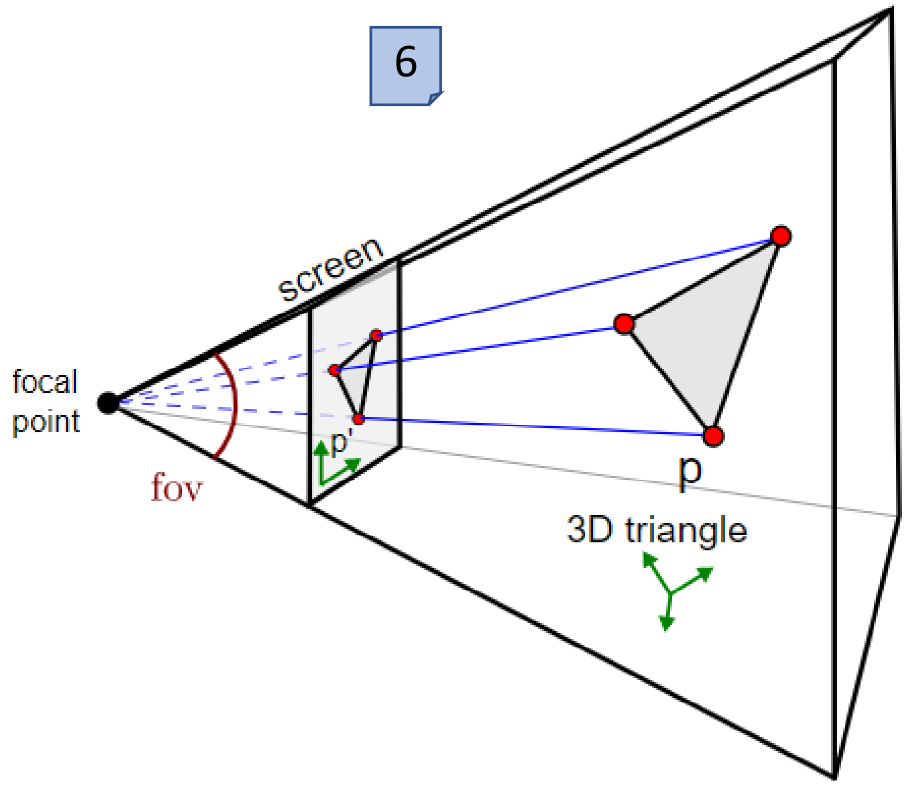
-> pixel reduction techniques, back-face culling, z-buffering…

https://devblogs.microsoft.com/directx/announcing-microsoft-directx-raytracing/

Méthode actuelle : Rastérisation



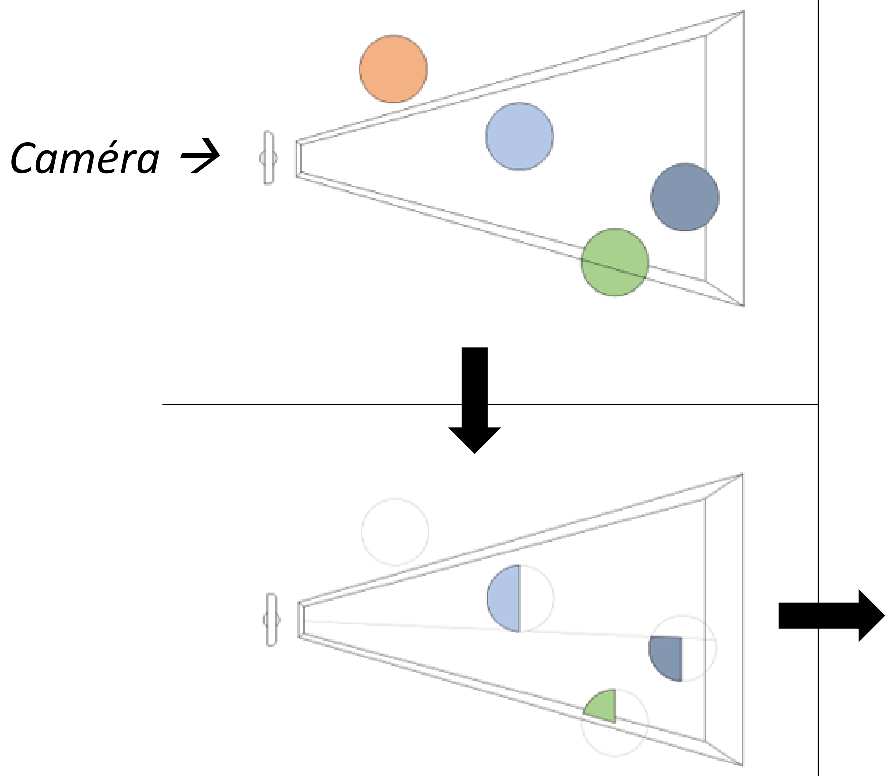


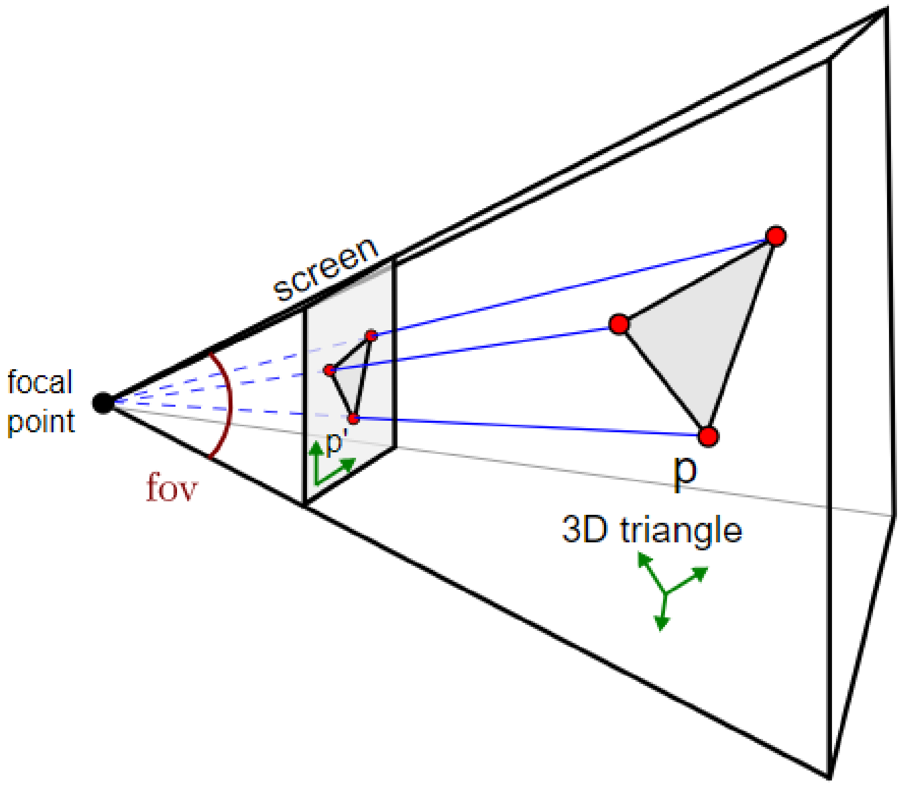


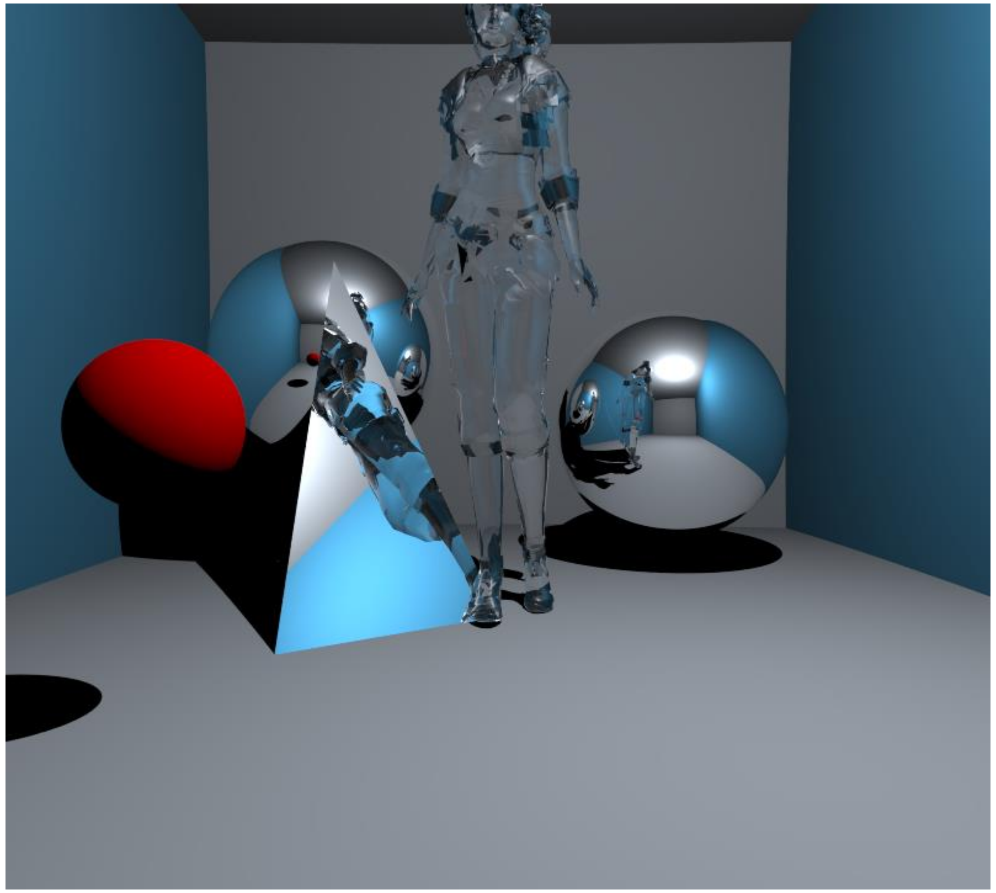
-> pixel reduction techniques, back-face culling, z-buffering…

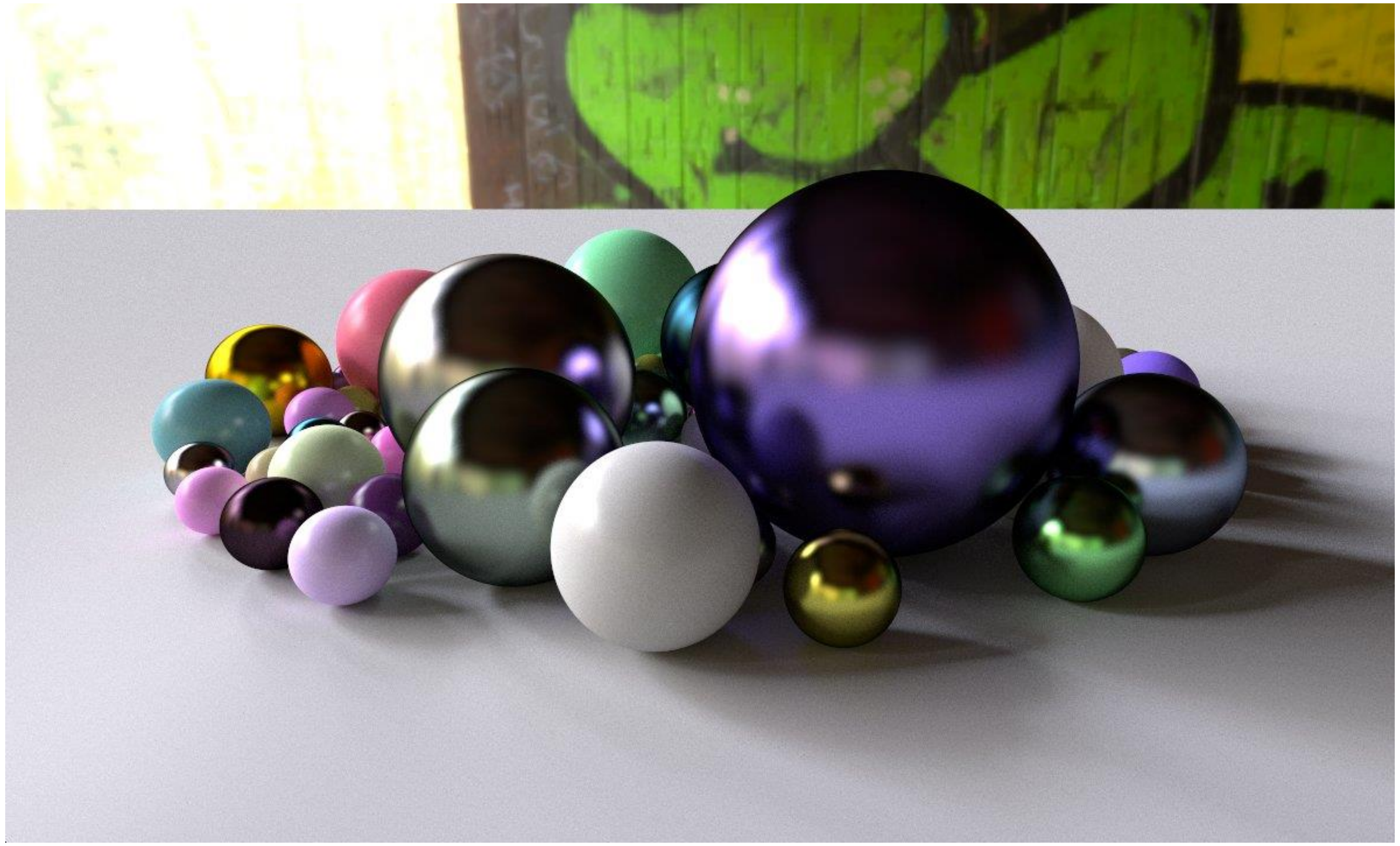
https://devblogs.microsoft.com/directx/announcing-microsoft-directx-raytracing/

Méthode actuelle : Rastérisation









Ray Tracing:

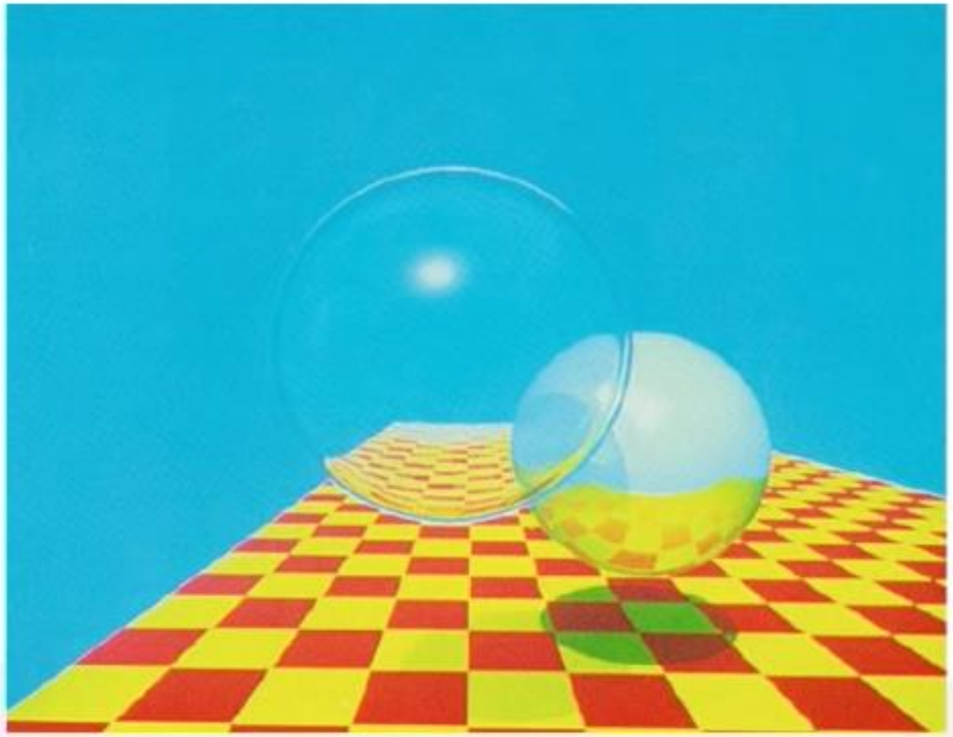
- **Refractions**
- **Reflexions**
- **Ombres douces**
- **Illumination globale**
- **…**

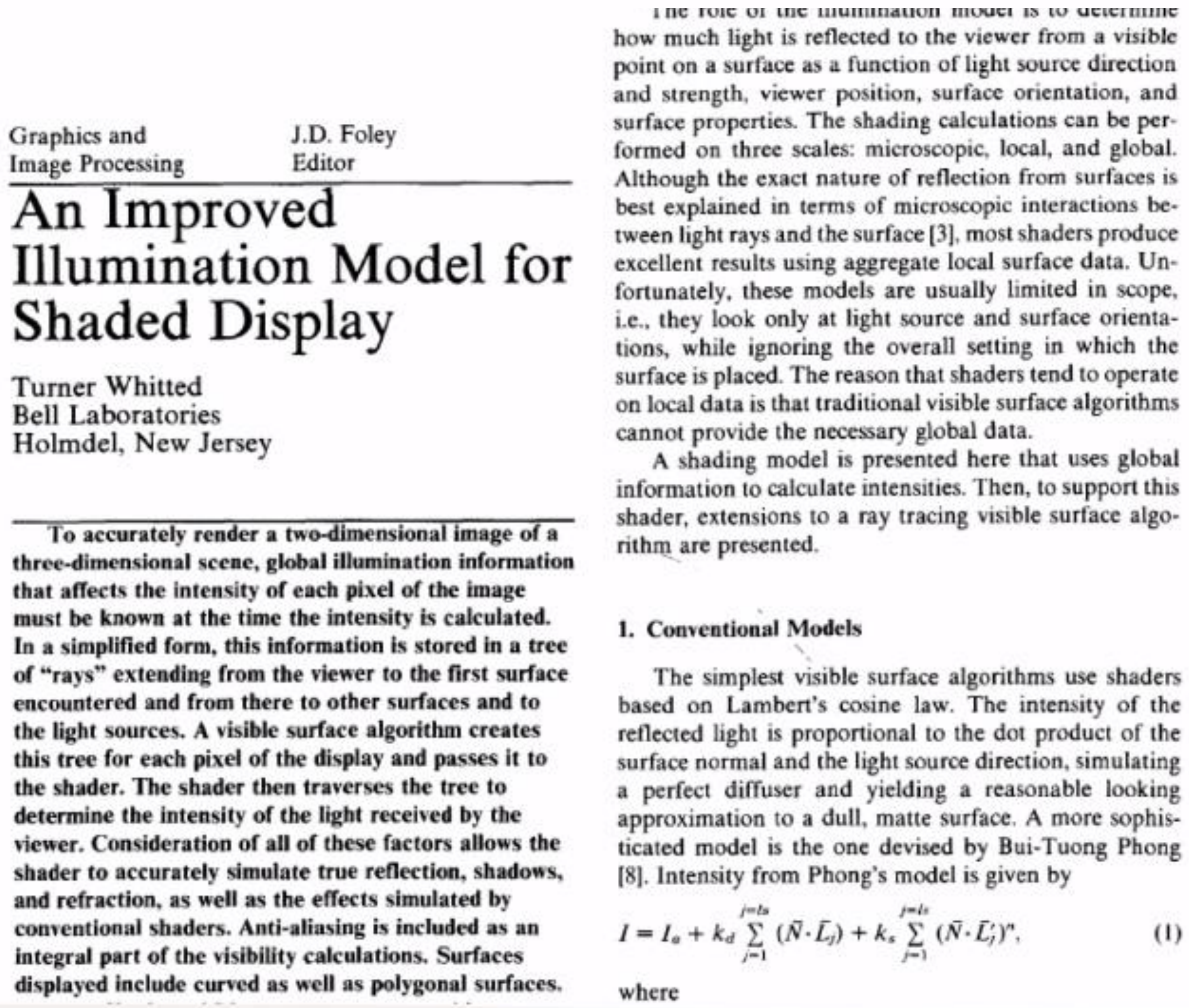
Turner Whitted 1980

included a bounding volume hierarchy

reflection and refraction

adaptive antialiasing





Graphics and Image Processing

J.D. Foley Editor

An Improved Illumination Model for Shaded Display

Turner Whitted
Bell Laboratories
Holmdel, New Jersey

To accurately render a two-dimensional image of a three-dimensional scene, global illumination information that affects the intensity of each pixel of the image must be known at the time the intensity is calculated. In a simplified form, this information is stored in a tree of "rays" extending from the viewer to the first surface encountered and from there to other surfaces and to the light sources. A visible surface algorithm creates this tree for each pixel of the display and passes it to the shader. The shader then traverses the tree to determine the intensity of the light received by the viewer. Consideration of all of these factors allows the shader to accurately simulate true reflection, shadows, and refraction, as well as the effects simulated by conventional shaders. Anti-aliasing is included as an integral part of the visibility calculations. Surfaces displayed include curved as well as polygonal surfaces.

The role of the illumination model is to determine how much light is reflected to the viewer from a visible point on a surface as a function of light source direction and strength, viewer position, surface orientation, and surface properties. The shading calculations can be performed on three scales: microscopic, local, and global. Although the exact nature of reflection from surfaces is best explained in terms of microscopic interactions between light rays and the surface [3], most shaders produce excellent results using aggregate local surface data. Unfortunately, these models are usually limited in scope, i.e., they look only at light source and surface orientations, while ignoring the overall setting in which the surface is placed. The reason that shaders tend to operate on local data is that traditional visible surface algorithms cannot provide the necessary global data.

A shading model is presented here that uses global information to calculate intensities. Then, to support this shader, extensions to a ray tracing visible surface algorithm are presented.

1. Conventional Models

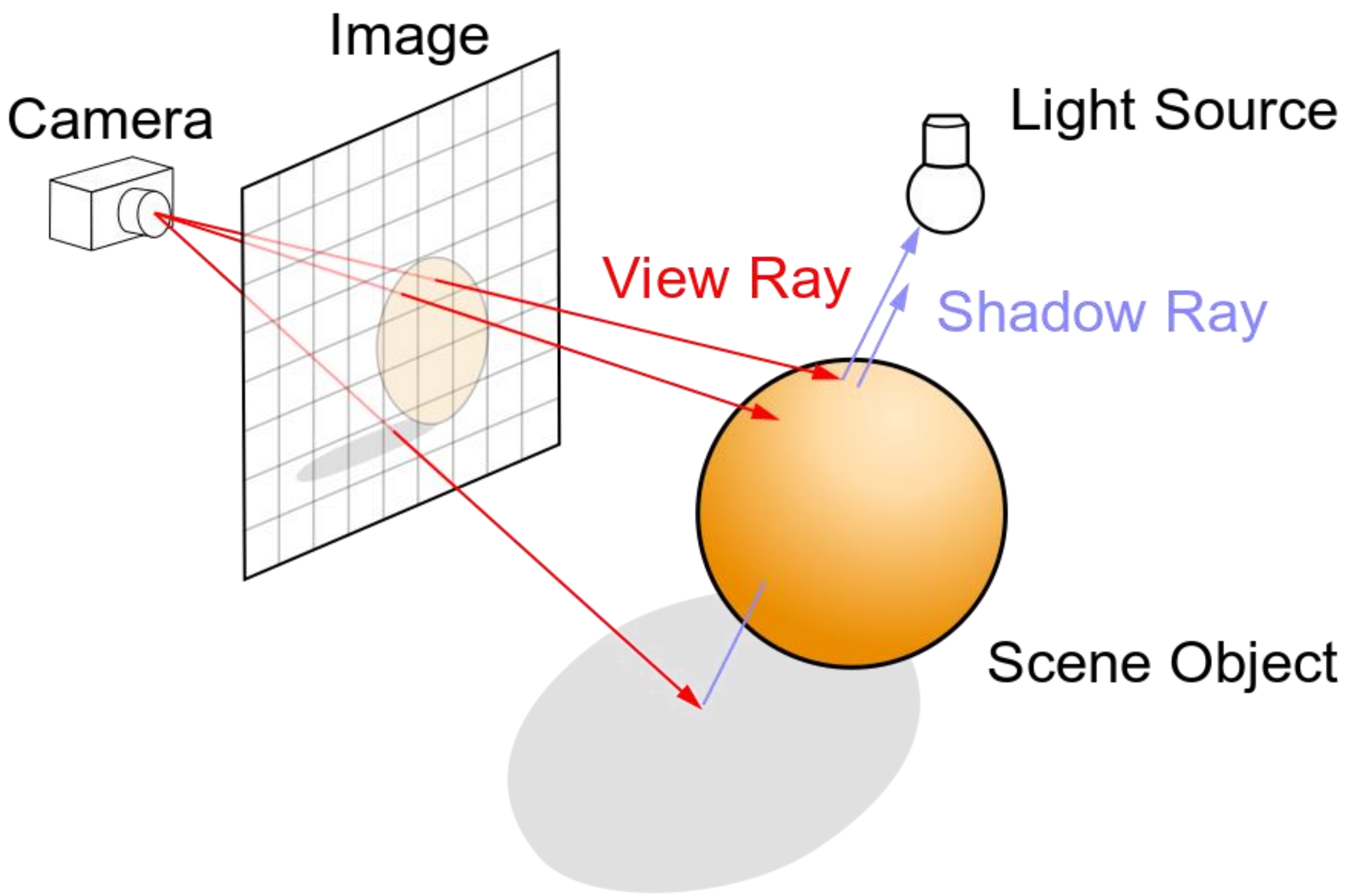
The simplest visible surface algorithms use shaders based on Lambert's cosine law. The intensity of the reflected light is proportional to the dot product of the surface normal and the light source direction, simulating a perfect diffuser and yielding a reasonable looking approximation to a dull, matte surface. A more sophisticated model is the one devised by Bui-Tuong Phong [8]. Intensity from Phong's model is given by

$$I = I_a + k_d \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}_j) + k_s \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}'_j)^n, \quad (1)$$

where

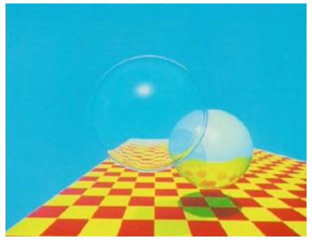
Principe

Rayon = Origine + Direction



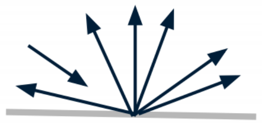
Ray Tracing initial papers

- **Whitted** 1980



- **Cook** 1984
  → Soft shadow

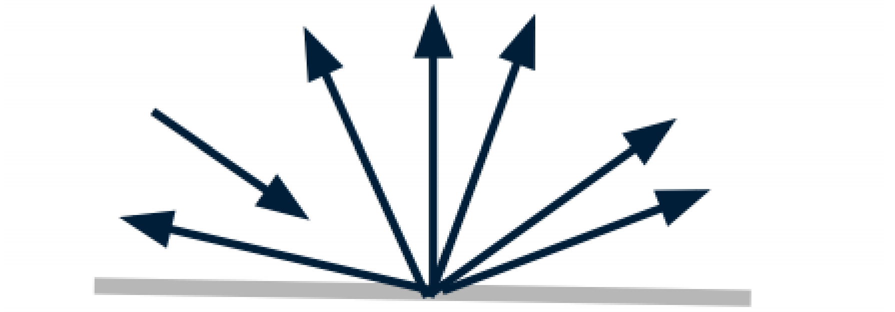
- **Kajiya** 1986
  → Global illumination



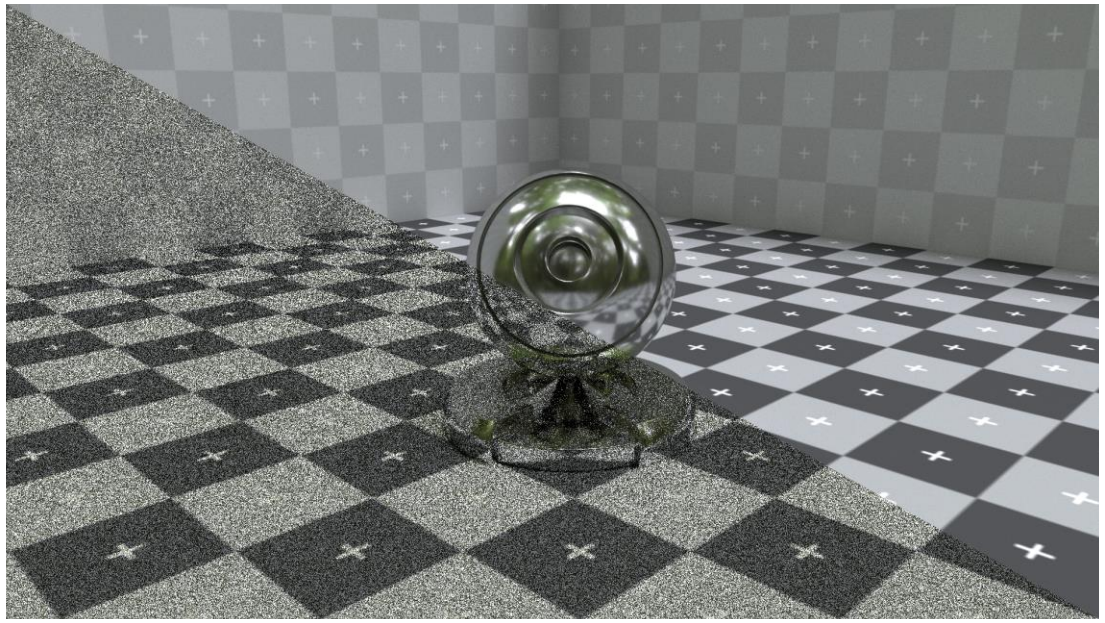
RenderTime 00:03.03
Peak Memory 1.54M
Sobol

Scene with Global Illumination

Global illumination



→ A chaque intersection de rayon, un autre rayon est lancé dans une direction aléatoire

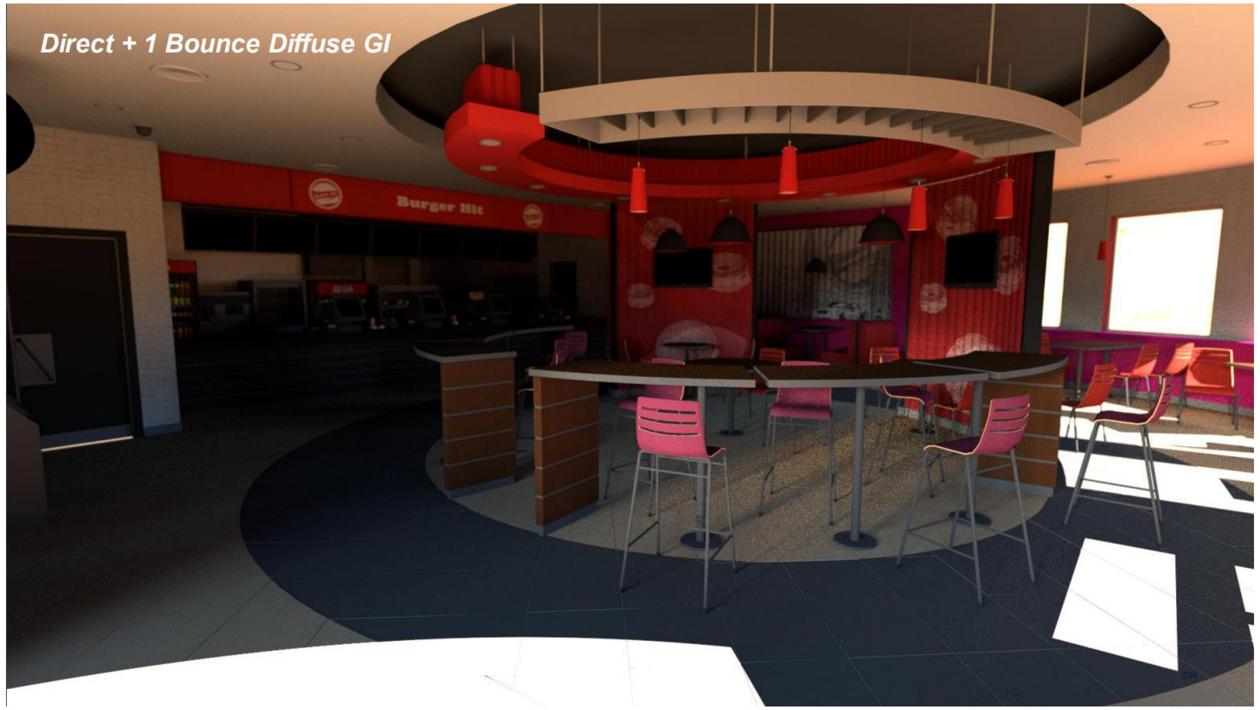


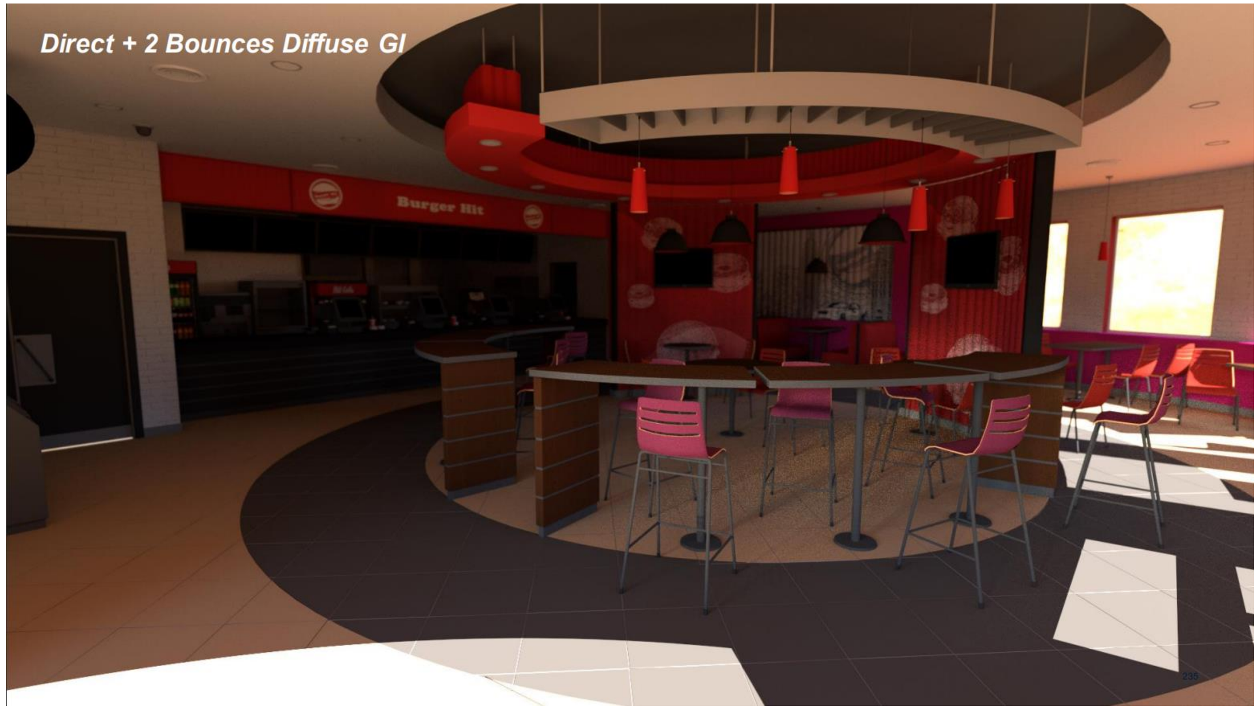
→ Trop peu de rayons supplémentaires = image bruitée

Global illumination

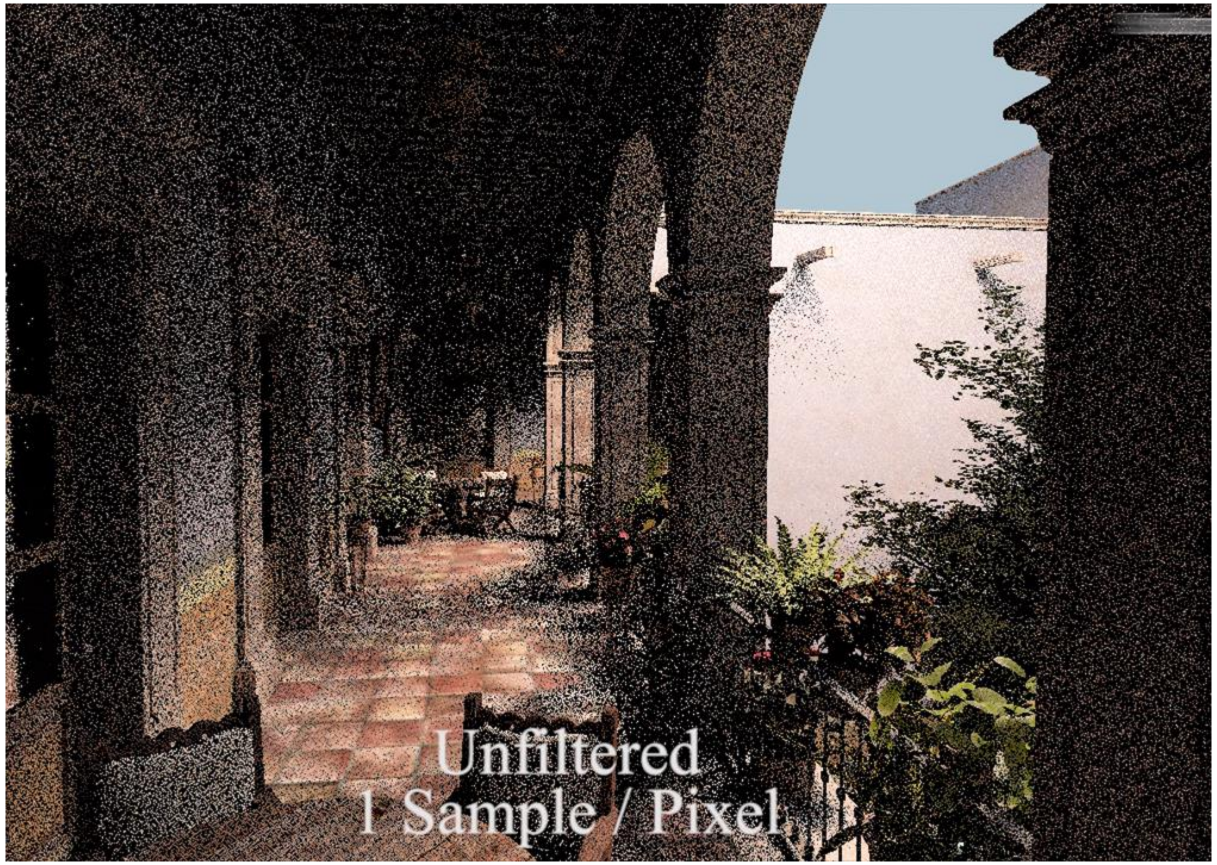
Plusieurs rebonds par pixel sont nécessaires.

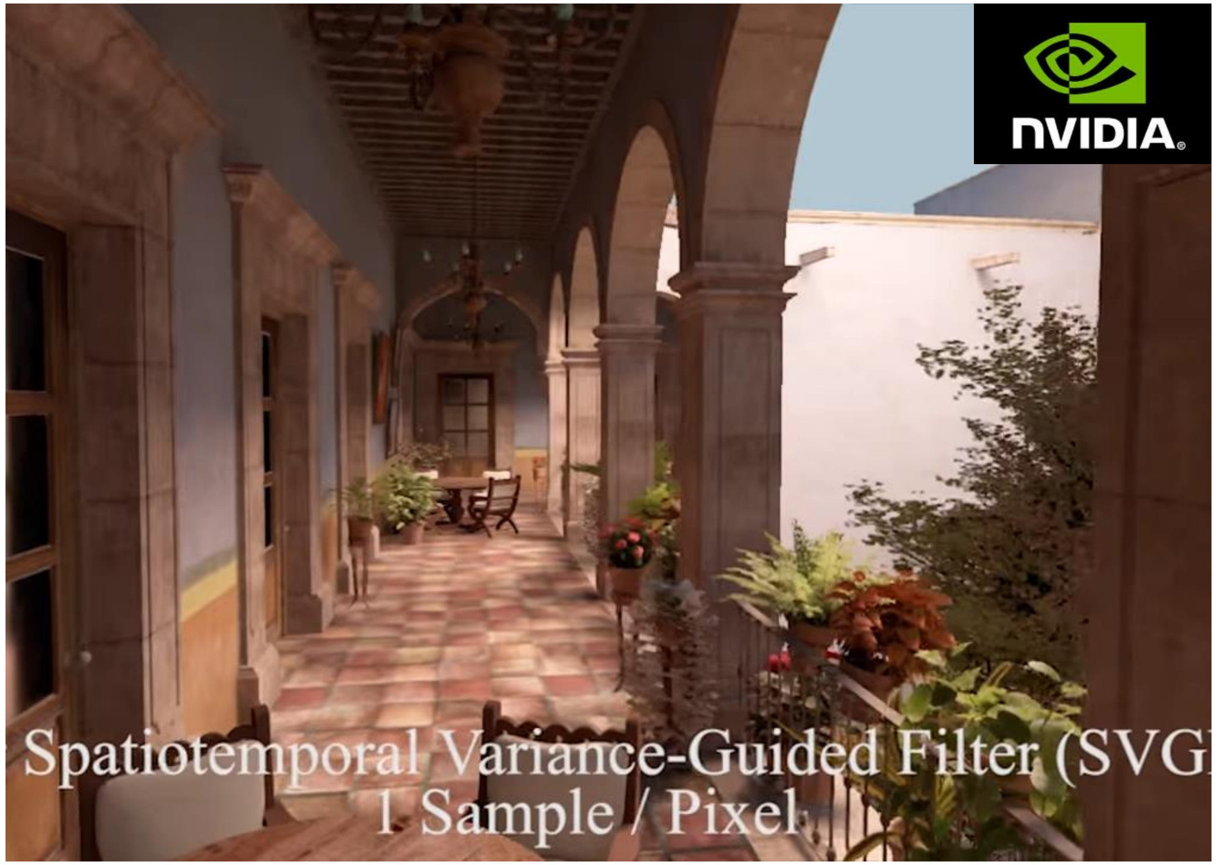
→ Converger en temps réel est impossible.





[2017][NVIDIA] Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder

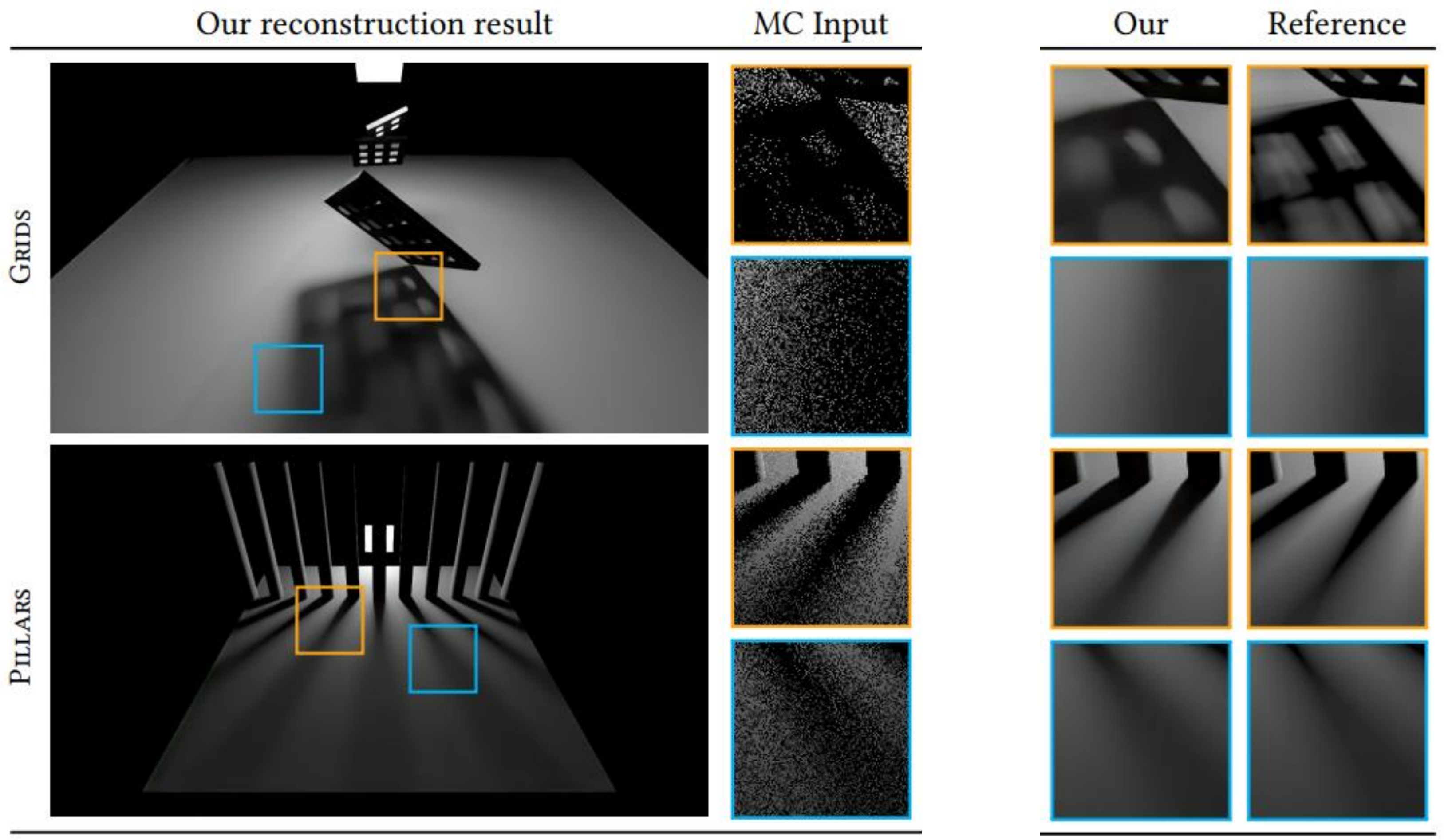




Denoising

Spatial AND Temporal filter
→ Image 1spp = 10ms
→ Filter = 10 ms

Denoising



[2017][NVIDIA] Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder

DIRECT X Raytracing

- Direct X : suite de bibliothèques de programmation pour le multimédia: jeux video.
  Etoffée depuis 1995, aujourd’hui en version DirectX 12, largement répandue et utilisée.

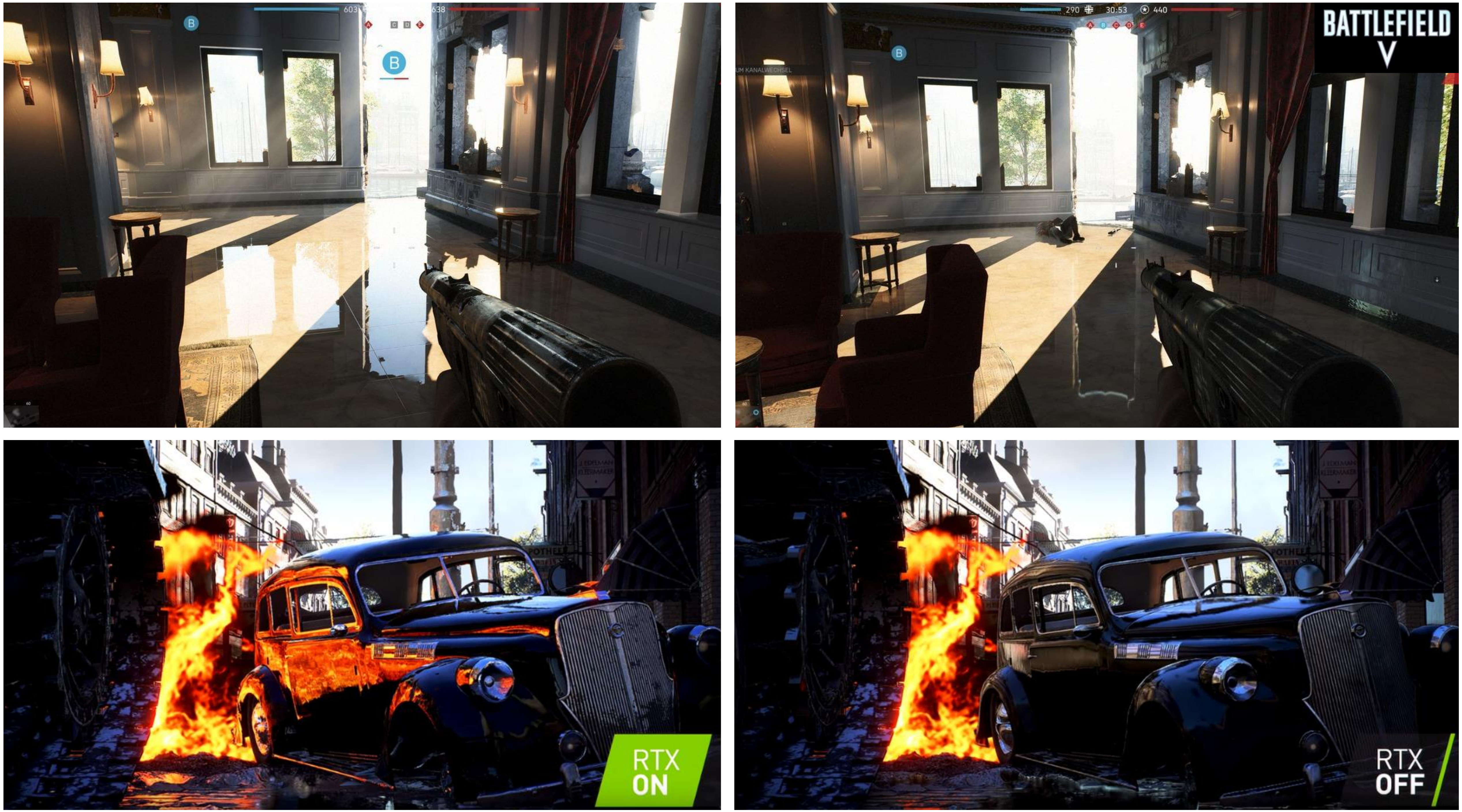
- En 2018, Microsoft annonce la sortie de DirectX Raytracing à la GDC (*Game Developers Conference*):
  - API intégrant toutes les techniques de base du Raytracing
  - Pipeline de shaders (programmes destinés à être executé sur GPU)
  - Encapsule la partie complexe touchant au GPU: allocation mémoire, transfert CPU/GPU

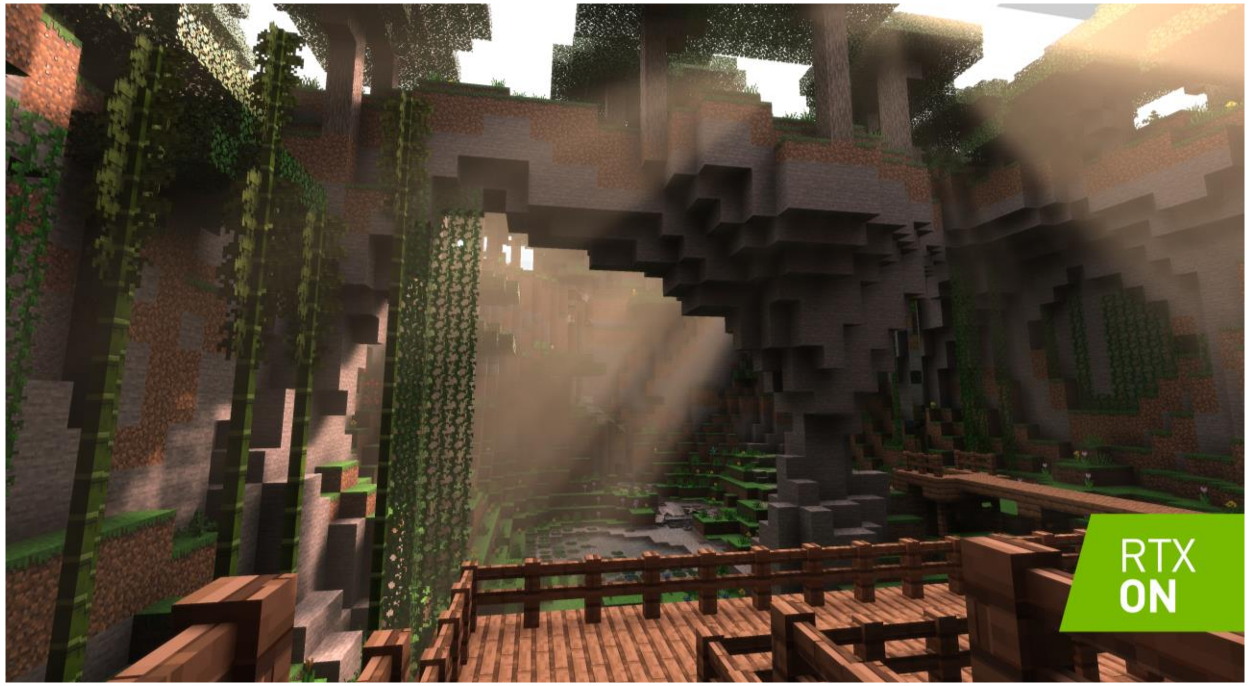
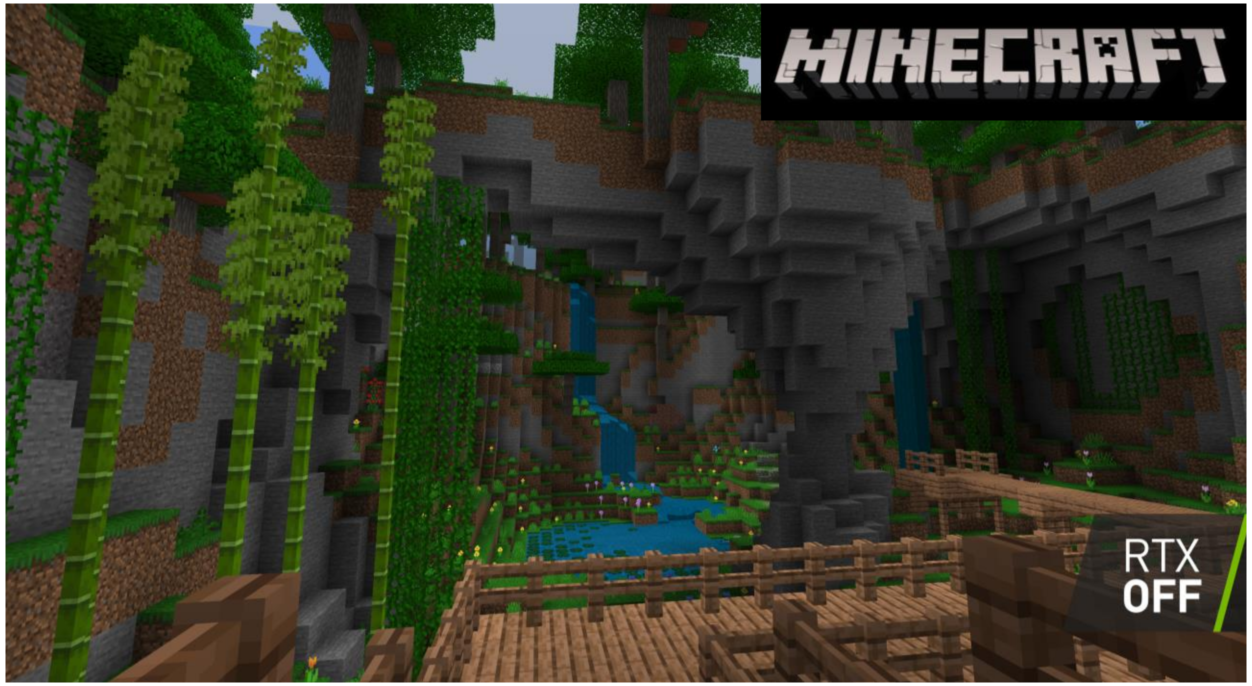
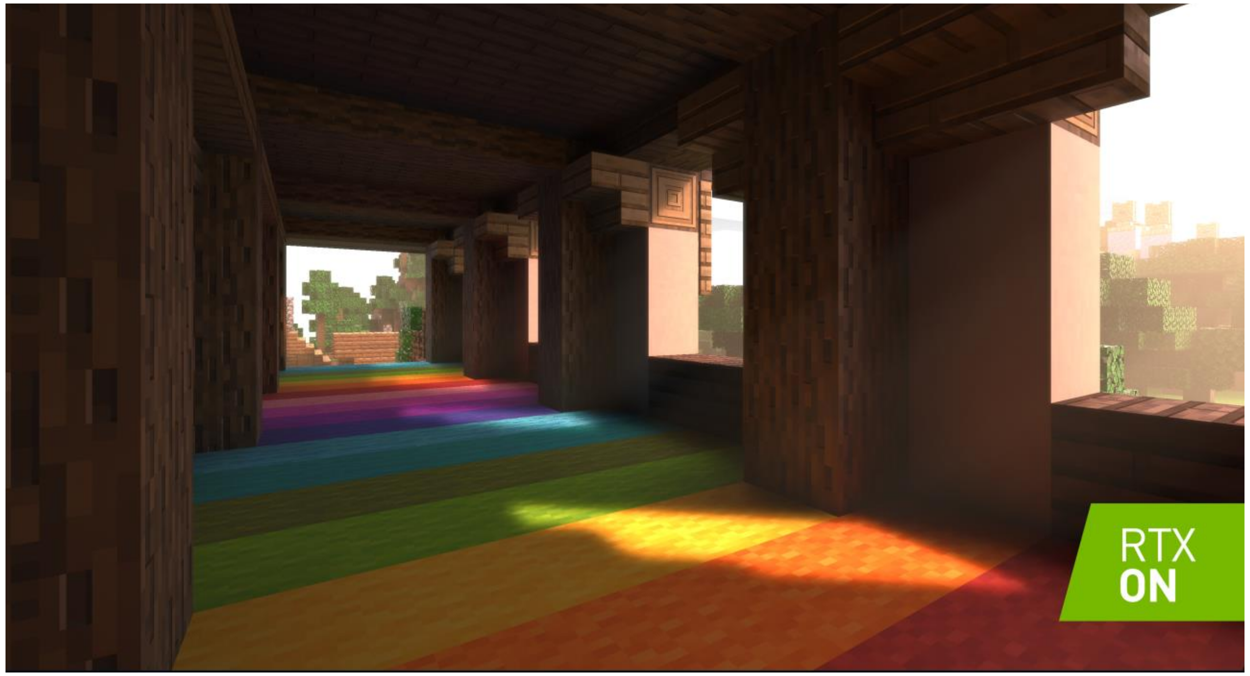
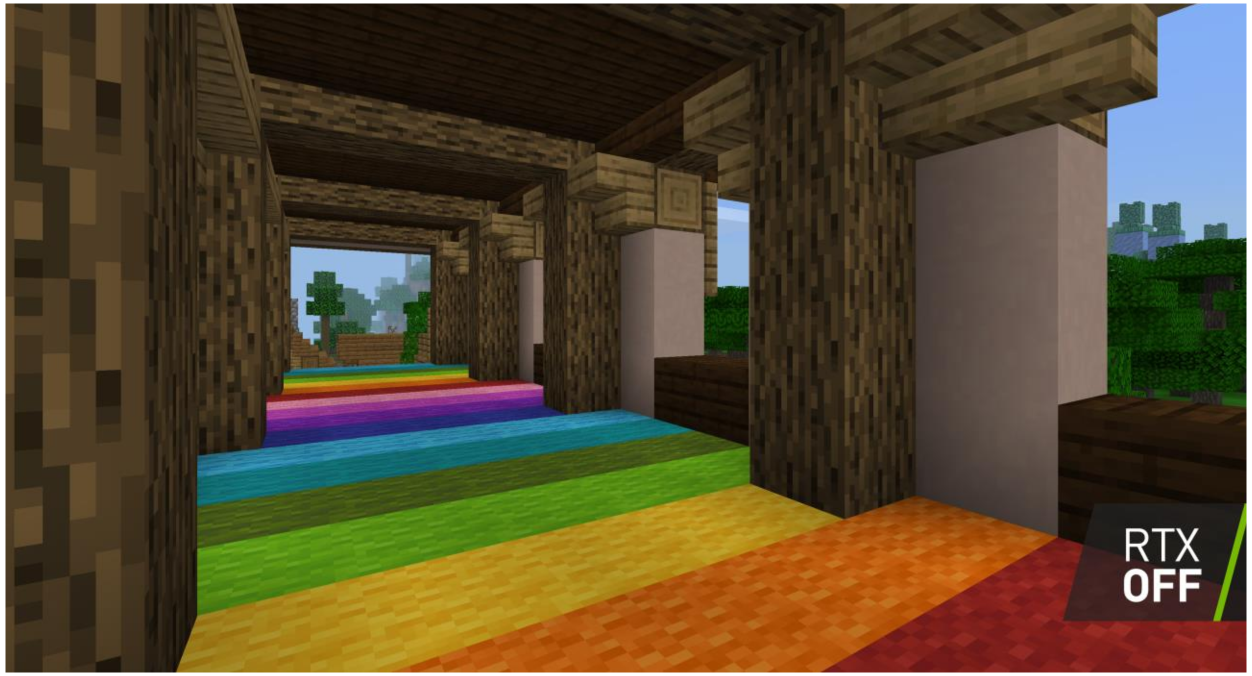
DIRECT X Raytracing

Futuremark, 3DMark

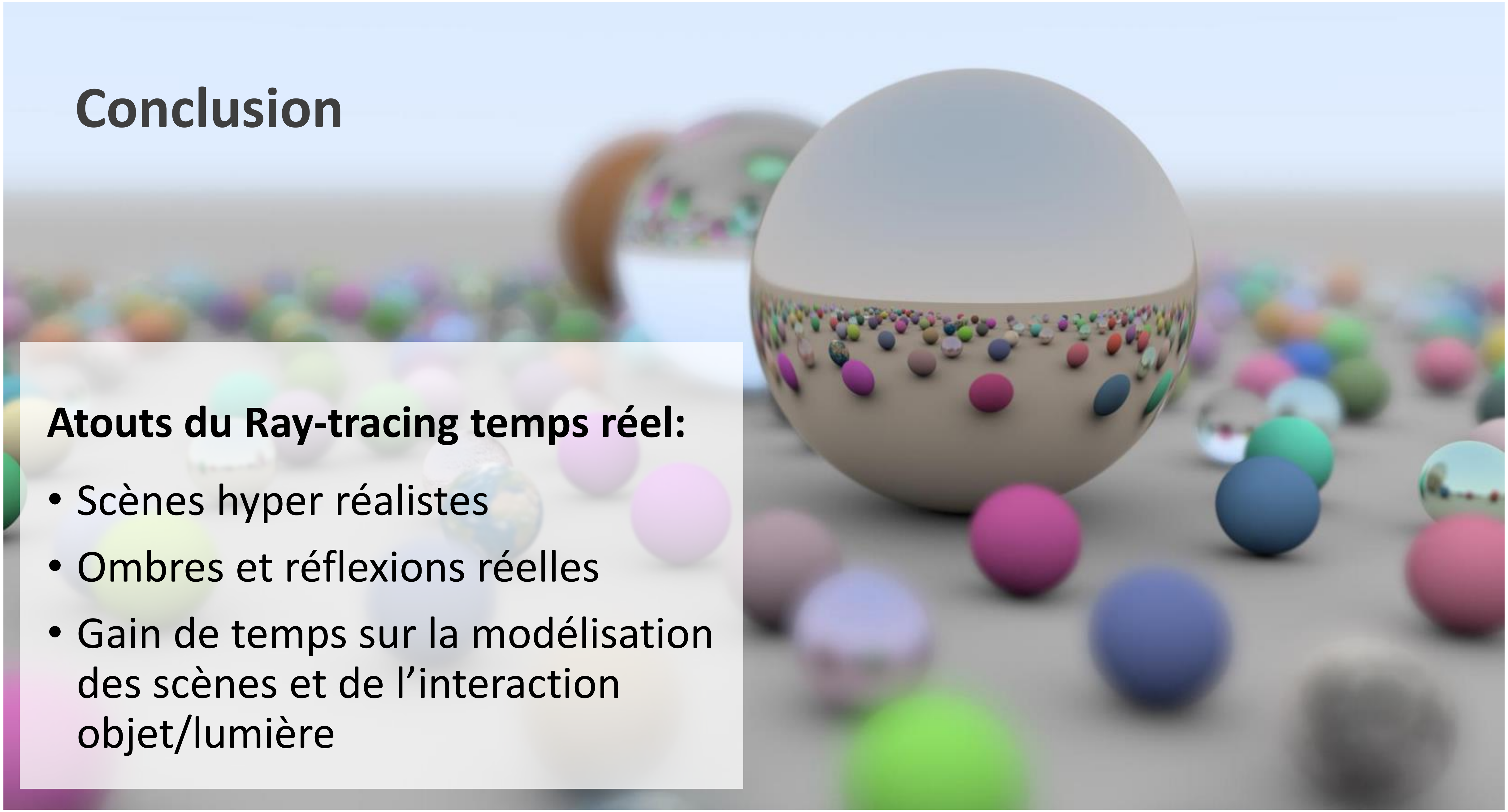
Unity Technologies, Unity Engine

- Déjà en place dans les grands moteurs de jeux
- A porté du grand public
- Permet de mixer Rastérisation et Raytracing, aujourd'hui indispensable.
- Présent sur la prochaine console Microsfot


603
638
290
30:53
440
UM KANALWECHSEL
BATTLEFIELD V
RTX ON
RTX OFF


RTX
ON

MINECRAFT
RTX
OFF

RTX
ON

RTX
OFF

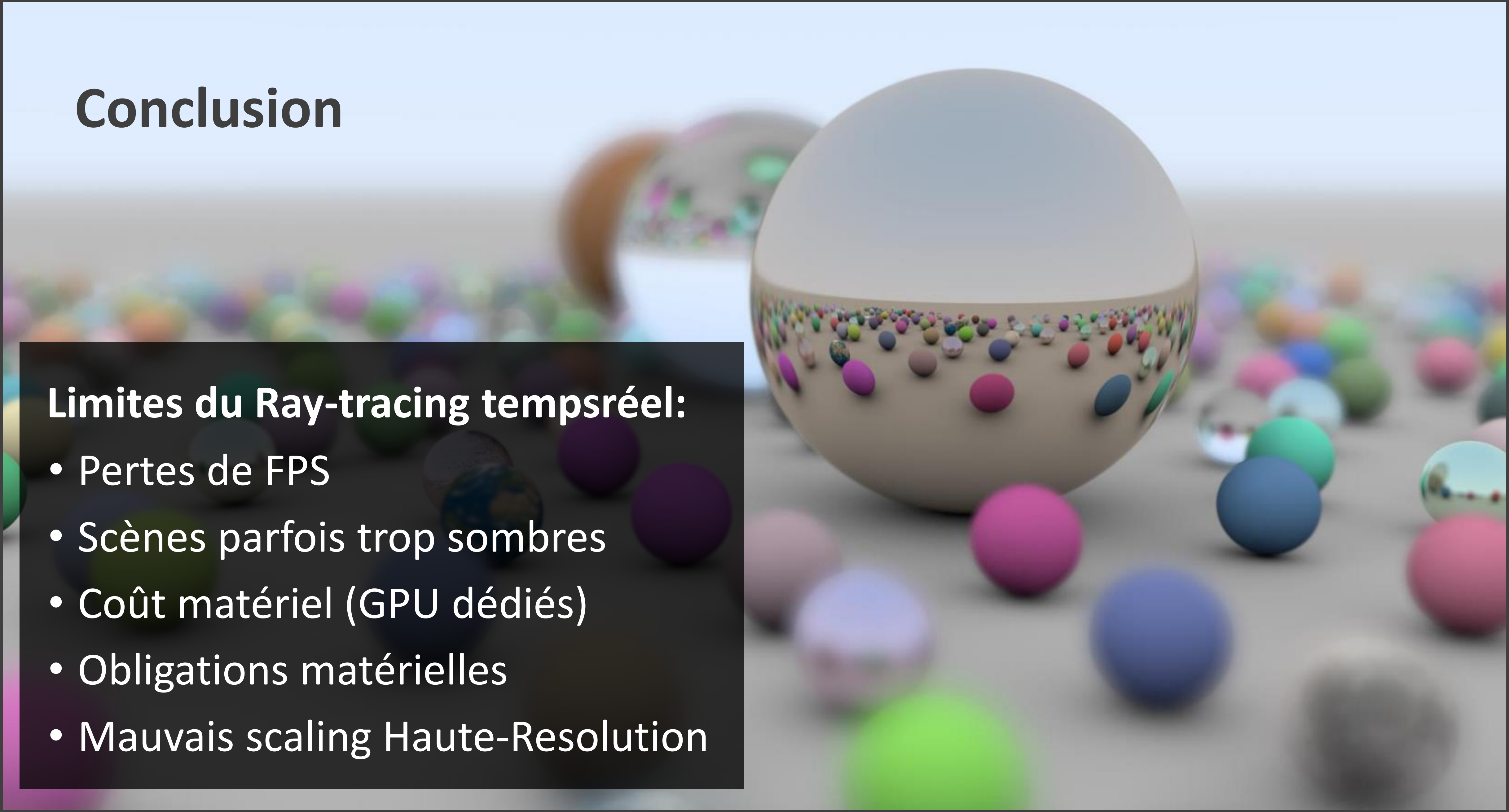
Conclusion



Atouts du Ray-tracing temps réel:

- Scènes hyper réalistes
- Ombres et réflexions réelles
- Gain de temps sur la modélisation des scènes et de l’interaction objet/lumière

Conclusion



Limites du Ray-tracing tempsréel:

- Pertes de FPS
- Scènes parfois trop sombres
- Coût matériel (GPU dédiés)
- Obligations matérielles
- Mauvais scaling Haute-Resolution

Conclusion

Futur du Ray-tracing temps réel:

- Techniques de Denoising
- Synergie Rastérisation/R-Tracing
- Hardware spécialisés
- Portabilité sur console

Images from http://intro-to-dxr.cwyman.org