

MV54 - Moteur de Rendu

Multi-pass

Julien Barbier

10 avril 2023

Multi-pass rendering : Le vrai avantage d'un moteur 3D

Deferred pass

Construction du FrameGraph de frame

Passe de Shadow

Convertir de HDR à SDR : le tonemapping

Conclusion

Multi-pass rendering : Le vrai avantage d'un moteur 3D

Multi-pass rendering

Définition

- Permet de faire le rendu finale sur plusieurs passes à chaque frame
- Permet d'appliquer des effets de rendus en screen space (SSAO, Vignetting, Anti-aliasing...)
- Permet de faire des optimisations de rendus avec les effets de lumières (deferred rendering)

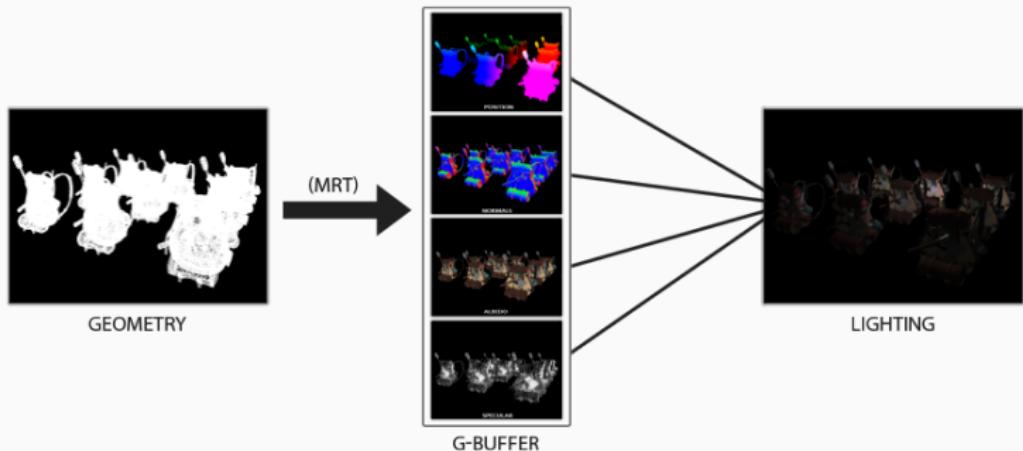


Figure 1 : Exemple de rendu multi-pass (deferred pass) (vient de LearnOpenGL
(<https://learnopengl.com/Advanced-Lighting/Deferred-Shading>))

Deferred pass

Deferred pass i

Objectif

- Permet de faire les calculs de lumières en screen space et non sur la géométrie
- Passage d'une complexité de $O(n*m)$ à $O(n+m)$
- Demande une passe de géométrie avant pour définir sur plusieurs targets les caractéristiques du matériaux.
- Existe différents types de deferred pass comme le LightPrePass (exige une passe après le calcul des lumières pour appliquer le résultat)

Deferred pass ii

Avantage

- Gain important si beaucoup de lumières
- Permet de déporter des calculs lourds sur 2 passes plus petites

Désavantage

- Gourmand en bande passante
- Marche difficilement avec les objets semi-transparents

GBuffer

Définition

- Réalise le packing des paramètres de matériaux
- Peut être utilisé pour packer d'autres paramètres (les ombres, la radiosity)
- Stocke les informations de position et de normal

MRT (Multiple Render Target)

Définition

- Possibilité de rendre sur plusieurs Render Target à la fois
- Différence avec le VPRT : Même view proj dans le vertex shader donc même pixel touché dans la rasterization.
- Permet de rendre différents paramètres sur différentes render target

GBuffer

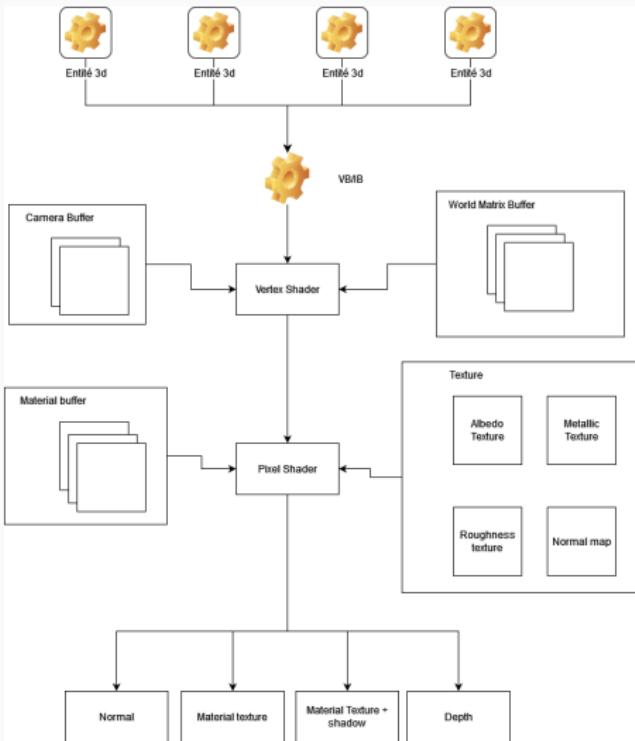


Figure 2 : Présentation d'une passe de GBuffer

Exemple de rendu de GBuffer ("Real-time lighting via Linked List" de Abdul Bezrati (Insomniac Games))



Figure 3 : Texture contenant la diffuse

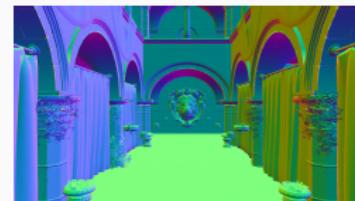


Figure 4 : Texture contenant les paramètres de normal

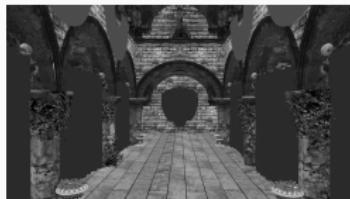


Figure 5 : Texture contenant les paramètres de Glossiness (1 - roughness)



Figure 6 : Texture contenant les paramètres de spécularité/reflectances

Deferred Rendering : Application des lumières

Application

- Deux possibilités :
 - Ne rendre que les éléments de lumières puis refaire une passe de géométrie pour les appliquer (Light Pre-Pass pipeline)
 - Tout rendre dans la même pass (deferred pass pure).

Définition

- Passe en screen space
- Calcul les différentes lumières et l'impact sur la scène
- Calcul dans le Pixel Shader

LightPrePass

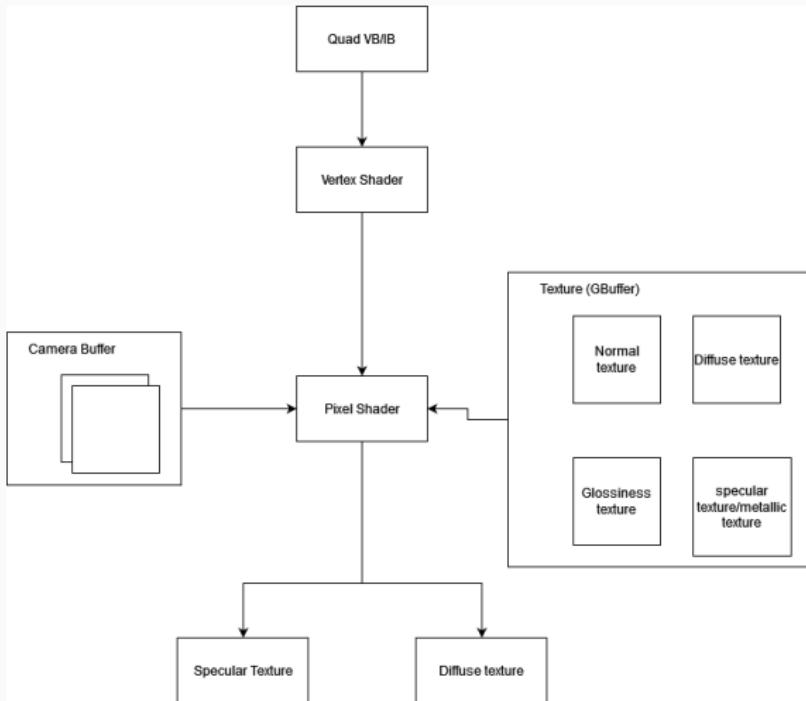


Figure 7 : Diagramme d'une passe LightPrePass

LightPrePass : Comment ça marche

```
resultDiffuse <- 0
resultSpecular <- 0
Get the coord of the pixel
Get the texcoord of the pixel

Unpack material parameter inside gbuffer texture
Unpack the normal vector inside the normal texture from gbuffer

Reconstruct world position with depth and invert view proj
for light in lights
    currentDiffuse = ComputeDiffuseTerm(light, normal, position, material)
    currentSpecular = ComputeSpecularTerm(light, normal, position, material)

    resultDiffuse += currentDiffuse
    resultSpecular += currentSpecular
endfor

resultDiffuse += GlobalIlluminationDiffuse()
resultSpecular += GlobalIlluminationSpecular()

return {resultDiffuse, resultSpecular} // MRT rendering
```

Pseudo-Code du calcul du LightPrePass

Problèmes

- Peut très rapidement atteindre la valeur maximal d'une texture RGBA8 (256 ou 1.0f)
- Provoque une image toute blanche + perte d'information

Solutions

- Faire des lumières avec des puissances faibles
- Le HDR

Définition

- Acronyme de High Definition Range
- Permet de dépasser la limite de 256 (ou 1.0 en normalisé)
- Correspond à des textures avec des tailles de bits entre 10 et 32 par channel
- Ne peut pas être rendu par une majorité de moniteurs

LightPrePass : exemple de rendu ("CryEngine 3 : reaching the speed of light" de Anton Kaplanyan (Crytek))



Figure 8 : Texture de spécularité du LightPrePass (normalisé)



Figure 9 : Texture de diffuse du LightPrePass

Application du LightPrePass dans la passe final de rendu des entités 3d

- Application du résultat du LightPrePass dans la passe final du rendu de la géométrie.
- Application du rendu des ombres
- Application du fog si présent

Résultat

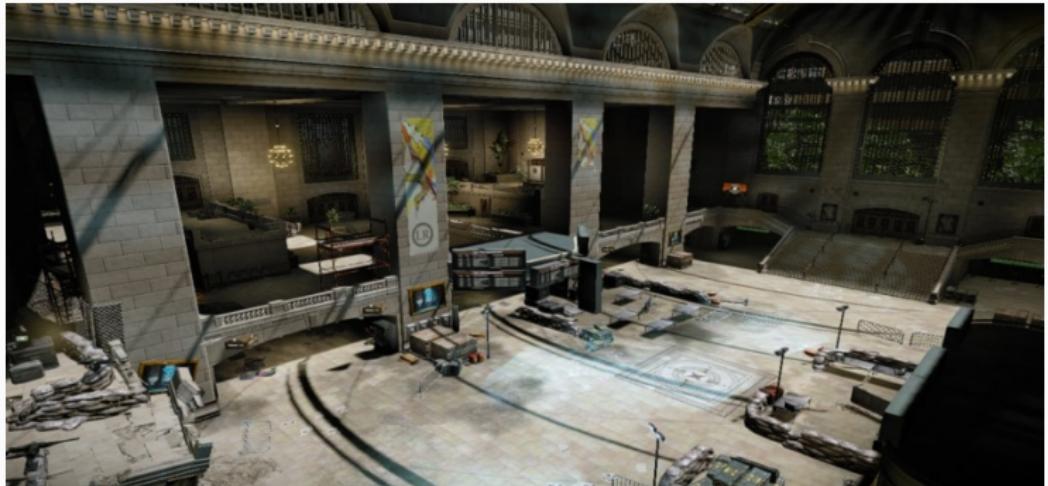


Figure 10 : Résultat final d'un Deferred Pass (avec application d'un tonemapping)

Architecture d'un FrameGraph d'un moteur 3D

Deferred Pass : une introduction au FrameGraph

- Nécessite plusieurs passes avec dépendance
- Nécessaire de réaliser une FrameGraph d'une frame.

Problème deferred pass avec HDR

- Utilisation du HDR : pas de possibilité de rendre directement sur l'écran
- Nécessité de réaliser une autre passe pour convertir l'image
- Manque la passe des ombres.

Construction du FrameGraph de frame

FrameGraph de frame

Définition

- Liste l'ensemble des passes de rendu et les interactions
- Obligatoire pour obtenir des scènes de rendus réalistes voir photo-réaliste
- Permet de voir les dépendances de chaque passe

FrameGraph du deferred pass

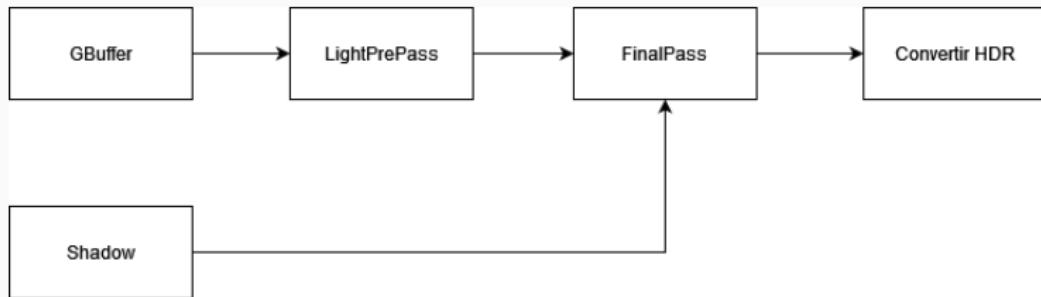


Figure 11 : Exemple de FrameGraph avec le LightPrePass

Types de passes de rendu

Existe 2 types de passes

- passe de géométrie : on rend les entité 3d de la scène
- screen pass : on rend directement dans une texture
 - pre-D3D11, OpenGL 4.3 : réalisé avec la FrameGraph rasterizer
 - D3D11 et ultérieure, et OpenGL 4.3 et ultérieur : réalisé en compute shader

FrameGraph du deferred pass avec distinction

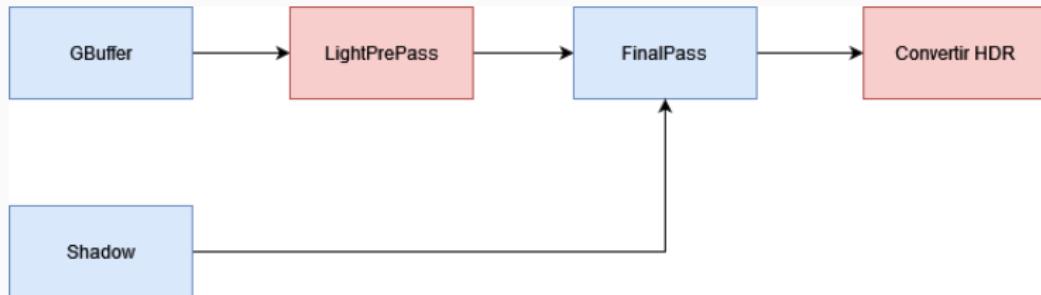


Figure 12 : Exemple de FrameGraph LightPrePass avec la distinction géométrie et screen pass (bleu : géométrie, rouge : screen pass)

Passe de Shadow

Passe de Shadow

Définition

- Passe de Géométrie
- Représente les ombres
 - zone où la lumière n'est pas appliquée à cause d'un élément.

Comment on fait ?

- Différents types de lumières = différentes façon de faire

Pass de shadow pour une lumière directionnel

- Réaliser une caméra qui se positionne sur la lumière
- Utiliser cette caméra comme caméra de rendu
- Utiliser une matrice de projection orthographique
- Utilisation d'un pixel shader minimaliste (voir depth only)

Représentation des shadow avec une directionnal light

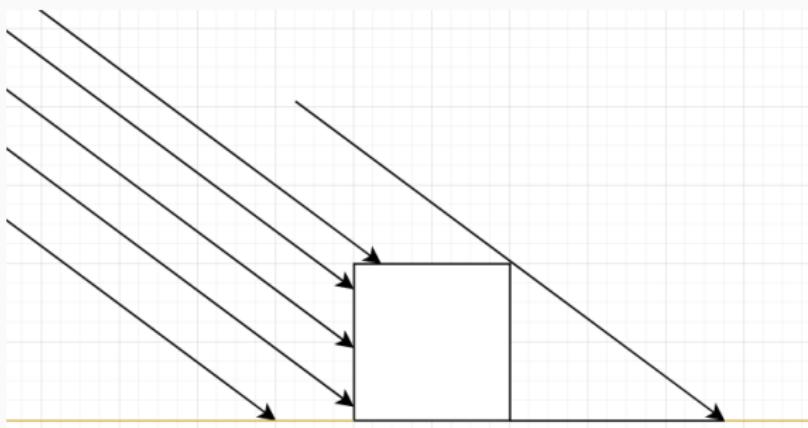


Figure 13 : Schéma présentant les shadows avec directional light

Exemple de shadow map

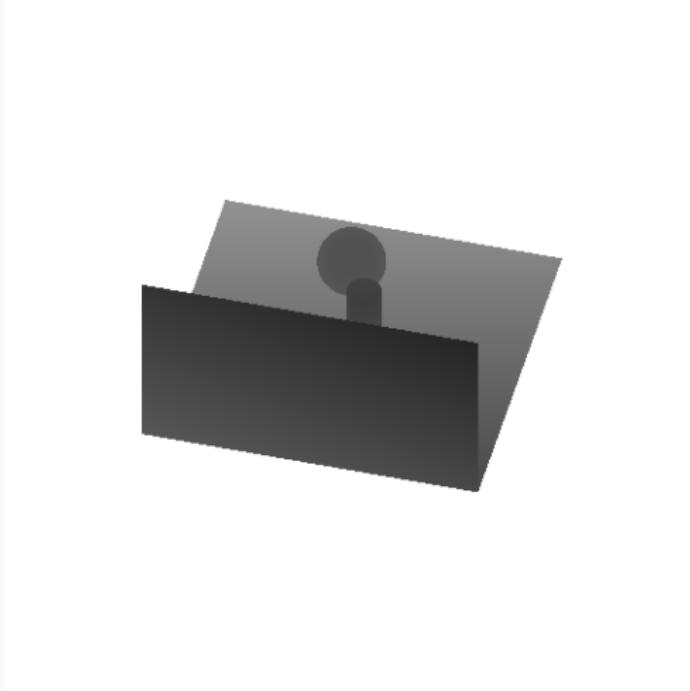


Figure 14 : Exemple de texture de shadow

Application de la shadow map

Application

- Transformer chaque vertex avec la matrice view proj de la shadow (light space) dans le vertex shader
- Dans le pixel shader, regarder si la profondeur en light space du pixel est supérieure à celle de la shadow
 - si oui : alors 0 (ou application de la lumière avec un multiplicateur < 1)
 - si non : application de la lumière

Résultat

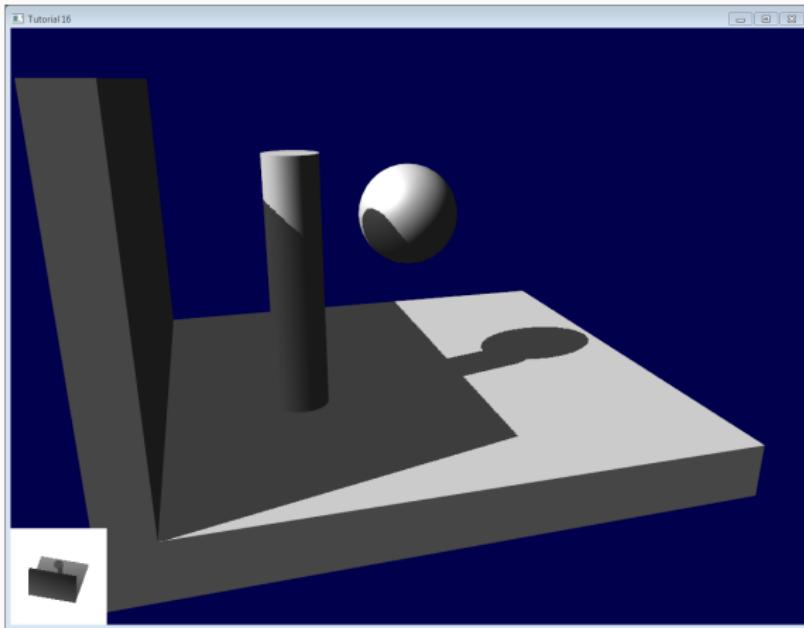


Figure 15 : Résultat du shadow mapping (venant de <http://www.opengl-tutorial.org/fr/intermediate-tutorials/tutorial-16-shadow-mapping/>)

Problème des shadow maps

Problèmes

- shadow acne : problème venant de l'(im)précision de la texture de profondeur utilisée pour faire le shadow mapping
- peter panning : problème venant de la correction du shadow acne

Résolutions

Bien paramétré le depth bias pour éviter le shadow acnée et le peter panning

- depth bias permet de modifier la profondeur.
- Ici on va faire reculer la profondeur pour éviter le shadow acnée.
- Peut être appliqué soit dans la passe de shadow elle-même soit dans la passe d'application de la shadow map.

Shadow acnée

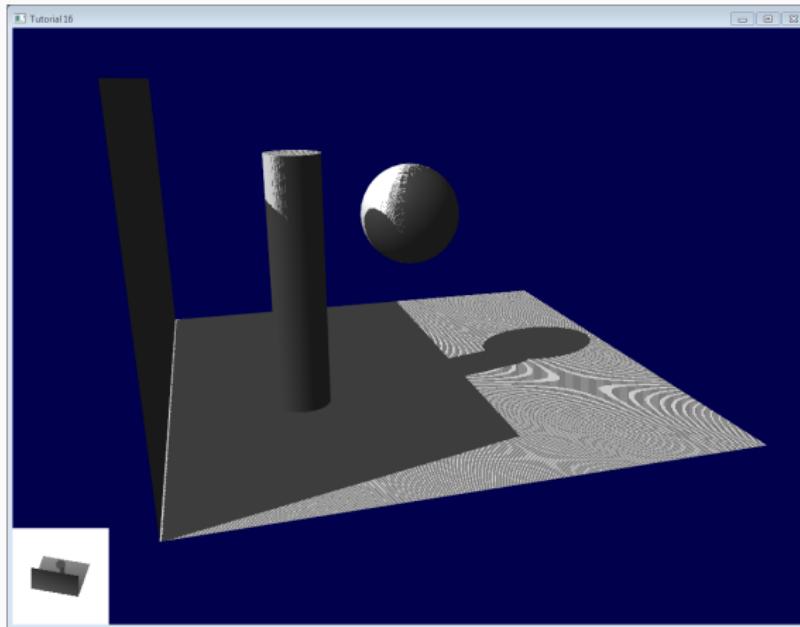


Figure 16 : Exemple de shadow acnée

Peter panning

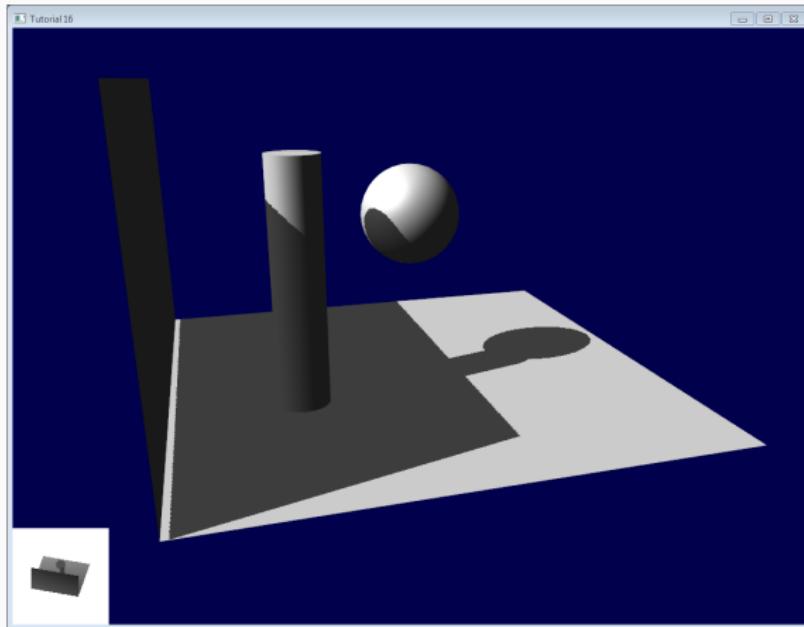


Figure 17 : Exemple de Peter Panning

Convertir de HDR à SDR : le tonemapping

FrameGraph du deferred pass avec distinction

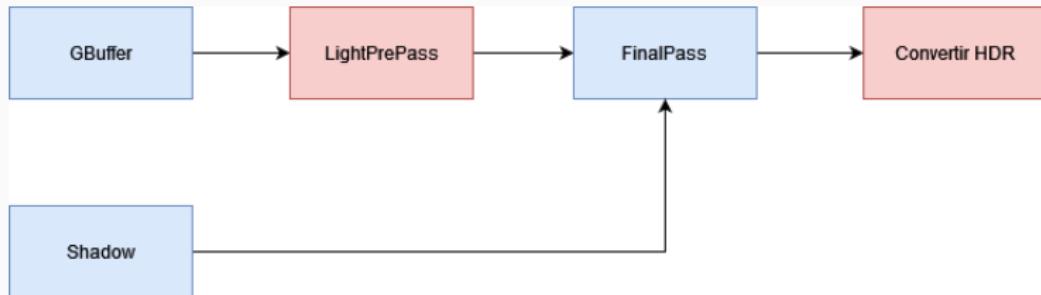


Figure 18 : Exemple de FrameGraph LightPrePass avec la distinction géométrie et screen pass (bleu : géométrie, rouge : screen pass)

Le Tonemapping

- Screen pass
- S'agit d'un post effect qui va permettre de simuler la conversion des valeurs du capteur de la caméra en une image SDR.
- Existe 2 grandes familles d'opérateurs
 - Local tonemapping
 - Applique différents paramètres en fonction des caractéristiques de l'image
 - Global tonemapping
 - Applique les mêmes paramètres à chaque pixel.
 - Encore très utilisé dans le temps réel

Caméra physiques : comment ça marche

Caractéristiques d'une caméra

- Une lentille
- Un capteur
- Un filtre numérique pour faire la conversion capteur-image

Paramètres de capture de l'image

- le shutter speed
- l'aperture
- l'ISO

Ces différents paramètres donnent l'exposition.

Exemple d'impact de l'exposition sur une image

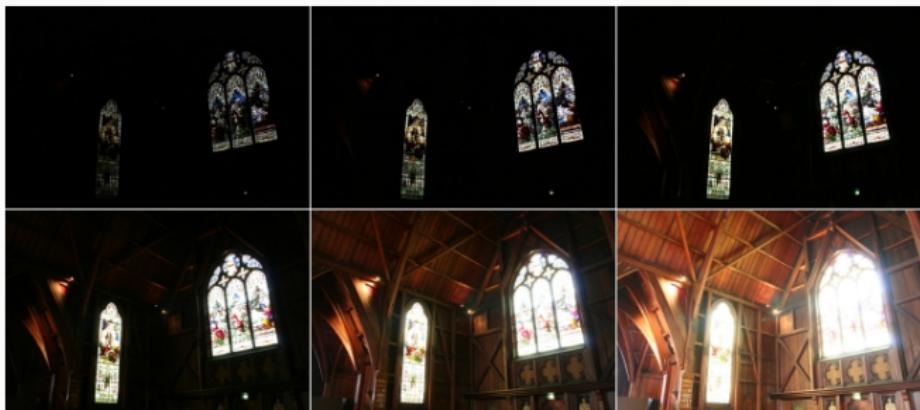


Figure 19 : Exemple d'application de différentes expositions (Par Deanpemberton commonswiki — Travail personnel, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=613689>)

L'opérateur de tonemapping

définition

- Réalise l'opération de conversion HDR vers SDR
- Peut prendre plusieurs paramètres dont l'exposition
- Existe différents opérateurs
 - Reinhard
 - Uncharted 2 tonemapping
 - Hable filmic
 - ACES

Exemple de tonemapping

```
float4 vColor = GetHDRColorFromHDRTexure();
vColor *= exposition;

float4 vSDRColor = ApplyTonemappingOperator(vColor);
return vSDRColor;
```

Pseudo code d'application du tonemapping

Exemple d'application du tonemapping

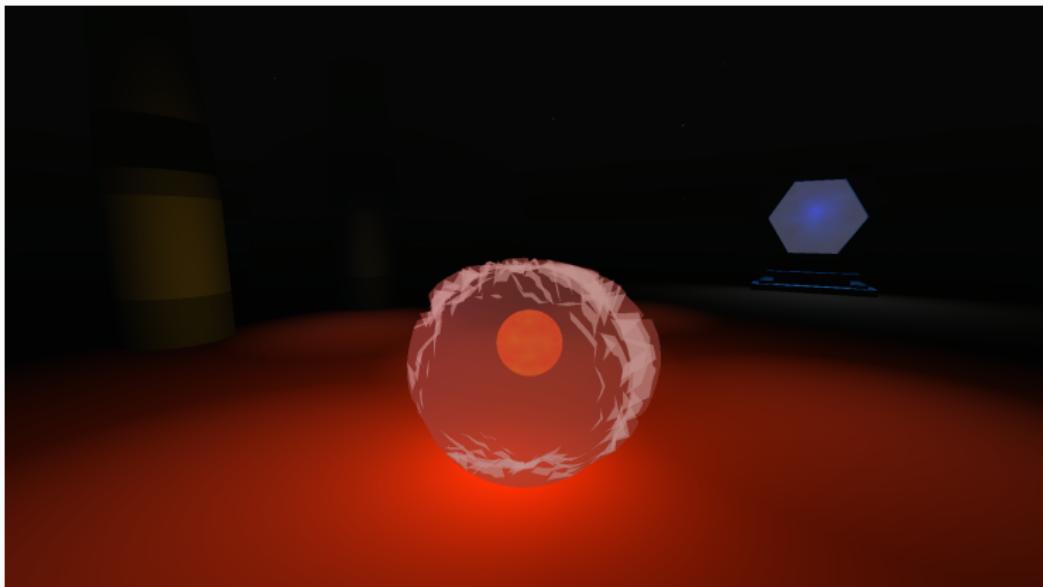


Figure 20 : Entrée du tonemapping

Exemple d'application du tonemapping

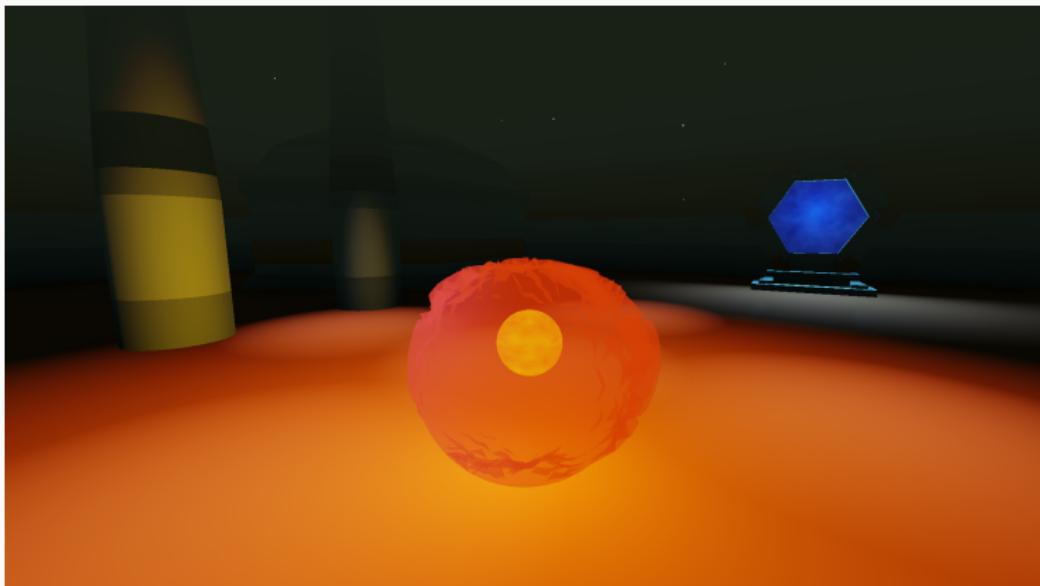


Figure 21 : Résultat du tonemapping

Exemple d'opérateur de tonemapping : Reinhard

Définition

- Opérateur simple de tonemapping

$$color_{sdr} = \frac{color_{hdr}}{color_{hdr} + 1}$$

La fonction de l'opérateur de tonemapping de Reinhard.

Avantage/Désavantage du tonemapping Reinhard

Avantage

- Facile à implémenter
- Léger

Désavantage

- Perte de contraste sur l'image
- Teinte grisâtre
- Très peu paramétrable (doit implémenter une variante pour vraiment le paramétrier)

Exemple d'image avec Reinhard appliqué

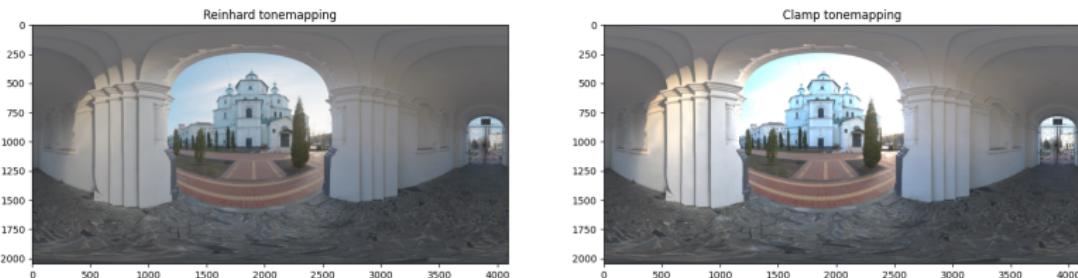


Figure 22 : Application à gauche du tonemapping Reinhard sur une image HDRI et à droite du tonemapping clamp de la même image

Conclusion

FrameGraph du cours

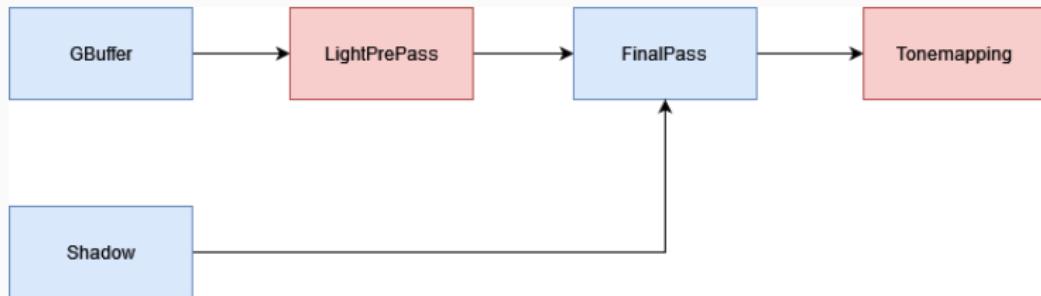


Figure 23 : FrameGraph du cours

Pour aller plus loin

- Les post effects
 - Anti-aliasing
 - Vignetting
 - Lens distortion
 - Depth of field
- L'ambiant occlusion et le SSAO
- Les calculs de lumières du PBR
- ...

Pour aller plus loin

Références du rendu

- <https://advances.realtimerendering.com/> ACM Siggraph (advances real-time rendering)
- <https://www.gdcvault.com/> GDC (plus orientée jeux)
- <https://www.ea.com/frostbite/news> Frostbite News (exclusivement sur le rendu de frostbite)
- <https://jcgt.org/> : Journal of Computer Graphics Techniques

Outil de debug de rendu

- Renderdoc
- Nsight Graphics
- Intel Graphics Debugger