Introduction to Computer Graphics

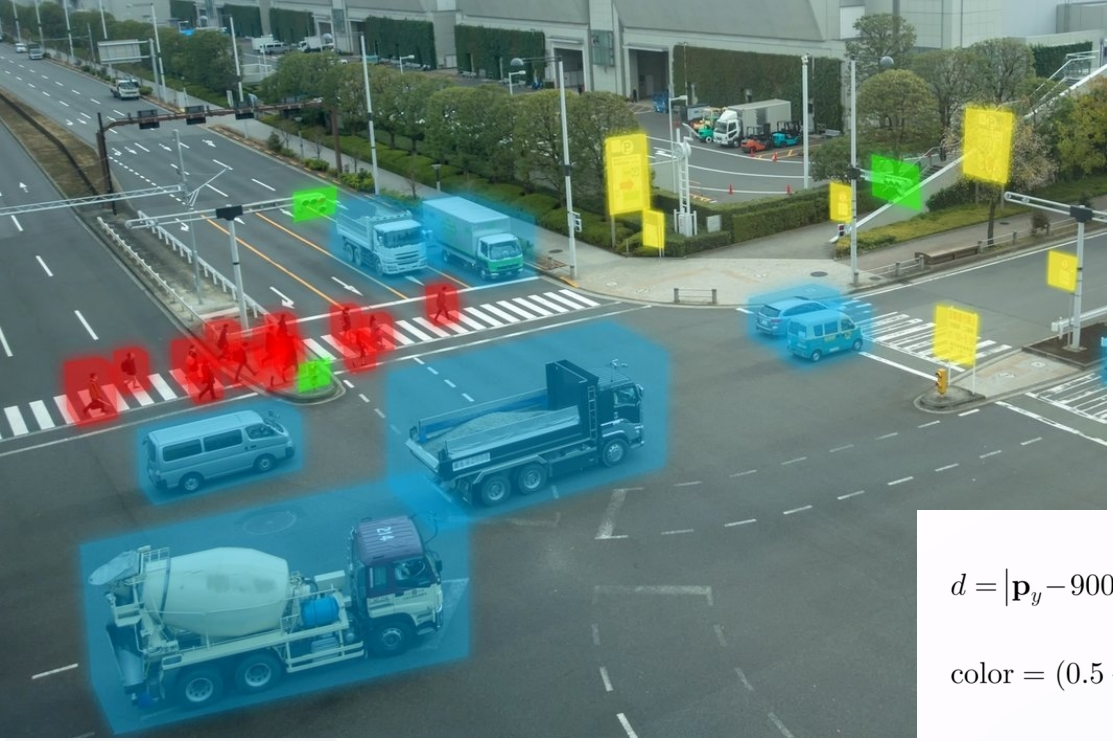
pour TSE - FISE 2 - INFO5 – Dévelopement Multimédia

Raphaël Chevasson

26 Janvier 2024

any question ?

→ raphael.chevasson@telecom-st-etienne.fr



Computer Vision :

Pixels d’une image → Modèle du monde

Graphics Programming :

Modèle du monde → Pixels d’une image

$$d = \left|\mathbf{p}_y - 900\right| - 40 + 400\,N\left(\frac{\mathbf{p}}{300}\right)$$

$$\text{color} = \left(0.5 + 0.5\,\nabla d \cdot \mathbf{s}\right)_+$$

$$\text{color} \to \lambda \cdot \text{color} + (1-\lambda)\cdot$$

$$\lambda = e^{-0.0005 \cdot t \cdot (1,2,4)^T}$$

Problème :

Combien de fois par secondes votre langage préféré peut-il effectuer un calcul simple ?

Réponse : ~10^9

Combien d'opérations faut-il faire pour calculer l'image d'un jeu vidéo en 4k60fps, si chaque pixel ne demande (que !) 10 opérations ?

Réponse : 4000*2000*60*10 = 5*10^9

Uh Oh...

Problème :

Combien de fois par secondes votre langage préféré peut-il effectuer un calcul simple ?

Réponse : ~10^9

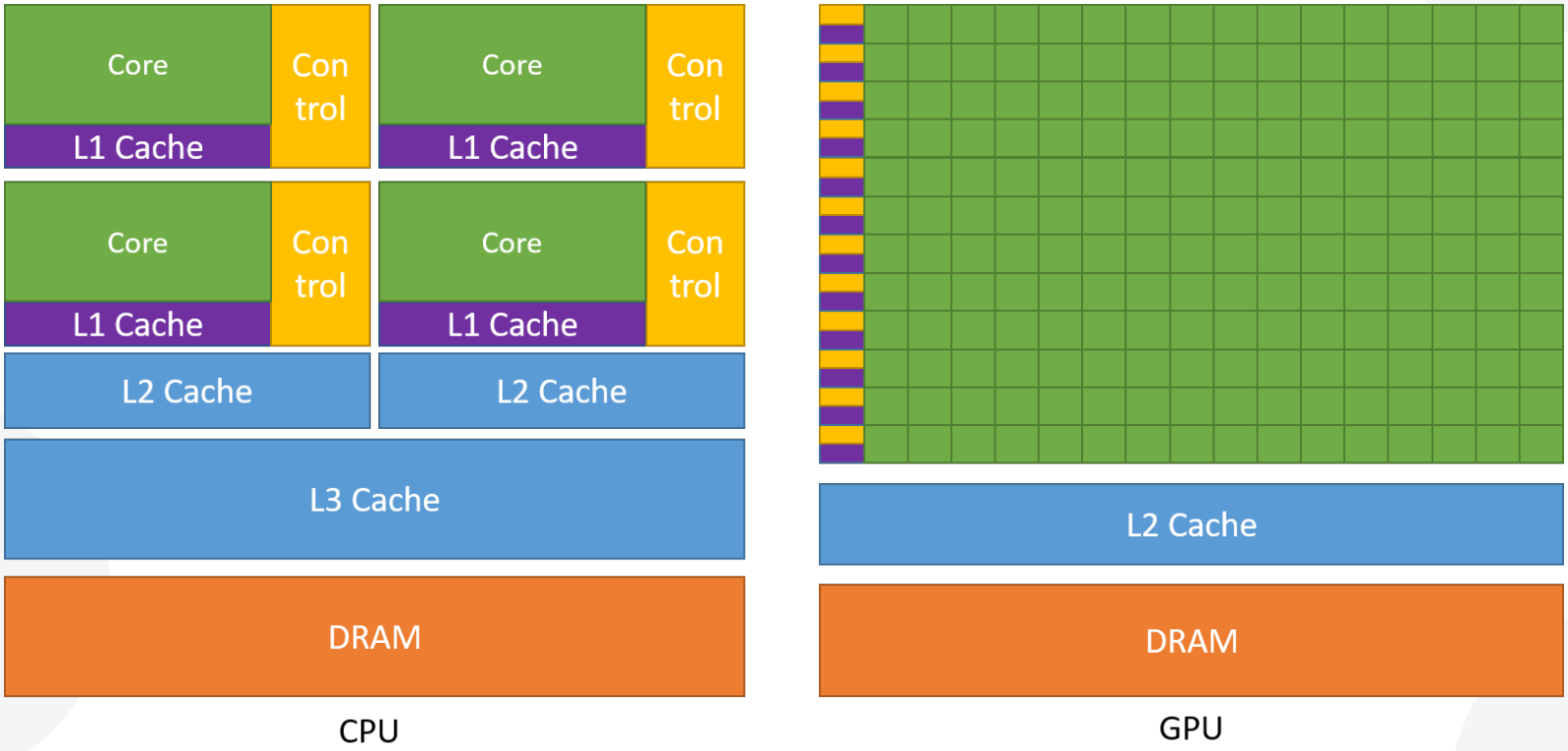
Combien d'opérations faut-il faire pour calculer l'image d'un film de 3h@24fps, si chaque pixel demande énormément opérations ?

Réponse : des années

Uh Oh...

Solution :

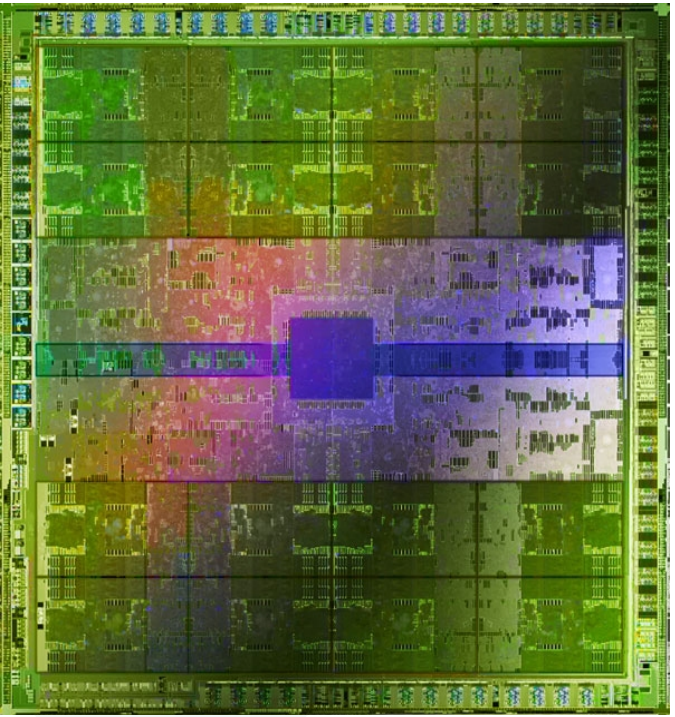
Les GPUs (Graphics Processing Units, = cartes graphiques) ont des **milliers** de coeurs très simples, qui sont parfait pour traiter une masse de tâches **indépendantes** en parallèle.



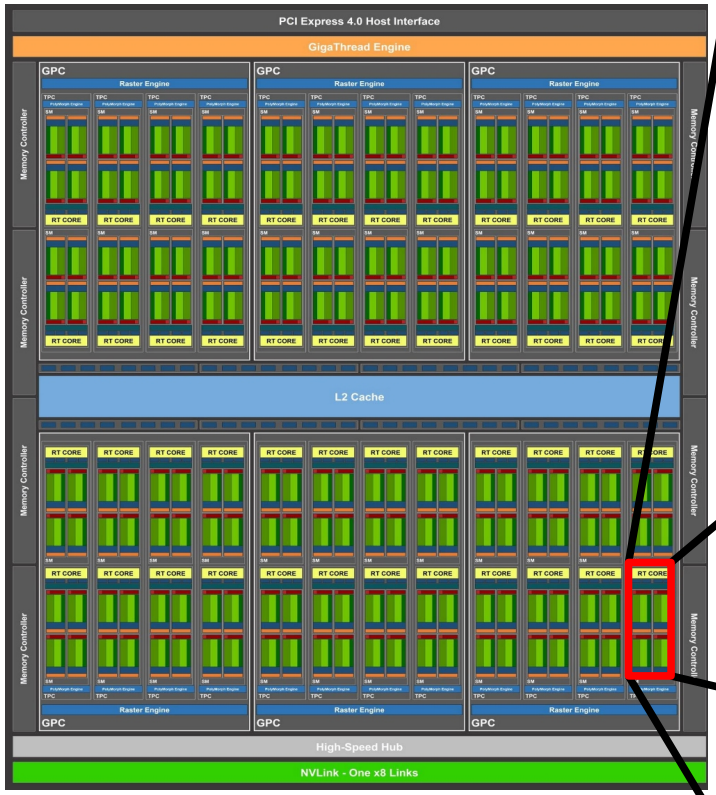
(Threads : 4-128) → (*100 !) (Threads : 700-15000)

slide bonus

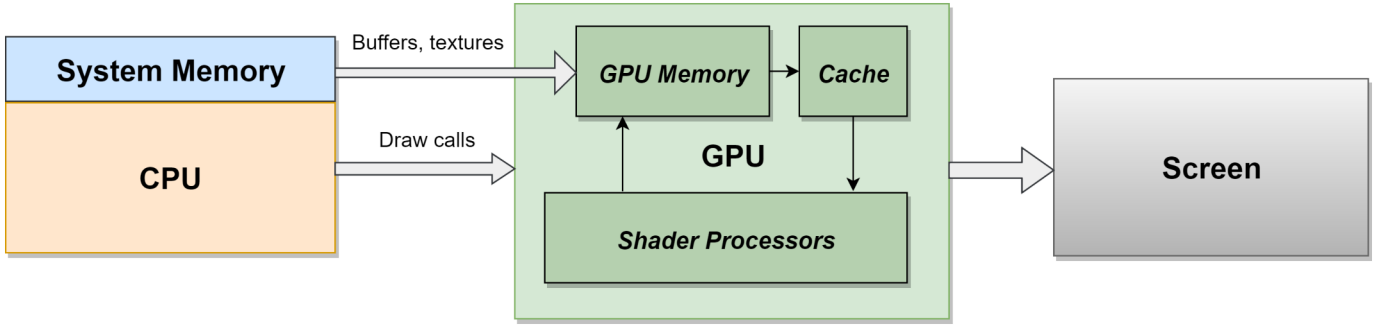
à quoi ressemble un vrai GPU



=



Le **GPU** est un co-processeur : il ne peut pas fonctionner tout seul.
Pour faire fonctionner le GPU, il faut lui **envoyer des ressources** (modèles 3Ds, textures…) et **des instructions** depuis un programme sur le **CPU**.



C’est le rôle des **API graphiques** !

Metal

Microsoft® DirectX®

Vulkan®

OpenGL®

Ce ne sont ni des langages, ni des implémentations, mais des **APIs**, i.e. **des standards**.
Elles sont ensuite **implémentées** dans les **drivers** et bibliothèques des vendeurs (nvidia, amd, intel, apple, qualcom…) que l’on **appelle** depuis notre langage préféré (souvent le **c++**).

slide bonus

Pourquoi OpenGL ? Comparons les API graphiques :

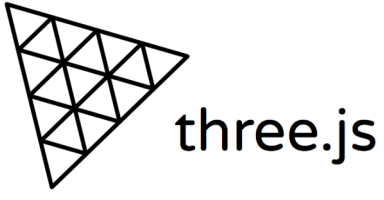
API	Difficulté	Fonctions poussées[1]	Compatpibilité				
			Web	Windows, xbox	Mac, ios	Linux, android, switch	
OpenGL	facile	2/5	WebGL	O	O	O	← Nous
Microsoft DirectX	difficile	5/5	X	O	X	pas natif	
Vulkan	difficile	5/5	X	O	pas natif	O	
Metal	difficile	5/5	X	X	O	X	
WebGPU	difficile	4/5	O[2]	O	O	O	

[1]Fonctions poussées : raytracing accéléré, calculs non graphiques comme les réseaux de neurones ou les simulations scientifiques, gestion manuelle de la mémoire, etc.

[2]Google Chrome, ou versions bêta des autres navigateurs

Il existe aussi des **API non graphiques** (pour le calcul parallèle : simulations scientifique, réseaux de neurones...)

Et des outils 3D **plus haut niveau** pour ceux qui ne veulent pas directement programmer le GPU

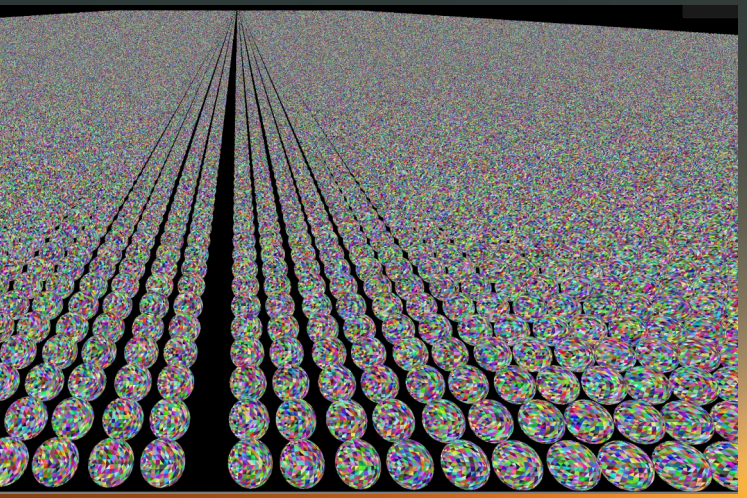


Ce que nous allons apprendre vous aidera à les comprendre !

On peut exécuter du code directemment sur le GPU.
Ces programmes s'appellent des **shaders** (ou des **kernels**).

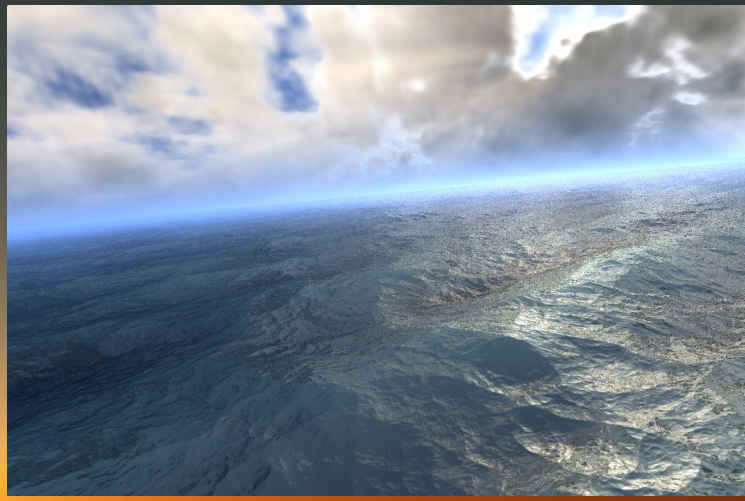
Ce code sera executé **indépendamment** et en parallèle, **sur chaque élément** de donnée fournis (par exemple, pour un pixel shader, le code va s'executer sur chaque pixel).

Il doit être codé dans un **langage GPU** (glsl, hlsl, wgsl...)



Exemple 1 : 100 Millions de triangles affichés à 4000fps

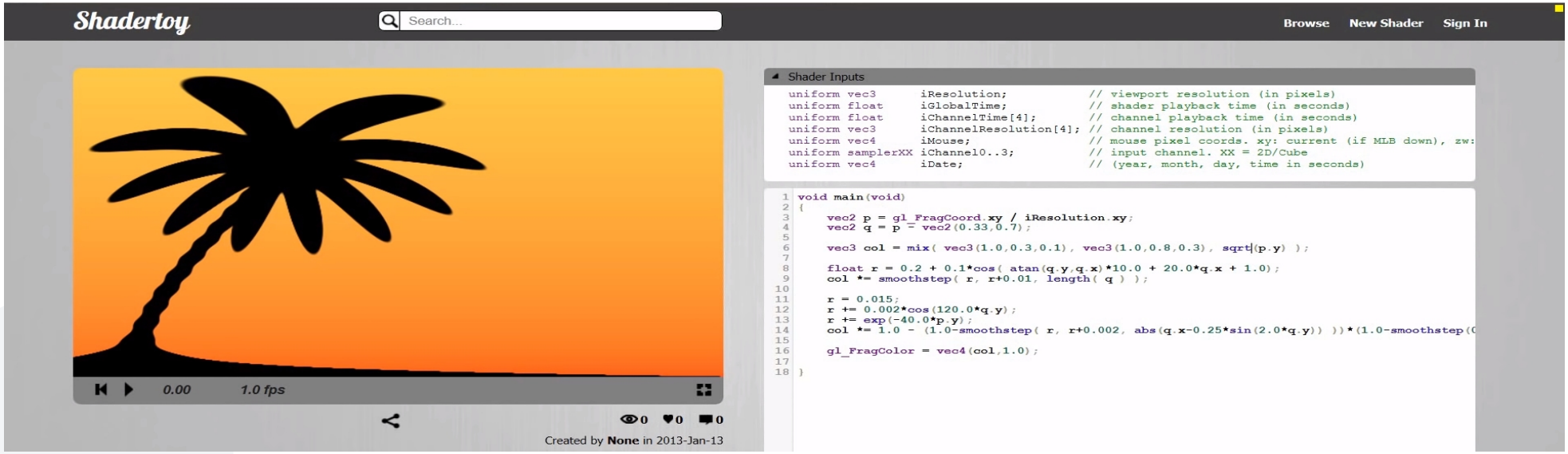
Exemple 2 : Calcul de l'atmosphère d'un coucher de soleil pour chaque pixel



Example 3 : Déformation des vagues et calcul des nuages

-- Iterlude : exemple de beaux shaders en pratique ! **--**

(inspectables et modifiables en live)



→ the book of shaders (lien)
→ shadertoy (lien)
→ vertexshaderart (lien)

Quel métier fait quoi ?

Artist : modèles 3D, textures, animations, environnements, etc

Technical artist : shaders (souvent en visual scripting), outils/plugins à destination des artistes

Graphics programmer : moteur de rendu (avec l'api graphique), shaders trop compliqués/spécifiques

→ souvent (malheureusement) le seul emploi stable et bien payé

Choose Your Class:

Graphics Programmer:
- Maximum Math
- Maximum CompSci
- Some Art (as a treat)
- Invents The Tools
- Specialist

Technical Artist:
- Medium Math
- Medium CompSci
- Medium Art
- Makes Tools Accessible
- Jack Of All Trades

General Artist:
- Minimal Math
- No CompSci
- Maximum Art
- Uses The Tools
- Specialist

Voici comment s’organise un code GPU :

Resources

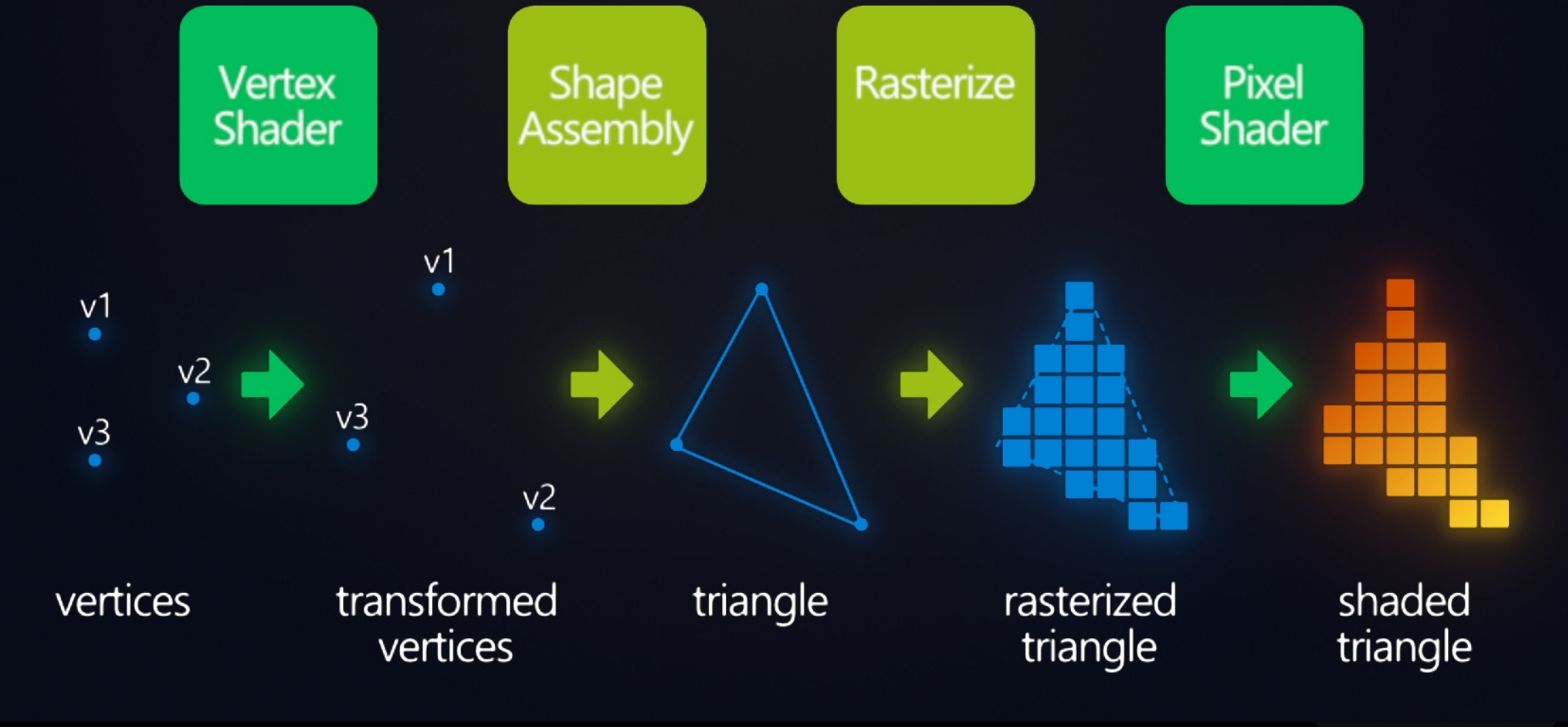
Graphics Pipeline

Render Target

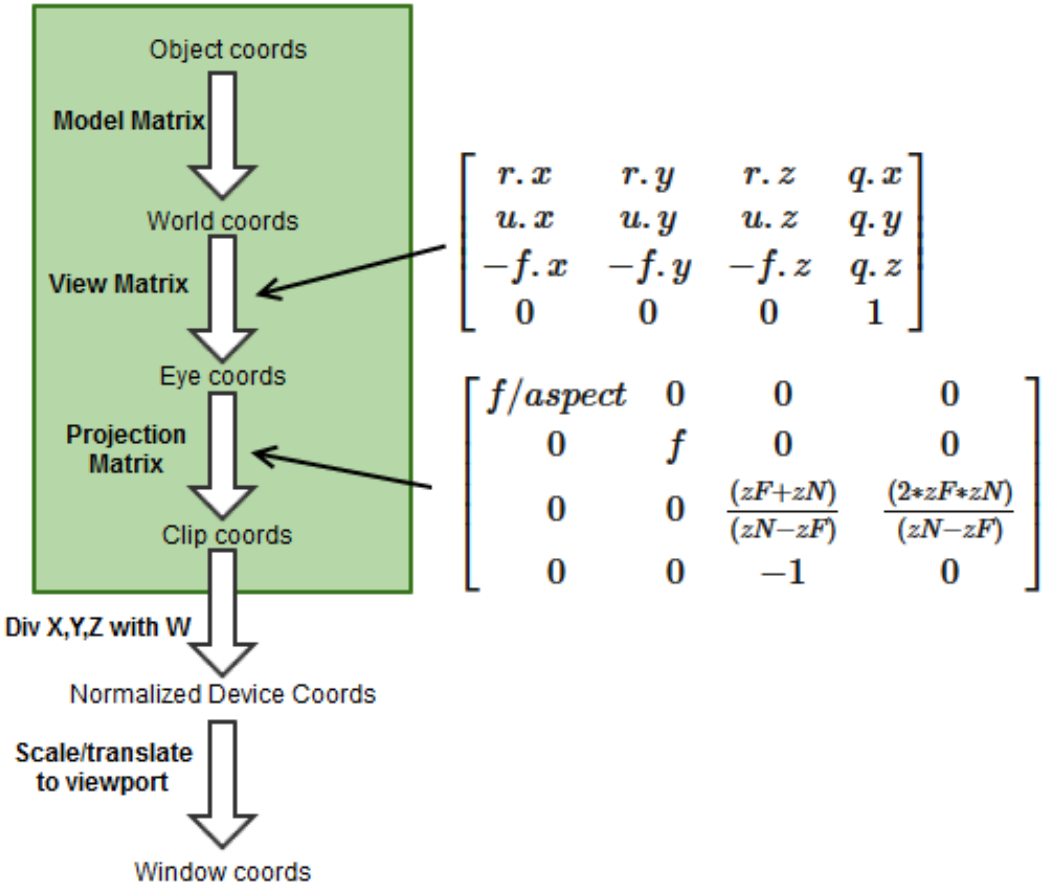
(image de sortie)

Cette pipeline applique des shaders à chaque élément des objets à afficher :

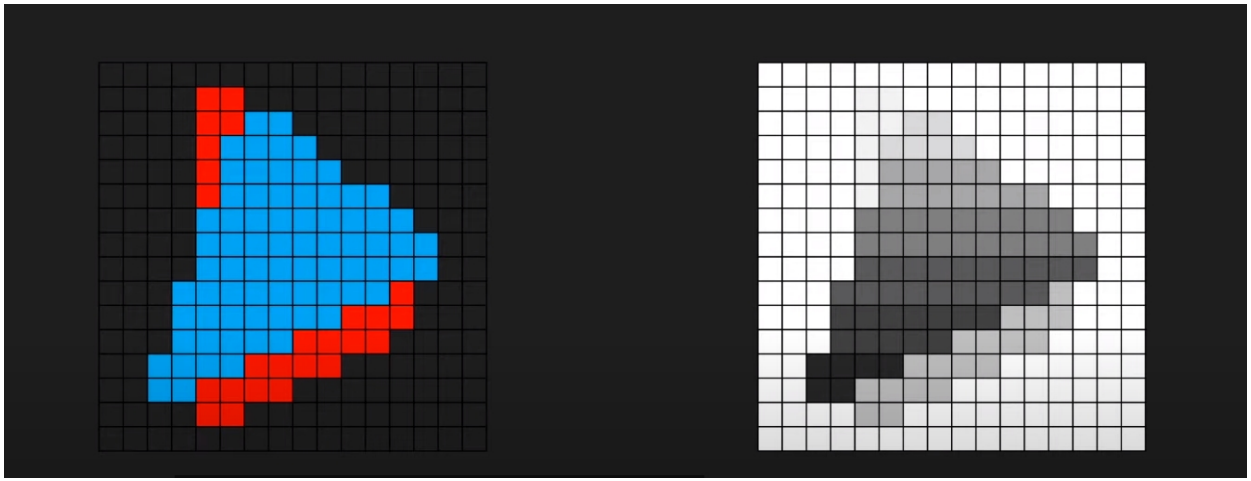
Programmable
Fixé



Étape 1 : placer les objets sur l'écran
(paramètre : le vertex shader)



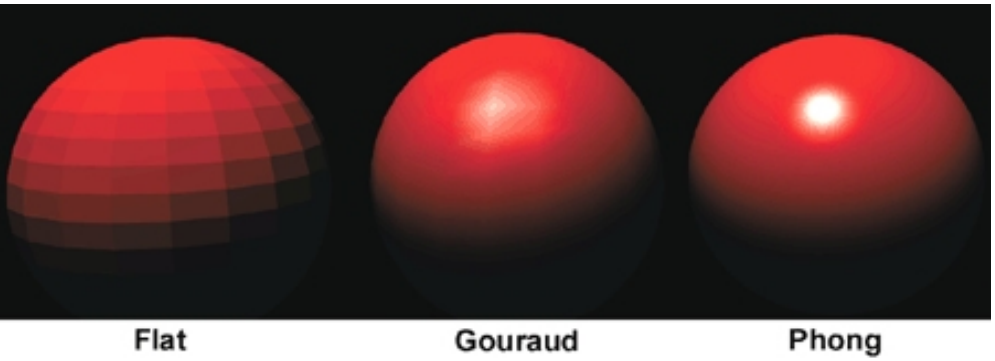
Étape 2 : tracer les formes
(paramètre : le type de primitive et d'antialiasing)

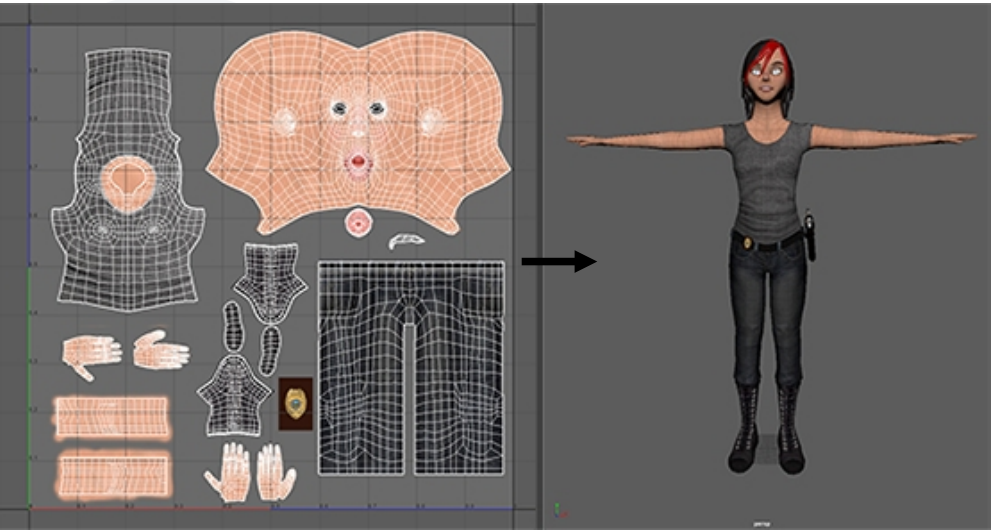


C'est la **rasterization** : prendre des **primitives** (triangles, lignes, lettres…) et la découper en **fragments** prêt à être colorés

NB : il existe d'autres méthodes alternatives :
Le raytracing, le raymarching, le splatting, la vieille méthode 2,5D de doom…

Étape 3 : colorer les fragments
(paramètre : le fragment/pixel shader)

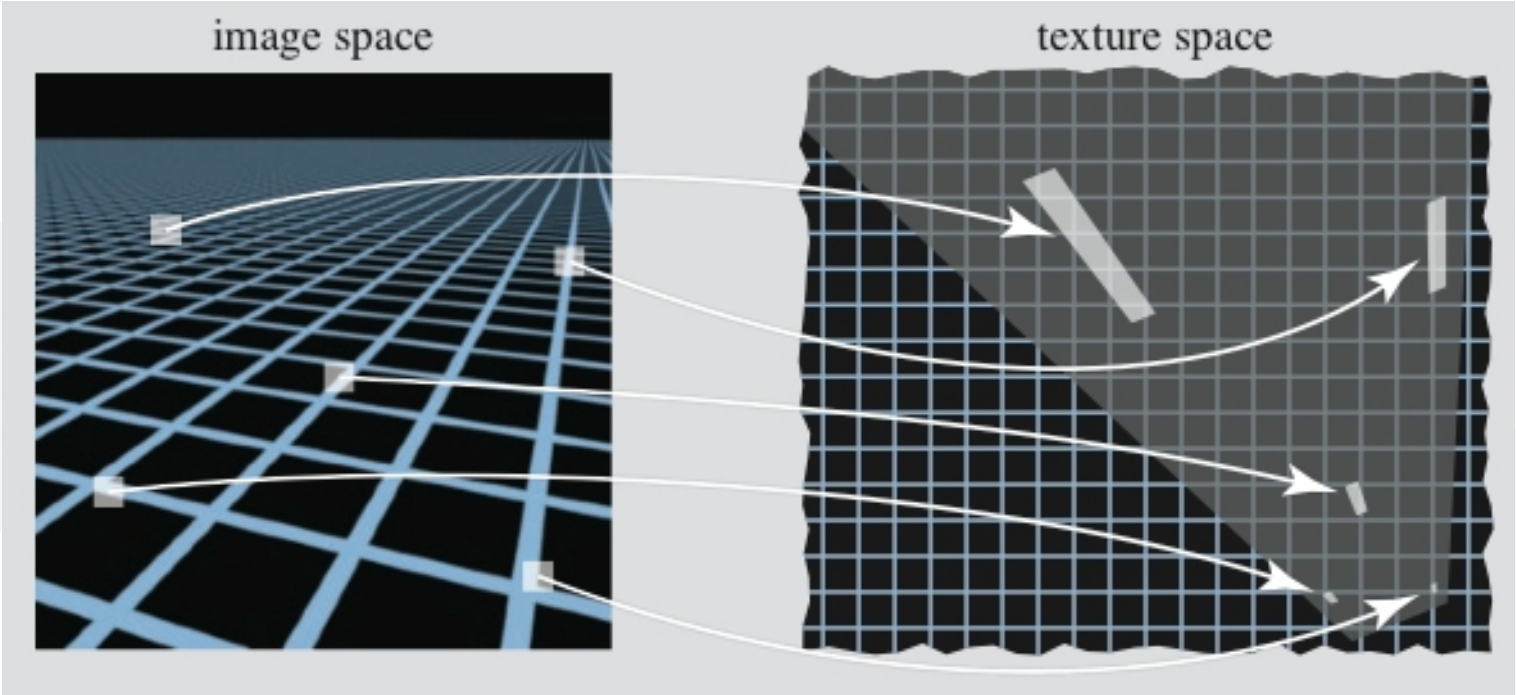




slide bonus

Qu’est-ce qu’un texel (vs. un pixel) ?

- Un **pixel** découle d’une coordonnée **image** (i,j), par exemple de l’écran.
- Un **texel** découle d’une coordonnée **texture** (u,v) sur laquelle tombe le fragment.

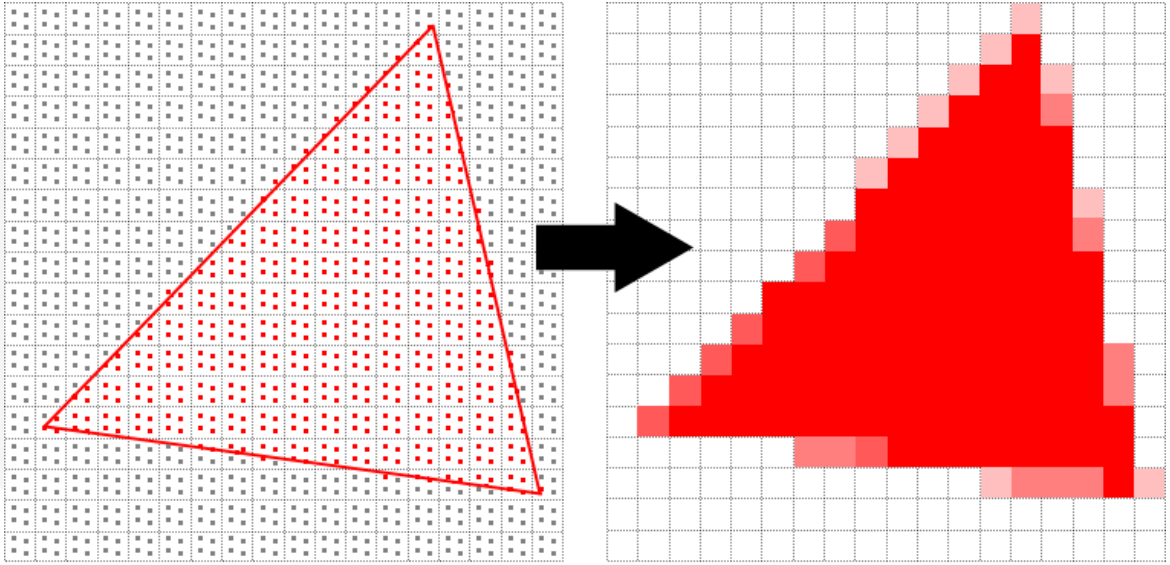


slide bonus

Qu'est-ce qu'un fragment (vs. un pixel) ?

- Un **fragment** contient toutes les **information** nécessaires pour le **futur calcul de la couleur** : position (i,j) sur l'écran, position (u,v) dans les textures de l'objet, etc.

  Ces informations sont issues de la rasterization d'une **primitive** (triangle, ligne, point…). Par exemple pour un triangle, on pondère les infos des **3 sommets**.



Using 4 samples per pixel (MSAAx4)

On peut aussi générer plusieurs fragments par pixels, par exemple pour l'antialiasing (MSAA) !

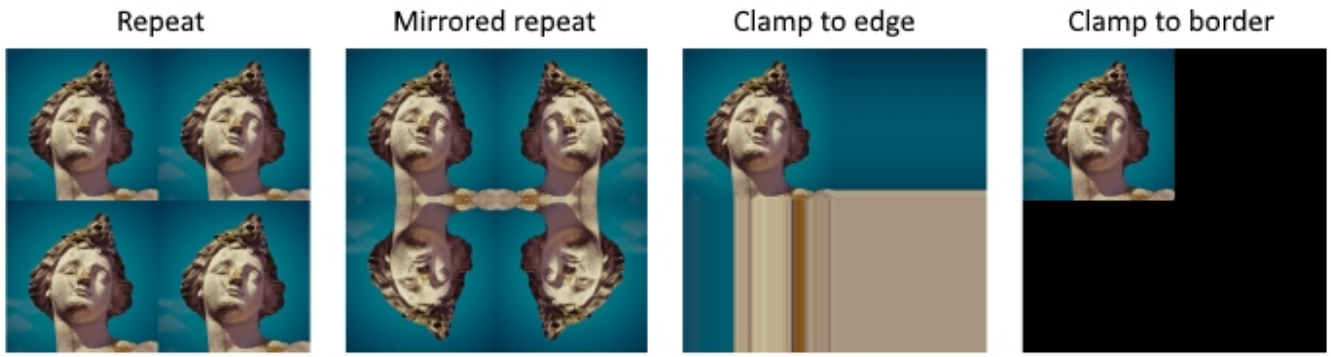
slide bonus

Qu’est-ce qu’une texture (vs. une image) ?

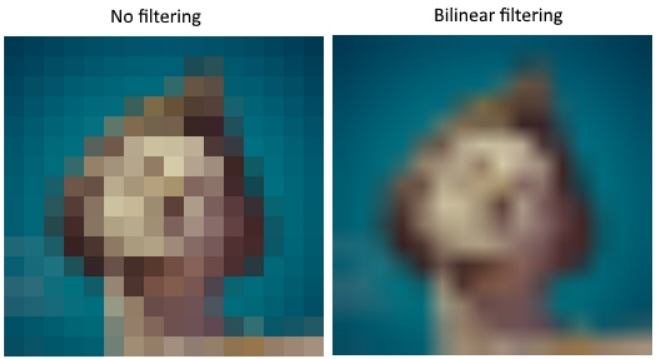
- une **texture** est une **image sur GPU +** des informations d’**échantillonage** :

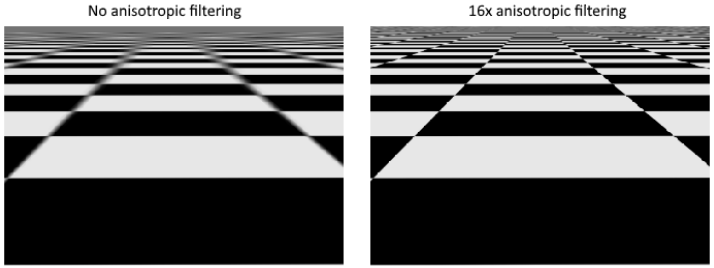
Que faire si…

…on tombe en dehors de l’image ?



…on tombe entre deux pixels (filtering) ?





-- Interlude final --

observons comment, en pratique, un vrai jeu construit une image !

→ GTA V - Graphics Study (lien)

Note : On peut réaliser ce genre d'étude sois-même à l'aide d'un débogueur GPU comme RenderDoc.

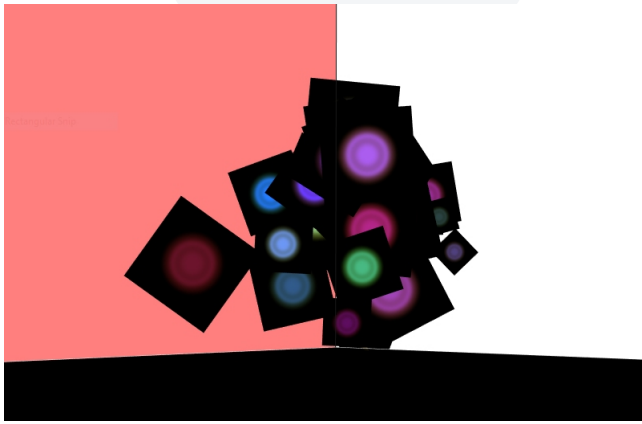
Resources

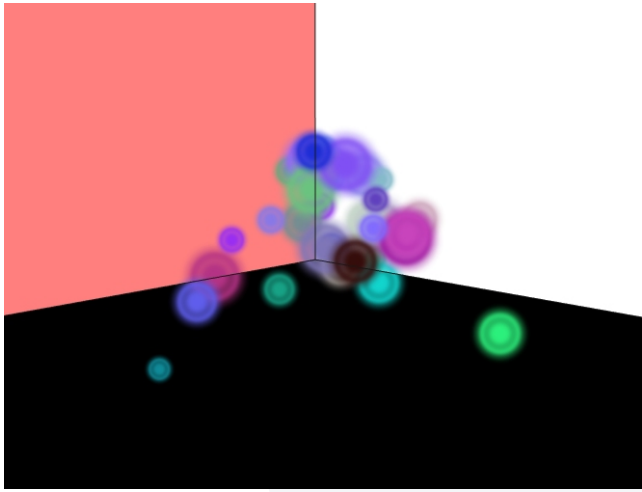
Voulez-vous que je mette en ligne une page avec toutes les ressources pour aller plus loin ?

Questions

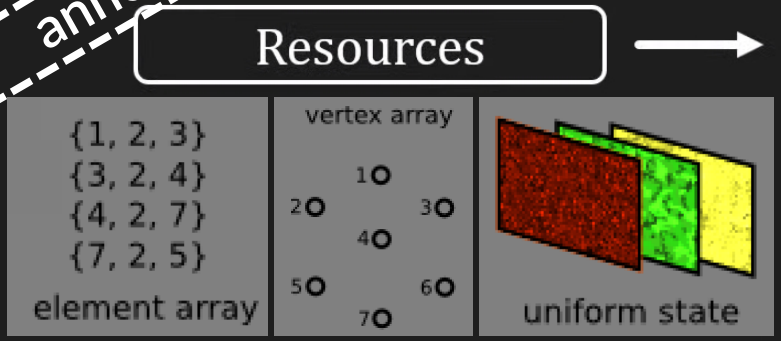
annexes

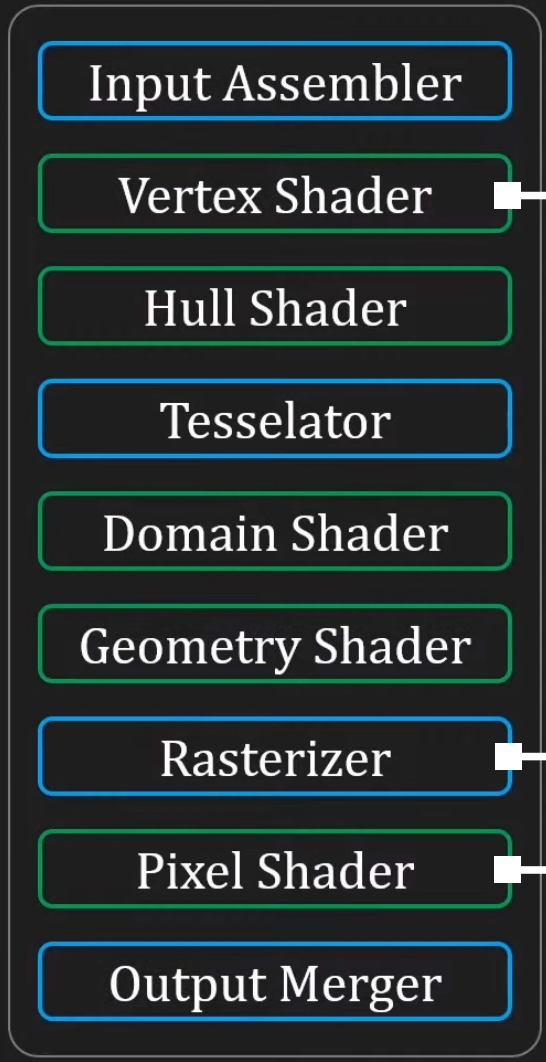
Étape 4 : fusionner avec l'image de sortie
(paramètre : blend mode)

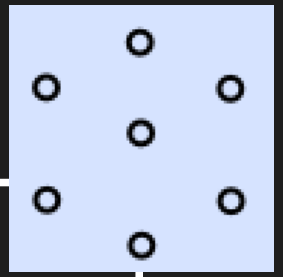


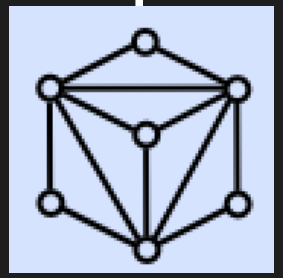


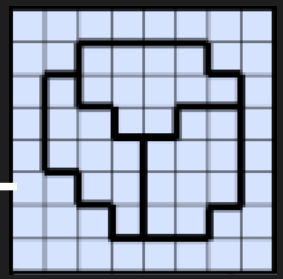
annexes

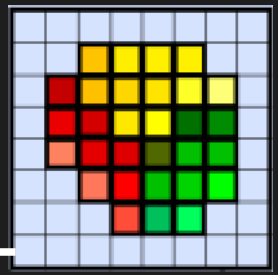




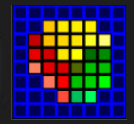








Render Target



La documentation des pipelines graphique mentionne de nombreux autres stages ; ils ne sont pas tous importants aujourd'hui.

- Programmable
- Fixé

annexes

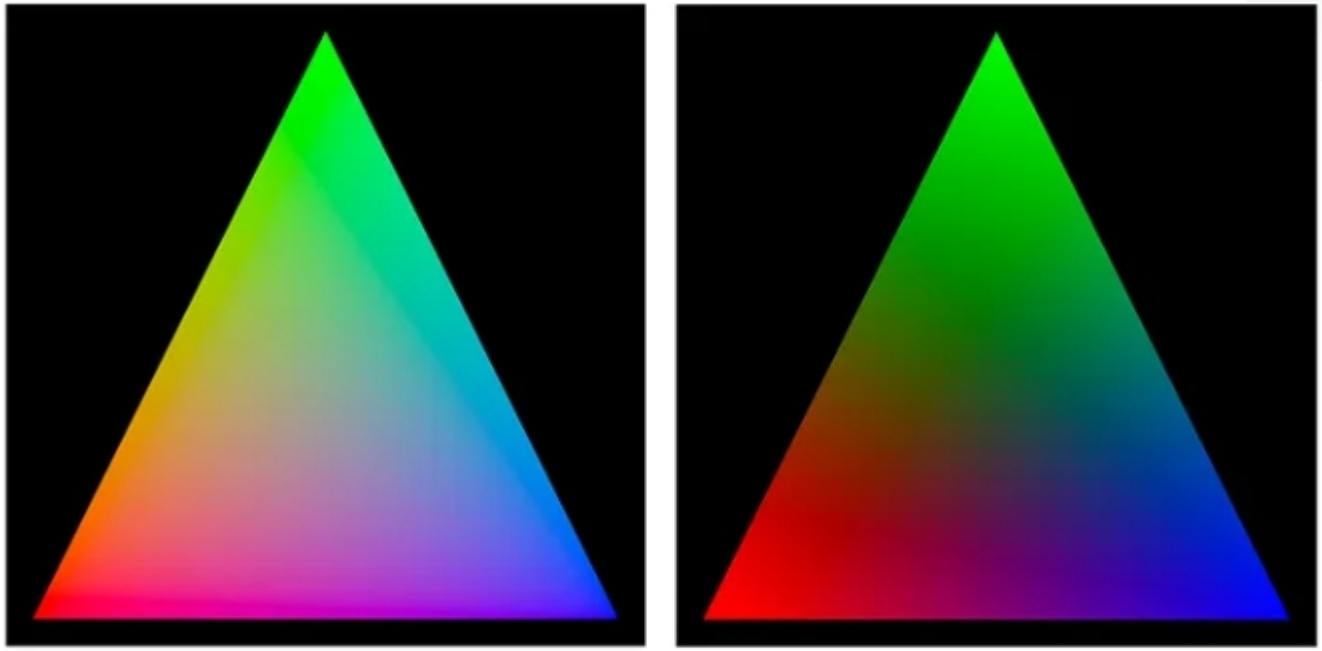
Comment calculer la lumière dans un jeu vidéo ?

→ comparaison graphique (lien)

- Forward rendering : pour chaque objet, pour chaque lumière,
  image += couleur(objet, lumière)
  \+ : rapide s'il n'y a que quelques lumières
  exemples : jeux mobiles et VR
- Defered rendering : pour chaque objet, écrire toutes les infos dans une image intermédiaire. Puis, pour chaque lumière,
  image += couleur(infos intermédiaires)
  \+ : permet de nombreuses lumières et des effets graphiques complexes avec les infos intermédiaires
  \- : cout mémoire, gère mal la transparance et le MSAA
  exemples : jeux console ou PC next gen

annexes

Attention à votre espace de couleur !



The colors at left were gamma encoded after interpolation; those on the right were not.

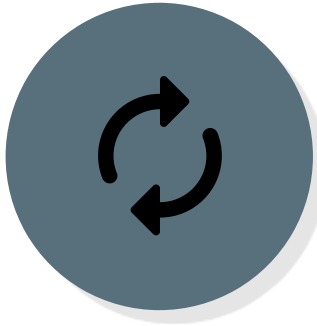
→ article sur la correction gamma avec OpenGL (lien)

--- Template ---

Lorem & Ipsum

Lorem

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Dolor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Lorem & Ipsum

- Lorem
- Ipsum
- Dolor
- Sit
- Amet

Illustrations by Pixeltrue on icons8

Lorem

Ipsum Dolor

> Id volutpat lacus laoreet non curabitur gravida arcu. Felis bibendum ut tristique et egestas quis ipsum suspendisse. Quam viverra orci sagittis eu. Risus commodo viverra maecenas accumsan lacus. Sit amet est placerat in egestas. Semper auctor neque vitae tempus quam pellentesque

Lorem

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Dolor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Sit

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Consectetur

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

01 Lorem

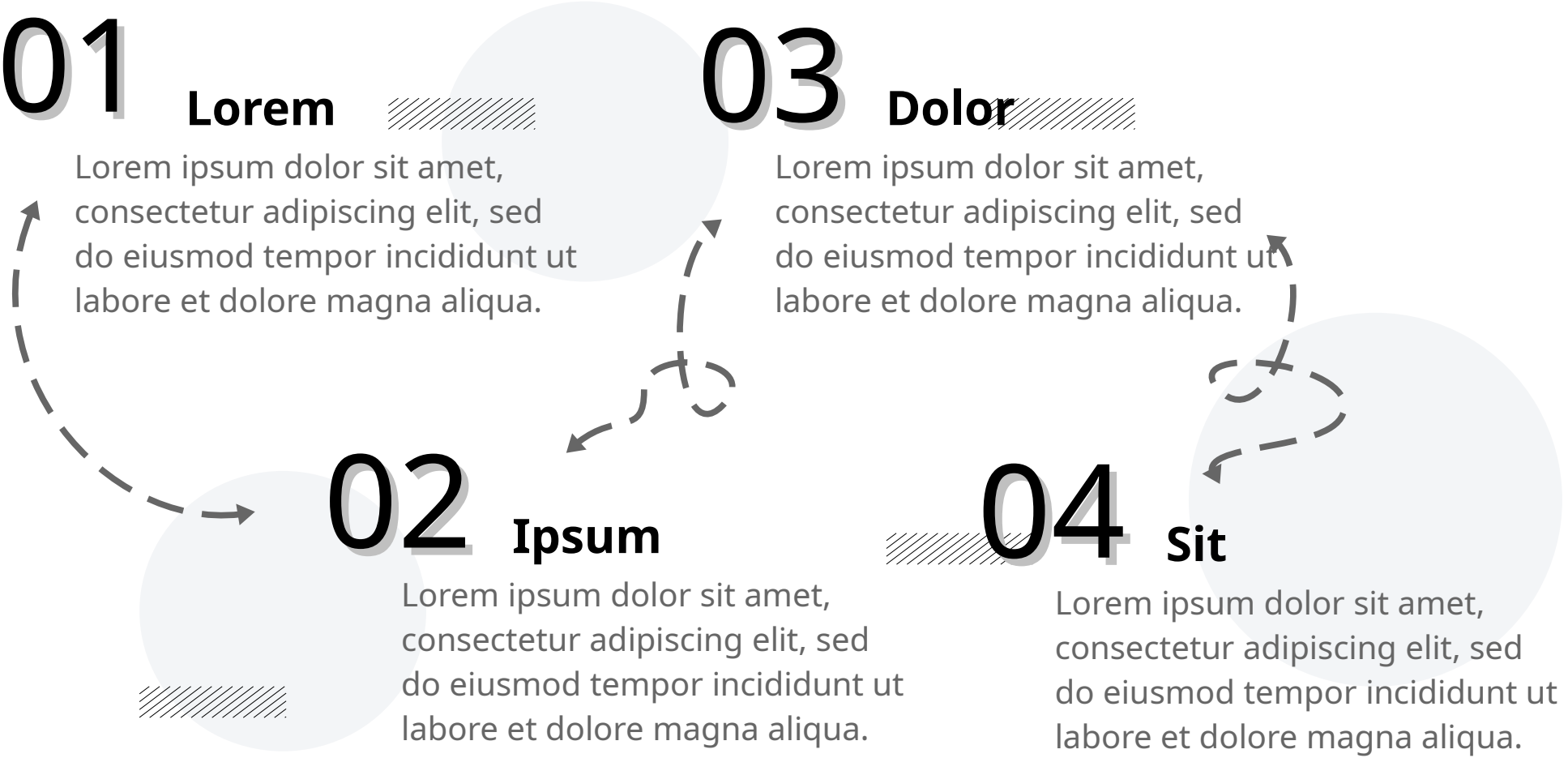
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

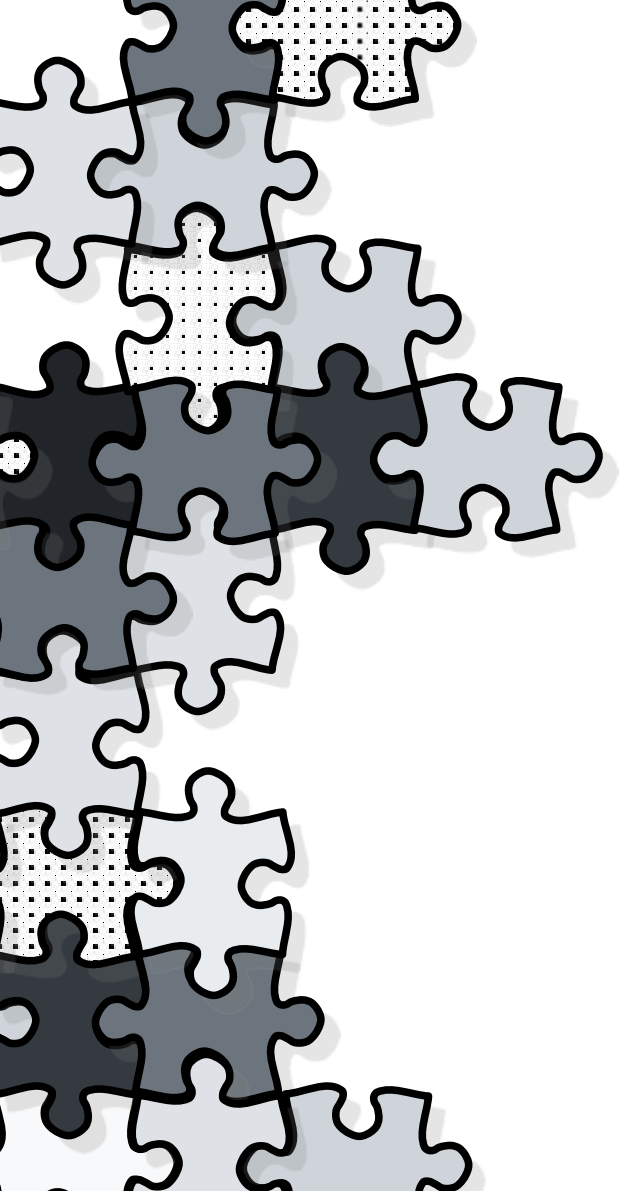
02 Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

03 Dolor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.


01
Lorem
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
02
Ipsum
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
03
Dolor
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
04
Sit
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



“

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

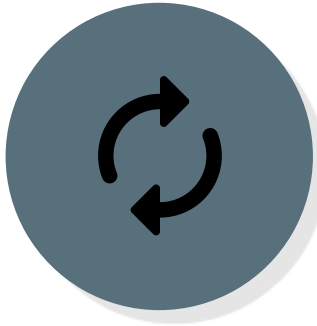
”

- Loremus

Lorem & Ipsum

Lorem

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Dolor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Lorem & Ipsum

Id volutpat lacus laoreet non curabitur gravida arcu. Felis bibendum ut tristique et egestas quis ipsum suspendisse. Quam viverra orci sagittis eu. Risus commodo viverra maecenas accumsan lacus. Sit amet est placerat in egestas. Semper auctor neque vitae tempus quam pellentesque

Id volutpat lacus laoreet non curabitur gravida arcu. Felis bibendum ut tristique et egestas quis ipsum suspendisse. Quam viverra orci sagittis eu. Risus commodo viverra

Photo by Dave Hoefler on Unsplash