Pipeline graphique

Djihane BABAHENINI

djihane.babahenini@univ-biskra.dz

20/03/2019

1.0

Légende

 Entrée du glossaire

 Abréviation

 Référence Bibliographique

 Référence générale

Table des matières

Objectifs

Ce chapitre vise à doter les étudiants des connaissances et compétences requises pour les rendre capable de :

- *Préciser* l'intérêt d'utilisation des GPUs pour la synthèse d'image.
- *Expliquer* le processus de visualisation d'une scène 3D.
- *Décomposer* le pipeline graphique en deux classes.
- *Identifier* les étapes fixes et programmables dans un pipeline graphique.

Pré-requis :

Pour pouvoir suivre et tirer le maximum de ce cours, l'étudiant doit connaître:

- Les algorithmes et les structures de données.
- L'architecture des ordinateurs.
- L'architecture parallèle.

Introduction

La synthèse d'image a connu une évolution importante ces derniers années et ses applications sont très variées, ce qui nécessite un large calcul des images de rendu. En effet, le CPU*p.36 p.36 p.36* de la machine permet de générer des images dans un temps de calcul et un coût mémoire très élevé. Depuis une dizaine d'années, la synthèse d'image prend une direction vers l'utilisation des cartes graphiques pour accélérer le temps de calcul d'une part et produire des images qualifiées réalistes d'autre part.

Pourquoi les GPUs ?

Les problèmes de génération des scènes 3D complexes sollicitent de plus en plus de puissance de calcul et les traitements à opérer sur les données sont de plus en plus exigeants. Les GPUs

sont attractifs de par leur grande puissance de calcul théorique et leur coût modeste.

En 2006, pour la première fois depuis leur invention, en raison des limites de dissipation de réchauffement, les processeurs ont atteint la limite des fréquences de fonctionnement possible. Afin de fournir des puces plus puissantes, les fabricants ont ensuite commencé à développer les processeurs multicoeurs, un chemin qui avait déjà été emprunté par les fabricants de cartes graphiques plus tôt. En 2012, NVIDIA a sorti les processeurs GK110 bénéficiant de 2880 coeurs avec une simple précision et 960 coeurs de double précision, pour une puissance de calcul de 6 Tflops en précision simple et 1.7 Tflops en précision

double.[p.38]

Actuellement, les supercalculateurs sont pourvus généralement avec de millions de coeurs dans les processeurs graphiques dédiés au calcul général.

Le processeur graphique permet des calculs complexes réalisés par la carte graphique. Il peut exécuter jusqu'à 1000 opérations en parallèle.

Les différents séries de cartes graphiques NVIDIA.

Exercice

II

[solution n°1 p.29]

Classez les types des cartes graphiques Nvidia selon leurs année de génération.

RIVA TNT2

GeForce 4 Ti

RIVA TNT

GeForce 2

GeForce FX

GeForce 3

GeForce 256

Visualisation d'une scène 3D

III

Une scène 3D est composée de plusieurs entités : les objets, les source lumineuses et la caméra. Chaque objet est correspond à un ensemble de polygones et une matière (couleur) qui définit son aspect. Les sources lumineuses permettent d'éclairer la scène 3D et sont utilisées pour calculer une image de rendu. Le rôle de la caméra virtuelle est de visualiser tous les objets de la scène sur un écran 2D.

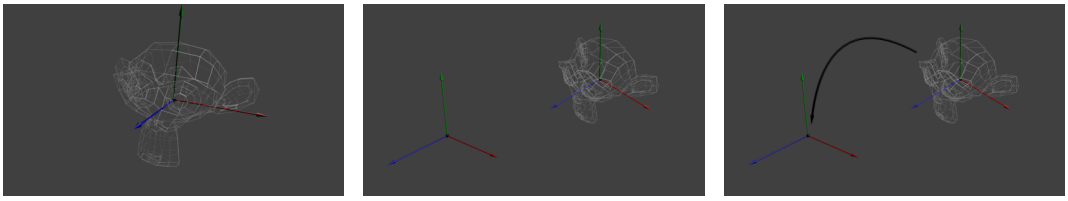
1. Principe

Un maillage est un ensemble de faces organisées en polygones qui définit un objet 3D.*p.38* Chaque polygone est constitué de sommets et d'arêtes.

Afin de faciliter la description des nombreux objets composant la scène, plusieurs repère géométriques sont utilisés :
p.38

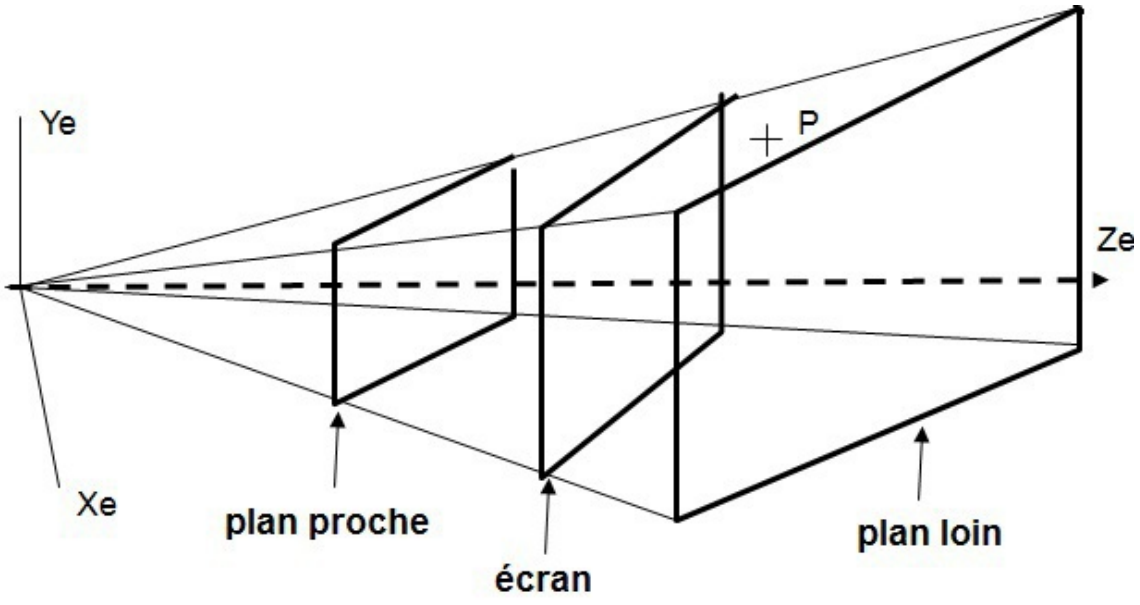
1. Les sommets des polygones d'un objet sont décrit dans un repère local à l'objet.
2. Les objets sont ensuite positionnés dans le repère global de la scène 3D.
3. Les sources lumineuses sont définies par une position (dans le repère global), une couleur, ainsi qu'un comportement : ponctuel, directionnel, spot.
4. La caméra permet de choisir un point de vue ainsi que la portion de la scène à visualiser. Elle est décrit par une position (dans le repère global), une direction d'observation et un angle d'ouverture (FOV).

Transformation d'un objet de repère local au repère global



2. Paramétrisation de la caméra

La visualisation d'une scène 3D consiste à projeter tous les objets sur un plan 2D (écran) nommé le *plan image.* Le modèle de caméra permet de calculer pour un point donné, à quel endroit dans l'image il sera visiblep.38 . Pour déterminer quelle partie se projette sur le plan image, on introduit la notion de *pyramide de vision* de la caméra.

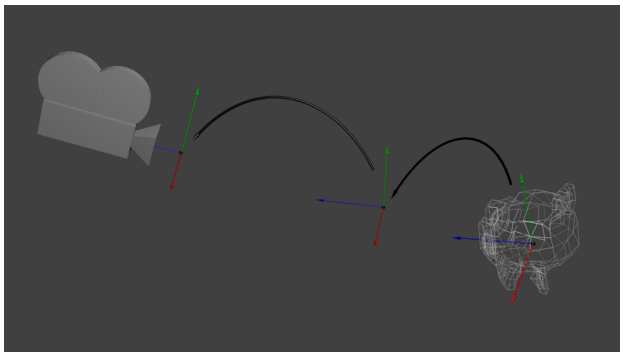


Pyramide de vision

Pour définir une caméra virtuelle, on a besoin de savoir sa position et sa direction.

2.1. Position de la caméra

La position par défaut de la caméra est à l'origine du repère de la scène. Elle est généralement un vecteur dans l'espace global. Pour visualiser un objet à partir d'un autre angle, on peut déplacer soit l'objet, ou bien la caméra, en appliquant les matrices de translation et de rotation.



Passage de l'espace monde à l'espace caméra.

2.2. Direction de la caméra

Le prochain vecteur requis est la direction de la caméra, par exemple. à quelle direction il pointe. Pour le moment, on laissons la caméra pointe vers l'origine de la scène: (0,0,0). Rappelez-vous que si on fait la soustraction de deux vecteurs, on obtient un vecteur qui fait la différence entre ces deux vecteurs. En soustrayant le vecteur de position de la caméra du vecteur d'origine de la scène, on obtient ainsi le vecteur de direction. La caméra pointe vers la direction z négative, on veut que le vecteur de direction pointe vers l'axe z positif de la caméra. Donc il faut inverser l'ordre de soustraction pour obtenir un vecteur pointant vers l'axe z positif de la caméra[p.37] .[p.37] [p.37]

2.3. Analogie avec la photographie

Voici un tableau qui compare entre une caméra réelle et une caméra virtuelle:

Caméra réelle	Caméra virtuelle	Type de transformation
Arranger les éléments de la scène à capturer	Composer une scène virtuelle	Transformation de modélisation
Positionner l'appariel photo	Positionner la caméra virtuelle	Transformation de vision
Régler la focale de l'appariel photo	Configurer une projection	Transformation de projection

Caméra réelle VS caméra virtuelle.

3. Exercice

[solution n°2 p.29]

Répondez par vrai ou faux :

3.1. Exercice

On projette les objets de la scène sur un plan nommé plan objet

- Vrai
- Faux

3.2. Exercice

Le pyramide de vision consiste à déterminer les parties visibles de la caméra

- Vrai
- Faux

3.3. Exercice

Un pyramide de vision est composé de deux plan écran et plan loin

- Vrai
- Faux

3.4. Exercice

Pour visualiser un objet, on doit passer d'un espace caméra à un espace monde

- Vrai
- Faux

3.5. Exercice

La caméra est caractérisée par une position et une direction

- Vrai
- Faux

3.6. Exercice

On projette les objets de la scène sur un plan nommé plan objet

- Vrai
- Faux

Pipeline graphique

IV

1. Concept de pipeline

Le rendu temps réel est un sous domaine de l'informatique graphique pour créer des images en temps interactif. En effet, les algorithme classiques de construction des images (lancer de rayons, tampon de profondeur) sur CPU ne sont pas efficace pour faire un rendu rapide surtout pour des scènes complexes. Depuis une dizaine d'années, le rendu est orienté vers l'exploitation des performances des cartes graphiques afin d'accélérer le temps de calcul. L'enchaînement des étapes exécutées sur le GPUp.36 est appelé *pipeline graphique.*

On appelle pipeline 3D la succession des opérations généralement réalisées par une carte graphique nécessaires au rendu d'un lot de données, le résultat de chaque opération est l'entrée de l'opération suivante.

Exemple : pipeline de construction de voiture

la voiture à la première étape de construction se déplace à la deuxième étape lorsque la voiture de la deuxième étape est prête à passer à la troisième étape, et ainsi de suite. À tout moment, dans le pipeline de montage, plusieurs voitures sont en construction, même si seulement un nombre réduit d'entre elles sort à chaque moment de la chaîne.p.37

Remarque

On parle généralement de *pipeline de rendu* suite aux étapes de rendu qui le constitue. La fonction principale du pipeline graphique de rendu est de générer ou rendre une image bidimensionnelle à partir d'informations géométriques tridimensionnelle.

Il existe deux grandes familles de pipeline graphique, le pipeline classique et le pipeline programmable.

2. Pipeline classique

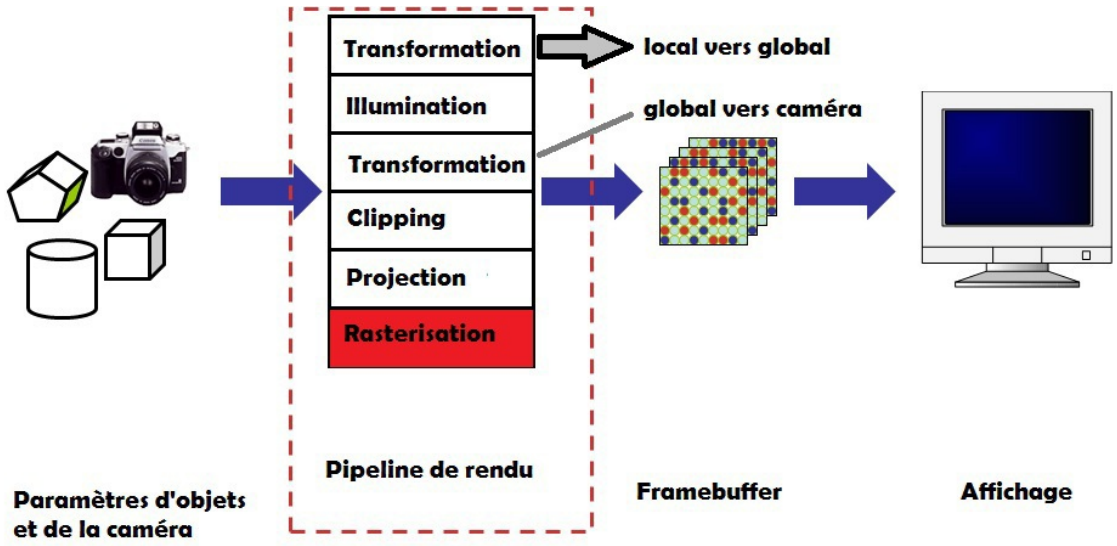
2.1. Définition

Un pipeline classique nommé aussi pipeline fixe est la suite d'étapes conduites à une image bidimensionnelles. D'après son nom, ces étapes ne peuvent pas être modifiées par le programmeur. Dans un pipeline graphique fixe, toutes ces fonctions sont codées (intégrées) dans la carte graphique, et il n'y a aucune intersection entre eux.

2.2. Etapes

Le pipeline de rendu commence son travaille lorsque tous les sommets de la primitive sont connus. L'image finale est construite en appliquant le pipeline à chaque primitive[p.38] . Le pipeline classique est divisé en 5 étapes, soit :

1. transformation modèle-vue.
2. éclairage et teinte des vertex.
3. tranquage (clipping) de vision.
4. projection.
5. rasterisation.



Pipeline de rendu classique

2.2.1. Étape 1 : transformation modèle-vue

Cette étape transforme les sommets (vertex) pour les positionner dans le repère de la caméra. Les cordonnées des vertex sont exprimées dans le repère local de l'objet mais le pipeline ne fonctionne que dans le repère de la caméra, un vertex v est transformé successivement par les matrices de modèle M et de projection P[p.38] . Où :

M : la matrice MODELVIEW qui positionne les sommets de l'objet dans le repère de la caméra.

P : la matrice PROJECTION qui détermine la projection réalisée par la caméra.

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = PM \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Exemple

p.38 Exemple de transformations p.38

```
1  /*description de la caméra*/
2  glMatrixMode(GL_PROJECTION);
3  glLoadIdentity();
4  gluPerspective(60, 1, 1, 20);
5
6  /*positionne la caméra*/
7  glMatrixMode(GL_MODELVIEW);
8  glLoadIdentity();
9  gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);
10
11 /*dessine le soleil (repère local)*/
12 soleil_affiche();
13
14 /*positionne la planète autour du soleil*/
15 glRotate(annee, 0, 1, 0);
16 glTranslate(orbite, 0, 0);
17 glRotate(jour, 0, 1, 0);
18
19 /*dessine une planète (repère local)*/
20 planete_affiche();
```

Dans un premier lieu, on fait la description de la caméra (lignes 1 à 9). La fonction glMatrixMode (GL_PROJECTION) indique que l'on souhaite manipuler la matrice de projection de la caméra. La projection est calculée par la fonction GLUT p.35 p.36 gluPerspective(60, 1, 1, 20) qui permet de générer une pyramide de vision d'ouverture 60°. Après la description de la caméra, il faut la donner une position et une direction, pour cela, on fait appel à la fonction gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0) qui donne la position de centre de projection (0, 0, 5), la position de point observé (0, 0, 0), et un vecteur (0, 1, 0) indiquant le haut de la scène (le z).

Maintenant, on passe aux positionnement des objets, on dessine le soleil dans son repère local (ligne 12), après on positionne la planète autour du soleil en faisant deux rotation et une translation (lignes 15 à 17). A la fin, on affiche la planète dans sa repère local.

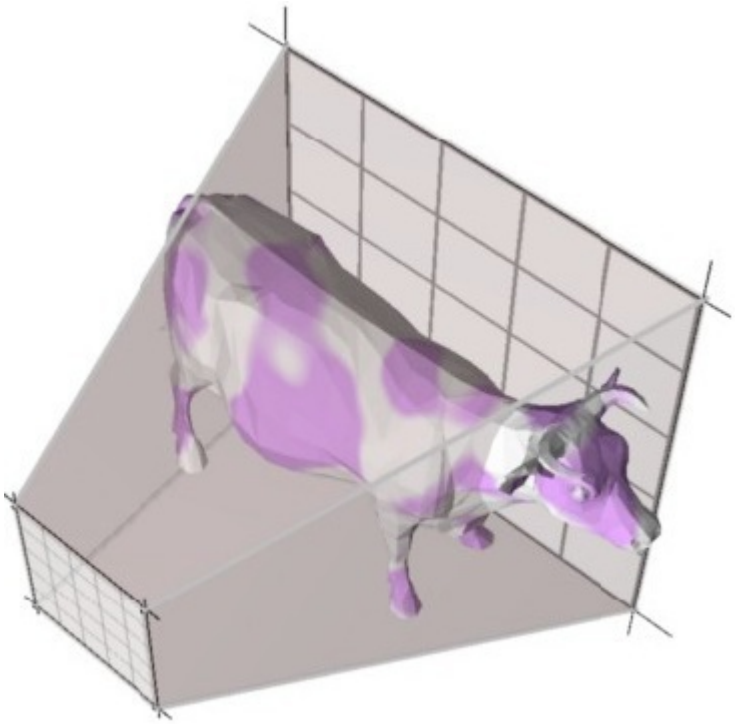
2.2.2. Étape 2 : éclairage et teinte des vertex

Cette étape consiste à appliquer le calcul des informations par vertex qui seront interpolées lors de la rasterisation. Ces informations englobent, par exemple, l'éclairage, la couleur du vertex ou le calcul des coordonnées de texture.

2.2.3. Étape 3 : clipping

Elle permet de calculer la contribution lumineuse uniquement des vertex des primitives qui sont à

l'intérieur de pyramide de vision, et de rejeter les vertex qui sont à l'extérieur et n'ont pas besoin d'être rendus.



Objet dans le volume de vue

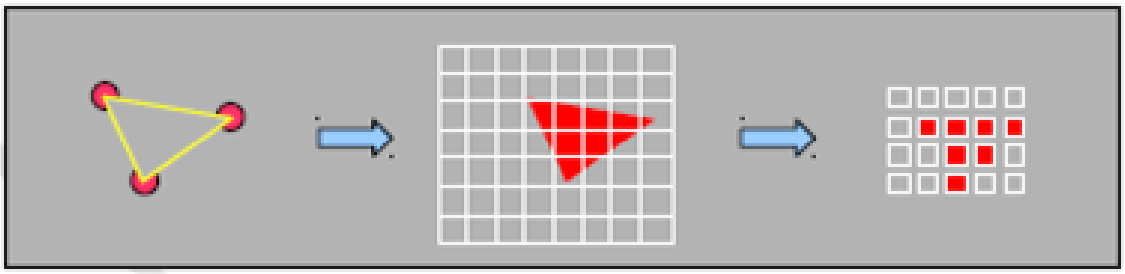
2.2.4. Étape 4 : projection

La projection consiste à prendre les vertex en espace caméra (transformés à l'étape de la transformation modèle-vue) et de les transformer, à l'aide de la matrice de projection, vers un espace normalisé cubique. Les limites de cet espace sont de (--1,-1,-1) à (1,1,1). Après la transformation de projection, les coordonnées des vertex sont dites être en coordonnées normalisées d'affichage. Notons aussi que la projection doit idéalement tenir compte de la dimension du canevas d'affichage en pixels afin de s'assurer que le passage des vertex en fragments, plus tard

dans le pipeline, correspondra aux bonnes coordonnées pixel[p.37] .

2.2.5. Étape 5 : rasterisation

Cette étape consiste à la construction des pixels à partir des vertex, pour les visualiser. La rastérisation des triangles consiste à prendre les triangles et à les convertir en fragments. Pour chaque fragment généré, on lui associe les données des vertex correspondants après les avoir interpolées. (Coordonnées textures, normale, couleur, etc.)[p.38] . Ces informations seront utilisées pour le calcul d'éclairage.



Rasterisation

2.3. Quelques effets

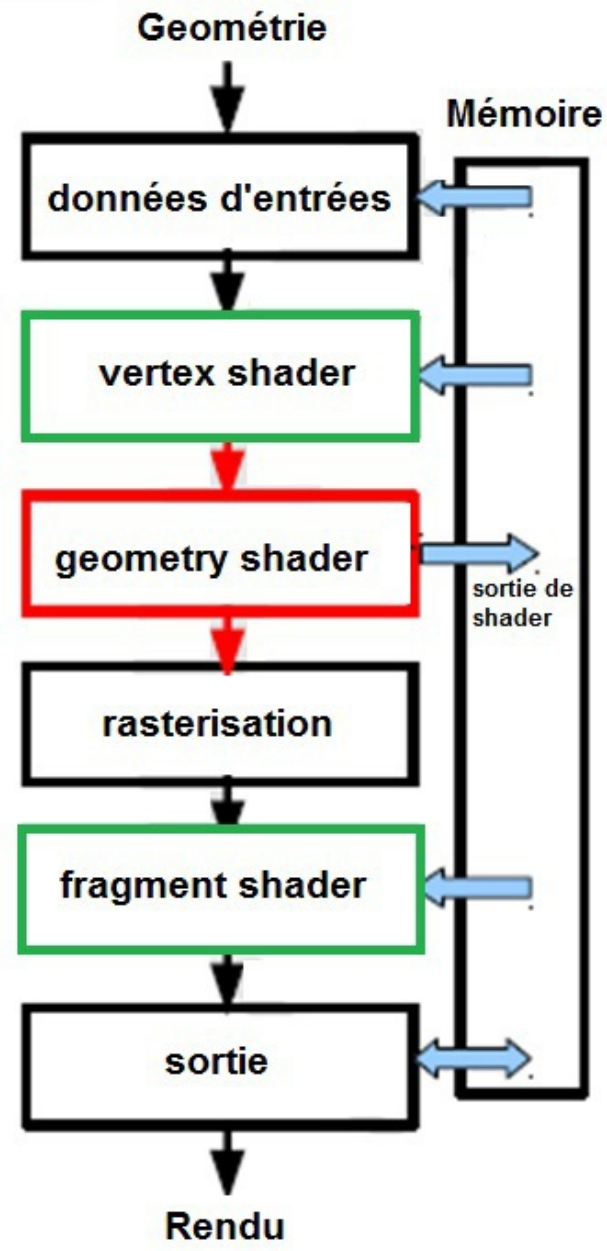
Le pipeline classique offre certains effets et méthodes telles que : le bump mappingp.35 , le système de particulep.35 , les images HDRsp.36 p.35 .

Quelques effets de pipeline classique

3. Pipeline programmable

3.1. Définition

Au différence de pipeline classique, le pipeline programmable des moderne GPUs, offre des fonctions qui peuvent être manipuler, gérer et modifier par le programmeur. L'utilité de pipeline programmable est de créer des méthodes de d'éclairage très évoluées en faisant des relations entre les différentes étapes de pipeline.



Pipeline programmable

3.2. Étapes

Dans le pipeline graphique programmable, on trouve trois nouveaux étapes qui sont entièrement gérées par le programmeur : le vertex shader[p.35] , le fragment shader, et la geometry shader.

1. *Vertex Shader :* transformation et éclairage.
2. *Geometry Shader :* assemblage des primitives.
3. *Fragment Shader :* ombrage et textures.

Les étapes de pipeline graphique programmable permettent de créer un programme pour paramétrer le rendu de la scène.

3.2.1. Étape 1 : vertex shader

Dans cette étape, le pipeline traite les sommets et leurs attributs. Les attributs des sommets sont composés d'une couleur, d'une normale et des coordonnées de texture. Cette étape prend un vertex à part pour travailler dessus[p.38] . Pour un triangle alors le vertex shader sera exécuté 3 fois.

Le rôle principal de vertex shader est de transformer les vertex (position, attributs) pour les positionner dans le repère de la caméra.

3.2.2. Étape 2 : geometry shader

Après la création d'un vertex shader pour chaque sommet de triangle de l'objet, l'étape de geometry shader consiste à l'assemblage des primitives de modèle 3D et de réaliser les opérations d'ajout / retrait de sommets et primitives.

3.2.3. Étape 3 : fragment shader

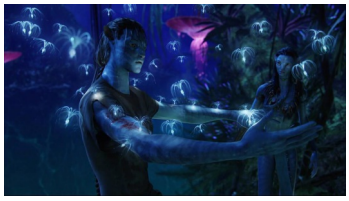
Cette étape permet de définir la couleur de chaque pixel de la forme délimitée par les vertex. Elle représente un pixel de l'image résultat ainsi que ces attributs.

Le rôle principal du fragment shader est de déterminer l'apparence de l'objet qui se projettent sur le pixel traité. La couleur, la transparence et la profondeur du fragment sont des paramètres nécessaires pour réaliser ce calcul[p.38] .

3.3. Quelques Effets

Le pipeline programmable (moderne) est souvent utilisé dans les jeux vidéo, la réalité virtuellep.35 , l'animation,.....

Quelques applications de pipeline programmable




4. Exercice

[solution n°3 p.30]

Retrouvez le type de pipeline correspond à ces tâches

transformation modèle-vue | assemblage des primitives | définir la couleur de chaque pixel

traiter les sommets et leurs attributs | rejeter les vertex qui sont à l'extérieur de pyramide de vision

éclairage et teinte des vertex | construction des pixels à partir des vertex

normalisation des coordonnées des vertex

Pipeline classique	Pipeline programmable

5. Pipeline classique vs pipeline programmable

Pipeline classique figé et limité :

Ne gère pas :

- Calculs d'ombres
- Réflexions, éclairement global
- La physique des scènes (collisions, mouvements...)

Pipeline programmable :

- BRDF plus sophistiquées
- Textures procédurales => matériaux plus réalistes
- Animation procédurale
- Anti Alissage

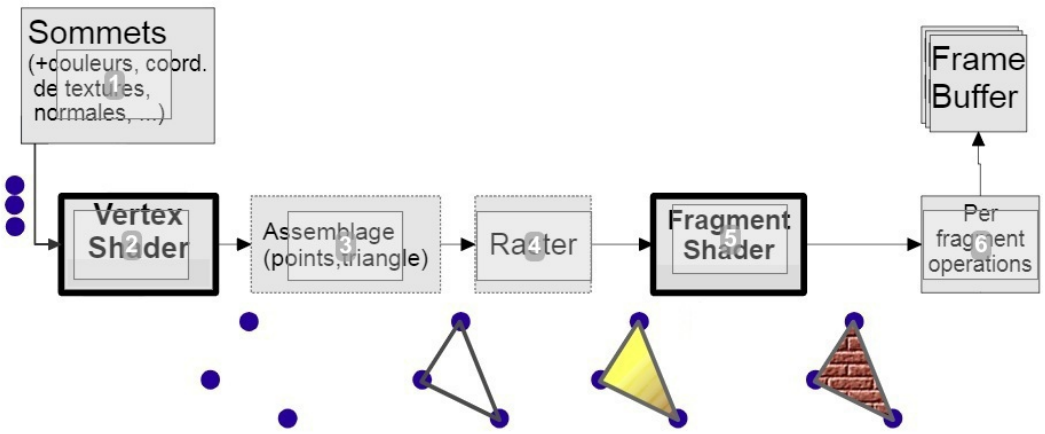
Ce tableau résume les différences entre un pipeline classique et pipeline programmable par rapport aux gestion de données, transformation de données et calcul de la couleur

	Pipeline classique	**Pipeline programmable**
Gestion des données	Envoi spécifique de sommet, normales, textures de coordonnées...	Tout est dans des buffers génériques. L'utilisation des données par « sommet » est spécifiée par leprogrammeur
Transformation de données	Pile de matrice. Fonctions pour les transformations et projections élémentaires	Entièrement spécifiées par le programmeur
Calcul de la couleur	Utilisation du modèle d'éclairement fixé. Accès aux données via des mots clés.	Écriture de modèle d'éclairement par le programmeur, On doit écrire un shader. Liberté d'écriture.

6. Exercice

[solution n°4 p.30]

Parmi les étapes ci-dessous, lesquelles illustrent les étapes programmables?

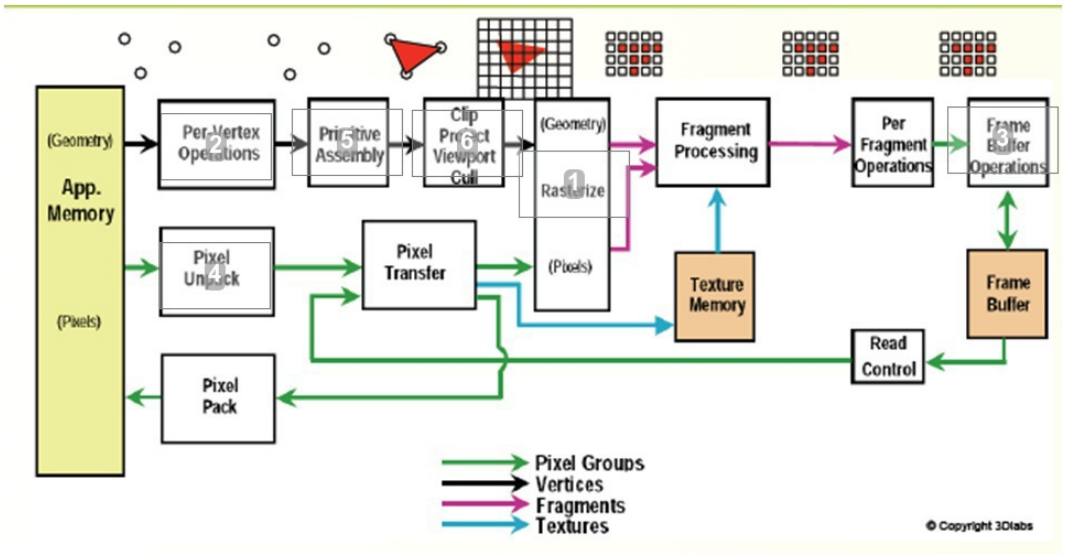


- ☐ 1. A
- ☐ 2. B
- ☐ 3. C
- ☐ 4. D
- ☐ 5. E
- ☐ 6. F

7. Exercice

[solution n°5 p.31]

Parmi les étapes ci-dessous, lesquelles illustrent les étapes de vertex shader?



- ☐ 1. A
- ☐ 2. B
- ☐ 3. C
- ☐ 4. D
- ☐ 5. E
- ☐ 6. F

Tester votre compréhension

V

Exercice

[solution n°6 p.32]

Le CPU offre :

- ☐ un coût mémoire élevé
- ☐ un temps de calcul interactif
- ☐ un temps de calcul élevé

Exercice

[solution n°7 p.32]

On utilise les GPUs pour :

- ☐ ajouter des précisions à un modèle
- ☐ calculer de manière précise l'éclairage
- ☐ modéliser une scène 3D
- ☐ accélérer le temps de calcul de rendu

Exercice

[solution n°8 p.33]

Un pipeline graphique de rendu est :

- ☐ une succession d'étapes indépendantes
- ☐ une succession d'étapes dépendantes
- ☐ ensemble de fonctions de rendu exécutables sur GPUs
- ☐ ensemble de fonctions de rendu exécutables sur CPUs

Exercice

[solution n°9 p.33]

Un pipeline classique contient :

- ☐ un geometry shader
- ☐ le clipping
- ☐ la projection
- ☐ un fragment shader
- ☐ la rasterisation

Exercice

[solution n°10 p.33]

Le fragment shader permet de :

- ☐ définir la couleur de chaque pixel
- ☐ calculer les sommets de primitives
- ☐ déterminer les transformations appliquées un un vertex
- ☐ calculer des méthodes d'éclairage

Exercice

[solution n°11 p.34]

En soustrayant le vecteur de position de la caméra du vecteur d'origine de la scène, on obtient :

Exercice

[solution n°12 p.34]

Le pipeline qu'on ne peut pas modifier ces étapes est nommé pipeline :

Exercice

[solution n°13 p.34]

L'étape d'assemblage de primitives est la :

Exercice

[solution n°14 p.34]

Un shader est un programme exécuté sur :

Exercice

[solution n°15 p.34]

Le domaine de la réalité virtuelle utilise un pipeline :

Conclusion

V

- Le pipeline graphique est un ensemble d'étapes dépendantes qui sert à exécuter une certaine partie de programme sur le GPU.
- Il existe deux grandes catégories de pipeline graphique (pipeline de rendu) qui est le pipeline classique et le pipeline programmable.
- La principale étape qu'elle a été ajouté dans le pipeline programmable est la geometry shader

Solutions des exercices

> **Solution** n°1

Exercice p. 8

Classez les types des cartes graphiques Nvidia selon leurs année de génération.

RIVA TNT | RIVA TNT2 | GeForce 256 | GeForce 2 | GeForce 3 | GeForce 4 Ti | GeForce FX

> **Solution** n°2

Exercice p. 11

Exercice

On projette les objets de la scène sur un plan nommé plan objet

- ○ Vrai
- ◉ Faux

Exercice

Le pyramide de vision consiste à déterminer les parties visibles de la caméra

- ◉ Vrai
- ○ Faux

Exercice

Un pyramide de vision est composé de deux plan écran et plan loin

- ○ Vrai
- ◉ Faux

Exercice

Pour visualiser un objet, on doit passer d'un espace caméra à un espace monde

- ○ Vrai
- ◉ Faux

Exercice

La caméra est caractérisée par une position et une direction

- ◉ Vrai
- ○ Faux

Exercice

On projette les objets de la scène sur un plan nommé plan objet

- ○ Vrai
- ◉ Faux

> **Solution** n°3

Exercice p. 21

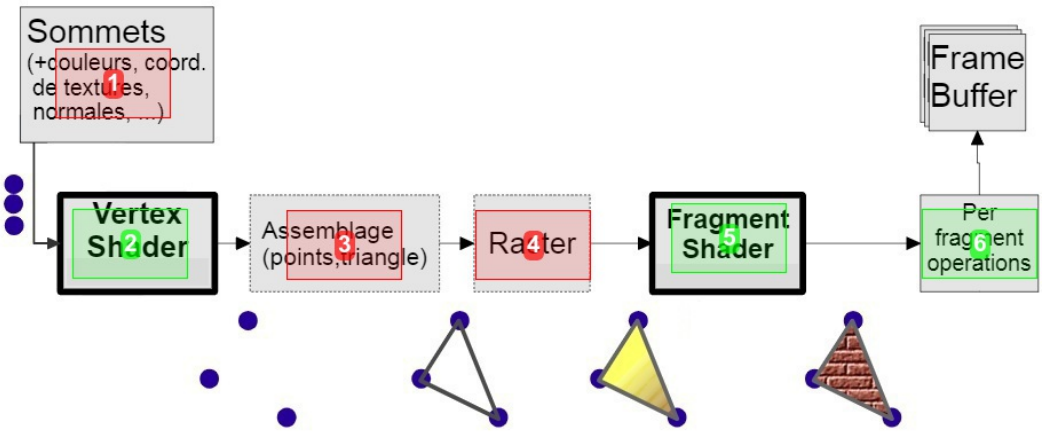
Retrouvez le type de pipeline correspond à ces tâches

Pipeline classique	Pipeline programmable
normalisation des coordonnées des vertex	définir la couleur de chaque pixel
éclairage et teinte des vertex	construction des pixels à partir des vertex
traiter les sommets et leurs attributs	assemblage des primitives
rejeter les vertex qui sont à l'extérieur de pyramide de vision	
transformation modèle-vue	

> **Solution** n°4

Exercice p. 23

Parmi les étapes ci-dessous, lesquelles illustrent les étapes programmables?



☐ 1

A

☑ 2

B

☐ 3

C

☐ 4

D

☑ 5

E

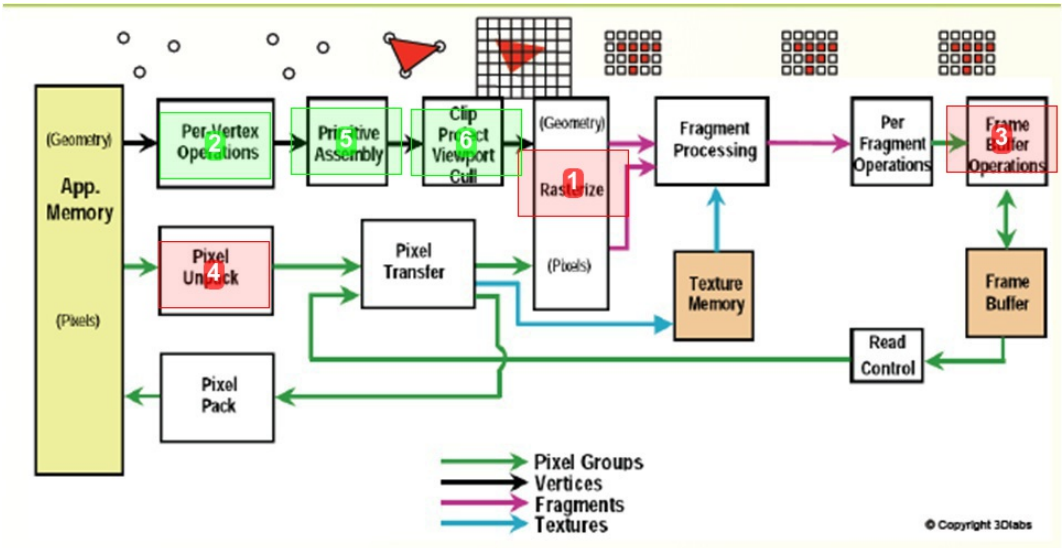
☑ 6

F

> **Solution** n°5

Exercice p. 24

Parmi les étapes ci-dessous, lesquelles illustrent les étapes de vertex shader?



- ☐ 1

A

- ☑ 2

B

- ☐ 3

C

- ☐ 4

D

- ☑ 5

E

- ☑ 6

F

> **Solution** n°6

Exercice p. 25

Le CPU offre :

- ☑ un coût mémoire élevé
- ☐ un temps de calcul interactif
- ☑ un temps de calcul élevé

> **Solution** n°7

Exercice p. 25

On utilise les GPUs pour :

- ☐ ajouter des précisions à un modèle
- ☑ calculer de manière précise l'éclairage
- ☐ modéliser une scène 3D
- ☑ accélérer le temps de calcul de rendu

> **Solution** n°8

Exercice p. 25

Un pipeline graphique de rendu est :

- ☐ une succession d'étapes indépendantes
- ☑ une succession d'étapes dépendantes
- ☑ ensemble de fonctions de rendu exécutables sur GPUs
- ☐ ensemble de fonctions de rendu exécutables sur CPUs

> **Solution** n°9

Exercice p. 26

Un pipeline classique contient :

- ☐ un geometry shader
- ☑ le clipping
- ☑ la projection
- ☐ un fragment shader
- ☑ la rasterisation

> **Solution** n°10

Exercice p. 26

Le fragment shader permet de :

- ☑ définir la couleur de chaque pixel

- ☐ calculer les sommets de primitives
- ☐ déterminer les transformations appliquées un un vertex
- ☑ calculer des méthodes d'éclairage

> **Solution** n°11

Exercice p. 26

En soustrayant le vecteur de position de la caméra du vecteur d'origine de la scène, on obtient :

la direction

> **Solution** n°12

Exercice p. 26

Le pipeline qu'on ne peut pas modifier ces étapes est nommé pipeline :

classique

> **Solution** n°13

Exercice p. 26

L'étape d'assemblage de primitives est la :

geometry shader

> **Solution** n°14

Exercice p. 26

Un shader est un programme exécuté sur :

GPU

> **Solution** n°15

Exercice p. 27

Le domaine de la réalité virtuelle utilise un pipeline :

programmable

Glossaire

Bump mapping

plaquage de relief (en français) est une technique utilisée en infographie et qui sert à donner du relief aux modèles 2D ou 3D, ou aux textures. Elle consiste à modifier la normale de la surface.

GLUT

bibliothèque qui offre des fonctions pour la gestion des fenêtres OpenGL.

HDR

un ensemble de techniques numériques permettant d'obtenir une grande plage dynamique dans une image. Son intérêt est de pouvoir représenter ou de mémoriser de nombreux niveaux d'intensité lumineuse dans une image.

Réalité virtuelle

nouvelle technologie qui permet de simuler la présence réelle d'une personne dans un environnement virtuelle.

Shader

programme exécuté sur le GPU

Système de particules

une technique qui permet de simuler de nombreux phénomènes naturels tels que feu, explosion, fumée, eau, nuage, poussière, neige, feux d'artifices.

Abréviations

CPU : Central Processing Unit

GLUT : OpenGL Utility Toolkit

GPU : Graphics Processing Unit

HDR : High Dynamic Range

Références

Camera https://learnopengl.com/Getting-started/Camera

PROCESSEURS GRAPHIQUE ET PIPELINE Olivier Vaillancourt, Olivier Godin, Automne 2014, Université de Sherbrooke

Bibliographie

V

Ramakrishnan Mukundan, *Advanced methods in computer graphics,* Springer, 2012

NOUR EL-HOUDA BENALIA, *Exploitation des potentialités des cartes graphiques à l'optimisation des algorithmes évolutionnaires,* thèse de Doctorat, Laboratoire LESIA, Université de Biskra, Algérie, 2015

Bernard Péroche et Dominique Bechmann, *Informatique graphique et rendu,* édition Hermes

François Sillion, *Synthèse d'image géographique,* édition hermes, 2002