



Programmation GPU

...

De Mecquenem Ninon
Falgarone Théo
Hubert Alexis
N'Goma Komb Mercia

Introduction

Problème d'amélioration des performances CPU (Central Processing Unit)



Paralléliser les calculs

Le GPU : Graphics Processing Unit

- Moins consommateurs et moins chers
- Stockage d'un très grand nombre de threads en parallèle directement sur le processeur
- Le GPU peut choisir un thread sans aucun délai

"High Performance Computing : are GPU going to take over ?", A. Nedelcoux, Université du SI, 2011.

Programmation GPU

- **nVIDIA CUDA**

- > Contient de nombreuses bibliothèques dans de nombreux langages
- > Uniquement compatible avec du matériel nVIDIA supportant CUDA

- **bibliothèque Microsoft C++ Accelerated Massive Parallelism (Microsoft C++ AMP)**

- > fonctionne grâce à DirectX
- > Supporte un grand nombre de matériels
- > S'exécute sur les systèmes Microsoft Windows

<https://blogs.msdn.microsoft.com/devpara/2011/09/02/introduction-la-programmation-gpu-part-1/>

Programmation GPU

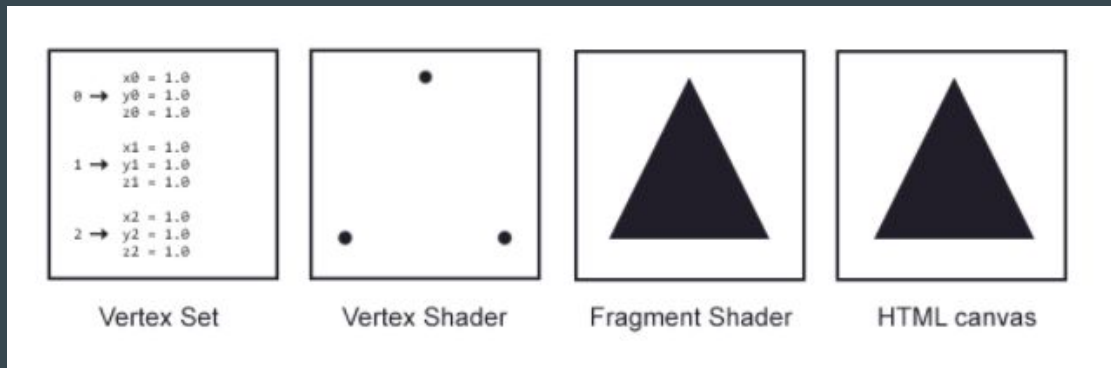
- **OpenGL** (Open Graphic Library)
 - > Interface de programmation
 - > Silicon Graphics
 - > Création de scènes complexes à partir de formes géométriques simples
- **OpenCL** (Open Computing Language)
 - > Une API et un langage de programmation
 - > Khronos Group
 - > Indépendance vis-à-vis de la plateforme et du matériel

WebGL - Introduction

- Dérivé de la bibliothèque OpenGL's ES 2.0 => Open source
- Design d'éléments graphiques interactifs 2D et 3D sur le Web
- Compatible avec de nombreux moteur de recherches (y compris sur smartphone)
- Programmation en JavaScript
- Pas besoin de compiler

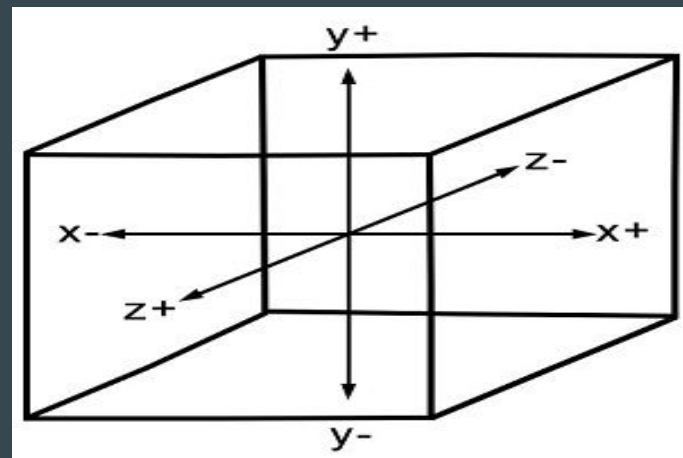
WebGL : les différentes étapes

- Préparer les canevas et le context WebGL
- Créer et compiler les Shader (vertex + fragment)
- Créer les Buffers et y associer les Shaders
- Dessiner les objets



WebGL : Préparer les canevas et le context WebGL

- HTML-5<canvas> : option facile et puissante pour dessiner des graphes, des images ou des animations en utilisant JavaScript
- Le contexte contient les paramètres (données, variables, conditions, etc.)
- Système de coordonnées compris entre -1 et 1 pour les trois dimensions



WebGL : Préparer les canevas et le context WebGL

```
<!DOCTYPE html>
<html>
  <body>
    <canvas id="glCanvas" width="1840" height="960"></canvas>
    <script src="solution.js"></script>
  </body>
</html>
```

```
var canvas = document.getElementById("glCanvas");
var gl = canvas.getContext("webgl");
```


WebGL : Créer et compiler les Shader

- Les shaders sont les programmes pour GPU.
- Ils définissent comment les sommets, les transformations, les matériaux, les lumières et la caméra interagissent les uns avec les autres.
- Deux types de shaders :
 - Vertex shader : objet et caméra
 - Fragment shader : couleur, texture et lumière, matériaux

WebGL : Créer et compiler les Shader

```
function createShader(gl, type, source) {  
  var shader = gl.createShader(type);  
  gl.shaderSource(shader, source);  
  gl.compileShader(shader);  
}
```

WebGL : Créer les Buffers et y associer les Shaders

- Les buffers sont les zones de mémoire de WebGL qui contiennent les données.
- Trois types de buffers :
 - vertex buffer : données correspondant à chaque vertex
 - index buffer : données correspondant aux indices des vertices
 - frame buffer : partie de la mémoire graphique qui contient les données de la scène

WebGL : Créer les Buffers et y associer les Shaders

```
// Create a buffer and put three 2d clip space points in it  
var positionBuffer = gl.createBuffer();  
  
// Bind it to ARRAY_BUFFER (think of it as ARRAY_BUFFER = positionBuffer)  
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);  
  
var positions = []; //TODO  
  
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(positions), gl.STATIC_DRAW);
```

WebGL : Dessin des objets

```
var primitiveType = gl.TRIANGLES;  
var offset = 0;  
var count = ; //TODO  
gl.drawArrays(primitiveType, offset, count);
```

Énoncé

- Compléter les *TODO* dans le fichier *temp.js*
- Construire une étoile à six branches dans le canvas
- Intégrer une couleur au fragment shader
- Liens utiles :

<https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html>

https://developer.mozilla.org/fr/docs/Web/API/WebGL_API/Tutorial/Commencer_avec_WebGL