Premiers pas avec Vulkan

Ce mini-TP a pour objectif de vous faire installer Vulkan sur vos machines, et d'exécuter un petit code permettant de lister les caractéristiques et les compat- ibilités de votre ordinateur.

1 Installation de Vulkan

1. Rendez-vous sur le site de vulkan.lunarg.com et dirigez-vous vers la page de l'installateur associé à votre OS (Windows, Linux, Mac).
2. Téléchargez la version la plus récente du SDK Installer.
3. Lancez l'installeur sur votre machine. Suivez les instructions et **notez bien le répertoire d'installation**.
4. Suivez les indications de l'installateur.

2 Création d'un premier code Vulkan

2.1 Préparation de l'environnement de travail

Nous allons maintenant préparer l'environnement de travail afin de pouvoir utiliser Vulkan dans ce projet.
Dans Visual Studio

1. Choisir Créer un projet.

2. Choisir un Projet vide.

3. A droite de votre écran se trouve une sous-fenêtre *Explorateur de solutions*. Sur le dossier nommé *Fichiers sources*, faites un clic-droit et choisissez *Ajouter...* → *Nouvel élément*.

4. Sélectionner *Fichier C++*, que vous nommerez main.cpp. Créez le fichier.

5. Ouvrez l'onglet *Projet* → *Propriétés*. Vérifiez bien que vous soyez dans le cas *Configuration : Toutes les configurations* et *Plateforme : Toutes les plateformes*. Dans la fenêtre des propriétés, ouvrez l'onglet *Propriétés de configuration* → *C/C++* → *Général* et positionnez votre curseur sur la ligne *Autres répertoires Include*.

6. Ajoutez le sous-répertoire Include situé à la racine de votre installation de Vulkan, typiquement sous la forme suivante : C:\VulkanSDK\1.X.XXX.X\Include

7. Toujours dans la fenêtre des propriétés de votre solution, dirigez-vous maintenant vers *Propriétés de configuration* → *Editeur de liens* → *Général* et dirigez-vous vers la ligne *Répertoires de bibliothèques supplémentaires*. Ajoutez cette fois-ci la racine des libraires Vulkan sur votre ordinateur, que vous trouverez probablement dans C:\VulkanSDK\1.X.XXX.X\Lib

8. Toujours dans l'*Editeur de liens*, dirigez-vous maintenant vers la sous- section *entrée*. Sur la première ligne nommée *Dépendances supplémen- taires*, ajoutez à l'aide de la flèche de droite, puis en cliquant ensuite sur <*Modifier...*> la valeur vulkan-1.lib

9. Validez et fermez la fenêtre.

2.2 Initialisation de Vulkan

Nous pouvons tout d'abord écrire un code C++ de la manière la plus classique : int main(){}. Vérifiez que ce simple code s'exécute bien et renvoie bien la valeur 0.

1. Ajoutez les différentes libraires nécessaires pour exécuter le code. NB : pour utiliser Vulkan, incluez <vulkan/vulkan.h>.

2. Pour que Vulkan fonctionne, vous devez avant toute chose créer une in- stance. Pour cela, vous devez déclarer un objet VkInstance qu'il n'est pas utile d'initialiser.
   NB : pour les plus pointilleux d'entre vous, il est normalement bon de prendre l'habitude en C++ d'éviter de déclarer un objet sans l'initialiser. Ceux qui souhaitent suivre l'adage peuvent initialiser leur instance Vulkan avec un nullptr, ou en utilisant le mot-clé VK_NULL_HANDLE.

3. Vous déclarerez également un objet VkInstanceCreateInfo, pour lequel l'initialiserez par {}.

4. Initialisez Vulkan. Pour cela, utilisez la fonction vkCreateInstance à laquelle vous donnerez trois arguments, tout d'abord **l'adresse** de votre objet VkInstance, un pointeur nul nullptr et enfin **l'adresse** de votre objet VkInstanceCreateInfo.
   Pour vous assurer que l'instanciation a réussi, la fonction doit renvoyer la valeur VK_SUCCESS. Vérifiez que ce soit bien le cas, et si non, renvoyez un message d'erreur et arrêtez le code.

5. Puisque nous travaillons avec des outils destinés au GPU, prenez garde de vous assurer que vous détruisiez tout ce que vous créez. Prenez désormais l'habitude en mettant dès à présent à la fin de votre code la commande vkDestroyInstance qui prendra en arguments votre instance Vulkan et un pointeur nul.

2.3 Nombre d'appareils disponibles sur la machine

Utilisons désormais Vulkan pour trouver les différents appareils disponibles et compatibles sur votre machine.

1. Utilisez la fonction vkEnumeratePhysicalDevices prenant trois argu- ments : votre instance Vulkan, **l'adresse** d'un entier et un pointeur nul.
2. Affichez la valeur de l'entier, correspondant au nombre d'appareils détectés et compatibles Vulkan sur votre machine. Appelez-moi lorsque vous serez à cette étape et dites-moi le nombre d'appareils compatibles.
3. Si ce nombre est non nul, vous pouvez continuer.

2.4 Liste des appareils disponibles sur la machine

1. Créez un vecteur à l'aide de la librairie C++ <vector> qui contiendra des objets de type VkPhysicalDevice, que vous initialiserez par le nombre d'appareils que vous avez trouvé à la question précédente.

2. Remplissez ce vecteur des appareils détectés par Vulkan en appelant une nouvelle fois la fonction vkEnumeratePhysicalDevices avec les mêmes arguments que précédemment, à l'exception du dernier pour lequel vous enverrez un pointeur vers la mémoire de votre vecteur, en utilisant le suffixe .data()

3. A l'aide d'une boucle parcourant la liste des appareils stockés dans votre vecteur, vous récupérerez et afficherez les propriétés de chacun en déclarant tout d'abord un objet VkPhysicalDeviceProperties, puis en utilisant la fonction vkGetPhysicalDeviceProperties, celle-ci prenant deux arguments : l'appareil détecté par Vulkan et **l'adresse** de votre objet stockant les propriétés de votre appareil.

4. Vous pouvez ensuite afficher le nom de l'appareil en y accédant par l'objet stockant ses propriétés à l'aide du suffixe .deviceName.