Ilkka Pokkinen

Vulkan-ohjelmointirajapinta

Metropolia Ammattikorkeakoulu

Tieto- ja viestintätekniikan tutkinto-ohjelma

Menetelmäopinnot, Pelisovellukset

Tutkielma

4.11.2020

Metropolia

Sisällys

1 Johdanto

Vulkan on grafiikkasuorittimen ohjaamiseen tarkoitettu ohjelmointirajapinta. The Khronos Group aloitti sen kehittämisen vuonna 2014, käyttäen sen pohjana AMD:n Mantle-rajapintaa. Vulkanin ensimmäinen versio julkaistiin vuonna 2016 ja Vulkan 1.2 vuonna 2020 (1; 2).

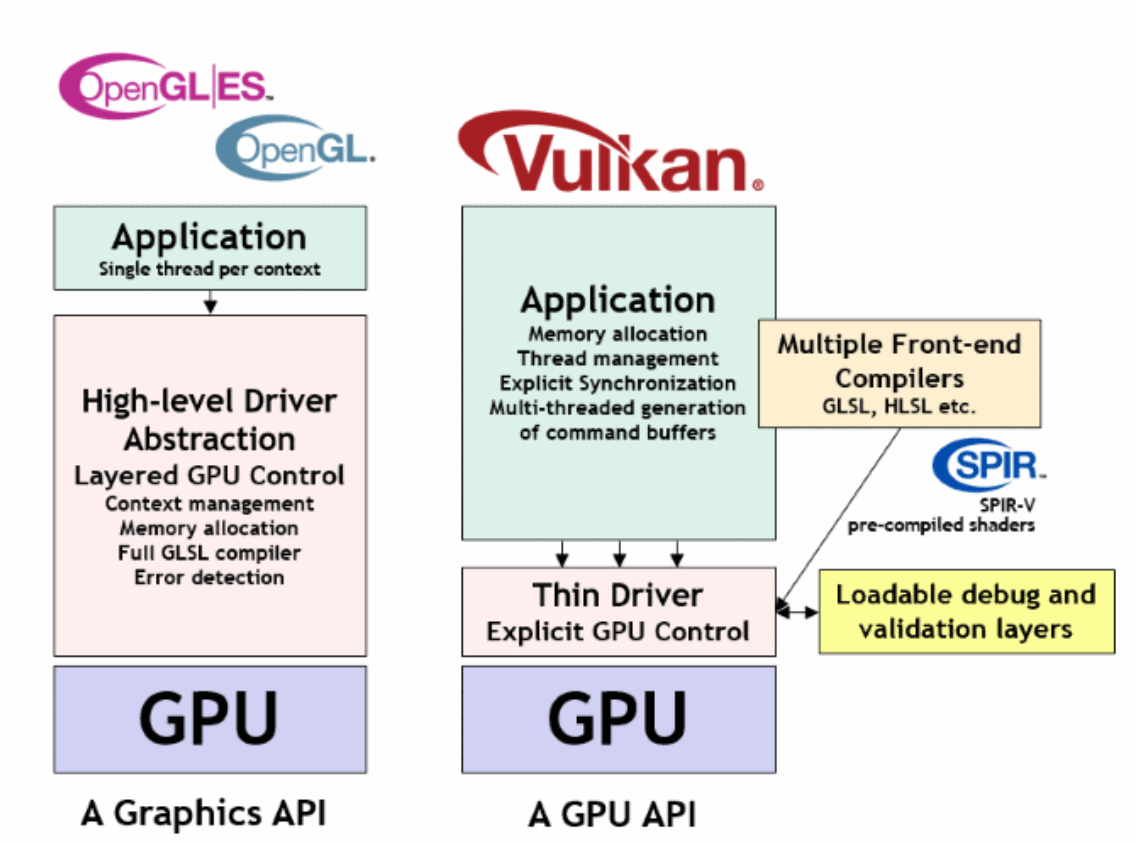
Tässä tutkielmassa esitellään Vulkan-rajapinta, sen käyttötarkoitukset, alustaminen ja suorituskykyeroja verrattuna OpenGL-rajapintaan. Tutkielma ei ole kaikenkattava eikä tutoriaali, vaan tarkoituksena on tutustua Vulkanin pääperiaatteisiin.

2 Vulkan-ohjelmointirajapinta

Vulkan on ohjelmointirajapinta, jonka pääkäyttötarkoitus on grafiikan piirtäminen näytönohjaimen avulla. Se on kuitenkin kohtuullisen monikäyttöinen, ja 2D- ja 3D-grafiikan esittämisen lisäksi sitä voidaan käyttää myös näytön ulkopuoliseen renderoimiseen, yleislaskentaan ja koneoppimisalgoritmien suorittamiseen näytönohjaimella (3). Vuonna 2020 Vulkan sai tuen säteenseurannalle (4).

Rajapinnalla on laaja tuki eri alustoilla. Vulkan-tuki on Windows 7-, 8- ja 10-käyttöjärjestelmillä, yleisillä Linux-distribuutioilla, Androidilla ja Nintendo Switch -pelikonsolilla. Applen Metal-grafiikkarajapinnalle on olemassa työkalut, jotka mahdollistavat Vulkan koodin ajamisen (3). Monet pelimoottorit, kuten Unity, Unreal Engine, Godot, CryEngine ja Source 2, tukevat Vulkania.

Vulkan antaa Khronos Groupin OpenGL-rajapintaa suuremman hallinnan grafiikkasuorittimen toimintaan pienentämällä ajureille kuuluvaa työmäärää ja paljastamalla laitetason abstraktion ohjelmoijalle (kuva 1). Tämä tekee Vulkanin käyttämisestä työläämpää verrattuna muihin korkeamman tason rajapintoihin.



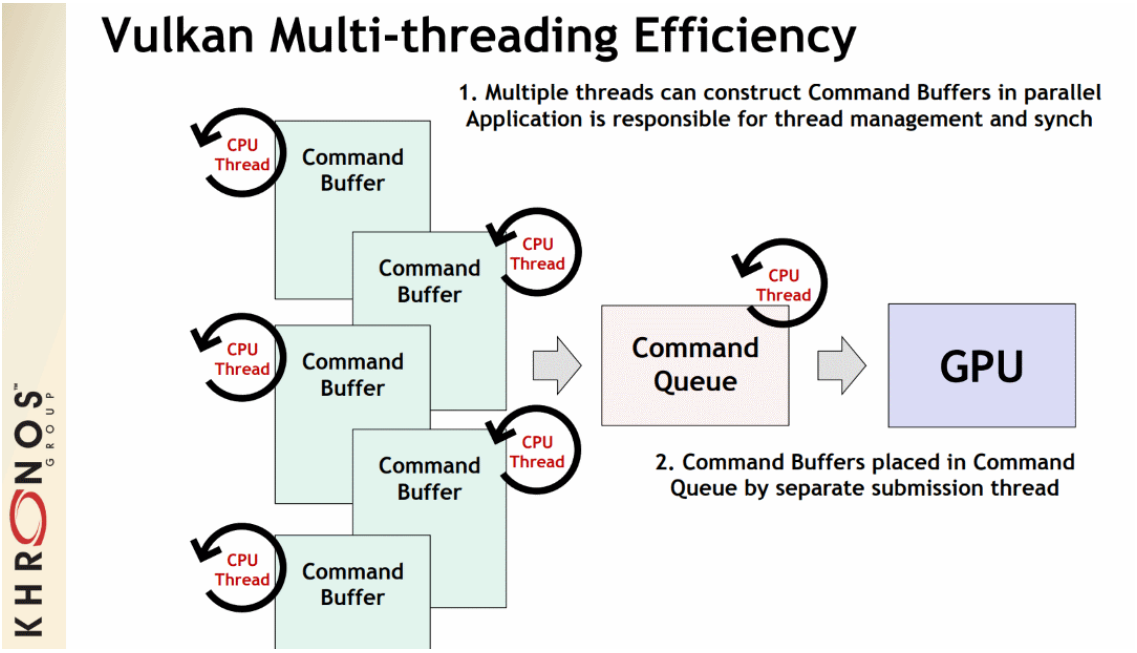
Kuva 1. OpenGL- ja Vulkan-ohjelmointirajapintojen sovellus- ja ajuritason eroavaisuudet. Vulkan-ajurit ovat keveämmät, mutta niiden tehtäviä on siirretty sovellukselle (3).

Vulkanissa on ominaisuus, jota kutsutaan yksinkertaisesti nimellä kerrokset (layers), jotka reitittävät Vulkan-funktiokutsuja kerrosten kautta. Kerroksia voidaan käyttää esimerkiksi virheidentarkistukseen ja lokitiedostojen kirjoittamiseen, ja ne voidaan helposti poistaa käytöstä sovelluksen julkaisuversiossa suorituskyvyn parantamiseksi. (5).

SPIR-V on The Khronos Groupin kehittämä varjostinkieli, jota Vulkan-rajapinta käyttää. Se on muodoltaan tavukoodia, ja näin se ei ole ihmiselle sellaisenaan luettavaa. SPIR-V-varjostinmoduuleja voi kirjoittaa GLSL-, HSSL- ja HLSL-varjostinkielillä, joille Khronos on luonut SPIR-V-kääntäjän (3; 6).

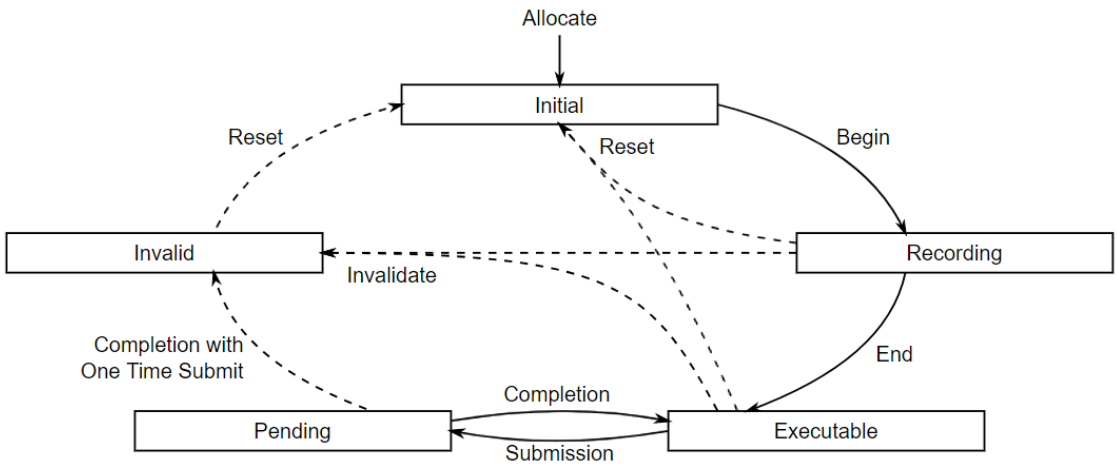
Komentopuskuri on Vulkan-olio, johon prosessori kirjoittaa näytönohjaimen komentoja. Kirjoittamisen jälkeen komentopuskuri lisätään näytönohjaimen komentojonoon, josta se menee näytönohjaimelle suoritettavaksi. Suorittamisen jälkeen komentopuskurin tila saattaa olla virheellinen (kuva 3).

Tietokoneiden trendinä on ollut laskentaytimien määrän kasvaminen ja rinnakkaisajon yleistyminen. Vulkanissa komentopuskureita voidaan kirjoittaa rinnakkain usealla prosessorin säikeellä (kuva 2).



Kuva 2. Vulkanin komentopuskureita voidaan kirjoittaa rinnakkain usealta säikeeltä. Yksi säie huolehtii puskureiden siirtämisestä näytönohjaimen komentojonoon (3).

Vulkan-rajapinnassa on määritelty ensisijainen ja toissijainen komentopuskuri. Toissijaista komentopuskuria ei lisätä komentojonoon, vaan se suoritetaan ensisijaisen komentopuskurin mukana. Samaa toissijaista komentopuskuria voi käyttää usea ensisijainen komentopuskuri. Toissijaisen komentopuskurin mennessä virheelliseen tilaan myös kaikki sitä käyttävät ensisijaiset komentopuskurit siirretään virheelliseen tilaan. (5).



Kuva 3. Komentopuskurin mahdolliset tilat ja siirtymät niiden välillä (5).

Komentopuskuri on alustuksen jälkeen alkutilassa, johon puskuri voi myös palata joidenkin komentojen jälkeen. Vulkan-funktio vkBeginCommandBuffer siirtää puskurin kirjoitustilaan. Tämän jälkeen komentopuskuri siirretään suoritusta odottavaan tilaan, jolloin puskuriin ei voi enää kirjoittaa. Komentopuskurin siirtäminen näytönohjaimen komentojonoon tai suorittavaksi ensisijaiselle komentopuskurille muuttaa komentopuskurin odotustilaan. Kun komentopuskuri on suoritettu, se siirtyy automaattisesti joko takaisin suoritusta odottavaan tilaan tai virhetilaan (kuva 3).

3 Vulkanin alustaminen

3.1 Instanssi

Ensimmäinen askel Vulkanin alustamisessa on VkInstance-olion luominen. Se sisältää tietoa sovelluksen ja Vulkanin versioista, nimistä, laajennuksista ja kerroksista. Vulkan-olioiden luomiseen tarvittava informaatio kerätään ensin rakenteeseen (esimerkkikoodi 1), joka annetaan luontifunktiolle parametrina. Instanssin luontifunktio VkCreateInstance alustaa myös Vulkan-kirjaston (5).

```
typedef struct VkInstanceCreateInfo {
    VkStructureType             sType;
    const void*                 pNext;
    VkInstanceCreateFlags       flags;
    const VkApplicationInfo*    pApplicationInfo;
    uint32_t                    enabledLayerCount;
    const char* const*          ppEnabledLayerNames;
    uint32_t                    enabledExtensionCount;
    const char* const*          ppEnabledExtensionNames;
```

```
} VkInstanceCreateInfo;
```

Esimerkkikoodi 1. VkInstance-olion luomisessa käytettävä rakenne (7).

Vulkanin alustariippumattomuuden takia se ei sisällä toimintoja ikkunan luomiseen. Jos tarkoituksena on näyttää grafiikkaa näytöllä, tarvitaan ikkunan luomiseen ulkopuolinen kirjasto, kuten GLFW. Grafiikan piirtoa varten joudutaan myös luomaan Vulkanin ikkunaa abstrahoiva olio VkSurfaceKHR.

3.2 Laitteen alustaminen

Vulkan-rajapinnalla ohjelmoitaessa joudutaan tarkistamaan käytettävä laite ja sen ominaisuudet. VkEnumeratePhysicalDevices-funktiolla haetaan lista fyysisistä laitteista, joita instanssi pystyy käyttämään.

Fyysiselle laitteelle on määritelty komentoja vastaanottava olio VkQueue, joita voi olla useita laitteen mukaan. Komentojono voi käsitellä yhtä tai useampaa neljästä operaatiotyypistä: grafiikka-, laskenta-, siirto- ja ”sparse binding” -operaatiot. Jos tarkoituksena on luoda 3D-grafiikkamoottori, tarvitaan komentojono ainakin kolmelle ensimmäiselle operaatiotyypille.

Kuvan näyttämiseen ruudulla laitteella tulee olla kuvapuskuri, jota Vulkanissa esitetään VkSwapchainKHR-oliolla. Kuvapuskurille tulee määritellä esimerkiksi kuva- ja väriformaatit ja esitysmoodi (esimerkkikoodi 2). Jos kuvan esitykseen tulee jokin muutos, esimerkiksi resoluutio vaihtuu, VkSwapchainKHR-olio joudutaan luomaan uudelleen.

```
typedef struct VkSwapchainCreateInfoKHR {
    VkStructureType                  sType;
    const void*                      pNext;
    VkSwapchainCreateFlagsKHR        flags;
    VkSurfaceKHR                     surface;
    uint32_t                         minImageCount;
    VkFormat                         imageFormat;
    VkColorSpaceKHR                  imageColorSpace;
    VkExtent2D                       imageExtent;
    uint32_t                         imageArrayLayers;
    VkImageUsageFlags                imageUsage;
    VkSharingMode                    imageSharingMode;
    uint32_t                         queueFamilyIndexCount;
    const uint32_t*                  pQueueFamilyIndices;
    VkSurfaceTransformFlagBitsKHR    preTransform;
    VkCompositeAlphaFlagBitsKHR      compositeAlpha;
    VkPresentModeKHR                 presentMode;
    VkBool32                         clipped;
    VkSwapchainKHR                   oldSwapchain;
} VkSwapchainCreateInfoKHR;
```

Esimerkkikoodi 2. VkSwapchainKHR-olion luomisessa käytettävä rakenne (7).

Esitysmoodi määrää, miten näytönohjain ottaa huomioon näytön vertikaalisen intervallin. Jotkin esitysmoodeista saattavat aiheuttaa kuvan repeytymistä näytöllä. Näytönohjaimen ominaisuudet määrittelevät, mitä esitysmoodeja voi käyttää. Esitysmoodeja ovat

- VK_PRESENT_MODE_IMMEDIATE_KHR
- VK_PRESENT_MODE_MAILBOX_KHR
- VK_PRESENT_MODE_FIFO_KHR
- VK_PRESENT_MODE_FIFO_RELAXED_KHR
- VK_PRESENT_MODE_SHARED_DEMAND_REFRESH_KHR
- VK_PRESENT_MODE_SHARED_CONTINUOUS_REFRESH_KHR.

4 Suorituskyky

Vulkan pystyy säikeistyksen avulla hyödyntämään käyttämätöntä näytönohjaimen laskentatehoa OpenGL-rajapintaa paremmin. Jos näytönohjaimen suorituskyvystä muodostuu pullonkaula, ei ole mitään hyötyä, että näytönohjaimelle voidaan kirjoittaa komentoja nopeammin prosessorin puolella.

Jos tiedetään, että kehitettävää sovellusta tullaan suorittamaan ympäristössä, jossa näytönohjaimen teho on rajallinen, on järkevää valita OpenGL-rajapinta Vulkanin sijaan. OpenGL voi jopa suoriutua Vulkania paremmin, jos näytönohjaimen laskentateho loppuu kesken (taulukko 1), ja se on Vulkania merkittävästi yksinkertaisempi rajapinta käyttää.

Taulukko 1. Suorituskykytestin tulokset. Testissä Vulkan- ja OpenGL-rajapinnat yhdellä säikeellä sekä Vulkan monisäikeistettynä. Yksikkö on kuvia sekunnissa. Suluissa on suhteellinen ero ylimpään riviin verrattuna. (8).

	252 000 verteksiä	**5,03 miljoonaa verteksiä**
Vulkan 1 säie	695	132
Vulkan 4 säiettä	1176 (+69 %)	143 (+8 %)
OpenGL 1 säie	772 (+11 %)	182 (+38 %)

Korkea resoluutio lisää näytönohjaimen työmäärää prosessoriin verrattuna. PC Gamer -julkaisun Doom-pelin suorituskykytestissä (taulukko 2) nähdään, että Vulkan- ja OpenGL -rajapinnoilla ei ole tällaisessa tilanteessa juurikaan eroja.

Taulukko 2. www.pcgamer.com Doom-pelin suorituskykytestin tulokset. Testi on ajettu eri näytönohjaimilla ja resoluutioilla. Yksikkö on kuvia sekunnissa. Suluissa Vulkan- ja OpenGL -rajapintojen suhteellinen ero (9).

	1920 x 1080	**2560 x 1440**	**3840 x 2160**
GTX 1080 OpenGL	162	125	63
GTX 1080 Vulkan	193 (+19 %)	136 (+9 %)	63 (+0 %)
GTX 1070 OpenGL	147	102	51
GTX 1070 Vulkan	161 (+10 %)	101 (-1 %)	50 (-2 %)

5 Yhteenveto

Vulkan on nykyaikainen matalan tason grafiikkaohjelmointirajapinta. Se pyrkii minimoimaan ajurit siirtämällä niiden toimintaa sovelluspuolelle. Vulkanin edut muihin rajapintoihin verrattuna ovat alustariippumattomuus ja tuki komentopuskurien säikeistykselle. Vulkan on kuitenkin vaikeampi rajapinta hallita ja käyttää kuin esimerkiksi OpenGL, ja se vaatii ohjelmoijalta syvällistä perehtymistä. Vulkan antaa abstraktion laitetason ominaisuuksiin, mikä laajentaa sen mahdollisia käyttötarkoituksia.

Vulkan voi parantaa suorituskykyä, jos prosessori ei pysty tarjoamaan näytönohjaimelle tarpeeksi komentoja tämän koko suorituskyvyn käyttämiseksi. Jos näytönohjain on sovelluksen pullonkaulana, Vulkan ei ole tehokkaampi muihin rajapintoihin verrattuna ja voi jopa heikentää suorituskykyä sen monimutkaisuuden takia.

Lähteet

1 Khronos Releases Vulkan 1.0 Specification. 2016. Verkkoaineisto. The Khronos Group. <https://www.khronos.org/news/press/khronos-releases-vulkan-1-0-specification>. 16.2.2016. Luettu 19.11.2020.

2 Khronos Group Releases Vulkan 1.2. 2020. Verkkoaineisto. The Khronos Group. <https://www.khronos.org/news/press/khronos-group-releases-vulkan-1.2>. 15.1.2020. Luettu 19.11.2020.

3 Vulkan-Guide. 2020. Verkkoaineisto. The Khronos Group. <https://github.com/KhronosGroup/Vulkan-Guide>. Päivitetty 3.11.2020. Luettu 19.11.2020.

4 Khronos Group Releases Vulkan Ray Tracing. 2020. Verkkoaineisto. The Khronos Group. <https://www.khronos.org/news/press/khronos-group-releases-vulkan-ray-tracing>. 17.3.2020. Luettu. 4.11.2020.

5 Vulkan 1.2 spesifikaatio. Verkkoaineisto. 2020. The Khronos Group. <https://www.khronos.org/registry/vulkan/specs/1.2/html/index.html>. Päivitetty 2.11.2020. Luettu 19.11.2020

6 Kessenich, John. 2015. An introduction to SPIR-V. Verkkoaineisto, LunarG. <https://www.khronos.org/registry/spir-v/papers/WhitePaper.pdf>. Luettu 19.11.2020.

7 Vulkan 1.2 manuaali. Verkkoaineisto. The Khronos Group. <https://www.khronos.org/registry/vulkan/specs/1.2-extensions/man/html/>. Luettu 19.11.2020.

8 Blackert, Axel. 2016. Evaluation of Multi-Threading in Vulkan. Opinnäytetyö. Linköping University, Department of Electrical Engineering, Information Coding. Digitala Vetenskapliga Arkivet -tietokanta.

9 Walton, Jarred. 2016. Doom benchmarks return: Vulkan vs. OpenGL. Verkkoaineisto. PC Gamer. <https://www.pcgamer.com/doom-benchmarks-return-vulkan-vs-opengl/>. 21.7.2016. Luettu 4.11.2020.