Referat API

**OpenGL**

OpenGL (Open Graphics Library) este o specificație a unui standard care definește un API (Application Programming Interface) multiplatformă foarte utilizat pentru programarea componentelor grafice 2D și 3D ale programelor de calculator. Interfața constă în peste 250 de apeluri diferite care folosesc la a desena pe ecranul calculatorului scene 3D complexe din primitive (din primitives, elemente simple). OpenGL a fost inițial dezvoltat de compania Silicon Graphics, Inc. (SGI) în 1992 și este foarte utilizat în grafică asistată de calculator, realitate virtuală, vizualizare științifică, simulări de zboruri sau jocuri pe calculator. Acest ultim domeniu este în strânsă competiție cu tehnologia DirectX de la Microsoft (compară OpenGL cu Direct3D). Proiectul OpenGL este condus de compania Khronos Group, un consorțiu tehnologic non-profit.

OpenGL are două scopuri principale:

* să mascheze (ascundă) complexitatea interfețelor cu diferite acceleratoare 3D, prin confruntarea programatorului cu un singur API uniform.
* să mascheze capabilitățile diferitelor platforme hardware, prin cerința ca toate implementările să accepte OpenGL ca un set complet (cu ajutorul emulării de software, dacă este necesar).

Avantaje:

* este o tehnologie foarte portabilă, față de DirectX și te lasă să îți păstrezi opțiunile făcute în cazul în care dorești să rulezi codul pe o altă platformă
* se adapteză ușor la noi componente hardware, de exumplu la plăcile video (GPU)
* poate să fie accesat folosind diverse limbaje de programare
* oferă independență față de protocoalele de rețea și de topologii

Dezavantaje:

* documentația este destul de greu de înțeles pentru cineva care nu este în domeniu și este destul de stufoasă
* implimentarea poate să difere mult din punct de vedere al cerințelor care trebuie să fie îndeplinite
* unele funcții din versiunile mai vechi nu mai sunt compatibile cu noile versiuni, deoarece există noi biblioteci care le înlocuiesc
* unele platforme s-ar putea să fie incompatibile cu noile versiuni ale OpenGL

**Vulkan**

Vulkan este API low-over, cross-platfor, standard deschis pentru grafică 3D și computing. Vulkan vizează aplicații 3D în timp real de înaltă performanță, cum ar fi jocurile video și mass-media interactică. În comparație cu OpenGL, Direct3D 11 și Metal, Vulkan este destinat să ofere performanțe mai ridicate și o utilizare mai echilibrată a procesorului și GPU-ului și oferă un API considerabil de nivel inferior și sarcini paralele pentru aplicație. Pe lângă utilizarea redusă a procesorului, Vulkan este conceput pentru a permite dezvoltatorilor să distribuie mai bine munca între mai multe nuclee de procesor. Vulkan a fost anunțat pentru prima dată de grupul non-profit Khronos la GDC 2015. API-ul Vulkan a fost inițial denumit „inițiativa OpenGL de următoarea generație” sau „OpenGL next” de către Khronos, însă utilizarea acestor nume a fost întreruptă când a fost anunțat Vulkan. Vulkan este derivat din și construit pe componente ale API-ului Mantle al AMD, care a fost donat de AMD lui Khronos cu intenția de a oferi lui Khronos o bază pe care să înceapă dezvoltarea unui API de nivel scăzut pe care să-l poată standardiza în întreaga industrie. Vulkan nu este compatibil înapoi cu OpenGL.Vulkan este destinat să ofere o varietate de avantaje față de alte API-uri, precum și predecesorul său, OpenGL. Vulkan oferă un cost mai redus, un control mai direct asupra GPU-ului și o utilizare mai redusă a procesorului. Conceptul general și setul de caracteristici ale Vulkan sunt similare cu Mantle adoptat ulterior de Microsoft cu Direct3D 12 și Apple cu Metal.

Avantaje:

* este mai bun ca OpenGL pentru platformele moderne precum cele mobile
* aplicația are control direct asupra GPU
* poate să fie accesat utilizând mai multe limbaje de programare
* este gândit pentru platforme multi-core

Dezavantaje:

* poate să nu fie compatibil cu hardware mai vechi
* pune multă responsabilitate pe aplicație precum alocarea memorie, sincronizarea CPU-GPU
* nu este destul de utilizat ca alte API (OpenGL, DirectX)

**Direct****X**

Microsoft DirectX este o colecție de API-uri pentru controlul funcțiilor multimedia, în special pentru programarea jocurilor video, pe platformele Microsoft. Inițial, numele acestor API-uri au început cu Direct, cum ar fi Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound și așa mai departe. Numele DirectX a fost inventat ca termen scurt pentru toate aceste API-uri (X-ul a fost folosit pentru numele anumitor API-uri) și a devenit în curând numele întregii colecții. Când Microsoft a stabilit mai târziu să dezvolte o consolă de jocuri, X constituit baza numelui consolei Xbox astfel indicându-se faptul că consola era bazată pe tehnologia DirectX.X inițial a fost folosit in continuare în denumirea API-urilor proiectate pentru Xbox, cum ar fi XInput și XACT, iar modelul DirectX a fost continuat pentru API-urile Windows, cum ar fi Direct2D și DirectWrite.

Direct3D (graficul 3D API din cadrul DirectX) este utilizat pe scară largă în dezvoltarea jocurilor video pentru Microsoft Windows și a gamei de console Xbox. Direct3D este, de asemenea, utilizat de alte aplicații software pentru vizualizare și sarcini grafice, cum ar fi ingineria CAD/CAM. Deoarece Direct3D este cea mai mediatizată componentă a DirectX-ului, este comun să se vadă numele "DirectX" și "Direct3D" folosite interschimbabil.

Kitul de dezvoltare software DirectX (SDK) este alcătuit din biblioteci runtime în formă binară redistribuitabilă, împreună cu documentația și anteturile care o însoțesc pentru utilizare în codificare. Inițial, runtime-urile au fost instalate numai de jocuri sau explicit de către utilizator. Windows 95 nu a lansat cu DirectX, dar DirectX a fost inclus cu Windows 95 OEM Service Release 2. Windows 98 și Windows NT 4.0 ambele au fost livrate împreună cu DirectX, la fel ca orice versiune de Windows lansată de atunci. SDK-ul este disponibil ca descărcare gratuită. În timp ce runtime-urile sunt proprietate, software-ul cu sursă închisă, codul sursă este furnizat pentru majoritatea mostrelor SDK. Începând cu lansarea Previzualizării pentru dezvoltatori Windows 8, SDK-ul DirectX a fost integrat în SDK-ul Windows.

DirectX oferă numeroase API-uri bune, deci are o gamă largă de avantaje față de mai multe domenii și sectoare și permite să vadă acele puncte plus ale acestuia, care sunt după cum urmează:

* Cu API-ul grafic Direct2D putem realiza un nivel ridicat de proiectare 2D și putem utiliza instrumentele sale ușor de manevrat pentru a obține rezultatul dorit în proiectul nostru nu numai acest lucru, ci putem face redare 3D în timp real cu Direct3D. Putem face, de asemenea, calcul general pentru unitățile de procesare grafică utilizând API-ul DirectCompute, care este un plus al DirectX și ne oferă soluții pentru multe probleme ale unității de procesare grafică.
* Pentru editarea și manipularea piesei audio sau a fișierului audio, putem merge atât la editare, fie că este vorba de un nivel ridicat sau de un nivel scăzut al acestuia. Pentru nivel înalt putem folosi cu XACT3 și pentru nivel scăzut puteți utiliza cu API-urile XAudio2 din DirectX.
* Urmărirea razelor în timp real este posibilă și cu API-ul DirectX Raytracing (DXR). Cu ceva timp ne confruntăm cu probleme în manipularea dispozitivelor de intrare cu sistemul nostru, astfel încât cu DirectInput API putem crea interfațarea diferitelor dispozitive de intrare, cum ar fi tastaturi, controlere de joc, mouse, joystick și așa mai departe.

Dezavantaje:

* În timpul proiectării imaginii de înaltă rezoluție, ne putem confrunta cu unele dificultăți, deoarece creează probleme în fidelitatea vizuală, iar spațiile dintre linii nu sunt aranjate corespunzător din cauza lipsei unor caracteristici.
* S-ar putea să se încălzească panourile de pe sisteme, deoarece plăcile grafice concepute pentru diferite API-uri nu funcționează în timp util cu unele speciale.
* Există o problemă când dorim să dezinstalăm DirectX, deoarece mediul Windows nu permite utilizatorului să îl dezinstaleze de pe computerul său, dar Microsoft a planificat să rezolve această problemă, care permite utilizatorului să facă acest lucru fără a-și pierde datele și viteza computerului.

Cum explicați modelul de automat cu stări finite al OpenGL și cum afectează acest lucru procesul de randare al scenei 3D de către biblioteca grafică/API?

Indiferent de ce realizezi în OpenGL acel lucru depinde de o funcție.De exemplu daca vrem să adaugăm textură avem nevoie de o funcție nouă. Acest lucru îi complică viața programatorului, deoarece trebuie să adauge funcții pentru fiecare lucru pe care dorește să îl implementeze. Acest lucru poate să devină greu de gestionat pe viitor dacă vorbim despre un program.