OpenGL

OpenGL (Open Graphics Library) este cel mai larg adoptat API grafic 2D și 3D din industrie, aducând mii de aplicații pe o mare varietate de platforme de calculatoare. Este independentă de sistemul de ferestre și de sistemul de operare, precum și transparentă față de rețea. Această interfață constă în aproximativ 150 de comenzi distincte pe care le utilizați pentru a specifica obiectele și operațiile necesare pentru a produce aplicații tridimensionale interactive.

OpenGL a fost creat pentru prima dată ca o alternativă deschisă și reproductibilă la Iris GL, care fusese API-ul grafic proprietar pe stațiile de lucru Silicon Graphics. Deși OpenGL a fost inițial similar în unele privințe cu IrisGL, lipsa unei specificații oficiale și a testelor de conformitate au făcut ca Iris GL să nu fie potrivit pentru o adoptare mai largă. Mark Segal și Kurt Akeley au fost autorii specificației OpenGL 1.0 care au încercat să oficializeze definiția unui API grafic util și a făcut ca implementarea și suportul terță parte non-SGI să fie viabile. O omisiune notabilă din versiunea 1.0 a API-ului a fost obiectele de textură. IrisGL a avut etape de definire și legare pentru tot felul de obiecte, inclusiv materiale, lumini, texturi și medii de textură. OpenGL a evitat aceste obiecte în favoarea schimbărilor incrementale de stare cu ideea că modificările colective ar putea fi încapsulate în listele de afișare. Aceasta a rămas filozofia, cu excepția faptului că obiectele de textură (glBindTexture) fără o etapă de definire distinctă sunt o parte cheie a API-ului.

OpenGL este concepută ca o interfață simplificată, independentă de hardware, care poate fi implementată pe numeroase platforme hardware diferite. Pentru a obține aceste calități, nu există comenzi pentru realizarea ferestrelor sarcini de fereastră sau de obținere a intrărilor utilizatorului sunt incluse în OpenGL; în schimb, trebuie să lucrați prin intermediul oricărui sistem de fereastră care controlează hardware-ul specific pe care îl utilizați. În mod similar, OpenGL nu oferă comenzi de nivel înalt pentru descrierea modelelor de obiecte tridimensionale. Astfel de comenzi v-ar putea permite să specificați forme relativ complicate, cum ar fi automobile, părți ale corpului, avioane sau molecule. Cu OpenGL, trebuie să construiți modelul dorit dintr-un mic set de primitive geometrice - puncte, linii și poligoane.

O bibliotecă sofisticată care să ofere aceste caracteristici ar putea fi cu siguranță construită pe OpenGL. Biblioteca OpenGL Utility Library (GLU) oferă multe dintre caracteristicile de modelare, cum ar fi suprafețele cuadrice și curbele și suprafețele NURBS. GLU este o parte standard a fiecărei implementări OpenGL. De asemenea, există un set de instrumente de nivel superior, orientat pe obiecte, Open Inventor, care este construit deasupra OpenGL și este disponibil separat pentru multe implementări ale OpenGL. (Pentru mai multe informații, consultați "OpenGL-Related Libraries". informații despre Open Inventor).

API-ul OpenGL este conceput pentru a aborda o gamă largă de tehnici avansate de redare grafică, cum ar fi cartografierea texturilor (capacitatea de a aplica o imagine pe o suprafață grafică), anti-aliasing, transparență, ceață, iluminare (capacitatea de a calcula

coloraţia suprafeţei atunci când se aplică diferite modele de iluminare pe suprafaţă de la una sau mai multe surse de lumină), umbrirea netedă (capacitatea de a calcula efectele de umbrire atunci când lumina loveşte o suprafaţă sub un anumit unghi, ceea ce determină diferenţe subtile de culoare pe suprafaţă), estomparea mişcării şi transformarea modelării (capacitatea de a modifica locaţia, dimensiunea şi perspectiva unui obiect în spaţiul de coordonate 3D).

Avantajele utilizării OpenGL:

- OpenGL este un standard industrial ghidat de un consorţiu independent, OpenGL Architecture Review Board, care supraveghează specificaţia OpenGL. Acesta face ca OpenGL să fie singurul standard grafic cu adevărat deschis, neutru din punct de vedere al furnizorului şi multiplatformă.
- OpenGL este stabil de mai bine de şapte ani pe o mare varietate de platforme. Adăugările la specificaţie sunt bine controlate, iar actualizările propuse sunt anunţate la timp pentru ca dezvoltatorii să adopte modificările. Cerinţele de compatibilitate retroactivă asigură faptul că aplicaţiile existente nu devin învechite.
- OpenGL oferă scalabilitate, cu aplicaţii care pot fi executate pe sisteme de la electronice de consum la PC-uri, staţii de lucru şi supercomputere. Prin urmare, aplicaţiile pot fi adaptate la orice clasă de maşini pe care dezvoltatorul alege să le vizeze.

Un dezavantaj al OpenGL este faptul că are o comunitate mai mică decât DirectX, astfel încât există mai puţin suport disponibil. În plus, OpenGL nu este la fel de bine adaptat pentru jocurile 3D ca DirectX.

Un program OpenGL este alcătuit din două shaders:

- Shaderul de vertex este executat (de obicei) o dată pentru fiecare vertex pe care dorim să îl desenăm. Acesta primeşte câteva atribute ca intrare, calculează poziţia acestui vertex în spaţiu şi o returnează într-o variabilă numită gl_Position. De asemenea, defineşte unele variaţii.
- Shaderul de fragmente este executat o singură dată pentru fiecare pixel care urmează să fie redat. Acesta primeşte ca intrare anumite variaţii, calculează culoarea acestui pixel şi o returnează într-o variabilă numită gl_FragColor.

Microsoft DirectX este o colecţie de interfeţe de programare a aplicaţiilor (API) pentru gestionarea sarcinilor legate de multimedia, în special programarea de jocuri şi video, pe platformele Microsoft. Iniţial, numele acestor API-uri începeau toate cu "Direct", cum ar fi Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound şi aşa mai departe. Numele DirectX a fost inventat ca un termen prescurtat pentru toate aceste API-uri (X reprezentând numele API-urilor) şi a devenit în curând numele colecţiei. Mai târziu, când Microsoft a intenţionat să dezvolte o consolă de jocuri, X a fost folosit ca bază a numelui Xbox pentru a indica faptul că consola se baza pe tehnologia DirectX. Iniţialitatea X a fost continuată în denumirea API-urilor concepute pentru Xbox, cum ar fi XInput şi Cross-platform Audio

Creation Tool (XACT), în timp ce modelul DirectX a fost continuat pentru API-urile Windows, cum ar fi Direct2D și DirectWrite.

Competiția dintre OpenGL și DirectX este probabil la fel de cunoscută ca și războaiele purtate între entuziaștii AMD și Intel. Acest subiect a aprins focul multor războaie de foc de-a lungul anilor și nu anticipez că acest lucru se va schimba prea curând.

Poate că cea mai evidentă diferență este că DirectX, spre deosebire de OpenGL, este mai mult decât un simplu API grafic. DirectX suportă sunet, muzică, intrare, rețele și multimedia. Pe de altă parte, OpenGL este strict un API grafic.

O diferență majoră este că OpenGL este cross-platform, iar DirectX este disponibil doar pe Windows și XBox. Dacă aveți nevoie să dezvoltați pentru mai mult decât Windows, OpenGL este calea de urmat.

Când vine vorba de capacitățile grafice, ambele API-uri se bazează pe utilizarea pipeline-ului grafic tradițional. Aceasta este aceeași conductă care a fost utilizată în jocurile pe calculator încă din primele zile ale graficii pe calculator.

Atât OpenGL, cât și DirectX descriu verticele ca fiind un set de date constând în coordonate în spațiu care definesc locația vertexului și orice alte date legate de vertex. Primitivele grafice, cum ar fi punctele, liniile și triunghiurile, sunt definite ca un set ordonat de vârfuri. Există diferențe în modul în care fiecare API gestionează modul în care sunt combinate vârfurile pentru a forma primitive, diferențe care sunt prezentate mai jos.

Vulkan este un standard deschis pentru grafică și calcul 3D, cu costuri reduse, cu API pentru mai multe platforme, Vulkan vizează aplicațiile grafice 3D de înaltă performanță în timp real, cum ar fi jocurile video și mediile interactive. Vulkan este menit să ofere performanțe mai ridicate și o utilizare mai eficientă a CPU și GPU în comparație cu API-urile mai vechi OpenGL și Direct3D 11. Oferă un API de nivel considerabil mai scăzut pentru aplicație decât API-urile mai vechi, ceea ce face ca Vulkan să fie comparabil cu Metal API de la Apple și Direct3D 12 de la Microsoft. În plus față de utilizarea mai redusă a CPU, Vulkan este conceput pentru a permite dezvoltatorilor să distribuie mai bine munca între mai multe nuclee CPU.

Atât OpenGL, cât și Vulkan sunt API-uri open source pentru platforme încrucișate, ceea ce înseamnă că ambele sunt gratuite pentru a le utiliza cu o mulțime de caracteristici bune și nu trebuie să vă faceți griji cu privire la niciun abonament sau plan pentru a le utiliza. Ambele API-uri au fost dezvoltate de același dezvoltator, iar Vulkan a fost introdus în 2015 ca API neprofitabil de către grupul Khronos la GDC și este denumit inițial "next generation OpenGL initiative" sau "OpenGL next", dar ulterior a fost schimbat în Vulkan.

Vulkan oferă o serie de avantaje, deoarece oferă un control direct puternic asupra GPU, reduce utilizarea CPU, precum și presiunea mai mică asupra acesteia. Pe scurt, conceptul general și caracteristicile sale sunt similare cu Mantle, care a fost adoptat și de Direct3D de la Microsoft Windows și Metal de la Apple. OpenGL generează tampoane de comenzi pentru mai multe fire de execuție și procesare simultană pentru același lucru la o conductă de

comenzi și, din acest motiv, dezvoltatorul nu trebuie să lucreze la întreținerea cadrului sau, dacă dorește să o facă, o poate face cu puțin efort.

OpenGL își creează propriul compilator pentru GLSL, care este limbajul de nivel înalt al acestuia, iar acest limbaj scrie shaders, ceea ce obligă driverul OpenGL să implementeze propriul compilator pentru acest limbaj și să execute aplicația în timp de execuție pentru a traduce shaders din program în codul mașinii GPU. În timp ce Vulkan dispune de shaders care au fost deja traduse în format binar intermediar și care se numesc SPIR-V (standard portable intermediate representation).

Cu Vulkan aveți o integrare mai bună a instrumentelor în comparație cu OpenGL, deoarece puteți activa validarea și diagnosticarea straturilor în mod independent. Datorită faptului că nu există o diferență de API-uri între versiunile pentru mobil și PC ale celor două programe, portarea jocurilor între aceste platforme încrucișate este ușoară.

În viitor, în OpenGL vor fi adăugate noi extensii, la fel ca în Vulkan, și anume NV_command_list, care este aceeași cu "GPU feeding paradigm" din Vulkan și care va îmbunătăți capacitatea de joc a OpenGL. Deoarece Vulkan a fost introdus ca următoarea generație de API pentru OpenGL, veți avea mai multe funcții și tehnici îmbunătățite pentru a realiza diferite tipuri de sarcini legate de domeniul său, pentru a obține cele mai bune rezultate în activitatea de proiect. Astfel, puteți explora cu ușurință funcțiile Vulkan față de OpenGL.

Un sistem eterogen este un sistem care utilizează mai multe procesoare sau nuclee pentru a îmbunătăți performanța. Procesoarele pot fi similare sau diferite, în funcție de sarcină. OpenCL este un cadru care ajută la scrierea de programe pentru sisteme eterogene. Prin urmare, programatorul poate utiliza OpenCL pentru a scrie programe pentru sisteme cu mai multe unități centrale de procesare, GPU, procesoare de semnal digital (DSP), rețele de porți programabile pe câmp (FPGA) etc. În plus, permite efectuarea de calcule paralele folosind paralelismul bazat pe sarcini și date.

Kernelul este o funcție care este executată pe un dispozitiv OpenCL. OpenCL definește o interfață de programare a aplicațiilor (API) pentru a permite programelor care rulează pe gazdă să lanseze kernel-uri pe dispozitive de calcul și să gestioneze memoria dispozitivului. În plus, oferă un limbaj similar cu C pentru a scrie programe. Dispune de API-uri pentru C, C++ și alte limbaje și tehnologii, cum ar fi Python, Java, Perl și NET etc.

Principala diferență între OpenGL și OpenCL este că OpenGL este utilizat pentru programarea grafică, în timp ce OpenCL este utilizat pentru calculul eterogen.

WebGL este abreviat ca Web Graphics Library (Biblioteca grafică web). Este concepută în principal pentru redarea graficii bidimensionale și a graficii tridimensionale interactive. Este un API Javascript care poate fi utilizat cu HTML5. Suportă platforme încrucișate și este disponibilă doar în limba engleză. Programele WebGL constau dintr-un cod de control care este scris în JavaScript. OpenGL se numește Open Graphics Library (Biblioteca grafică deschisă). Este menționată ca o interfață de programare a aplicațiilor între limbaje și

platforme pentru redarea graficii vectoriale bidimensionale și tridimensionale. OpenGL oferă multe funcționalități, cum ar fi extensiile.

Diferențele dintre OpenGL și WebGL sunt:

- WebGL se bazează pe OpenGL ES, căruia îi lipsesc multe dintre caracteristicile pe care le are OpenGL obișnuit, cum ar fi faptul că suportă doar shaderi de vertex și de fragmente. OpenGL dispune de caracteristici care nu se regăsesc în WebGL, cum ar fi shadere de geometrie, shadere de teselare și shadere de calcul.
- WebGL este utilizat în principal pentru browsere. OpenGL are nevoie de drivere native și este orientat în principal către instalarea de software.
- WebGL este utilizat pentru aplicații web, iar OpenGL este utilizat pentru multe jocuri video.
- WebGL este mai ușor de învățat și de dezvoltat aplicații. OpenGL poate fi învățat cu ușurință dacă odată ce sunteți familiarizat cu WebGL.
- În WebGL, se poate falsifica textura 3D cu ajutorul utilizării texturii 2D. În OpenGL, nu este necesar să se facă acest lucru, deoarece dispune de o mulțime de caracteristici, cum ar fi geometria și shaderii.
- În WebGL, este forțat să învețe să utilizeze de la început shaders și buffere. În OpenGL, nu este așa.
- WebGL are o curbă de învățare mai mică, deoarece are mai puține caracteristici. OpenGL are o curbă de învățare mai mare, deoarece are o mulțime de caracteristici, inclusiv WebGL.
- WebGL se bazează pe OpenGL ES 2, care nu este OpenGL pur și simplu. OpenGL ES este un subset al OpenGL. OpenGL ES are mai puține capabilități și este foarte simplu pentru un utilizator. OpenGL are o mulțime de capabilități și este dificil de utilizat.

Contextul OpenGL conține informații utilizate de sistemul de redare. Aceste informații se numesc stări, ceea ce a dat naștere la afirmația că OpenGL este o "mașină de stări finite". O bucată de stare este pur și simplu o valoare stocată în contextul OpenGL.

O mașină cu stări finite (FSM) sau un automat cu stări finite (FSA, plural: automate), un automat finit sau pur și simplu o mașină cu stări, este un model matematic de calcul. Este o mașină abstractă care se poate afla în exact una dintre un număr finit de stări la un moment dat. FSM poate trece de la o stare la alta ca răspuns la anumite intrări; trecerea de la o stare la alta se numește tranziție. Un FSM este definit de o listă a stărilor sale, de starea inițială și de intrările care declanșează fiecare tranziție. Mașinile cu stare finită sunt de două tipuri - mașini cu stare finită deterministe și mașini cu stare finită nedeterministe. O mașină cu stare finită deterministă poate fi construită în mod echivalent cu orice mașină nedeterministă.

Fiecare bucată individuală de stare din contextul OpenGL poate fi identificată în mod unic. Documentul de specificații OpenGL are un tabel masiv la sfârșit, care definește care sunt acești identificatori și care este starea pe care pot fi utilizați pentru a fi interogați. Nu toți

identificatorii sunt la fel de simpli ca o enumerare, deoarece starea OpenGL poate fi în matrici, unde fiecare intrare trebuie interogată individual.

Atunci când se creează un context, fiecare element de stare este inițializat la o valoare implicită bine definită. Tabelul de stare definește care este valoarea inițială pentru fiecare element de stare OpenGL. Reținem că acest lucru nu înseamnă că un context proaspăt creat poate fi utilizat pentru randare fără o anumită configurare.

Funcțiile OpenGL pot fi grupate în 3 mari categorii: funcții care stabilesc starea în context, funcții care interoghează starea și funcții care redau având în vedere starea curentă a contextului. Funcțiile care interoghează starea utilizează identificatorul unic pentru a denumi valoarea de stare pe care o recuperează. Tabelul de stare din specificații indică ce funcții sunt utilizate pentru a interoga fiecare valoare de stare.

OpenGL nu se mai află în dezvoltare activă: în timp ce între 2001 și 2014 specificația OpenGL a fost actualizată în mare parte anual, două versiuni (3.1 și 3.2) având loc în 2009 și trei (3.3, 4.0 și 4.1) în 2010, cea mai recentă specificație OpenGL 4.6 a fost lansată în 2017, după o pauză de trei ani, și s-a limitat la includerea a unsprezece extensii ARB și EXT existente în profilul de bază.

Dezvoltarea activă a OpenGL a fost abandonată în favoarea API Vulkan, lansată în 2016 și cu numele de cod glNext în timpul dezvoltării inițiale. În 2017, Khronos Group a anunțat că OpenGL ES nu va avea versiuni noi și de atunci s-a concentrat pe dezvoltarea Vulkan și a altor tehnologii. Ca urmare, anumite capabilități oferite de GPU-urile moderne, de exemplu ray tracing, nu sunt suportate de OpenGL.