10.19279/TVZ.PD.2016-4-2-03

RAZLIKE IZMEĐU OPENGL I DIRECT3D PROGRAMSKIH SUČELJA

DIFFERENCES BETWEEN THE OPENGL AND DIRECT3D PROGRAMMING INTERFACES

Andrija Bernik[1], Vedran Bergovec[1], Zvonimir Sabati[2]

[1]Sveučilište Sjever – Sveučilišni centar Varaždin
[2]Fakultet organizacije i informatike

Sažetak

U ovom radu prikazane su i objašnjene razlike između dva najpopularnija grafička programska sučelja počevši od njihovih izoliranih početaka 1992. godine, preko njihovih čestih srazova u područjima performansi, jednostavnosti korištenja, pokušaja upadanja u ciljane skupine onog drugog i neuspjelog pokušaja spajanja u jedan proizvod. Ovaj rad također se bavi raznim tehnikama i problemima koje spomenuti API-evi moraju savladati i nadvladati da mi se izdigli iznad konkurencije. Na kraju rada prikazana je raspodjela spomenutih API-eva prema određenim korisničkim skupinama i objašnjeno je zašto su baš te skupine odabrale određeni API u svom radu.

Ključne riječi: *OpenGL, Direct3D, API, Marshalling, Driver Overhead*

Abstract

This paper presents and explains the differences between two of the most popular graphic Application Programming Interfaces (API-s) starting from their isolated inceptions in 1992, through their frequent competitive collisions in the area of performance, ease of use, attempts to penetrate the other's target group of users and a failed attempt to merge the API-s into a single product. This paper also deals with various techniques they must master and obstacles they must overcome to rise above the competition. The end of this paper explains which user types prefer which API and explains why that is the case.

Keywords: *OpenGL, Direct3D, API, Marshalling, Driver Overhead*

1. Uvod

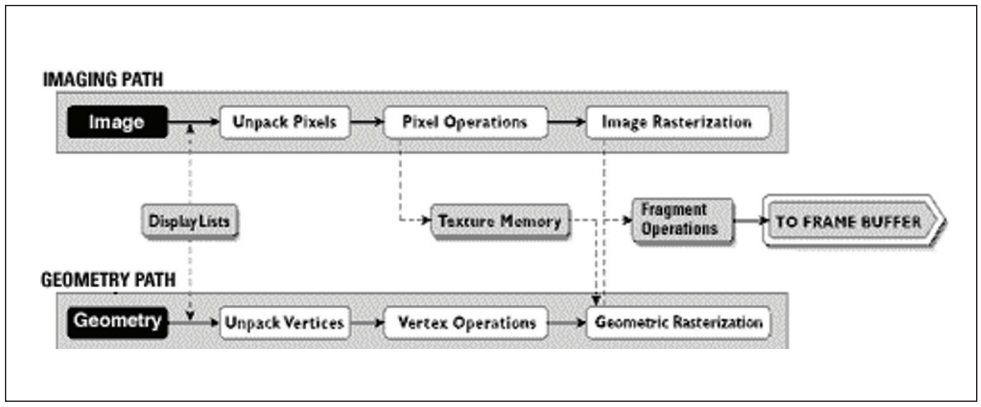
1. *Introduction*

Direct3D i OpenGL su konkurentska programska sučelja (API) koja se mogu koristiti u aplikacijama za prikazivanje 2D i 3D računalne grafike. Hardverska akceleracija tog procesa je učestalo koristi od otprilike 1999. Od 2005. grafičke procesorske jedinice (GPU) gotovo uvijek koriste oba API-a. Direct3D aplikacije se najčešće koriste na Microsoft Windows platformi dok je OpenGL otvoreni standard i koristi se na širokom spektru platformi. Glavna razlika je da OpenGL procesuira 2D i 3D grafiku dok DIrect3D samo 3D grafiku, no postoje i druge, manje očite razlike. Direct3D je API koji posjeduje Microsoft te koristi hardversku akceleraciju GPU-a ukoliko grafička kartica posjeduje tu mogućnost. Dizajnirao ga je Microsoft za korištenje na Windows operativnom sustavu ali se može koristiti na drugim operativnim sustavim korištenje Win32 API sloja poput Wine-a ali uz manji set mogućnosti.
OpenGL je otvoreni standard API-a koji se koristi na većini suvremenih operativnih sustava poput Windowsa, Mac OS X-a i Linuxa

2. OpenGL

2. *OpenGL*

OpenGL je primarno okruženje za razvoj prijenosnih i interaktivnih 2D i 3D grafičkih aplikacija. Od kada je uveden 1992[1]., OpenGL je postao najrašireniji API za razvijanje 2D i 3D grafičkih aplikacija, dovodeći tisuće aplikacija na širok spektar računalnih platformi. OpenGL



Slika 1
Shema slijeda programiranja OpenGL-a [1]

Figure 1
OpenGL programming scheme [1]

potiče inovacije i ubrzava razvoj aplikacija koristeći veliku količinu funkcija za renderiranje, mapiranje tekstura, specijalne efekte i ostale vizualizacijske funkcije.

2.1. Dostupnost

2.1. *Availability*

OpenGL je dostupan na svim UNIX radnim stanicama te se isporučuje sa svim Windowsima i MacOS računalima. Nijedan drugi grafički API ne radi na širem broju hardverskih platformi i softverskih okruženja. OpenGL se poziva iz sljedećih programskih jezika: Ada, C, C++, Fortran, Python, Perl i Java i u njima omogućuje potpunu neovisnost od mrežnih protokola i topologija.

2.2. Upravljanje

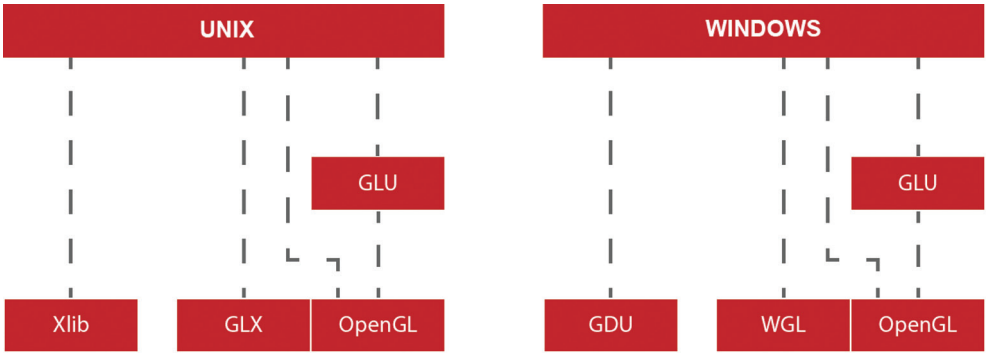
2.2. *Governing*

Odbor za ocjenjivanje OpenGL arhitekture (ARB) je bio neovisni konzorcij osnovan 1992 koji je vodio budućnost OpenGL-a, predlagao i odobravao promjene specifikacija, novih verzija i testiranja kompatibilnosti. 2006. ARB postaje „OpenGL Working Group“ u nadležnosti tvrtke „Khronos Group“, konzorcija za otvorene API-eve. Odbor za karakterizaciju performansi OpenGL-a je nezavisna organizacija osnovana za stvaranje i održavanje OpenGL mjerila performansi te objavljuje rezultate na svojoj web stranici: www.specbench.org/gpc/ opc.static/ index.html.

2.3. Povijest

2.3. *History*

OpenGL je u početku stvoren kao otvorena alternativa Iris GL-u koji bio vlasnički API korišten na radnim stanicama Silikonske Doline[2]. Iako je u početku OpenGL bio sličan Iris GL-u, zbog nedostatka dokumentacije i testova kompatibilnosti Iris GL nije bio prikladan širokoj publici. Mark Segal i Kurt Akeley napisali su specifikacije OpenGL-a 1.0 kojima je cilj bio normalizacija definicije korisnog grafičkog API-a te koje su omogućile cross-platform implementacije i podršku od trećih strana. OpenGL je prošao mnogobrojne revizije koje su najčešće bile inkrementalni dodatci gdje su se ekstenzije API-a postepeno dodavale u njegov temeljni dio.



Slika 2
HIJERARHIJA API-A [1]

Figure 2
API Hierarchy [1]

OpenGL 2.0 uključuje bitan dodatak - OpenGL Shading Language (GLSL), jezik sličan C-u koji omogućuje transformacije i razdjeljivanje stadija sjenčanja.
OpenGL 3.0 dodaje koncept označavanje nepotrebnih funkcija za brisanje u sljedećim verzijama.

3. Direct3D

3. *Direct3D*

Direct3D je grafički API korišten za Microsoft Windows operativne sisteme[3]. Koristi se za iscrtavanje trodimenzionalne grafike u aplikacijama koje zahtijevaju visoke performanse, primjerice u igrama. Direct3D koristi hardversko ubrzavanje ukoliko mu grafička kartica to omogućava. Direct3D otvara mogućnost korištenja naprednih mogućnosti 3D grafičkog hardvera poput Z-buffering, W-buffering, Stencil buffering, spatial anti-aliasing, texture blending, clipping, Culling itd…
Direct3D sadrži mnoge naredbe za programiranje 3D grafike no od verzije 8 preuzima odgovornost za iscrtavanje 2D grafike.

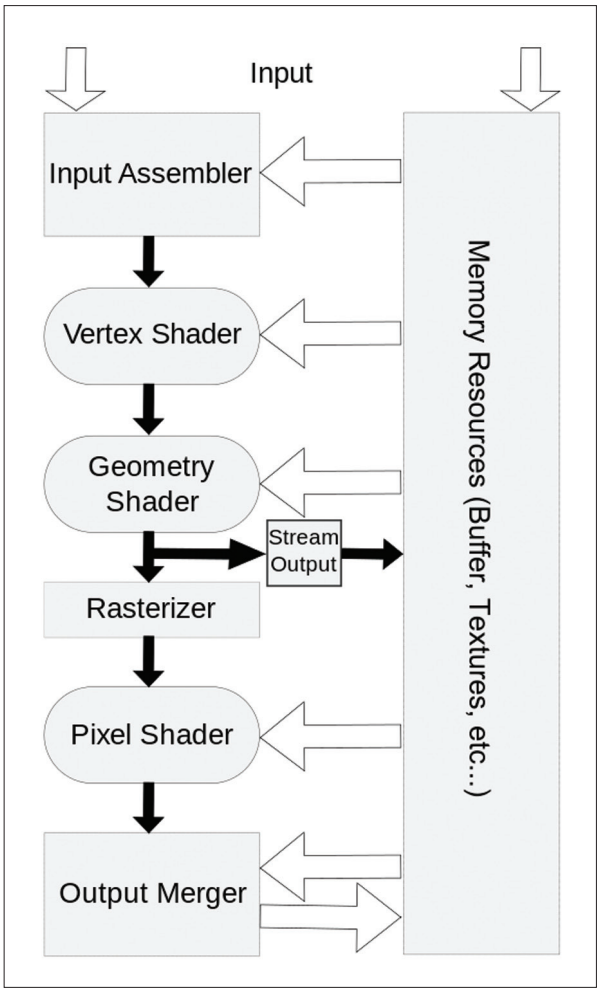
3.1. Početci

3.1. *Inception*

1992, Servan Keondjian osniva tvrtku „Render Morphics“[4] koja razvija 3D grafički API nazvan „Reality Lab“ koji se koristio za medicinske prikaze i CAD software koji je izdan u dvije verzije. Microsoft kupuje Render Morphics 1995, te dovodi Keondjiana s kojim razvija 3D grafički engine za Windows 95 što rezultira u prvoj verziji Direct3D-a koja se isporučuje u DirectX 2.0 i DirectX 3.0.
Prva verzija Direct3D je koristila tzv. „suzdržani“ i „izravan“ mod 3D API-a. Suzdržani mod Direct3D-a je naišao na negodovanje programera koji su tražili veću kontrolu nad grafičkim hardverom te su se zbog toga samo dvije igre temeljene na suzdržanom modu Direct3D prodale u značajnom broju, „Lego Island“ i „Lego Rock Raiders“ te Microsoft nije nudio suzdržan mod od verzije 3.0. Izravan mod Direct3D-a bio je temeljen na programskom modelu „izvršne međumemorije (buffer)“ za koji se Microsoft nadao da će proizvođači hardvera direktno podržavati. Izvršna međumemorija je bila zamišljena kao rezervirani prostor na hardverskoj memoriji namijenjen razdvajanju od strane hardvera da bi prikazao 3D sliku. Nažalost, bili su vrlo nespretni za programiranje te time unazadili prihvaćanje API-a i poticali pozive na prihvaćanje OpenGL API-a kao službenog za izradu 3D igara i profesionalnih aplikacija. Umjesto toga, Microsoft je odlučio unaprjeđivati Direct3D, ne samo da bi bio konkurentan OpenGL-u nego i ostalim zatvorenim API-ima poput 3dfx-ovog „Glide“.

3.2. Koraci

3.2. *Steps*



Slika 3 *Direct3D proces* [9]

Figure 3 *Direct3D process* [9]

Microsoft Direct3D API definira proces konverzije krupe vertexa, tekstura, međumemorija i njihovih stanja u sliku na ekranu. Taj proces se sastoji od nekoliko koraka, u verziji 11[3] to su:

1. **Input Assembler** – čita podatke o vertexima iz vertex međumemorije i šalje dalje
2. **Vertex Shader** – izvodi operacije poput transformacija i osvjetljenja na pojedinačnim vertexima
3. **Hull Shader** – izvodi operacije na kontrolnim točkama objekta i generira dodatne podatke o njima
4. **Tasselation Stage** – dijeli geometriju u cilju boljeg prikaza objekta
5. **Domain Shader** – izvodi operacija na pojedinačnim vertexima, kao i u drugom koraku
6. **Geometry Shader** – procesuira primitivne oblike poput trokuta, točaka ili linija te određuje da li će ih odbaciti ili iz njih generirat još jedan ili više oblika
7. **Stream Output** – ispisuje rezultate prijašnjih koraka u memoriju zbog čega se podaci mogu vratit natrag u proces
8. **Rasterizer** – pretvara primitivne oblike u piksele te uz to može obavljati druge zadatke poput odbacivanje nevidljivih dijelova objekata ili pretvaranja podataka o vertexima u podatke o pikselima
9. **Pixel Shader** – određuje konačnu boju pixela
10. **Output Merger** – spaja različite vrste prijašnjih podataka i od njih radi konačnu sliku

4. Kompatibilnost

4. *Compatibility*

4.1. Direct3D

4.1. *Direct3D*

Direct3D u vlasništvu Microsofta je službeno dio samo Microsoft Windows operativnih sustava uključujući ugrađene verzije korištene u Xbox igraćim konzolama i Sega Dreamcast-u. Nekoliko verzija Direct3D API-a stvorene su od trećih strana te su većinski funkcionalne, npr. Wine za operativne sustave slične Unix-u te Cedega, vlasnička verzija Wine-a. Ipak, taj proces razvoja takvih API-eva je usporen zbog ovisnosti DirectX-a i drugi dijelova Windows-a te iz razloga što Direct3D zbog svoje zatvorenosti zahtijeva težak obrnuti inženjering.

4.2. OpenGL

4.2. *OpenGL*

OpenGL posjeduje implementacije dostupne na mnogim platformama uključujući Microsoftov Windows, sustavima temeljenim na UNIX-u poput Mac OS X, GNU/Linux. Nintendo i Sony su za svoje konzole razvili vlastite API-eve temeljene na OpenGL-u s kojim dijele mnoge elemente no nisu identični. Izvedenica OpenGL-a je izabrana za temeljni API za Android, Blackberry OS, iOS i Symbian.
Microsoftov driver za OpenGL pruža hardversku akceleraciju za Windows Vistu i novije dok je podrška obustavljena za Windows XP. Akceleracija za OpenGL je postignuta korisničkom instalacijom drivera razvijenih od strane proizvođača GPU-a tako da je instalacija aktualnih drivera dovoljna za OpenGL podršku. Google od nedavno omogućuje direktno pretvaranje OpenGL poziva u DirectX 9 u svrhu kompatibilnosti između proizvođača korištenjem WebGL-a (varijanta OpenGL-a koji radi u browseru).

5. Jednostavnost upotrebe

5. *Ease of use*

5.1. Direct3D

5.1. *Direct3D*

Prva verzija Direct3D-a je 1996. godine izazvala negodovanje developera jer su i najjednostavnije radnje zahtijevale stvaranje i postavljanje objekata zvanih „execute buffer“ dok se jednostavne promjene stanje u OpenGL-u provode jednom funkcijom.
U drugoj verziji, Direct3D 5 (nazvan tako jer je bio dio DirectX-a 5) odbacio je stvaranje execute buffera korištenjem DrawPrimitive API-a, no i dalje se smatrao nezgrapnim za rad. No mnoga negodovanja programera nisu spriječila Microsoft da nastavi razvijati API.
No danas se i najveći kritičari slažu da je

Direct3D dosegao, čak i prestigao OpenGL u području mogućnosti i jednostavnosti korištenja.

5.2. OpenGL

5.2. *OpenGL*

OpenGL je API stvoren za rad s C jezikom iako se može koristiti s drugim programskim jezicima. OpenGL se kroz vrijeme razvio iz primitivnog „state machine[8]“ modela, gdje apstraktan stroj, u ovom slučaju OpenGL, može imati konačan broj stanja, i može biti u samo jednom stanju u danom trenutku, u objektno orijentiran sustav.

5.3. Usporedba

5.3. *Comparison*

Direct3D je zamišljen kao sustav za virtualizaciju 3D hardverskih sustava te time omogućuje programeru da se ne zamara pripremanjem hardvera za svoj rad dok se OpenGL dizajniran kao sustav za iscrtavanje grafike (rendering) ubrzan grafičkim hardverom čije se ubrzanje može emulirati softverom. Ta dva API-a su dizajnirana u potpuno različitim filozofijama.
Zbog toga postoje temeljne razlike u njihovom radu. Direct3D očekuje od aplikacije da barata hardverskim resursima dok se OpenGL bavi njihovim iskorištavanjem. OpenGL zbog toga omogućuje lakši razvoj aplikacija no izradu drivera kompliciranija, ako želimo da driver dobro radi.
Sa radom u Direct3D-u, developer aplikacije mora sam omogućiti optimalno korištenje hardverskih resursa, no razvoj grafičkih drivera je jednostavniji a developer ima mogućnost alociranja hardverskih resursa tamo gdje mu najviše trebaju.
Usprkos tim razlikama, dva API-a omogućuju gotovo identičnu razinu funkcionalnosti. Proizvođači hardvera i softvera generalno vrlo brzo prihvaćaju promjene u DirectX-u, čiji je Direct3D dio dok nove mogućnosti OpenGL-a stvaraju prvenstveno proizvođači hardvera, koje se kasnije dodaju u standardnu inačicu OpenGL-a.

6. Performanse

6. *Performance*

Nedugo nakon predstavljanja oba API-a oko 1995, započeli su tzv. „API ratovi“ u kojima se većinom raspravljalo o tome koji API nudi veće performanse jer su cijene odvojenih grafičkih procesora bile vrlo visoke.

6.1. Rane rasprave

6.1. *Early discussion*

U početku su se aplikacije poput AutoCAD-a i igri poput Quake-a morale optimizirati za rad na različitim grafičkim podsustavima dok proizvođači nisu počeli ugrađivati grafičke ubrzivače na svoje proizvode kompatibilne s OpenGL. Zbog toga su developeri aplikacija masovno počeli prihvaćati OpenGL kao standard no s rođenjem multitasking operativnih sustava potreba za takvim ubrzivačima je nestala.
U to vrijeme je Microsoft prikazivao Direct3D kao brži od konkurencije na temelju usporedbe performansi na svojim sustavima. Krivnju za niske performanse OpenGL-a Microsoft je svalio na rigorozne specifikacije i sukladnost potrebne za OpenGL. Taj je stav promijenjen kada je OpenGL zajednica 1996. razvila softversku implementaciju OpenGL-a za Windows nazvanu CosmoGL koja je u raznim demonstracijama pokazivala jednake ili čak više performanse od Direct3D-a što je značilo da krivnja za loše performanse OpenGL-a na Windowsima nije na OpenGL-u nego na Microsoftovoj implementaciji OpenGL-a u Windows sustav.
No razvoj jeftinog grafičkog hardvera učinio je cijelu raspravu o performansama API-a irelevantnom jer su i najjeftiniji grafički podsustavi omogućavali više performanse od najbržih procesorskih grafičkih implementacija potpomognutim Direct3D i OpenGL API-ima.

6.2. Marshalling

6.2. *Marshalling*

Korisničke aplikacije pokreću se u modu operativnog sustava zvanog „user mode“. Svaka aplikacija pokrenuta u user modu dobiva svoj zasebni proces i svoje zasebno mjesto u memoriji računa, izolirana od drugih aplikacija tako da ako se pojedinačna aplikacija sruši, ostale nisu pod utjecajem tog pada. Također, većina drivera i ostalih sistemskih procesa radi u modu zvanom „kernel mode“. U kernel modu operativni

sustav pokreče aplikacije koje dijele zajednički memorijski prostor zbog čega postoji mogućnost kompromitiranja podataka između aplikacije te ako se jedna aplikacija u kernel modu sruši, ruši se cijeli operativni sustav.
Direct3D API koristi drivere isporučene od strane proizvođača od kojih svi rade u kernel modu, dok se driveri za OpenGL pokreću djelomično u kernel a djelomično u user modu. Kada OpenGL iz user moda poziva funkciju dostupnu samo u kernel modu, CPU se mora prebaciti u kernel mod rada. Taj proces je spor, traje nekoliko milisekundi i u to vrijeme procesor nije u mogućnosti raditi bilo kakve operacije. Zbog toga je smanjenje broja tih operacije bitno za optimizaciju rada sustava. Na primjer, ako je naredbeni buffer procesora pun podataka za prikaz, API može višak tih podataka spremiti na privremeni buffer i kada je naredbeni buffer na procesoru gotovo prazan, sve podatke iz privremenog buffera koji je u user modu može prebaciti u naredbeni buffer čime se provodi samo jedan prijelaz iz user u kernel mod rada. Taj proces zove se marshalling[6].
S obzirom da su svi driveri za Direct3D rađeni za rad u kernel modu, proizvođači drivera nemaju mogućnost korištenja user moda te se marshalling ne može postići što znači da se za svaku naredbu mora prijeći u kernel mod što opet traje i onemogućava procesor. No Direct3D uključen u Windows Vistu omogućava dio drivera da rade u user modu, time omogućava i marshalling te su performanse između API-a opet na gotovo jednakoj razini. [5]

6.3. Utrka za nultu stopu driver overhead-a

6.3. *Race to zero driver overhead*

Driver overhead[7] – količina radnji koje CPU mora napraviti da bi pripremio naredbe za obradu u GPU-u
Neki od razloga povečanja driver overheada [10]:

- posljedica poštovanja API “ugovora” od strane CPU-a koji sačinjavaju API
- potvrđivanje
- izbjegavanje grešaka

Rješenja za daljnje smanjivanje driver overheada [10]:
Buffer storage - umetanje višestrukog broja objekata u svaku jedinicu privremene memorije
Texture arrays - određivanje višestrukog broja tekstura u jednom informacijskom slijedu. U jedan slijed može primiti više teksutura koje se apliciraju na zajednički 3D objekt, no omogučuje postavljanja tekstura sa različitim dimenzijama, formatima i slično.
Multi-Draw indirect - tehnika iscrtavanja scena s velikom količinom sličnih objekata kojom se izbjegava korištenje posebnih funkcijskih poziva za svaki potrebni objekt
AMD je svojim Mantle API-em pokrenuo raspravu o moderniziranju grafičkih API-eva moderniziranjem apstraktnih koncepata koji bi trebali reflektirati operacije unutar GPU-a i od tada su i Microsoft i OpenGL proizvođači počeli prikazivati svoje vizije smanjenja ili potpune eliminacija driver overhead-a.
Microsoft planira takva rješenja predstaviti u DirectX-u 12 (planiran za prosinac 2015.) dok je OpenGL 2014. objavio listu obaveznih specifikacija i značajki za OpenGL 4.3 i 4.4 ili nekih već dostupnih u ekstenzijama. Tom su prilikom razvili i svoj program za mjerenje performansi koji je pokazao OpenGL kao višestruko brži sustav od DirectX 11.

7. Extenzije

7. *Extensions*

Jedna od najvećih razlika između spomenutih API-a je OpenGL-ov mehanizam ekstenzija. Svaki developer koji modificira OpengGL za svoje potrebe može dodavati mogućnosti koje druge verzije nemaju poput novih funkcija, načina za prijenos podataka na GPU itd. To omogućuje da se nove ekstenzije rano otkriju no često zbunjuje developere ako različiti API-evi koriste slične ekstenzije. Mnoge takve ekstenzije se redovito standardiziraju ili čak postanu dio temeljnog OpenGL API-a.
S druge strane, Direct3D API isporučuje samo jedan proizvođač, Microsoft, što omogućuje konzistentniji API ali onemogućuje dodatne funkcionalnosti potrebne drugim stranama. Doduše, Direct3D podržava korištenje ekstenzija za formatiranje tekstura koje se dana često koriste, što prije nije bio slučaj.
Ekstenzija koju podržavaju i OpenGL i Direct3D je S3 Texture Compression (S3TC DXTn ili

DXTC). Navedena ekstenzija služi kompresiji tekstura u hardverski akceleriranim grafičkim sustavima. Navedena ekstenzija kompatibilna je s DirectX 6.0 i novijim te OpenGL 1.3 i novijim.

8. Korisnici

8. *Users*

OpenGL je uvijek bio više nastrojen profesionalnim grafičkim implementacijama nego Direct3D koji je kao dio DirectX-a uvijek bio više fokusiran na igre. Trenutačnu se oba API-a dovoljno funkcionalno preklapaju da bi se oba mogla koristiti za često korištene radnje u oba područja. Jedino je operativni sustav onaj koji diktira koji će se API koristiti; Direct3D za Windowse, OpenGL za gotovo sve ostale sustave.

8.1. Profesionalci

8.1. *Professionals*

U jednom je periodu dio profesionalnih grafičkih kartica podržavao isključivo OpenGL API dok sada gotovo sve uz OpenGL podržavaju i Direct3D. Za dio toga je zaslužan prijelaz profesionalnih aplikacija sa operativnih sustava temeljenih na Unix-u na Windowse.
Glavni razlog dominacije OpenGL a u profesionalnim okruženjima bile su performanse. Mnoge profesionalne aplikacije su u početcima bile temeljene na IRIS GL-u na kojem je OpenGL temeljen za sustave sa mnogo naprednijim i bržim komponentama od tadašnjih PC-eva. Kasnije su mnoge od tih aplikacija prebačene na OpenGL i u isto su vrijeme korisnički PC-evi postajali sve sposobniji u smislu procesorske i grafičke snage omogućujući time pokretanje i korištenje profesionalnih aplikacija.
Konkurentna cijena je u međuvremenu razbila dominaciju OpenGL-a na tržištu no razvijena baza OpenGL korisnika je omogućila da su i OpenGL i Direct3D razumne opcije za razvoj aplikacija.
Još jedan razlog za dominaciju OpenGL-a je marketing i dizajn. Direct3D se nikad nije promovirao kao API za profesionalce već kao API za razvoj aplikacija niskih performansi i zahtjeva namijenjen prosječnim korisnicima, prvenstveno za razvoj igara na relativno jeftinom hardveru. OpenGL je API namijenjen širokom rasponu aplikacija i grafičkog hardvera od jeftinih grafičkih kartica do profesionalnih i znanstvenih grafičkih aplikacija i prikaza daleko izvan potreba prosječnog korisnika i setom mogućnosti koji nije bio specifično namijenjen jednoj vrsti korisnika.
Developeri igara rijetko imaju zahtjeve API-em širokim kao što zahtijevaju developeri profesionalnih sustava. Rijetke igre koriste napredne mogućnosti koje im API omogućuje a koje su od presudne važnosti u snimanju filmova i modeliranju.
U jednom trenutku su Microsoft i SGI (tvorac OpenGL-a) pokušali spojiti DirectX i OpenGL u API zvan „Fahrenheit graphics API[8]“ koji je bio zamišljen kao API koji zadovoljava visoke uvjete profesionalnih aplikacija dok je istovremeno omogućavao široku podršku manje zahtjevnih developera poput one za DirectX. Na kraju je Microsoft odustao od projekta zbog premalo resursa uloženih u svoj dio projekta. Njihov potez je shvaćen kao način da se dio developera prebaci na njihovu platformu, dok bi u slučaju uspjeha projekta Microsoft izgubio i taj dio sljedbenika jer bi Fahrenheit tada postao standard u grafičkim API-ima.

8.2. Igrači

8.2. *Gamers*

U rano doba 3D igranja, najbitnije stavke kod proizvodnje grafičkih kartica bile su performanse i pouzdanost i driveri su se posebno pisali za svakog proizvođača kartica no tijekom godina su se OpenGL i DirectX iskristalizirali kao slojevi iznad grafičkog hardvera, prvenstveno zbog grafičkih resursa koji moraju biti dostupni na svim platformama grafičkih sustava.
U početku proizvođači kartica nisu razvijali pune drivere za OpenGL zbog dva razloga:
Većina grafičkih kartica nije podržavala sve funkcije OpenGL API-a
Mnogim proizvođačima je bilo mukotrpno implemetirati puni OpenGL driver s dobrom funkcionalnošću i performansama
Umjesto toga su pisali MiniGL, lakšu verziju OpenGL-a koja je bila sposobna pokrenuti GLQuake. Pravi OpenGL driveri su se razvijali usporedno s grafičkim hardverom, što se dogodilo u vrijeme DirectX 6 i DirectX 7.
Konzole obično koriste vlastite API-eve, često izvedene iz OpenGL-a, optimizirane za maksimalne performanse na određenoj konzoli

te su zbog toga usporedbe između Direct3D-a i OpenGL-a rezervirane za PC platformu.

9. Zaključak

9. *Conclusion*

Iako su OpenGL i Direct3D krenuli iz različitih izvora i namjena, često su se u povijesti susretali kao konkurencija, u jednom trenutku skoro čak i postali jedan proizvod, danas se nalaze na suprotnim stranama korisničkog spektra, jedan za profesionalce, drugi za igrače, s tim da povremeno prelaze u domene onog drugog. Kakva god njihova publika u danom trenutku bila, nedvojbeno je da su kroz sve godine svog postojanja tjerali jedan drugoga na napredak.

10. Reference

10. *References*

[1] OpenGL – About, dostupno: https://www.opengl.org/about/

[2] OpenGL – History, dostupno: https://www.opengl.org/wiki/History_of_OpenGL

[3] Wikipedia – Direc3D, dostupno: https://en.wikipedia.org/wiki/Direct3D

[4] Alex stJohn – the Evolution of Direct3D, dostupno: http://www.alexstjohn.com/WP/2013/07/22/the-evolution-of-direct3d/

[5] Stach Overflow – What is Object Marshalling, dostupno: http://stackoverflow.com/questions/154185/what-is-object-marshalling

[6] Jesper Mosegaard – While we wait – Approaching Zero Driver Overhead, dostupno: http://cg.alexandra.dk/?p=3778

[7] Fact Indeks – Fahrenheit Graphics API, dostupno: http://www.fact-index.com/f/fa/fahrenheit_graphics_api.html

[8] Wikipedia – Finite-state Machine, dostupno: https://en.wikipedia.org/wiki/Finite-state_machine

[9] Wikipedia – D3D Pipeline, dostupno: https://en.wikipedia.org/wiki/File:D3D_Pipeline.svg

[10] Slideshare - Approaching zero driver overhead, dostupno: www.slideshare.net/ CassEveritt/approaching-zero-driver-overhead

11. Slike

11. *Figures*

Slika 1 Shema slijeda programiranja OpenGL-a[1]

Figure 1 OpenGL programming scheme[1]

Dostupno na: https://www.opengl.org/about/images/Flow.gif

Slika 2 HIJERARHIJA API-A[1]

Figure 2 API Hierarchy[1]

Slika 2 je autorska.

Slika 3 Direct3D proces[9]

Figure 3 Direct3D process[9]

Dostupno na: https://en.wikipedia.org/wiki/File:D3D_Pipeline.svg

AUTOR · *AUTHOR*

Andrija Bernik – nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 2, No. 2, 2014.

Zvonimir Sabati – nepromjenjena biografija nalazi se u časopisu Polytechnic & Design Vol. 2, No. 2, 2014.