Teknik bakom tredimensionella datorgrafiken Direct3D

Samuli Ketola

Examensarbete
Medieteknik
2011

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Medieteknik
Identifikationsnummer:	3234
Författare:	Samuli Ketola
Arbetets namn:	Teknik bakom tredimensionella datorgrafiken Direct3D
Handledare (Arcada):	Johnny Biström
Uppdragsgivare:	

Direct3D är en av de mest använda applikationsprogrammeringsgränssnitt för grafiska applikationer på datorer idag. Det är en delkomponent i applikationsprogrammeringsgränssnittet DirectX och är producerad av Microsoft. Arbetet beskriver ur en teknisk synvinkel hur Direct3D fungerar och hur det används i programvara samt mellan programvara och hårdvara. Det börjar med en inledning och sedan introduceras en översikt om datorgrafik för att bättre förstå de avancerade egenskaper som inkluderas i DirectX och Direct3D. Dessa presenteras senare i arbetet. Huvudsakliga målet är att tekniskt bevisa hur Direct3D ritar grafik på skärmen samt presentera de nya visuella tekniker som inkluderats i nyaste versionen Direct3D 11. Arbetet har avgränsats så att det handlar endast om de grafiska egenskaperna i Direct3D, några avvikelser måste dock göras för att ge läsaren en bättre helhets förståelse om i fråga varande egenskaper. Som metod användes främst litteraturforskning och viktigaste litteraturen har varit Microsoft applikationsutvecklingssidor. Övriga viktiga källor har varit grafiska exempel, tekniska artiklar, böcker som behandlar grafik och för en liten del Wikipedia. Resultat som man har kommit till är att de visuella egenskaperna i Direct3D 11 blir allt mer avancerade och realistiska samt att programmeringen har blivit mera flexibel och bättre hanterbar. Därför är potentialen enorm. Arbetet avslutas med en kort jämförelse mellan Direct3D och dess största konkurrent OpenGL och sedan slutsatser.

Nyckelord:	Datorgrafik DirectX Direct3D Rendering
Sidantal:	
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Media technology
Identification number:	3234
Author:	Samuli Ketola
Title:	Teknik bakom tredimensionella datorgrafiken Direct3D Technique behind the three dimensional computer graphics Direct3D
Supervisor (Arcada):	Johnny Biström
Commissioned by:	

Direct3D is one of the most widely used application programming interfaces for graphical applications on PCs today. It is a component of the application programming interface, DirectX, which is produced by Microsoft. The work describes in technical terms how Direct3D works and how it is used in software and between software and hardware. It begins with an introduction and then presents an overview of computer graphics to better understand the advanced features that are included in Direct3D and DirectX. They are presented later in this work. The main goal is to technically prove how Direct3D is drawing graphics on the screen and present the new visual technologies that are included in the newest version Direct3D 11. The work has been limited so that it only deals with the graphical features of Direct3D. Some deviations had to be done though to give the reader a better overall understanding of those characteristics. The method used was for the most part literature research and the most important literature has been Microsoft application development website. Other important sources have been graphical examples, technical articles, books that deal with graphics and for a small part Wikipedia. Results are that the visual features in Direct3D 11 are becoming more advanced and realistic, and that the programming has become more flexible and more manageable. Because of that, the potential is enormous. The work ends with a brief comparison between Direct3D and OpenGL (its largest competitor) and conclusions.

Keywords:	Computer graphics DirectX Direct3D Rendering
Number of pages:	
Language:	Swedish
Date of acceptance:	

INNEHÅLL / CONTENTS

Figurer

Tabeller

ORDLISTA

Detta är en ordlista för termer inom datorgrafik och datorvetenskap, ord som förekommer kan nämligen ha någon annan mening i annat bruk.

API Application Programming Interface, applikationsprogrammeringsgränssnitt är en regeluppsättning för hur en viss programvara kan kommunicera med en annan programvara.

Animering En sekvens av två- eller tredimensionella konstverk eller figurer i snabb följd som skapar en rörlig bild.

Block av data En sekvens av byten och bitar som har en nominell längd. Används för att underlätta och påskynda behandlingen av en dataström för datorprogrammet som tar emot det.

CAD (Computer Aided Design) Termen avser digitalt baserad (skapad med en dator) design och skapande av tekniska ritningar som används för det mesta inom konstruktion och arkitektur.

DDI (Device Driver Interface) Gränssnitt för drivrutiner för en apparat.

Destruktiv komprimering Förstorande komprimering, utnyttjar och använder de defekter som finns i människans uppfattningsförmåga.

Emulate Att duplicera (”emulera”, görs av en emulator) funktioner av ett system med att använda ett annat system, så att det andra systemet uppför sig så som det första.

GDI (Graphics Interface Device) Kärnelement i ett operativsystem som ansvarar om representation av grafiska objekt och sedan överför dem till output medium så som monitorer och printrar.

Geometri (datorvetenskap) Studerar geometriska och matematiska algoritmer för presentation av grafik.

HAL (Hardware Abstraction Layer) HAL är en implementering i mjukvara som sköter kommunikationen mellan applikationer och hårdvara.

Hårdvaruacceleration Användning av hårdvara för att genomföra funktioner snabbare än vad det är möjligt i mjukvara som allmänt körs av processorn.

Icke-destruktiv komprimering Felfri kompression, komprimerar på ett sådant sätt att all ursprunglig data går att återskapa.

Interpolation En metod för att generera nya datapunkter från en diskret mängd av befintliga datapunkter.

Lookup Avser letande av ett objekt i en datastruktur som sedan passar till någon specificerad egenskap.

Paging Ett minneshanteringsschema i operativsystem med hjälp av vilken datorn kan spara och återfå data från sekundära lagringsmetoder för att sedan förbrukas i primärminnet.

Polygon Tvådimensionell form som modelleras och sparas inom sin egen databas.

Primitiv (språk) De mest simpla element som är tillgängliga i programmeringsspråk.

Primitiv (geometri) De mest simpla slag av figurer inom datorgrafik.

Rendering Den beräkning som ett datorprogram utför för att framställa en bild eller animering utifrån en scen eller modell.

Runtime library (runtime bibliotek) I datorprogrammering är runtime library ett speciellt program bibliotek som används av en kompilator för att implementera funktioner i programmeringsspråk under exekvering (runtime) av ett datorprogram.

Shader En samling av mjukvaruinstruktioner som används huvudsakligen för beräkning av rendering effekter på grafisk hårdvara.

Sprite En tvådimensionell bild eller animation som integreras i en större scen.

Swap Refererar till att ömsesidigt byta ut värden av två variabler.

Textur En datamängd som bestämmer en ytas utseende och egenskaper. Texturer består av flera texel.

Transform Konvertering av rumsliga koordinater (tredimensionella objekt) för tvådimensionell vy.

Tråd (thread) En exekveringsenhet inom en process som styrs av operativsystemet.

Vertex En datastruktur som beskriver en punkt i två- eller tredimensionell rymd. Objekt ansluts med hjälp av tabeller av flata ytor (typiskt trianglar) och vertex information definierar läget samt övriga attribut av hörnen på ytorna.

1 INLEDNING

År 1992 skapade Servan Keondjian ett företag som hette RenderMorphics. Företaget utvecklade ett applikationsprogrammeringsgränssnitt (senare API) för tredimensionellt grafik som kallades Reality Lab. Reality Lab användes för visuell medicinsk representation samt CAD bruk. Microsoft köpte RenderMorphics år 1995 och tog med Servan Keondjian för att implementera en grafisk tredimensionell motor för Windows 95. Det resulterade till första versionen av Direct3D och allt efter det är historia.

Direct3D är en API som används i tredimensionell grafik då prestanda är mycket viktigt.

1.1 Syfte och målsättning

Syftet med detta arbete är att ur en teknisk synvinkel klarlägga hur Direct3D fungerar och hur det används i programvara samt mellan programvara och hårdvara. Arkitektur och nya egenskaper i Direct3D beskrivs noggrant samt exempelgrafik presenteras. Direct3D är tills vidare en API som används endast i Microsoft plattformer (t.ex. Xbox och Windows). Kompatibiliteten för hårdvara och mjukvara studeras och beskrivs grundligt. De största skillnaderna och viktigaste milstolparna inom utvecklingen av Direct3D skall också tas upp från och med DirectX 2.0 (1996) till DirectX 11 (2009). Det inkluderas även en kort jämförelse mellan Direct3D och dess största konkurrent OpenGL.

Målsättning med arbetet är att förklara hur Direct3D ritar ut grafik på skärmen samt presentera de nya avancerade tekniker som inkluderats i nyaste versionen Direct3D 11. Från och med en översikt om datorgrafik till arkitektur och egenskaper av Direct3D.

1.2 Metoder

Jag har studerat jämförelser och litteratur ur olika versioner av DirectX och Direct3D och hur de fungerar i olika spel och applikationer. Viktigaste litteraturen var Microsoft hemsidor samt deras applikationsutvecklingssidor. Ett par bra exempel på dessa är

Microsoft DirectX Developer Center och Windows DirectX Graphics Documentation. Där finns det mycket pålitligt information på grund av att DirectX och Direct3D är en Microsoft produkt. Jag har studerat också flera artiklar, presentationer (från olika konferenstillfällen), böcker, grafik samt övriga textdokument för att hitta information som är relevant och pålitligt. Studierna gav mig en bättre inblick och förståelse om teknikerna bakom Direct3D och samlade informationen utnyttjades i examensarbetet.

Två grafiska demonstrationer som studerades var Unigine Heaven Benchmark och Unreal Engine 3 Realtime Demonstration (”Samaritan”). I dessa används det nämligen en stor del av de nya egenskaperna i Direct3D 11 samt vad man kan åstadkomma med dessa. De gav mig en bättre översikt och förståelse om hur de visuella egenskaperna i Direct3D 11 ser ut och fungerar i praktiken.

1.3 Avgränsning

Arbetet skall handla om hur Direct3D används inom datorgrafik. Egenskaper av Direct3D används nämligen i dagens högupplösningskonsoler också, tekniker inom dessa är dock mycket liknande. Det har studerats hur DirectX kunde effektivt fungera i användargränssnittet Linux/Unix, till exempel med hjälp av programmet Wine. Enligt vad jag har studerat så har man har inte kommit fram till något ordentligt slutresultat och därför skall jag inte inkludera detta i examensarbetet. Arbetet har vidare avgränsats så att jag har undersökt och skrivit endast om den grafiska delen av DirectX, och i huvudsak handlar det om Direct3D och dess nya egenskaper och tekniker. Det är dock nödvändigt att ta upp några andra API:n också för att i DirectX jobbar dessa ofta hand i hand. DirectX har en del API:n som berör audio och ljud men tekniker inom dessa skiljer radikalt från de grafiska API:n och därför skall deras egenskaper och tekniker inte undersökas och förklaras. Elektriska egenskaper för hårdvara samt programmeringsspråk (och matematiska formler) för skapande av Direct3D grafik presenteras inte heller.

2 ÖVERSIKT OM DATORGRAFIK

Idag berör datorgrafik och datorer otaliga aspekter av våra dagliga liv. Man ser datorgrafik i spel, visuella effekter, television, tidningar, reklam, inom presentation av material och så vidare. Datorgrafik beskriver i allmänhet allt skapande och manipulering av visuella ting med en dator. Visuellt material som används kan vara antingen helt syntetiskt eller tillverkad med manipulering av detta. Man kunde säga att nästan allt på datorer som inte är text eller ljud involverar datorgrafik. Därför är det en nödvändighet att förstå grunderna av datorgrafik före man studerar noggrannare på ämnet.

Termen datorgrafik kan referera till flera olika saker:

- Presentation och manipulering av bildinformation med en dator
- Tekniker som används för att skapa och manipulera bilder
- Figurer/Bilder som skapats
- Underkategorier inom datorvetenskap som studerar metoder för att på ett digitalt sätt syntetisera och manipulera visuellt information med datorer. Exempel på tre allmänna underkategorier:
    - Geometri: studerar olika sätt för att presentera och framställa ytor
    - Animering: studerar olika sätt för att presentera och manipulera rörelse
    - Rendering: studerar algoritmer för att reproducera ljusförflyttning

Det finns två typer av datorgrafik som används huvudsakligen, dessa är tvådimensionellt datorgrafik och tredimensionellt datorgrafik.

2.1 2D datorgrafik

2D datorgrafik är med andra ord tvådimensionella digitalbilder eller text och tekniker som används för att producera dessa. Det omfattar två dimensioner, längd och bredd.
2D datorgrafik är en teknik som används brett inom applikationer som är menade för ritande och utskrivande. Traditionella exempel på dessa är typografi, kartografi, tekniskt ritande och annonsering. Två vanliga tekniker som används inom 2D datorgrafik är rastergrafik och vektorgrafik.

2.1.1 Rastergrafik

I rastergrafik består en bild av en mängd pixlar (bildpunkt) som är placerade inom ett fyrkantigt rutnät där varje pixel har en specifik färg. Rastergrafik kunde också kallas till bitmapgrafik. Egenskaper för bilder inom grafiken är definierade med bredd och längd samt mängden bit per pixel. Färgdjupheten för bilder är alltså definierad med mängden bitar som används för att presentera färgen för en singel pixel (ju fler bitar, desto bättre kvalitet). Mest använda färgrymden är RGB, där det används rött, grönt och blått vilka blandas för att få en specifik färg. Det används i nästan all elektronisk grafisk presentation. En annan mycket använd färgrymd är CMYK (cyan, magenta, gul och svart eller nyckelfärg) som används i färgtryckeri och dylikt.

Bilder inom rastergrafik är upplösningsberoende, så de kan icke skalas upp oändligt utan att kvaliteten blir synligt försämrad. Bilder sparas i filer med varierande format, beroende på hur bra man vill att kvaliteten proportionellt skall bli.

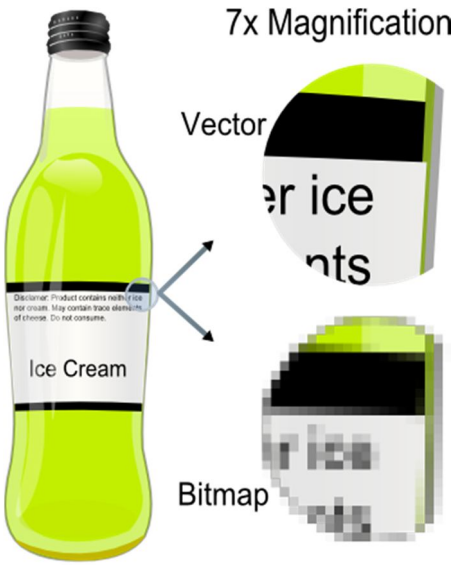
Universellt använda format:

- Icke komprimerade (bäst kvalitet): BMP, TIFF
- Destruktivt komprimerade (bra kvalitet): JPEG
- Icke-destruktivt komprimerade (medelmåttig kvalitet): GIF, PNG

En alternativ teknik som används inom 2D datorgrafik är vektorgrafik.

2.1.2 Vektorgrafik

Till skillnad från rastergrafik består vektorgrafik av geometriska objekt, så som punkter, linjer, cirklar, längdbågar, polygoner och så vidare. Deras form och egenskaper presenteras med hjälp av matematiska funktioner, och på grund av detta kan man i teorin skala vektorgrafik hur stort som helst utan att bildkvaliteten försämras.



Figur 1. 7x förstoring, skillnad mellan Rastergrafik och Vektorgrafik. (Wikimedia 2011)

I princip görs all form av 3D-modellering med hjälp av vektorbaserad teknik.

En vektorbild konverteras av grafikkortet till en rasterbild innan den presenteras på skärmen, detta kallas till rasterisation.

2.2 3D datorgrafik

3D datorgrafik är tredimensionell representation av geometriskt data som är sparad i datorer för att genomföra uträkningar och ritande av 2D bilder. I 3D datorgrafik används det liknande algoritmer som i vektorgrafik för visuell presentation av tredimensionella fysiska objekt. Rastergrafik används igen för den slutliga bild som ritas på skärmen.

Processen för hur 3D datorgrafik skapas kunde delas i tre grundläggande faser.

2.2.1 Modellering

Modellering beskriver processen för hur formen av ett objekt skall se ut. Det inkluderar utveckling av matematisk representation av tredimensionella objekt (eller ytor) med hjälp av specialiserad mjukvara. Fysisk simulering är ett annat sätt att skapa tredimensionella modeller.

Tredimensionella modeller presenteras ofta visuellt med hjälp av en så kallad järntråds modell (wire-frame model). Dessa skapas med att specificera varje hörn av ett objekt där två matematiskt fortgående ytor möts, eller med att ansluta flera vertex för ett objekt med linjer eller kurvor.

2.2.2 Planering och animering

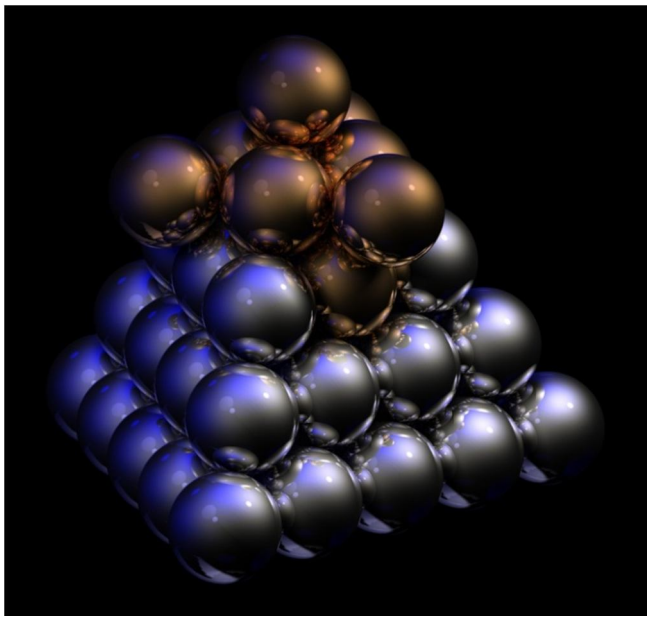
Planering är en viktig steg då man skapar 3D datorgrafik för att man måste tänka på ett flertal saker för att nå den resultat som önskas. Objekt måste placeras ut inom en scen förrän de ritas på skärmen, detta definierar rumsliga förhållanden mellan objekt inom en scen inklusive läge och storlek. Man måste också planera färger, plandjupet (hur långt framåt man ser i scenen), hur texturer skall se ut och så vidare.

Animation igen hänvisar till den temporära beskrivningen av ett objekt, med andra ord hur objektet ändras och rör på sig över tid. Animation kan skapas till exempel med hjälp av nyckelbilder, motion capture (”fångande av rörelse”), invers kinematik och fysisk simulation. Dessa tekniker används ofta i samverkan med varandra.

2.2.3 Rendering

Rendering är den beräkning som ett datorprogram utför för att konverterar en modell eller en scen (datafil) till en bild eller animation. En scen-fil kan innehålla flera objekt med strikt definierade datastruktur. Det kunde exempelvis inkludera information om ljus, geometri, texturer, reflektioner och skuggning som alla tillsammans sedan definierar den virtuella scenen.

I rendering används endera simulering av ljusförflyttning för att få fotorealistiska bilder, eller någon sort av stil så som i t.ex. icke-fotorealistisk ritande av bilder. De två grundläggande operationer inom realistisk rendering är ljusförflyttning (hur mycket ljus flyttas från en plats till en annan), och ljusutspridning (hur ytor reagerar med ljus).



Figur 2. Exempel på ett renderat tredimensionellt objekt. (Wikimedia 2011)

Fastän processer och tekniker inom rendering varierar så krävs det allmänt en grafisk pipeline samt en GPU (grafikprocessor) för att skapa en bild eller animation från en modell eller en scen. Direct3D och OpenGL är två märkvärdiga exempel på en grafisk pipeline, och dessa presenteras noggrannare senare i arbetet.

2.3 Mapping

Mapping är ett ord som ofta förekommer då det talas om datorgrafik. Det är en process där man ”mappar” eller kartlägger olika element för en datamodell och/eller en 3D-modell. Dessa element skapas oftast med hjälp av olika koordinatsystem och koordinater/punkter. Element som används kan vara t.ex. en vertex för en polygon. Egenskaper av mapping implementeras ofta också i hårdvara.

Det finns flera olika mapping tekniker som används idag, de mest allmänna är:

- Texture mapping: En metod som används för att addera detaljer, ytor, färger eller texturer för en grafisk tredimensionell modell. Metod som används för att bestämma färgen för en pixel i en texture map kallas för texture filtering eller texturfiltrering.

- Bump mapping: Teknik för simulering av skrynklor och knöl på ytan av ett objekt.
- Normal mapping: Teknik som används för att förfalska belysningen av knöl och bucklor, detaljer adderas utan att använda flera polygoner.
- Environment mapping: Teknik där man använder belysningsegenskaper av en omgivning för att skapa reflektioner och dylikt.
- Cube mapping: En metod av environment mapping som använder en sex sidig kub för map formen.
- Displacement mapping: En alternativ teknik för att skapa en effekt där aktuella geometriska positioner av punkter över en yta med texturer förflyttas enligt värden som textur funktionerna evaluerar för varje punkt på ytan.
- Parallax mapping: En förbättring till bump mapping och texture mapping som möjliggör skapande av en realistisk illusion av djup.
- Relief mapping: En teknik som används för att rendera detaljer för ytor av tredimensionella objekt noggrant och effektivt.

För övrigt finns det olika så kallade ”mappar” som bestämmer egenskaper så som belysning (lightmap) och höjd (heightmap) för ytor. I grafisk tredimensionell texturfiltrering räknas det också ut så kallade mipmap som är optimerade samlingar av bilder som åtföljer en huvudsaklig textur för att höja rendering hastigheten och förminska fel i utjämning.

2.4 Stereoskopisk 3D

Det har introducerats tekniker för stereoskopiskt tredimensionellt datorgrafik som har blivit mycket populära idag. Idén är att skapa eller förbättra en illusion av djup för tredimensionellt grafik. Det presenteras så att man visar bilden på skärmen i två skilda tvådimensionella bilder, en för båda ögonen. Dessa bilder är sedan kombinerade i hjärnan för att ge en förnimmelse av tredimensionellt djup. Tekniken används till exempel i en stor del av dagens Direct3D spel och filmer. I datorbruk krävs en mängd kompatibla komponenter för att stereoskopisk 3D skulle fungera effektivt. Dessa är stereoskopiska glasögon, en 3D kompatibel skärm samt en kompatibel grafikkort.

Det har tillverkats en flertal stereoskopiska tekniker inom de närmaste årtionden, men NVIDIA 3D Vision kit som presenterades år 2008 introducerade tekniken för konsumenter och datorspel. (Coles, 2009)

Stereoskopisk 3D fungerar redan mycket bra idag och det möjliggör ett relativt realistiskt och trovärdigt tredimensionellt djup.

2.5 Hårdvara

En dator består av flera fysiska komponenter som kallas hårdvara för datorn. Dessa komponenter behandlar information för ett datorsystem. Hårdvara för en modern dator inkluderar i allmänhet en moderkort, strömkälla, Central Processing Unit (CPU, senare centralprocessor), grafikkort, Random Access Memory (senare primärminne), hårddisk (sekundärminne), optisk läsare, skärm, tangentbord, mus och möjligtvis övrigt tilläggs hårdvara.

2.6 Mjukvara

Mjukvara är en samling datorprogram och relaterad data som sedan ger instruktioner för vad datorn skall göra och hur det skall göras. Applikationsprogrammeringsgränssnitt, applikationer, spel och olika CAD (Computer Aided Design) program är vanliga exempel på mjukvara. Visuella effekter, animationer, och simulationer i dessa är igen vanliga exempel på för vad man kan återkomma med hjälp av mjukvara.

2.7 Övergång från hårdvara till mjukvara

De två komponenter som huvudsakligen tar hand om överföringen av data är centralprocessorn och grafikkortet. Övrig hårdvara som krävs är moderkort och strömkälla för uppkoppling och ström, primärminne för temporärt sparande av information (exempelvis pixelinformation och hela bilder), sekundärminne för lagring av information och en monitor för att se det slutliga resultatet.

Region i minnen där data sparas temporärt då det flyttas från en plats till en annan kallas till buffert. Dessa används ofta för att jämna ut skillnader i ojämna överföringshastigheter. Det finns buffert för flera olika ändamål.

I grafisk presentation används det huvudsakligen följande tekniker:

- Rambuffert: En video output apparat som kör en bildskärm från en minnesbuffert inklusive en hel ram av data. Information i minnesbuffert består vanligtvis av färg egenskaper för varje pixel på skärmen.
- Z-buffert (eller djuphets buffert): Hanterar bilddjups koordinater i tredimensionell grafik, görs vanligtvis i hårdvara men det är möjligt i mjukvara också. Använder djupet av genererade pixlar som skapas då ett objekt renderas.
- Stencil buffert: Används oftast för att avgränsa arean i rendering. Stencil buffert fungerar per-pixel och arbetar med heltalsvärden. Använder också övriga avancerade tekniker och har ett starkt förhållande till z-buffert i renderingspipelinen.

I datorgrafik används det idag multipla buffertar också. Det är alltså användning av mera än en buffert för att hålla ett block av data, så att man ser på skärmen en fullständig version av data istället för en partiellt uppdaterad version. I datorgrafik används det ofta dubbel buffert (double buffering) eller trippel buffert (triple buffering) för att rita grafik snabbt och utan förvrängningar, brytningar, gnistor och så vidare.

Centralprocessor är den komponent i ett datorsystem som transporterar och exekverar maskininstruktioner av ett datorprogram, och är den primära element för datorns funktioner. Den transporterar varje instruktion (inklusive typ av data, registrering, adressering med mera) av program i sekvenser för att genomföra fundamentala aritmetiska, logiska och input/output operationer samt beräkningar för ett datorsystem. Då man vill exekvera och rita krävande grafiska uppgifter på skärmen så räcker inte centralprocessorns prestanda alltid nödvändigtvis till. Det är då som grafikkort behövs. En grafikkort har en GPU (Graphics Processing Unit, senare grafikprocessor) inbyggt i sig. Det är en specialiserad krets som är kapabel för att rita grafik mycket snabbare än en central processor. Idén bakom grafikkortet är alltså att underlätta de grafiska uppgifter som annars skulle göras av processorn. De moderna grafikkort är mycket effektiva i manipulering

av datorgrafik för att kunna följa och orka behandla de kontinuerligt utvecklade tekniker.

För att behandla och använda information i det slutliga skedet så installeras ett operativsystem (t.ex. Microsoft Windows) och en mängd mjukvaruapplikationer för att nå dessa resultat som önskas.

3 DIRECTX

Första versionen av applikationsprogrammeringsgränssnittet DirectX presenterades år 1995 (Eisler, 2006). Det påhittades av tre Microsoft arbetare Alex St. John, Craig Eisler och Eric Engström. Ursprungliga idén var att komma på ett visst sätt för multimedia att kommunicera mellan datorprogram och hårdvara och det var ganska långt just det som resultatet blev. DirectX används idag vidsträckt inom utvecklande av datorspel samt inom visuella och grafiska uppgifter för mjukvara och video. Man kunde säga att DirectX har blivit en samling av regler för hur datorgrafik ritas på bildskärmen.

3.1 Komponenter

Namnet DirectX är en så kallad förkortning för en mängd applikationsprogrammeringsgränssnitt där ”X” står för det specifika API namnet. Varje API har en definierad uppgift i DirectX samlingen. Direct3D är i allmänhet den mest populära och utspridda komponent av DirectX, och därför är det ofta vanligt att se dessa två namn gå hand i hand. DirectX har dock ett flertal andra grafiska API:n också, så som DirectDraw och Direct2D.

Bokstaven ”X” inkluderas också i namnet av Microsoft spelkonsol Xbox för att indikera att den är baserad på DirectX teknologi.

3.1.1 DirectDraw

DirectDraw är en av föregångaren för Direct3D och Direct2D. Det är en tvådimensionell API för 2D rendering och stöder inte tredimensionell hårdvaruacceleration. Den

kunde användas för ritande av tredimensionellt grafik, men prestandan skulle vara mycket trög jämfört med t.ex. Direct3D som igen stöder tredimensionell hårdvaruacceleration. DirectDraw är fortfarande en komponent i DirectX fastän den icke utvecklas mer. Istället används det ofta egenskaper av Direct3D för 2D rendering, och för största delen är detta API nu ersatt med Direct2D.

3.1.2 Direct2D

Direct2D är en grafisk API som använder vektorbaserade tekniker för tvådimensionell rendering. Den erbjuder hög kvalitet och snabb prestanda medan den är kompatibel med API:n så som Direct3D, DirectDraw och GDI/GDI+ (Graphics Device Interface, medium för ett grafiskt gränssnitt). Direct2D stöder hårdvaruacceleration.

Andra komponenter som används för grafiska uppgifter inom DirectX är DirectCompute för värdering av grafikprocessorn, DirectWrite för fonter och DirectX Media familjen för att förvandla media till en lämplig form för internet. Övriga komponenter berör antingen ljud eller sedan används de för varierande input/output tekniker eller olika gränssnitts operationer.

3.2 Tillverkning

För tillverkning används det DirectX Software Development Kit (senare SDK). Det är en uppsättning utvecklingsverktyg som möjliggör mjukvaru utvecklaren att bygga applikationer för DirectX bruk. Det består av ett flertal runtime bibliotek, dokumentation samt en mängd headerfiler (inkluderingsfil) för programmeringsbruk. Programmeringsspråk som används är C++, Visual Basic eller HLSL (High Level Shader Language) samt dess referensdokumentation som specificerar karakteristik för språket. I skapande av grafik för Direct3D så används det ofta D3DX som är ett bibliotek av verktyg. Det är planerat för att utföra vanliga matematiska uträkningar, anslutning av shader funktioner och för flera övriga komplicerade uppgifter.

3.3 Kompatibilitet och utveckling

Sedan år 1995 har DirectX utvecklats från och med version 1.0 till version 11 som presenterades år 2008. Första versionen var en SDK för Windows spel, det gav en möjlighet för alla versioner av Microsoft Windows (börjande från Windows 95) att förenas med multimedia som kräver högt prestationskraft.

API:n måste kommunicera med hårdvara, och det görs med hjälp av drivrutiner. Hårdvarutillverkaren måste skriva dessa särskilt för varje version av DirectX drivrutins gränssnitt, och sedan testa varje enskild bit av hårdvara för att göra dem kompatibla med DirectX.

DirectX används tills vidare endast på Microsoft plattformer. Olika versioner av Microsoft Windows har inkluderat och stött olika versioner av DirectX. Det har gett möjlighet för nyare versioner av operativsystem att fortsätta köra applikationer som är planerade för tidigare versioner av DirectX tills dessa versioner sedan kan steg för steg uppdateras för nyare API:n, drivrutiner och hårdvara.

Förrän DirectX 10 var DirectX runtime planerad att vara bakåt kompatibel med äldre drivrutiner. Det betyder att nyare versioner av API:n fungerade med äldre versioner, det kunde dock försämra funktionaliteten. När DirectX 10 gavs ut så gjorde Microsoft beslut att det var dags för en drastisk uppdatering. Det kom ut en ny drivrutinsmodell som ledde till att stöd för äldre Windows operativsystem, så som populära Windows XP, icke längre fanns. DirectX 10 gavs ut endast för Windows Vista och senare versioner. Microsoft gav ut obligatoriska specifikationer för DirectX 10 och det var en bra sak för utvecklaren och programmeraren. De behövde inte mera tänka på vad för hårdvara de tillverkar för och på den producentspecifika koden.

DirectX 11 är den nyaste versionen av DirectX och fungerar på Windows Vista och Windows 7 samt Microsoft framtida gränssnitt. Karakteristiken för hårdvaru och API funktioner är liknande som i DirectX 10. Kompatibiliteten har dock förbättrats en bit för äldre versioner av DirectX samt hårdvara. De märkvärdigaste uppdateringarna gjordes dock för DirectX grafiska API Direct3D och dess egenskaper.

4 DIRECT3D

Direct3D är ett applikationsprogrammeringsgränssnitt för tredimensionellt grafik. Det används för tredimensionell rendering i applikationer där prestation är mycket viktigt. Direct3D använder hårdvaruacceleration om det är tillgängligt på grafikkortet. Det ger en möjlighet för hårdvaruacceleration antingen för hela Direct3D pipeline eller endast för partiell acceleration. Direct3D innehåller avancerade grafiska egenskaper för tredimensionell grafisk hårdvara, inklusive tessellation, kantutjämning, z-buffering, atmosfäriska effekter och så vidare. Direct3D kan implementeras fullständigt på en skärm (då skapar det hela grafiska output för en bildskärm/apparat) eller i ett fönster (då kommunicerar det med GDI för att skapa grafiskt output). Integrering med övriga DirectX teknologier utrustar Direct3D för en mängd övriga tekniker också, så som sprite egenskaper.

Direct3D erbjuder full vertex emulering för egenskaper som hårdvaran inte stöder eller innebär, men ingen pixel emulering. Direct3D gör alltså ingen emulering om mjukvara som är programmerad med Direct3D kräver någon egenskap och grafikkortet icke stöder det. Trots det så värderar och renderar den polygoner och texturer av 3D-modell, det leder dock ofta till sänkt kvalitet och försämrad prestation jämfört med ekvivalent hårdvara.

Microsoft strävar kontinuerligt efter att uppdatera Direct3D för att stöda de senaste teknologier som används i grafikkort.

4.1 Utveckling

Första versionen av Direct3D introducerades i DirectX version 2.0 samt 3.0. Direct3D implementerade ursprungligen både ”behållande tillstånd” (retained mode) och ”omedelbar tillstånd” (immediate mode) i tredimensionella API:n. Behållande tillståndet använde gamla samt icke flexibla tekniker. Utvecklaren krävde mer direkt kontroll över hårdvarans aktiviteter som detta tillstånd inte kunde erbjuda. Microsoft lämnade behållande tillståndet bort efter DirectX 3.0. Första versionen av omedelbara tillståndet baserade sig på programmeringsmodell som använde exekverande buffert. Exekverande buf-

fert allokeras i hårdvaruminnen och delas ut med hjälp av hårdvara för att utföra tredimensionell rendering. Omedelbara tillståndet används fortfarande idag, uppgraderingar och nya tekniker har dock presenterats en hel del.

Efter version 3.0 har det inom de två senaste årtionden utvecklats flera versioner av Direct3D ända tills version 11. För att bättre förstå de egenskaper och tekniker som inkluderas och används i Direct3D 11 så är det bra att vara medveten om de märkvärdigaste uppdateringar som gjorts från och med version 5.0 till version 10.1.

I Direct3D 5.0 introducerades DrawPrimitive API som eliminerade applikationernas nödvändighet att konstruera exekverande buffert. Det inkluderar metoder för att rendera sekvenser av icke indexerade geometriska primitiv av en specifik typ från gällande hårdvara.

Direct3D 6.0 introducerade ett antal egenskaper för att hantera hårdvara som användes då det släpptes ut. Som exempel multitexturer som gav möjlighet till användning av fler än en textur samtidigt på en polygon. Geometriska pipelinen var också optimerad samt ett valfritt hanteringsverktyg för texturer presenterades för att förenkla programmeringen.

I Direct3D 7.0 introducerades DirectDraw Surface (.dds) filformatet för sparande av komprimerade och icke komprimerade texturer. Det inkluderade också stöd för hårdvaruacceleration för transformationer samt ljusplacering och förmågan att utdela vertex buffrar i hårdvaruminnen.

Direct3D 8.0 introducerade vertex shader, pixel shader, texture mapping samt övriga kraftiga grafiska tredimensionella egenskaper.

Direct3D 9.0 inkluderade bättre stöd för grafisk hårdvara samt en ny version av HLSL. Stöd för flyttals textur format samt lookup för texturer i vertex shader fanns också med. I denna version presenterades också WDDM (Windows Display Driver Model) som gav en möjlighet till en hel del fördelar för operativsystemet Windows Vista och senare Microsoft operativsystem. WDDM är alltså arkitektur för drivrutiner för en grafikkort.

Det gav möjligheten för virtualisering och paging av grafikkortets minne till systemets eget hårdvaruminne, som sedan tillät avbrytning och schematisering av grafiska operationer samt fördelning av DirectX ytor över olika processer.

Direct3D 10 adderade flera egenskaper inklusive geometriska shader samt en uppdatering till shader model (shader model 4.0). Grafiska pipelinen, egenskaper för programmering samt effektiviteten för användning av processorn förbättrades också. Efter Direct3D 10 presenterades Direct3D 10.1 som inkluderade små uppdateringar så som en del bildkvalitetsstandarder för grafikkorts tillverkare som sedan gav en möjlighet för mer flexibel utveckling och bättre kontroll över befintliga egenskaper (t.ex. kantutjämning).

Direct3D 11 är tillsvidare den nyaste versionen av API:n som grundar sig på infrastruktursförbättringar som skapades i Direct3D 10 och Direct3D 10.1. Tessellation, flertråds rendering (multithreaded rendering), Compute Shader samt förbättringar för GPGPU (General purpose computing on graphics processing units) är märkvärdiga uppdateringar som inkluderades. GPGPU är en teknik för grafikprocessorn att genomföra beräkningar som traditionellt utförs av centralprocessorn.

4.2 Arkitektur

Målet med Direct3D är att genomföra kommunikationen mellan grafiska applikationer och drivrutiner för grafisk hårdvara. Det presenteras som ett abstraktionslager på en nivå som är jämförbar med Graphics Device Interface (GDI). Direct3D inkluderar dock en hel del egenskaper som saknas i GDI. Det förses med ett gränssnitt för varje tredimensionell funktion i grafikkort. Omedelbara tillståndet som Direct3D använder presenterar tre huvudsakliga abstraktioner, dessa är apparater, resurser samt swap kedjor (swap chains). Apparater är ansvariga för rendering av tredimensionella scener och resurser igen är specifik data som används under rendering. Swap kedjor är serien av virtuella rambuffert som utnyttjas av grafikkort och grafiska API:n för stabilisering av mängden rutor per sekund samt flera övriga funktioner.

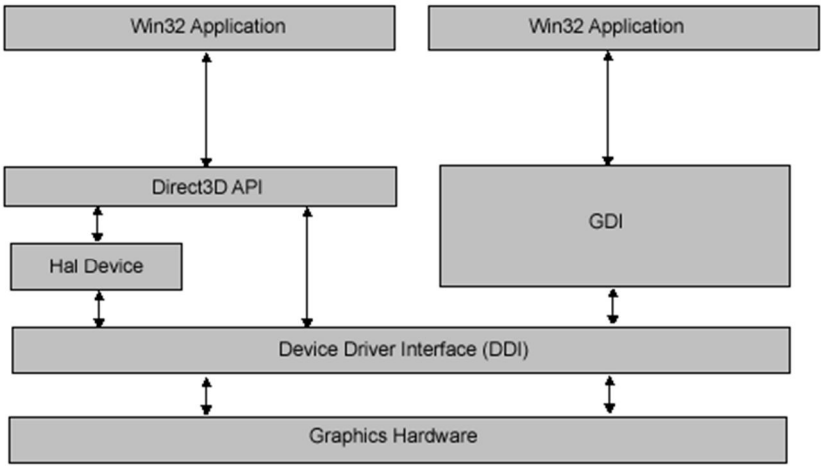
Apparater utrustas med ett gränssnitt för olika renderingsegenskaper och varje apparat inkluderar åtminstone en swap kedja. En Direct3D apparat delar ut och förstör objekt, renderar primitiv samt kommunicerar med hårdvara och drivrutiner för hårdvara. Det finns fyra typer av apparater:

- HAL (Hardware abstraction layer) apparat är de primära apparat som stöder hårdvaruacceleration för funktioner i grafiska pipelinen, baserad på egenskaper som hårdvaran stöder.
- Referensapparat som simulerar nya funktioner som inte ännu är tillgängliga i hårdvaran (kräver SDK). Kunde också kallas för referensrasteriserare.
- Null referensapparat som gör egentligen ingenting. Används då SDK icke är installerat och datorn gör en begäran på en referens apparat.
- Mjukvaruapparat som genomför rasterisering i mjukvara

Dessutom innehåller apparater ett urval av resurser. Det finns två grundtyper av resurser, dessa som måste återskapas då en apparat startas om och dessa som inte måste återskapas. Varje resurs inkluderar minst fyra definitioner:

- ”Type” som bestämmer hurdan resurs skall användas. Som exempel yta (surface), volym (volume), texturer för ytor (surface texture) och vertex buffert.
- ”Pool” som beskriver hur resursen styrs av runtime samt var det sparas.
- ”Format” som beskriver layouten av resursdata (t.ex. bit mängden) i minnet.
- ”Usage” som beskriver hur resursen används av applikationen.

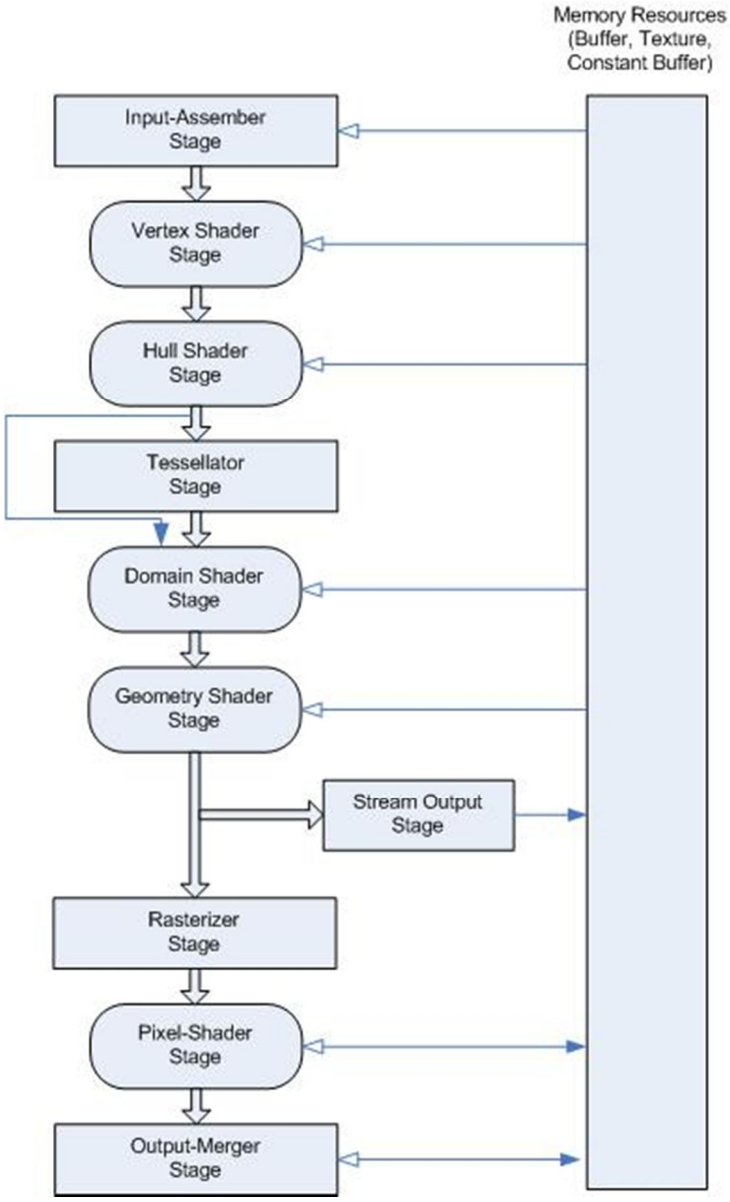
Följande diagram visar relationen mellan en Windows applikation, Direct3D, GDI och hårdvara.



Figur 3. Diagram av relationer mellan Direct3D och andra system komponenter. (Microsoft 2011)

4.3 Pipeline

Programmerbara Direct3D pipelinen är planerad för skapande av grafik i realtid för applikationer. Det definierar processen för konvertering av texturer, vertex data, buffrar och så vidare till en bild på skärmen. Följande diagram visar varje programmerbart steg i dataflödet från och med input till output.



Figur 4. Diagram av dataflödet i programmerbara Direct3D pipelinen. (Microsoft 2011)

Varje steg kan konfigureras med hjälp av Direct3D API:n. Steg som inkluderar shader element är programmerbara med hjälp av programmeringsspråket HLSL och den nyaste versionen av Shader Model som är Shader Model 5.0. Detta gör pipelinen mycket anpassningsbar och flexibel. Varje steg har ett specifikt ändamål. De tre nya stegen som introducerades i Direct3D 11 är Hull-Shader steget, Tesselator steget och Domain steget.

4.3.1 Input-Assembler

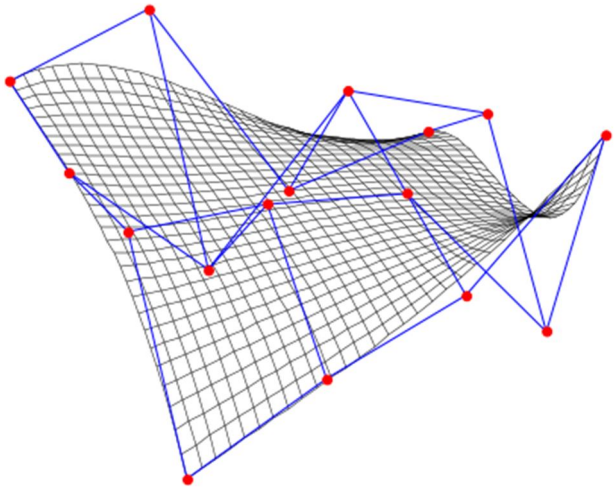
Input-Assembler (senare IA) steget ansvarar för leverering av data (trianglar, linjer, punkter) till pipelinen. Ändamålet är att läsa primitivdata från den användarfyllda bufferten och samla data i flera primitiv som sedan används av andra pipeline steg. IA steget kan utdela flera vertex till flera olika typer av primitiv (så som linjelistor, remsor av trianglar eller primitiv med grannmatris). Medan IA församlar primitivdata så har det ett annat ändamål också. Det befäster värden som är skapade av ett system (t.ex. textsträng) för att göra shader steg mer effektiva. Varje shader steg konstrueras från en allmän shader kärna, och shader kärnan använder värden som är skapade av ett system (exempelvis primitiv ID, vertex ID) så att ett annat shader steg sedan kan förminska behandlingen endast till dessa primitiv och vertex som icke ännu har använts. Så snart som IA steget har läst data från minnet så skickas det till Vertex-Shader steget.

4.3.2 Vertex-Shader

Vertex-Shader (senare VS) steget tar hand om vertex funktioner. En VS tar emot alltid en singel input vertex och producerar en singel output vertex. Steget behandlar enskilda vertex som skickas från IA och utför sedan operationer så som transformationer och ljuseffekter för dessa. Om ingen modifikation krävs så skapas det en ”pass-through” VS (”genompasserings” VS) i grafiska pipeline. VS steget måste alltid vara aktivt för att pipeline funktioner kunde exekveras.

4.3.3 Hull-Shader

Hull-Shader (senare HS) steget gör ”patchar” eller så kallade lappningar för ytor. Lappningar är rutnät som består av kontrollpunkter, dessa kontrollpunkter definierar lappningens form. Punkter kan placeras var som helst och med hjälp av dessa kan också formen av en objekt i teorin vara vad som helst. HS producerar output kontrollpunkter från input kontrollpunkter, båda ligger mellan 1 och 32 punkter. Dimensioner och användning av kontrollpunkter bestäms av användaren. Det utför också en del uträkningar för enskilda lappningar för att leverera data till tesselator steget och Domain-Shader steget.



Figur 5. Exempel på ett rutnät som består av kontrollpunkter, detta kallas till ”Bézier surface” eller ”Bézier patch”. (Wikimedia 2011)

4.3.4 Tesselator

Tesselator steget delar större definitionsmängder (eller domain) i underavdelningar av mindre objekt. Med hjälp av att implementera tessellation i grafisk hårdvara så kan en grafisk pipeline evaluera modeller med mindre detaljer (mindre antal polygoner) och rendera dessa med mera detaljer. Mjukvarutessellation är möjligt, man kan dock få en mycket större mängd av visuella detaljer med hjälp av tessellation som implementeras i hårdvara. Tessellation använder grafikprocessorn för att räkna ut en mera detaljerad yta från en yta som består av kvadratiska lappningar, triangel lappningar eller höjdkurvor. För att approximera ytan med högre detalj så delas varje lappning i underavdelningar av trianglar, punkter eller linjer. Tesselatorn opererar en gång per lappning och täcker en simpel domän i en normaliserad (noll till ett) koordinatsystem. Output för tesselator steget är koordinater samt topologi för ytor.

4.3.5 Domain-Shader

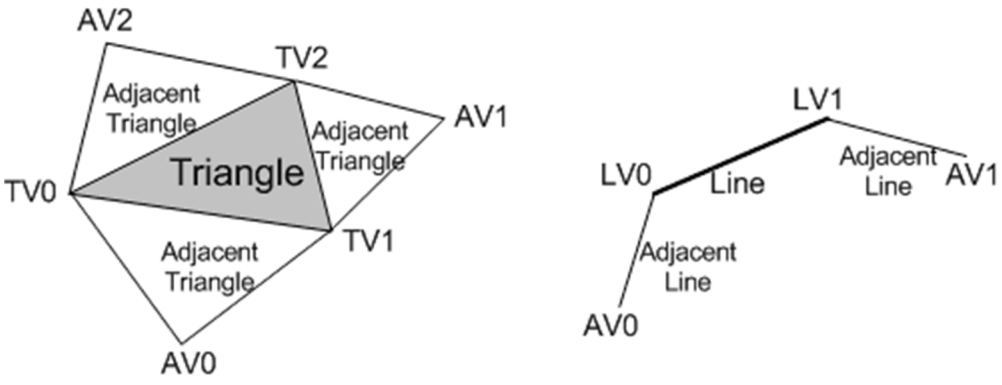
Domain-Shader (senare DS) steget räknar ut vertex positionen av en punkt i en underavdelning, det sker i output steget och/eller lappningen. DS körs en gång per tesselator stegets output punkt och har endast läsrättigheter för tesselator stegets koordinater, HS stegets output lappning samt dess konstanter. Output är positionen av en vertex. När

DS steget är färdigt så slutförs också tessellationen och pipeline data fortsätter till nästa pipeline steget.

4.3.6 Geometry-Shader

Geometry-Shader (senare GS) steget utför processer för hela primitiv. Input i detta är en fullständig primitiv som består till exempel av tre stycken vertex för en triangel, två för en linje eller en för en punk. Varje primitiv kan också inkludera vertex data för vilken som helst bredvid liggande primitiv med grannmatris. GS kan förneka en input primitiv eller sedan skicka ut en eller flera primitiv. Det använder applikationsspecifikt shader kod med flera vertex som en input och har förmågan att generera flera vertex i output.

Följande illustration är ett exempel på en triangel och en linje med flera vertex i grannmatris.



Figur 6. Illustration av en triangel och en linje med vertex i grannmatris. (Microsoft 2011)

Tabell 1. Förklaring till förkortningar i Figur 5.

TV	Triangel vertex
AV	Vertex i grannmatris
LV	Linje vertex

Till GS kan det implementeras algoritmer för att vidare utvidga och generera egenskaper som skickas ut. GS output kan matas in till rasteriserings steget och/eller till en vertex buffert i minnet med hjälp av Stream-Output steget.

4.3.7 Stream-Output

Stream-Output (senare SO) steget är planerat för en primitivs dataflöde (då det är på väg för rasterisering) från pipelinen till primärminnet. Meningen med SO steget är att ständigt strömma vertex data från GS steget (eller VS steget om GS steget är inaktiv) till en eller flera buffert i minnet. Data som skickas till primärminnet kan cirkuleras tillbaka till pipelinen som input data eller förbrukas på nytt av Centralprocessorn. Data kan strömmas ut och/eller skickas in för rasterisering.

4.3.8 Rasterizer

Rasterizer steget eller ”Rasteriserings steget” ansvarar för klippning och tillredning av en primitiv för pixel shader och bestämmer hur pixel shader skall anropas. Steget konverterar vektor information (tillsammans med former eller primitiv) till en raster bild (tillsammans med pixelinformation) för att ställa ut tredimensionellt grafik i realtid. Varje primitiv förvandlas till pixlar inom rasterisering och per-vertex värden interpoleras genom varje primitiv.

4.3.9 Pixel-Shader

Pixel-Shader (senare PS) steget tar emot interpolerad data för en primitiv och skapar data för pixlar, så som färger. Det möjliggör rika effekter så som skapande av per-pixel belysning samt postprocesser. PS är ett program som kombinerar konstanta variabler, data för texturer, interpolerade värden för varje vertex och övrigt data för att skapa en output. Rasteriseringssteget anropar PS en gång för varje pixel som är täckt av en primitiv. PS inkluderar också övriga funktioner så som producering eller derivering av kvantitet på tanke om skärmrymden x och y. Exempelvis värdering och uträkning av mängden av detaljer för texturer.

4.3.10 Output-Merger

Output-Merger (senare OM) steget ansvarar för anslutande av olika typer av output data (så som pixel shader värden och information om djupet) inklusive rendering ändamålet samt buffert information för att sedan skapa den slutliga pipeline resultaten. Buffert in-

formation kan inkludera information och/eller koordinater om t.ex. bilddjupet (styrs av ”depth buffer” eller ”z-buffer”) eller schablonen (styrs av stencil buffert). OM steget genererar den slutliga pixel färgen med hjälp av egenskaper i pipeline tillståndet, så som pixel data genererad med PS, innehållet av rendering ändamål och innehållet av flera djup/schablon buffert. OM steget är det slutliga steget som fastställer vilka pixlar är synliga samt sammanblandar de slutliga färger för pixlar.

4.4 Egenskaper i Direct3D 11

Direct3D 11 introducerade nya egenskaper för programmerbara pipelinen för att förbättra grafiska egenskaper som används i realtid i mjukvaruapplikationer (huvudsakligen i spel). Förbättringar för en hel del egenskaper gjordes också. Till nya egenskaper inkluderas tessellation, shader egenskaper (som inkluderas i Shader Model 5.0), stöd för flera trådar (multithreading), bättre stöd för hårdvara och så vidare.

Egenskaper som finns i tidigare versioner av Direct3D (t.ex. Direct3D 10) finns antingen inkluderade i Direct3D 11 eller sedan används det mera avancerade tekniker som har ersatt de äldre. Det finns också tekniker från äldre versioner av Direct3D som har lämnats bort på grund av att man har kommit på nya sätt att fullborda egenskaper som förr gjordes på annat sätt.

Grafiska egenskaper så som anti-aliasing (kantutjämning), texturfiltrering (i detta fall anisotropisk filtrering) samt ambient occlusion är populära tekniker som inkluderades i senaste versioner och stöds av Direct3D 11 också.

Anti-aliasing eller kantutjämning är en renderingsteknik som används för att skapa en sken av högre upplösning än vad som egentligen används. Skärpan på linjer ”tunnas” ut genom att ändra på färger i närliggande pixlar och på detta sätt ser man jämnare kant i scenen. Det finns varierande tekniker som används för kantutjämning men för grafiskt bruk används det ofta idag multisampling eller supersampling. Simpelt sagt så använder de olika tekniker som bestämmer färgen för varje singel pixel på skärmen med hjälp av flera pixlar och sedan kombinerar dessa. Detta möjliggör ännu jämnare kanter. En annan relativt ny kantutjämnings metod är transparent (genomskinlig) kantutjämning. Transpa-

rent kantutjämning fungerar så att det tar kantutjämnings mönster från polygoner där det används transparenta texturer för att skapa effekter. Dessa effekter syns sedan genom objekt så som staket eller träd.

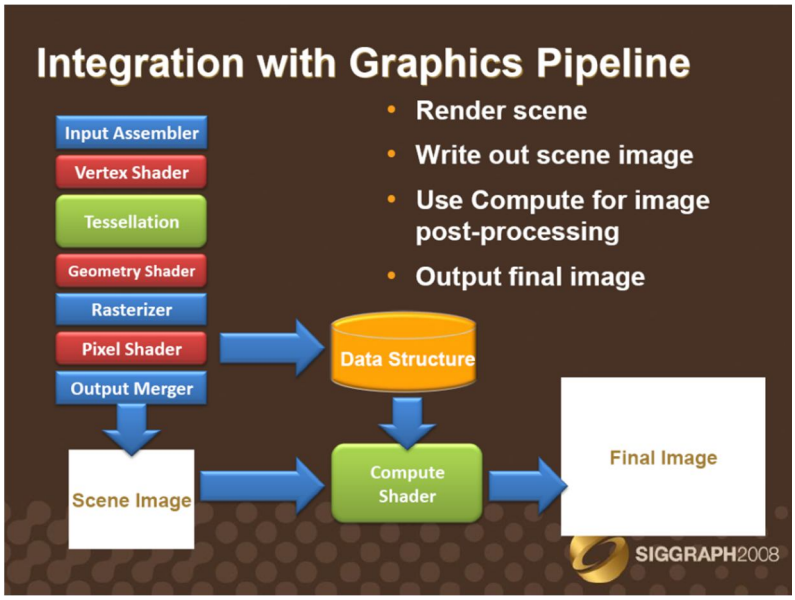
Anisotropic filtering eller anisotropisk filtrering är en metod för att förbättra kvaliteten av texturer på ytor som är i sned vinkel i avseende till vart projekteringen av texturer framställs (icke vinkelrätt). Det används mapping tekniker så som mipmapping för att genomföra uträkningar som sedan bestämmer hur bra kvaliteten blir.

Ambient occlusion är en teknik för att skapa realism i scenen via belysning och skuggor. Det räknar ut hur ljus faller på objekt för att skapa skuggor och hur egenskaper av ljus och skuggor ändras då distans på objekt i scenen ändras. Ambient occlusion tekniker försöker avgöra var en punkt i en scen existerar relativt till andra punkter samt effekten som ljusen har till detta och andra delar av scenen (också till dessa delar som direkt blockeras av ljusresursen). En ny teknik inom ambient occlusion är High Definition Ambient Occlusion och målet med det är att skapa skuggor i plats där det är svårt för ljuset att nås. Dessa mål kan nås med hjälp av flera sampel från djuphetsbuffert och accelereras till exempel med att utnyttja Compute Shader.

4.4.1 Compute shader

Compute shader (”värderings shader”) är en programmerbar shader planerad för parallell bearbetning av data. Compute shader teknologi kallas också till DirectCompute teknologi. Teknologin tillåter grafikprocessorn att använda flera operationer samtidigt. Det använder liknande tekniker för input och output åtkomst som övriga programmerbara pipeline shader funktioner men är inte direkt ansluten till dessa. En compute shader stöder GPGPU med hög hastighet och kan utnyttja parallella processorer på grafikkort. Det kan använda minnesfördelning och trådsynkroniseringsegenskaper för att ge mera effektiva parallella programmeringsmetoder. En compute shader integreras i Direct3D och är tillgänglig med hjälp av en Direct3D apparat. Med hjälp av en Direct3D apparat kan compute shader direkt fördela minnesresurser med andra grafiska shader.

Compute shader möjliggör mer generella datastrukturer och algoritmer för att skapa bättre grafik för interaktiv media och datorspel. Ändamålet kan vara postprocess egenskaper, fysiska effekt (partiklar, rök, vatten osv.), artificiell intelligens, animationer och så vidare.



Figur 7.Compute shader integrering med grafiska Direct3D pipeline (Boyd, Chas.2008)

4.4.2 Dynamic Shader Linking

Renderande system måste arbeta med invecklad beskaffenhet då de styr shader funktioner för att ge en möjlighet till optimerad shader kod. Det är en enorm utmaning för att shader funktioner måste stöda ett urval av material i en renderad scen mellan varierande hårdvaru konfigurationer. Direct3D 11 och shader model 5.0 introducerade därför ett konstruerad och objektorienterad programmeringsspråk som kallas Dynamic Shader Linking. Det förses med runtime stöd av shader kommunikation för att hjälpa utvecklaren att programmera shader egenskaper och shader ombyten.

4.4.3 Multithreading

Datorsystem med flera kärnor (med centralprocessorer med flera kärnor) har blivit mycket allmänna idag. Direct3D 11 inkluderar stöd för flera trådar (kallas till mul-

tithreading) som möjliggör effektiv samverkan mellan dessa i Centralprocessorn och Direct3D 11 API:n. Fastän det har funnits processorer med flera kärnor och trådar en relativt lång tid redan så har det använts endast en singel tråd för rendering. Övriga trådar har använts för ljud, fysik, dekompression av resurser och så vidare. Rendering är en mycket tung process för både centralprocessorn och grafikprocessorn, stöd för flera trådar underlättar dessa. Resurser skapades i föregående versioner i en renderings tråd. I multithreading används trådar som kan läsa samt dekomprimera texturer från hårddisken och fylla in resurser (Direct3D objekt) som igen skapas i huvudtråden.

I multithreading kan en programmerare skapa en objekt (för en apparat) per tråd som sedan används för att ladda resurser. Synkronisering i apparatens funktioner är mer ekonomiskt för centralprocessorn och möjliggör snabba exekveringar.

4.4.4 Tessellation

Tessellation är en teknik som kan användas för rendering av en enstaka modell med varierande nivå av detaljer. Detta skapar en geometriskt noggrann modell som är beroende av detaljer som behövs för scenen. Tessellation implementeras i grafikprocessorn för beräkning av jämnare ytor. Grafikprocessorn genererar mera polygoner (ofta trianglar) från existerande geometriska egenskaper och använder tessellations motorn som är en del av integrerade kretsen i grafikkort. Tessellation görs för en 3D-modell och inte för flata ytor. Det kan användas för flera olika ändamål och möjliggör skapandet av mer noggranna, realistiska och runda ytor.

Potentialiteten är enorm. Tack vare tessellation är det möjligt att implementera nivån av detaljer direkt till grafikprocessorn utan att använda normal mapping. Det tillåter mycket detaljerade modeller, alltså i teori flera miljoner polygoner istället för cirka tiotusentals som ofta används i dagens applikationer och spel.
(Lloyd, 2010)

Figur 8. Yta av vatten utan tessellation. (Lloyd Case, Maximum PC 2010)

Figur 9. Yta av vatten med tessellation.(Case Lloyd, Maximum PC 2010)

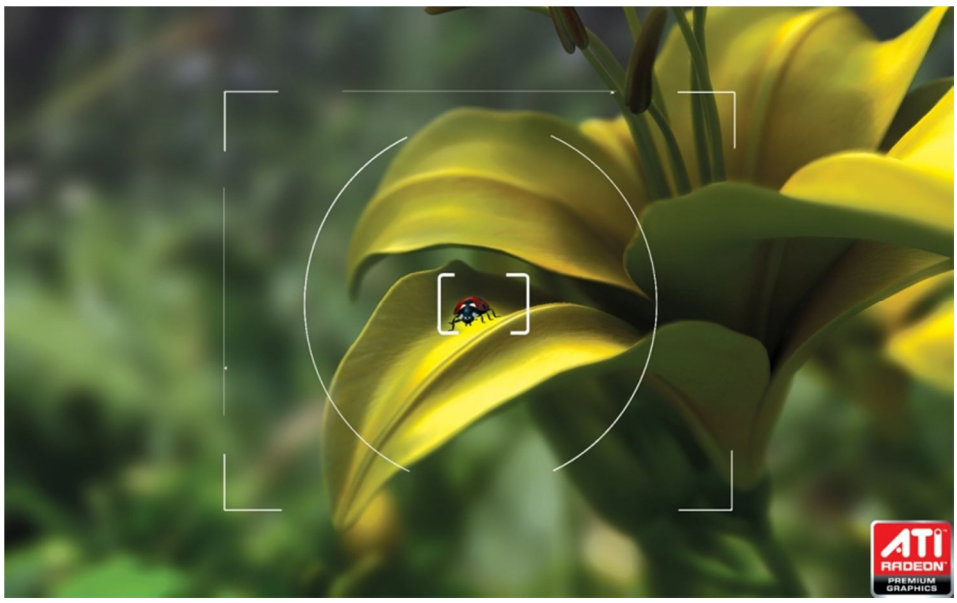
4.4.5 Effekter i postprocesseringen

Effekter i postprocess är sådana effekter som skapas för tredimensionella bild efter att de har renderats. Tekniken presenterades redan i DirectX 9 men har inte använts i applikationer så mycket förrän DirectX 10 och nu i DirectX 11 har det blivit allt mer vanligare. Nya tekniker har introducerats och ett brett urval av effekter är möjliga; som exempel plandjupet (depth of field), partiklar på skärmen, värme/vatten distorsion, dyna-

misk suddighet och ”bokeh”. Bokeh är suddigheten som man ser i fotografier på icke fokuserade areor. Många av dessa effekter kan användas för att addera realism till scenen, så som värme ovanför eld. Andra effekter kan igen göra scen mindre realistiska men mera cinematiska, så som plandjupet och bokeh effekter.

Postprocesserade effekter skapas oftast med hjälp av shader program som finns i Direct3D API:n. Det finns dock program med hjälp av vilka man kan skapa postprocesseffekter fastän de icke nödvändigtvis hör till existerande grafiska API:n. Grafikprocessorn i sig själv måste ändå alltid vara programmerbar för dessa egenskaper.

I Direct3D 11 kan effekter i postprocesseringen användas i samverkan med varandra och det kan skapas multipla effekter i en scen på grund av att shader rutiner kan anropas då de krävs. Detta ger utvecklaren möjligheter för t.ex. granulerad kontroll över effekter och hjälper användaren att fokusera på det som är nära och/eller långt borta på skärmen (så som man skulle ta bild med en kamera).



Figur 10. Exempel på ett fokuserat postprocesseffekt med kort plandjup och mycket bokeh. (Case Lloyd, Maximum PC 2010)

Figur 11. Exempel på ett fokuserat postprocesseffekt med längre plandjup och mindre bokeh (jämfört med figur 10). (Case Lloyd, Maximum PC 2010)

4.5 Övriga egenskaper

Några andra exempel på nya egenskaper i Direct3D 11 är en del utvidgade texturer, minnes stöd för fyra gigabyte (eller mera) för resurser samt output förbättringar. Övriga egenskaper och förbättringar är för största delen sådana som behandlar programmeringsspråket och underlättar programmering.

Tekniker som presenterades är endast de märkvärdiga visuella egenskaper och förbättringar som introducerades i Direct3D 11 och de är egentligen endast ett skrap på ytan om man tänker på helheten av Direct3D.

4.5.1 Prestanda

Direct3D 11 är planerad att stöda både ny och existerande hårdvara, från och med DirectX 9 till DirectX 11. En Direct3D kompatibel apparat utför operationer för belysning, transformationer, rasterisering och så vidare. I Direct3D 11 presenterades en ny teknik som består av så kallade ”feature levels”. Feature level är en noggrant definierad uppsättning av grafikprocessorns funktionaliteter. Därför är det möjligt med hjälp av

feature level egenskaper att köra en Direct3D applikation fastän Direct3D versionen på hårdvaran icke stöder det. Det krävs dock hårdvara med relativt hög prestanda (”high-end”) för att framställa och klara av de avancerade grafiska egenskaper som Direct3D 11 erbjuder idag, beroende på kvaliteten samt grafiska krav för applikationen som man använder. Upplösningar som används i grafisk presentation har blivit mycket höga idag och det kräver också mera av grafikkortet och grafikprocessorn. Höga upplösningar behöver mycket matematiska uträkningar (och inkluderar mera pixlar) och därför krävs det mera prestationskraft desto högre upplösningen är. Nya egenskaper i Direct3D 11 kan också ha stora effekter på prestandan, men nya generationen av grafikkort ger en möjlighet till en mycket bättre visuell precision medan prestationen hålls på bra nivå.

4.6 Framtid

Direct3D 11 var ett stort steg för tredimensionellt datorgrafik och det verkar som att höga krav på kvaliteten i visuella egenskaper och realism blir alltmer vanligt i applikationer och datorspel. Man ser effekter som man förr såg endast i film och TV, och programmerare har börjat använda effekter för att skapa deras applikationer liknande till dessa.

Microsoft har inte gett ut eller avslöjat egenskaper som kommer att finnas med i framtidens versioner av Direct3D. Det beror troligtvis på det att Direct3D 11 är tills vidare en så ny produkt och dess potentiella egenskaper har ännu så mycket att ge till applikations- och spelindustrin.

En teknik som man dock har hört mycket tal om och som några till och med förväntade för Direct3D 11 är Ray Tracing. Dessa förväntningar fullbordades inte, och det finns inget fakta om de kommer någonsin att fullbordas för Direct3D bruk. Man kunde dock tänka sig att det vore en av framtidens tekniker. Ray tracing används redan i effekter (skapade med datorgrafik) som man ser i TV reklamer och filmer. Det är en teknik för skapande av en bild med hjälp av att spåra (”trace”) hur ljus förflyttas genom pixlar som syns på skärmen och sedan simulera effekter då de möter virtuella objekt. Ray tracing är kapabel att simulera varierande optiska effekter så som reflektioner, refraktioner, ljusutspridning och kromatiska effekt. Tekniken möjliggör produktion av visuellt realism av

mycket högt grad, ofta högre än typiska rendering metoder men med en förlust på prestation. Istället för att behandla polygoner så fungerar ray tracing så att det spårar en väg (via ett imaginärt öga) genom varje pixel i en virtuell skärm och räknar sedan färgen av objektet som är synlig genom detta. Scener beskrivs antingen matematiskt av en programmerare eller av en visuell artist (med hjälp av förmedlande verktyg). Scener kan också inkludera data från bilder och modeller.

Möjligtvis den största orsaken till att Ray tracing icke ännu har slått igenom i grafiska applikationer och spelindustrin är dess (mycket) höga prestandakrav. Ray tracing använder mycket mera resurser än polygon-baserad rendering för att man borde förvandla pixlar till polygoner och det kräver mycket arbete jämfört med att gå från polygoner till pixlar då det krävs relativt lite uträkningar.

Figur 12. Visuellt realistisk grafik skapad med Ray tracing. (Wikimedia, 2008)

4.7 Direct3D mot OpenGL

Före DirectX så hade Microsoft inkluderat OpenGL i sin Windows NT plattform. OpenGL krävde då ”high-end” hårdvara och fokuserade på ingenjörsvetenskap samt CAD bruk. Direct3D planerades att bli en approximativ partner till OpenGL, uppmärksamheten riktades för spelbruk. När tredimensionellt spelande blev mer populärt så utvecklades OpenGL för ha bättre stöd för programmeringstekniker i interaktiva multimedia applikationer. Det gav utvecklaren en möjlighet att välja mellan OpenGL eller Direct3D som den grafiska 3D API:n för sina applikationer. Hellre än att adoptera OpenGL som en API för spelbruk så valde Microsoft att fortsätta förbättra Direct3D. Efter det har båda utvecklats till och med var de är idag, tävlandet har varit hårt.

Många tror att Direct3D och OpenGL är samma sak. Teknikerna i dessa skiljer sig dock drastiskt. Direct3D är planerat för virtualisering av 3D hårdvarugränssnitt, så programmeraren är inte bunden till hårdvaran. OpenGL igen är planerat till en 3D accelererad hårdvarusystem som implementeras i mjukvara/programvara. Så simpelt sagt, Direct3D antar att applikationen styr hårdvaruresurser och OpenGL gör implementeringar för detta.

Många spel använder tills vidare OpenGL, men det är svårt att förneka att en stor del av utvecklaren för ”high-end” PC-spel gillar främst DirectX idag. OpenGL är dock populärt för professionella 3D-utvecklare samt i mobilutveckling. Medan nya versioner av OpenGL har hållits uppdaterade med några egenskaper som finns i DirectX, måste de vanligtvis ändå implementeras med hjälp av extensioner i stället för den huvudsakliga API:n. Ett exempel på en sådan teknik är Geometry Shader. Microsoft har allt efter fortsatt att skapa avsevärda uppdateringar för att förbättra API:n (oberoende av kompatibilitets förlust), medan OpenGL har gjort ungefär tvärtom på grund av kompatibilitets orsaker. Man kunde säga att det är trögheten (alltså höga prestationskrav) för Direct3D som gör att programmeraren vill stanna kvar med OpenGL.
(Hardwidge 2011)

5 SLUTSATSER

Programmerbara shader och grafiska egenskaper tog ett stort steg framåt med DirectX 11. Om programmerare förr önskade att fylla flera effekter till en scen så blev shader programmet enormt och problematiskt. Nu kan de anropa shader rutiner då de behövs. Det ger en möjlighet till effektivt skrivande av shader program och mera framgångsrikt användning av effekter. Utvecklare har redan använt de visuella förmåner som Direct3D 11 erbjuder så som man ser i dagens applikationer och spel.

Jämförelser beträffande prestanda mot visuella effekter roterade förr runt anti-aliasing och anisotropisk filtrering. DirectX 10 adderade några nya metoder för applikationsutvecklarnas arsenal men dessa hade många negativa effekter på prestanda. Nya egenskaper i DirectX 11 kan förstås också påverka prestandan men på grund av nya generationen av grafikkort är det möjligt att köra mycket högre visuellt precision medan prestandan hålls resonabel. Bra nyheter för detta är också intrycket som Direct3D 11 har gjort för DirectX 11 kompetenta grafikprocessorer. Inverkan på utvecklingen har varit snabbt och populariteten i spelindustrin har varit enormt inom senaste tider.

Med hjälp av nya generationen av DirectX 11 egenskaper och hårdvara är det möjligt att njuta av högre kvalitet av visuella förmåner mer än någonsin förr. Det beror nu på utvecklaren och programmeraren samt hur effektivt de utnyttjar dessa egenskaper. Man kunde i alla fall tro att det vore möjligt för Direct3D att utvecklas till den huvudsakliga programmeringsomgivningen för applikationer och spel i framtiden.

KÄLLOR

Boyd, Chas. SIGGRAPH 2008. DirectX 11, Compute Shader [www]. Hämtat 6.4.2011.
http://s08.idav.ucdavis.edu/boyd-dx11-compute-shader.pdf

Coles, Olin. 2009. NVIDIA GeForce 3D Vision Gaming Kit [www]. Hämtat 6.4.2011.
http://benchmarkreviews.com/index.php?option=com_content&task=view&id=276&Itemid=58

Eisler, Craig. 2006. DirectX Then and Now (Part1) [www]. Hämtat 11.4.2011.
http://craig.theeislers.com/2006/02/directx_then_and_now_part_1.php

Epic Games. 2011. Epic Games Releases March 2011 Unreal Development Kit Beta. Hämtat 11.4.2011.
http://www.unrealengine.com/news/epic_games_releases_march_2011_unreal_development_kit_beta/

Fedy, Abi-Chahla. 2008. OpenGL 3 & DirectX11: The War Is Over. Hämtat 2.4.2011.
http://www.tomshardware.com/reviews/opengl-directx,2019.html

Franklin, Curt. 2008. How 3-D Graphics Work [www]. Hämtat 1.4.2011.
http://computer.howstuffworks.com/3dgraphics.htm

Hardwidge, Ben. 2011. Carmack: Direct3D is now better than OpenGL. Hämtat 9.5.2011.
http://www.bit-tech.net/news/gaming/2011/03/11/carmack-directx-better-opengl/1

Lloyd, Case. 2010. What DirectX 11 is, and What It Means to You [www]. Hämtat 11.4.2011.
http://www.maximumpc.com/article/features/directx_11_deconstructed?page=0,0

Luna, Frank D. 2006. Introduction to 3D game programming with DirectX 9.0c: a shader approach [www]. USA: Wordware Publishing, Inc. 617s.
ISBN-13: 978-1-59822-016-2
ISBN-10: 1-59822-016-0
Hämtat 16.2.2011.
http://www.google.com/books?hl=fi&lr=&id=is6Wc3hKSWIC&oi=fnd&pg=PR5&dq=ISBN-13:+978-1-59822-016-2&ots=_R2NlgaLco&sig=IP9dCfoH7rthNDl4Yfh60-DfQ8Y#v=onepage&q=ISBN-13%3A%20978-1-59822-016-2&f=false

Microsoft. 2010. DirectX Developer Center [www]. Hämtat 10.11.2010.
http://msdn.microsoft.com/en-us/directx/default.aspx

Microsoft. 2008. DirectX Software Development Kit [www]. Hämtat 13.4.2011.
http://www.microsoft.com/downloads/en/details.aspx?FamilyId=5493F76A-6D37-478D-BA17-28B1CCA4865A&displaylang=en

Microsoft. 2011. Direct3D Architecture (Direct3D 9) [www]. Hämtat 19.4.2011.
http://msdn.microsoft.com/en-us/library/bb219679%28v=vs.85%29.aspx

Microsoft. 2011. Direct3D 11 Features [www]. Hämtat 19.4.2011
http://msdn.microsoft.com/fi-fi/library/ff476342%28v=VS.85%29.aspx#Full

Microsoft. 2011. Programming Guide for Direct3D 11 [www]. Hämtat 18.4.2011.
http://msdn.microsoft.com/fi-fi/library/ff476345%28v=VS.85%29.aspx

Microsoft. 2011. Shader Stages [www]. Hämtat 2.4.2011.
http://msdn.microsoft.com/fi-fi/library/bb205146%28v=VS.85%29.aspx

Microsoft. 2010. Windows DirectX Graphics Documentation [www]. Hämtat 10.11.2010.
http://msdn.microsoft.com/en-us/library/ee663301%28v=VS.85%29.aspx

OpenGL. 2011. The Industry's Foundation for High Performance Graphics [www] Hämtat 10.11.2011.
http://www.opengl.org/documentation/

Shirley, Peter; Marschner, Steve. 2009. Fundamentals of Computer Graphics – Third Edition. USA: Library of Congress Cataloging-in-Publication Data. 752s.
ISBN 978-1-56881-469-8

Smalley, Tim. 2008. DirectX 11: A look at what's coming [www]. 11.4.2011.
http://www.bit-tech.net/bits/2008/09/17/directx-11-a-look-at-what-s-coming/1

Tyson, Jeff; Tracy V. Wilson. How Graphics Cards Work [www]. Hämtat 1.4.2011.
http://computer.howstuffworks.com/graphics-card.htm

Unigine. 2011. Heaven Benchmark [www]. Hämtat 11.4.2011.
http://unigine.com/download/

Wikimedia. 2008. Bézier_surface_example.svg [www]. Hämtat 4.5.2011.
http://upload.wikimedia.org/wikipedia/commons/b/bf/B%C3%A9zier_surface_example.svg

Wikimedia. 2011. Cannonball_stack_with_FCC_unit_cell.jpg [www]. Hämtat 3.4.2011.
http://upload.wikimedia.org/wikipedia/commons/e/e9/Cannonball_stack_with_FCC_unit_cell.jpg

Wikimedia. 2008. Glasses_800_edit.png [www]. Hämtat 6.4.2011.
http://en.wikipedia.org/wiki/File:Glasses_800_edit.png

Wikimedia. 2011. VectorBitmapExample.svg [www]. Hämtat 16.2.2011.
http://upload.wikimedia.org/wikipedia/en/a/aa/VectorBitmapExample.svg

Wikipedia. 2010. Comparison of OpenGL and Direct3D [www]. Hämtat 10.11.2010. Uppdaterad 10.11.2010.

http://en.wikipedia.org/wiki/Comparison_of_OpenGL_and_Direct3D

Wikipedia. 2010. Computer graphics (computer science) [www]. Hämtat 26.1.2011. http://en.wikipedia.org/wiki/Computer_graphics_%28computer_science%29

Wikipedia. 2010. DirectX [www]. Hämtat 10.11.2010. Uppdaterad 3.11.2010. http://en.wikipedia.org/wiki/DirectX

Wikipedia. 2010. Microsoft Direct3D [www]. Hämtat 10.11.2010. Uppdaterad 3.11.2010. http://en.wikipedia.org/wiki/Microsoft_Direct3D

Wikipedia. 2011. Raster graphics [www]. Hämtat 16.2.2011. Uppdaterad 12.2.2011. http://en.wikipedia.org/wiki/Raster_graphics

Wikipedia. 2011. Vector graphics [www]. Hämtat 16.2.2011. Uppdaterad 15.2.2011. http://en.wikipedia.org/wiki/Vector_graphics

Wikipedia. 2011. 2D Computer graphics [www]. Hämtat 16.2.2011. Uppdaterad 15.2.2011. http://en.wikipedia.org/wiki/2D_computer_graphics

BILAGOR