TECHNICKÁ UNIVERZITA V LIBERCI
FAKULTA STROJNÍ

DIPLOMOVÁ PRÁCE

3D VIZUALIZACE VĚTRNÉ ELEKTRÁRNY

Pavel Novák

Liberec 2003

Technická univerzita v Liberci
Fakulta strojní
Obor 23-40-8
Automatizované systémy řízení ve strojírenství

3D Vizualizace větrné elektrárny

Pavel Novák

Vedoucí diplomové práce: ing. Michal Moučka
Technická univerzita Liberec
Konzultant: Viktor Priesel
Compic Praha

Rozsah práce a příloh:

Počet stran textu:	58
Počet obrázků:	38
Počet tabulek:	3
Počet grafů:	2
Počet příloh:	1 CD

Bude zadaní

Téma:

3D Vizualizace větrné elektrárny

Anotace :

Tato diplomová práce se zabývá 3D vizualizací větrné elektrárny pro zobrazení vstupních a výstupních veličin. Její součástí je i respektování možných havarijních stavů během provozu.

Subject:

3D Visualisation of wind power plant

Annotation:

This Diploma work deals with 3D visualisation of wind power plants to depict input and output quantities. His part is also respecting possible crash situations during working.

Prohlášení:

Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.

V Liberci, dne 23.května 2003

Pavel Novák

Poděkování:

Na tomto místě bych rád poděkoval ing. Michalu Moučkovi, za jeho ochotu a důvěru, kterou mi umožnil tuto práci začít a dodělat i přes její náročnost. V neposlední řadě mi poskytoval také konzultace, což bylo významnou pomocí při vlastní tvorbě práce.

V Liberci, dne 23.května 2003

OBSAH

1. Úvod:

Počítačová grafika se v dnešních dobách dostala do drtivé většiny všech oborů. K jejímu tak rozsáhlému rozšíření vede zejména snadná rekonstrukčnost, snadné provádění změn a zejména úspora vynaložených financí při různých simulacích.

Každé jednotlivé rozhraní se odlišuje podle možnosti využití, podle použitého programovacího jazyka a také podle hardwaru. Například pro zobrazování grafického uživatelského rozhraní pod systémem Windows se používá GUI (Graphical User Interface). Při programování na platformě .NET pro komunikaci s uživatelem použijeme fce. GDI+, což je grafický subsystém Windows a jsou pro něj použitelné programovací jazyky jako např. C# nebo Visual Basic .NET. Oproti tomu, DirectX je použitelné pro práci s veškerými multimediálními aplikacemi, úzce spolupracuje s hardwarem a při zobrazování grafiky je mnohem rychlejší pro náročné aplikace z důvodů rozdílné architektury rozhraní.

Při zobrazování aplikací pro technickou praxi bez potřeby jejich animace se používá CAD systémů pro 2D, nebo 3D řešení, což jsou asi nejpoužívanější programy pro podporu návrhu součástí ve strojírenství, návrhu budov a jejich statických výpočtech ve stavebnictví. Při jejich animaci se nejčastěji užívá rozhraní OpenGL, což je standard pro rozhraní podporující 3D vykreslování a grafickou akceleraci. Tato softwarová knihovna má obrovskou výhodu pro všestrannost použití na systémech jako: Windows, MacOS,Linux a Unix systémy.

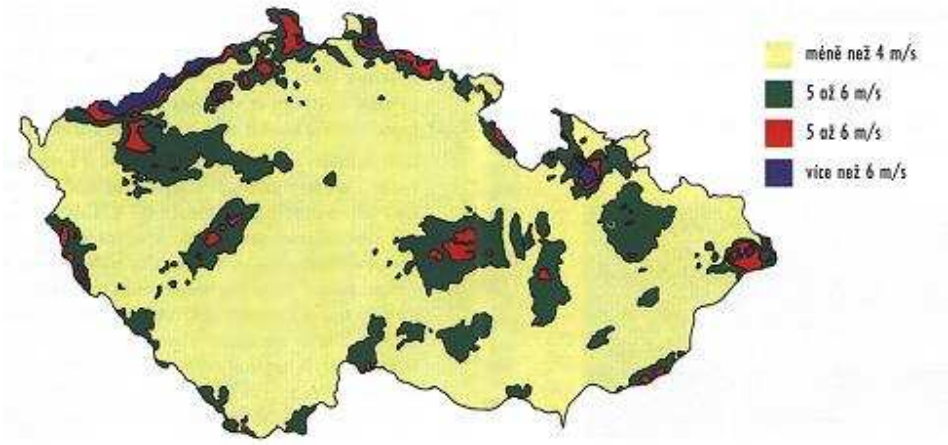
Počítačová grafika zasáhla do většiny oborů a není v mých silách na úvod rozebrat veškeré detaily. Při zmínění hlavních výhod na začátku tohoto odstavce, vede jistě také používání počítačové grafiky k výraznému zrychlení práce po připojení veškerých počítačových subsystémů dohromady v jednu integraci, kdy počítače řídí většinu prací pro zefektivnění výroby.

2.Větrná elektrárna,problémy, využití, technické parametry

Větrná energetika je obnovitelným zdrojem energie, ale může být použita pouze tam, kde má vítr alespoň průměrnou roční rychlost 4 m/s . Jde použít i při menších rychlostech , ale to se spíše vyplatí jiný zdroj energie.

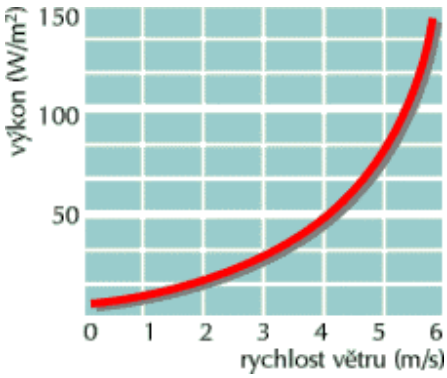
Energie z větru se samozřejmě nedá využívat všude.

- Vyrábí celkem veliké množství hluku
- Narušují krajinu svým vzhledem



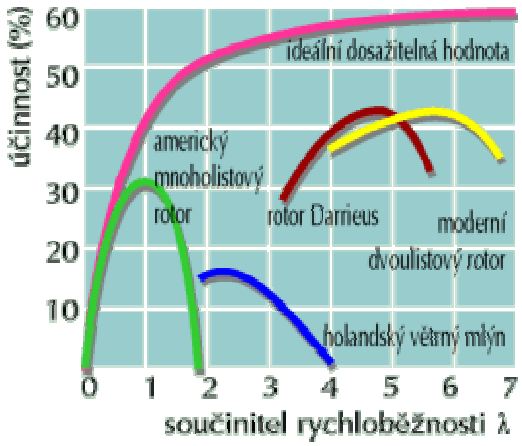
Obr.1: Průměrná rychlost větru v České republice

Slunce otepluje nerovnoměrně zemi a předává asi 2 % dopadajícího záření na vítr. Technicky z tohoto zdroje lze získat 26 000 TWh elektřiny ročně. Přitom svět vyrábí jen 9000 TWh ročně. Celkový výkon větrných elektráren je jen 2 000 MW .



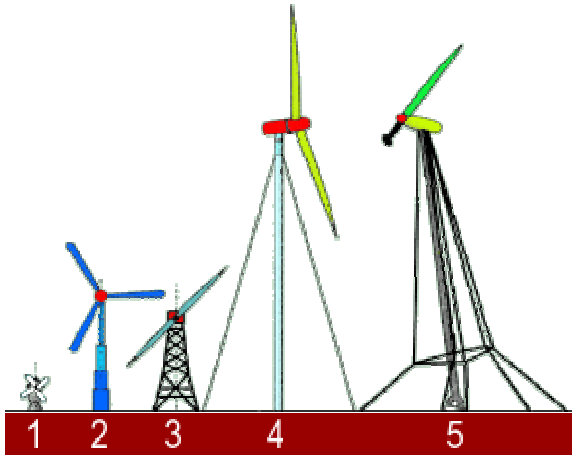
Obr.2: Závislost rychlosti větru na vyráběném výkonu

Maximální teoretická účinnost je 59% , ale kvůli tření atd. dosahují dnešní elektrárny pouze účinnosti 45% . Účinnost závisí především na tvaru lopatek.



Obr.3: Graf účinnosti

Zde je porovnání některých typů větrných elektráren:



Obr.4: Typy větrných elektráren

Porovnání několika možných velikostí a typů větrných elektráren:

1) malý typ o výkonu kolem 90 kW,

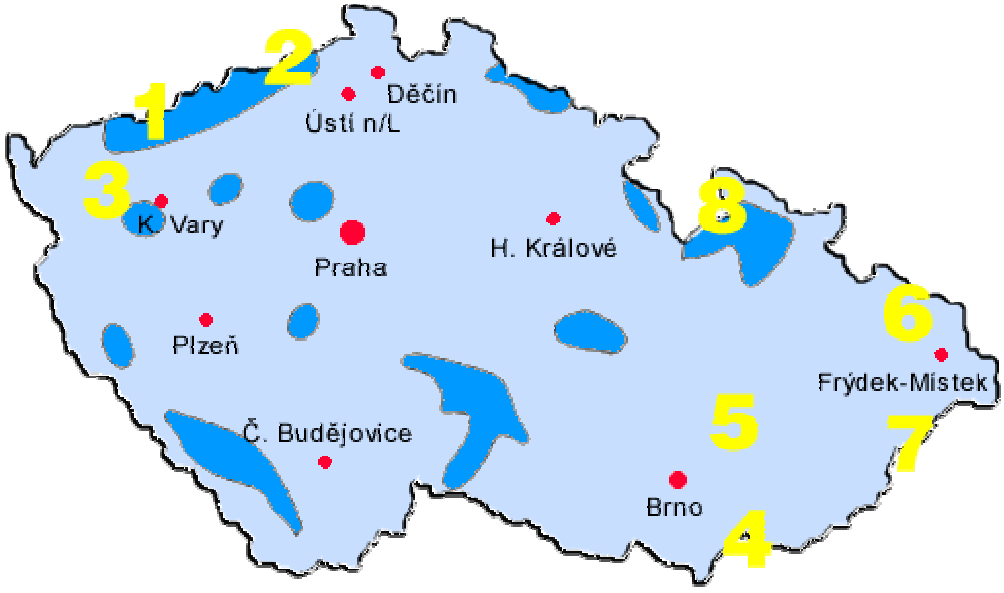
2) dánská větrná elektrárna TVIND s třílistou vrtulí o výkonu 2 MW,

3) dvoulistá větrná elektrárna ze Severní Karoliny,

4) německá větrná elektrárna GROWIAN o výkonu 2 až 3 MW,

5) německý projekt jednolisté větrné elektrárny o výkonu až 10 MW.

Země Evropských společenství přijaly program rozvoje větrné energetiky v roce 1980. Začaly stanovením technických a hospodářských možností v jednotlivých členských zemích a zpracováním jejich větrných energetických atlasů. Současný výkon evropských větrných agregátů přesáhl 500 MW. Na základě úspěšných projektů, zvláště v Dánsku, Nizozemsku, Německu a Velké Británii, rozhodla Evropská unie do roku 2000 zvýšit instalovaný výkon na 4 000 MW - dvojnásobek dnešního celosvětového výkonu. Do roku 2010 pak počítá EU s 25 000 MW, což je zhruba dvojnásobek veškerého dnešního instalovaného výkonu v ČR, a v roce 2030 hodlá dosáhnout dokonce 100 000 MW. To je výkon, který má pokrývat 20 % celkové západoevropské spotřeby elektrické energie!



Obr.5: Umístění největších větrných elektráren u nás

Větrná mapa.

Čísla přibližně označují místa, kde v roce 1994 stálo osm větších větrných turbín o instalovaném výkonu větším než 50 kW (v závorkách výrobce nebo dodavatel):

1) Boží Dar v Krušných horách - 75 kW (Vítkovice)
2) Dlouhá Louka nad Osekem v Krušných horách - 315 kW (Energovars z Dobré)-největší u nás
3) Hory u Karlových Var - 75 kW (Vítkovice)
4) Hrubá Vrbka u Kuželova v podhůří Bílých Karpat - 175 kW (Windpower, Dánsko)
5) Strabenice v podhůří Chřibů - 315 kW (Vítkovice)
6) Stařič u Frýdku-Místku - různé výkony (zkušební polygon firmy Energovars)
7) Bílý Kříž v Beskydech - 60 kW (Tacke, SRN)
8) Mravenečník u Loučné nad Desnou v podhůří Jeseníků - 250 kW (Worldwind, Dánsko)

Technické parametry pro typ **BW-15**

Obr.6: Typ BW-15

Základní parametry:

Jmenovitý výkon	15 kW
Jmenovité napětí	3x400V, 50Hz
Typ generátoru	asynchronní, 4 pólový
Vrtule	třílistá závětrná s aktivním natáčením listů o průměru 8,5m
Materiál vrtule	sklolaminát
Jmenovité otáčky vrtule	125 ot/min
Hmotnost jednoho listu	44kg
Hmotnost strojovny	750kg
Připojovací rychlost větru	4,5m/s
Odpojovací rychlost větru	24m/s
Stožár	trubkový o výšce 15 m, s kotvením ve třech kotvících bodech, roztečná kružnice kotvících bodů je 11m od paty stožáru
Hmotnost stožáru	2010kg
Natáčení do směru větru	natáčení strojovny do směru větru je pasivní
Hmotnost strojovny	620kg bez vrtulových listů
Brzdný systém	provozní brždění natáčením listů vrtule, bezpečnostní brždění elektromagneticky odbrzďovanou brzdou na asynchronním generátoru
Řídící systém	řízení větrné elektrárny zajišťuje programovatelný logický automat, který zajišťuje automatický provoz větrné elektrárny bez účasti lidské obsluhy. Automaticky připojuje větrnou elektrárnu k síti při vhodných povětrnostních podmínkách a odpojuje ji při poruchách nebo slabém či silném větru

Elektrické parametry:

Elektrárna je určena pro dodávku elektrické energie do elektrické sítě 3PEN 50Hz 400/230V TN-C. Je vybavena asynchronním generátorem SIEMENS 7BA 160L 04,Pn = 15 kW, Un = 400V, In = 28,5 A, nn=1540 ot/min. Připojení elektrárny k elektrické síti kabelem CYKY 5Cx6 mm^2 do vzdálenosti 50m. Hlavní jistič větrné elektrárny má jmenovitou hodnotu 32A.

Elektrické přístroje pro spínání, jištění, ovládání větrné elektrárny jsou soustředěné ve zděném pilíři u paty větrné elektrárny. Z tohoto rozvaděče jsou vedeny kabely do strojovny větrné elektrárny v příhradovém stožáru.

Ovládání větrné elektrárny zajišťuje programovatelný logický automat, který řídí okamžik připojení větrné elektrárny k elektrické síti, pomocí dalších přístrojů sleduje stav elektrické sítě, měří otáčky a výkon asynchronního generátoru a prostřednictvím meteorologických čidel měří rychlost a směr větru. Řídící systém umožňuje zobrazení provozních parametrů na displeji a vyvolání statických dat o provozu. Provoz větrné elektrárny je plně automatický a nevyžaduje trvalou obsluhu.

Řídící systém větrné elektrárny zajišťuje automatický provoz bez nutnosti trvalé lidské obsluhy. Automaticky ovládá start, provoz a odstavení větrné elektrárny včetně hlídání stavu elektrické sítě (napětí a kmitočet). Pomocí klávesnice s displejem je možné zjistit okamžité hodnoty výkonu rychlosti větru a otáček. Z hodnot uložených v paměti lze získat informace o předešlém provozu. Větrná elektrárna je vybavena ochranou před přepětím při úderu blesku.

Umístění větrné elektrárny:

Větrná elektrárna z principu své funkce musí být umístěna tam, kde jsou dobré větrné podmínky. Výstavba větrné elektrárny je rentabilní, je-li průměrná roční rychlost větru ve výšce 10 m nad okolním terénem alespoň 4,5 m/s (roční výroba el. energie cca 10 000 kWh). Tuto hodnotu lze zjistit minimálně půlročním měřením rychlosti větru nebo přibližně odhadnout pomocí dlouhodobých měření meteorologických veličin na nejbližších meteorologických stanicích nebo letištích.

V okolí místa instalace větrné elektrárny se nesmí do vzdálenosti cca 100m vyskytovat překážky jako jsou vysoké stromy, stavby, keře a podobně. Jejich přítomnost zvyšuje turbulence a snižuje výkon větrné elektrárny.

Pro jednotlivé průměrné rychlosti větru v m/s za rok může větrná elektrárna vyrobit následující množství elektrické energie:

Průměrná rychlost větru za rok (m/s)	Množství vyrobené el. energie za rok v (kWh)
4,0	**7.000**
4,5	**10.000**
5,0	**14.000**
5,5	**19.000**
6,0	**24.000**
6,5	**30.000**

Příklad využití vyrobené elektrické energie:

Elektrickou energii vyrobenou větrnou elektrárnou je možné ekonomicky zužitkovat například v těchto příkladech:

- ohřev užitkové vody (zemědělské farmy, hotely, restaurace, motoresty, rodinné domky, domy s nájemními byty)
- ohřev vody v plaveckých bazénech
- ohřev vody pro teplovodní vytápěcí systémy (penziony, hotely, motoresty, rodinné domky)
- úspora elektrické energie odebírané z elektrické sítě pro provoz chladících a mrazících systémů (zemědělské farmy, chlazení mlékáren , motoresty, restaurace, hotely - chladící a mrazící boxy)
- malé podnikatelské provozovny - úspora elektrické energie odebírané z elektrické sítě
- pily, truhlářské provozovny - úspora elektrické energie odebírané z elektrické sítě
- pily, truhlářské provozovny - aktivní vysoušení dřeva
- nabíjení akumulátorových vozíků

3.Možnosti vizualizace při použití počítačové grafiky v technické praxi

Při volbě grafického rozhraní jsem měl možnost velkého výběru. Z rozsáhlého množství grafických rozhraní jsem odložil rozhraní, která nefungují pod platformou Windows z důvodů snadné tvorby práce jak doma, tak i ve škole.Pod mým vybraným systémem se jevily pro použití na vizualizaci dvě dostupné technologie z důvodů možnosti jejich nastudování z internetu.Jedná se o DirectX a OpenGL.
Jako programovací jazyk jsem zvolil C++ a prostudoval jsem si rozhraní OpenGL.

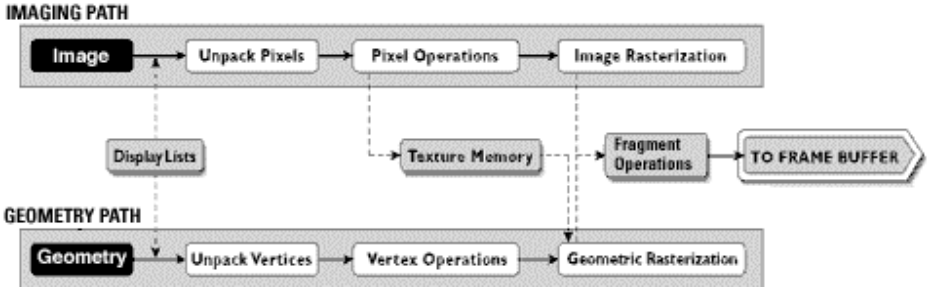
3.1 OpenGL

OpenGL je prostředí pro vývoj přenositelných , interaktivních 2D a 3D grafických aplikací. Od jeho představení v roce 1992 se OpenGL stalo nejpoužívanější, podporující 2D a 3D grafické aplikační programové rozhraní(API) přinášející mnohé možnosti vývoje na velké množině platforem. OpenGL zakládá pro vývoj rychlých aplikací spolupráci s rozsáhlými renderovacími funkcemi, přiřazujícími textury ,speciálními efekty a dalšími vizualizačními funkcemi.

Vývojáři mohou pocítit vliv síly OpenGL skrz oblíbené desktopové aplikace a na pracovních stanicích postavené platformy, zajišťující široký vývoj.
Jakákoliv vizuální aplikace požadující maximální výkonnosti od 3D animace po CAD vizuální simulaci může využít možnosti OpenGL. Tyto možnosti využívají vývojáři v rozdílných odvětvích jako jsou: vysílání,CAD/CAM/CAE, zábavním průmyslu,zobrazování v medicíně, ve virtuální realitě a umožňují produkovat a zobrazovat neuvěřitelně podmanivou 2D a 3D grafiku.

Výhody pro vývojáře:

OpenGL je průmyslovým široce otevřeným standardem podporující více platforem.
Již více než sedm let je k dispozici OpenGL implementace a její neustálá aktualizace je dobře kontrolována, aby jí vývojáři zahrnuli včas do svých aplikací.
Zpětná kompatibilita vyžaduje ujištění, že existující aplikace nezůstane zastaralá.
OpenGL jakožto přenositelný a spolehlivý systém produkuje shodné vizuální výsledky na kterémkoliv OpenGL API-podporujícím hardware. Velkou výhodou tohoto rozhraní je jistě jeho „oloupatelností“, což umožňuje aby aplikace běhaly na systémech počínaje spotřebitelskou elektronikou, přes PC a pracovními stanicemi, a konče na superpočítačích. Aplikace se může svou škálovatelností přizpůsobit jakékoliv třídě stroje na kterém poběží.
OpenGL obsahuje intuitivní a dobře strukturované příkazy.



Obr.7: The OpenGL Visualization Programming Pipeline

Pro použití v mém příkladu jsem použil DirectX z důvodů lepších studijních materiálů přístupných na síti internet a jeho použité komponenty jsou popsány v následujících kapitolách.

4.DirectX, .NET a jejich popis

O DIRECTX:Microsoft DirectX® je skupina technologií navržených pro vytváření programů pod operačními systémy Windows, jako ideální platforma pro běh a zobrazování aplikací bohatých na multimediální prvky, jako grafika v plném barevném spektru, video, 3-D animace a prostorový zvuk.

Microsoft Direct3D je sestrojený, aby umožnil programovat hry světové třídy a interaktivní 3D aplikace pod systémem Microsoft Windows. Představuje kreslící rozhraní, podporující 3D akcelerátory a různé grafické karty. Direct3D poskytuje excelentní grafiku pro hry a ostatní grafické aplikace pod Windows 95 a další včetně nejnovějších verzí Windows XP.

Významné rysy Direct3D:

- Nastavitelný depth buffering (z-buffer nebo w-buffer)
- Flat a Goraud stínování
- Velký počet různorodých světel
- Plná podpora materiálů a textur s počítáním mipmapping
- Robustní softwarové emulační ovladače
- Transformace a clipping
- Hardwarová nezávislost
- Plná podpora Windows 95,Windows 98,Windows ME,Windows 2000 a Windows XP
- Podpora Intel MMX architektury, Intel Streaming Single-Instruction,Multiple-Data(SIMD),SSE a AMD 3DNow architektura
- Podpora Hardware Abstraction Layer (HAL) - přímá práce se zobrazovacím hardware - maximální výkon
- Podpora clipping v okenních nebo full-screen aplikacích
- Podpora 3D z-bufferů
- Přístup k image-stretching hardware
- Paralelní přístup k standardním a rozšířeným zobrazovacím paměťovým místům
- Další rysy

Tyto rysy vám umožňují psát aplikace, které lehko předčí standardní Windows GDI, nebo GDI+ a grafické MS-DOS aplikace. Jádro Direct3D je založeno na vertexech, polygonech a příkazech kterými se ovládají.To umožňuje přímý přístup k transformacím, osvětlení, rasterizaci 3D graphics pipeline. Pokud hardware nepodporuje akceleraci renderování, Direct3D nabízí širokou softwarovou emulaci.

Direct3D poskytuje jednoduché a přímočaré metody na nastavení renderování 3D scény. Umožňují aplikacím renderovat jeden nebo více objektů ve scéně jednoduchým voláním funkce DrawPrimitive. Direct3D umožňuje spojení s 3D hardwarem. Je třeba vykonat explicitní volání pro transformace a osvětlení, je třeba nastavit všechny nutné matice a je třeba zjistit druh a schopnosti hardwaru. Tolik k popisu DirectX jako rozhraní.

V textu se budou vyskytovat slova jako renderování, což je vykreslení 3D scény na obrazovku, dále pixel je bod zobrazovacího zařízení a vertex znamená bod v 3D prostoru.

DirectX jako multimediální nástroj v sobě neobsahuje jenom podporu grafiky, ale je rozděleno do následujících oblastí:

4.1Rozdělení DirectX

- DirectX Graphics je spojený Microsoft DirectDraw a Microsoft Direct3D do API, které můžeme použít pro všechno grafické programování. Komponenta obsahuje Direct3DX knihovnu, která zjednodušuje mnohé grafické úlohy.

- DirectX Audio je spojený Microsoft DirectSound a Microsoft DirectMusic do API, které můžeme použít pro všechno audio programování.

- Microsoft DirectInput poskytuje podporu různých vstupních zařízení, které počítá s podporou force-feedback technologie.

- Microsoft DirectPlay poskytuje podporu multiplayer networked games.

- Microsoft DirectShow poskytuje podporu pro vysoko kvalitní zachytávání a reprodukci multimedia streams.

- Microsoft DirectSetup je jednoduché API které poskytuje instalaci DirectX komponentů.

DirectX nástroje:

- DirectX Caps Viewer zjišťuje ovladače a schopnosti pro všechny komponenty DirectX.

- DirectX Control Panel Application-Můžeme použít Microsoft DirectX Control Panel utility, která je nainstalovaná s SDK, na přezkoušení, na modifikování vlastností DirectX komponentů.

- DirectX Diagnostic Tool-Microsoft DirectX Diagnostic Tool zahrnuje informace o systému a DirectX komponentech používajících testů, které zajišťují, že komponenty pracují správně.

4.2 Architektura DirectX

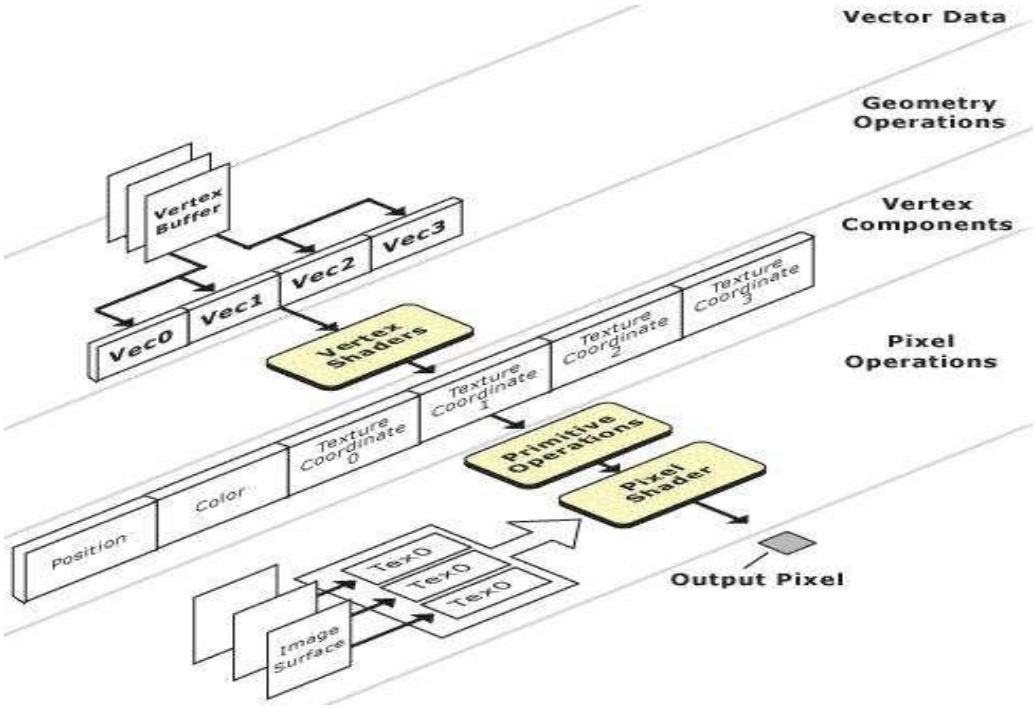
Tato část obsahuje základní informace o vztahu mezi Microsoft Direct3D komponentami a zbytkem DirectX, operačním systémem a hardwarem.

- Úvod do architektury Direct3D
- Hardware Abstraction Layer (HAL)
- Integrace systému

Úvod do architektury Direct3D

Existují dvě programovatelné sekce Microsoft Direct3D architektury:

Vertex shaders a pixel shaders. Vertex shaders je určený pro práci s vertexy a vektory. Pixel shaders je použitý po volání DrawPrimitive a je určený na operace s pixely. Následující zjednodušený diagram ilustruje architekturu Direct3D. Je neúplný, například Direct3D podporuje osm množin textur a souřadnic textur.

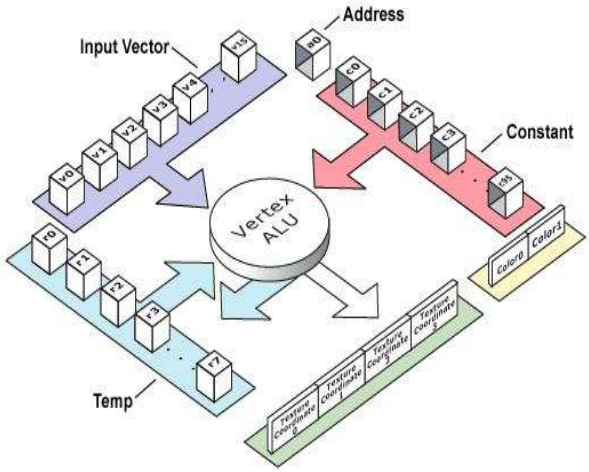


Obr.8: Architektura Direct3D

Jak je vidět z diagramu, vertex buffery kopírují data do vertex shaderu. Vertex shader vykoná geometrické operace použitím instrukcí definovaných Direct3DX vertex shader assemblerem. Data uložené do vertex shaderu nejsou obsažené ve vertex bufferech. Po zpracování vertexů jsou vykreslené pomocí DrawPrimitive. V tomto bodě je každý pixel kresleného primitiva poslaný do pixel shaderu,počítáním pozice, barvy, textury a souřadnic textur.

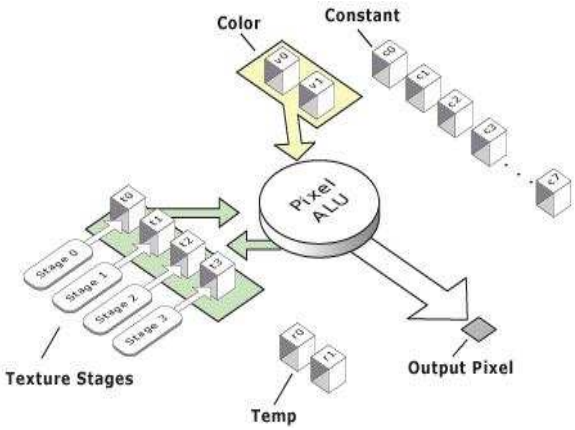
Pixel shader uskuteční operace s pixely použitím instrukcí definovaných Direct3D pixel shader assemblerem.

Ke zpracování údajů pixel shaderem je potřeba povrch textur a souřadnice textur, které kontrolují nanášení textur na povrch. Výstup je pixel zapsaný do frame bufferu.



Obr.9: Architektura Vertex Shaderu

Vstupující data do vertex dat přicházejí z proudů (stream) vertex dat a konstanty jsou načítané pomocí vertex shader deklarátoru. Následující diagram ukazuje architekturu pixel shaderu:



Obr.10: Architektura Pixel shaderu

Vstupující data do pixel shaderu jsou vertexy (homogenní souřadnice). Difusní a odrazová barva přichází z registru barev (v0 a v1). Souřadnice textur a textury přicházejí z registru textur(t0,t1,t2 a t3), z texture stages a vertex dat.

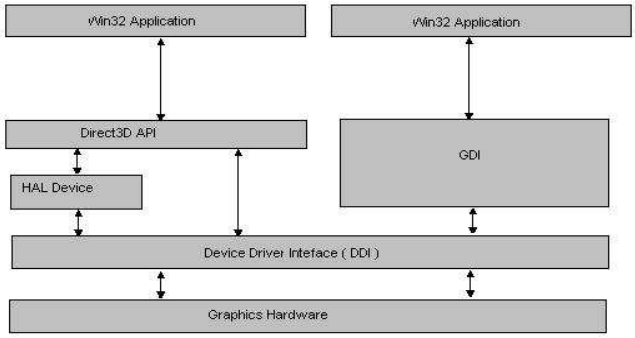
Hardware Abstraction Layer (HAL)

Direct3D poskytuje nezávislost na zařízení prostřednictvím Hardware Abstraction Layer (HAL). HAL je interface, poskytovaný výrobcem zařízení, který Direct3D používá na přímou práci s obrazovým hardware. Aplikace nikdy nepracuje přímo s HAL. Infrastrukturu, kterou poskytuje HAL , Direct3D sestaví do konzistentních skupin interface a metod, které aplikace použije na zobrazení grafiky. Výrobce zařízení implementuje HAL požitím 16-bitového a 32-bitového kódu pod Windows. Pod Windows NT a 2000 je HAL již vždy implementovaný 32-bitovým kódem.

HAL může být část zobrazovacího ovladače, nebo dynamicky linovaná knihovna (DLL), která komunikuje se zobrazovacím ovladačem přes interface definovaný výrobcem. Direct3D HAL je implementovaný výrobcem čipů, desek, nebo výrobcem zařízení (original equipment manufacturer OEM). HAL implementuje jen kód závislý na zařízení a neposkytuje emulaci. Pokud nějaká funkce není podporovaná hardware, HAL jí neuvede mezi schopnosti hardware.

V DirectX 8.0 HAL může mít tři odlišné vertex processing módy: software vertex processing, hardware vertex processing a smíšený vertex processing. Pravý mód je odlišný u různých HAL zařízeních.

Integrace systému



Obr.11: Vztah mezi Direct3D, GDI, HAL a hardwarem

Jak ukazuje diagram Direct3D aplikace existují po boku GDI aplikací a obě dvě mají přístup ke grafickému hardwaru přes ovladač pro grafickou kartu. Narozdíl od GDI Direct3D dokáže využít některé rysy hardwaru pomocí HAL. HAL zařízení poskytují hardware akceleraci založenou na rysech podporovaných grafickou kartou. Softwarová emulace, i když nepodporuje všechny prvky v DirectX 8.0 , vám zajistí , že aplikace vždy najde pracující zařízení na jiném systému. Avšak v mnohých případech musí být tato aplikace připravená na snížení funkčnosti. Schopnosti softwarové emulace je také možné zjistit, stejně jako schopnosti hardware.

Direct3D Rendering Pipeline

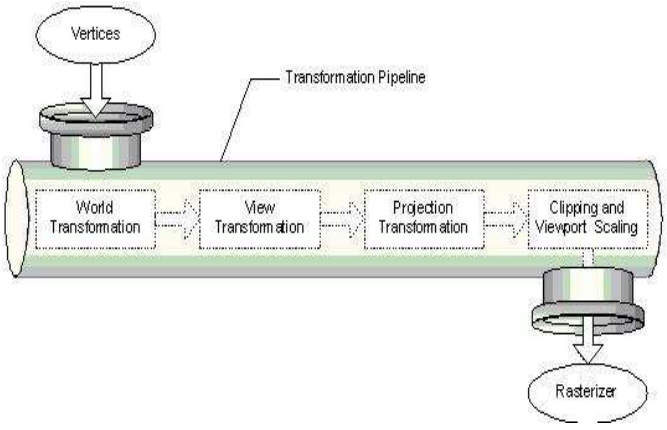
Direct3D rendering pipeline je série procesů, které specifikují chování vertexových transformací, osvětlení, nanášení pixelů a textur. Když navrhneme 3D aplikaci, definujeme svět v jakýchkoliv jednotkách.

Aplikace odešle popis tohoto světa do Direct3D. Tento popis obsahuje velikosti a relativní pozice všech objektů ve světě, pozice a orientace kamer. Direct3D transformuje tyto popisy (vertexy) do série pixelů na obrazovce. První krok tohoto procesu – transformace geometrie , která převede objekty do 2D obrázku – se nazývá geometry pipeline, někdy také transformation pipeline. Po zpracování vertexů jsou pixel data použité na uskutečnění multitexturingu a na nanášení mlhy. Potom nastanou alfa, stencil a hloubkové testy. Nakonec je rasterizérem vykonané zobrazení do frame-bufferu. Následující témata vysvětlují kompletní rendering pipeline. Vysvětlují klíčové koncepce a zabíhají od těchto konceptů k jejich protějškům v Direct3D API.

- Ustálené Vertex a Pixel zpracování
- Programovatelné Vertex a Pixel zpracování
- Převedení Vertex dat do Vertex procesoru

Ustálené Vertex a pixel zpracování

Ta část Direct3D, která zpracovává geometrii, se nazývá transformation engine. Umísťuje objekty a kamery ve světě, projektuje vertexy pro zobrazení na obrazovku a zobrazuje vertexy na viewport. Transformation engine uskutečňuje osvětlení a zjišťuje difúzní a odrazové komponenty každého vertexu. Geometry pipeline má jako vstup vertexy. Transformation engine aplikuje tři transformace – světovou (world), pohledovou (view) a projekční (projection) transformaci – na vertexy, zobrazí vertexy a vloží všechno do rasterizéru.



Obr.12: Geometry pipeline

Na začátku řetězce (pipeline) nejsou aplikované žádné transformace, takže všechny vertexy jsou deklarované relativně k souřadnému systému- ten představuje lokální východiskový bod a orientaci. Tato orientace souřadnic se často vztahuje k modelovacímu prostoru, individuální souřadnice se nazývají souřadnice modelu.

První etapa geometry pipeline transformuje vertexy modelu z jeho lokálního souřadného systému do systému, který používají všechny objekty ve scéně (například kolo od auta může mít svůj vlastní souřadný systém, ale když toto kolo chceme zobrazit spolu s autem a okolím, musíme ho transformovat do globálního souřadného systému). Proces reorientace vertexů se nazývá světová transformace. Tato nová orientace se často vztahuje ke světovému prostoru a každý vertex ve světovém prostoru je deklarovaný použitím světových souřadnic.

V další etapě jsou vertexy v 3D světě orientované vzhledem na kameru. To znamená, že aplikace si vybere pozici kamery (point-of-view) ve scéně a světové souřadnice jsou přemístěné a otočené okolo kamery, čím se světový prostor změní na prostor kamery. Tato transformace se nazývá pohledová.

Další etapa je projekční transformace. V této části řetězce jsou objekty obvykle měřítkované v závislosti od jejich vzdálenosti od pozorovatele (kamery) tak, aby vytvářeli iluzi hloubky ve scéně. Bližší objekty se objeví větší než vzdálené, atd..

V posledním kroku řetězce, jakýkoli vertex, který nebude viditelný na obrazovce je vymazaný, takže rasterizér se nezabírá výpočty barev a stínováním něčeho, co nebude vidět. Tento proces se nazývá clipping. Po clippingu jsou zvýšené vertexy měřítkované podle parametrů viewportu a konvertované do souřadnic obrazovky. Výsledné vertexy – viditelné na obrazovce po rasterizaci – existují v prostoru obrazovky (screen space).

4.3 Popis Direct3D a jeho funkcionality

Tato kapitola vám přiblíží programování s Microsoft Direct 3D. Další informace jsou rozdělené do následujících částí:

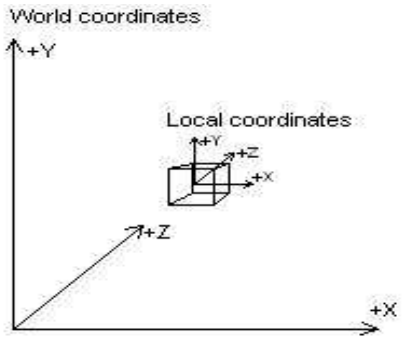
- Transformace a osvětlení
- Viewporty a clipping
- Rasterizace

Světová transformace (World transformation)

Popis světové transformace obsahuje základní koncepty.

Co je to světová transformace:

Světová transformace mění souřadnice z modelového prostoru, kde jsou vertexy definované relativně vzhledem ke globální vztažné soustavě, ve které jsou všechny objekty. V podstatě světová transformace umísťuje model do světa. Následující diagram ukazuje vztah mezi souřadným systémem světa a souřadným systémem modelu.



Obr.13: Světová transformace

Světová transformace může obsahovat kombinace posunutí, rotace a změny velikosti

Pohledová transformace

Popis pohledové transformace obsahuje základní koncepty a detaily.

Co je to pohledová transformace:

Pohledová transformace umísťuje pozorovatele do světového prostoru, transformujíc vertexy do prostoru kamery. V prostoru kamery je kamera vztažná soustava, dívající se (orientovaná) ve směru kladné části osy z.Direct3D používá levotočivý souřadný systém, takže kladná část osy z směřuje dovnitř scény. Pohledová matice vlastně přemístí objekty ve světě kolem kamery.

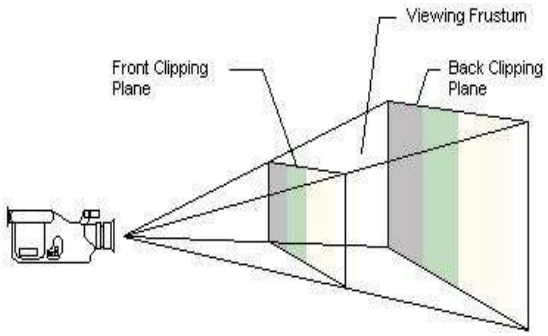
Existuje mnoho způsobů jak vytvořit pohledovou matici. Ve všech případech má kamera nějakou logickou pozici a orientaci ve světovém prostoru, který představuje startovací bod na vytvoření pohledové matice. Pohledová matice posouvá a otáčí objekty do prostoru kamery, kde představuje kamera vztažnou soustavu.

Projekční transformace (projection Transformation)

Tato transformace je mezi ostatními nejkomplikovanější.

Prostor viditelnosti je ve scéně umístěný relativně k výřezu kamery. Tvar tohoto objemu vyplývá, jako jsou objekty projektované z prostoru kamery na obrazovku. Nejběžnější typ projekce je perspektivní projekce – zodpovídá za to, že bližší objekty se objeví větší než objekty vzdálené. Prostor viditelnosti je možno zobrazit jako pyramidu, přičemž kamera je v jejím vrcholu.

Tato pyramida je ořezaná přední a zadní clipping rovinou (clipping plane). Prostor viditelnosti je právě mezi těmito rovinami. Objekty jsou viditelné jen, když se nacházejí právě v tomto prostoru.



Obr.14: Prostor viditelnosti

Co je to projekční transformace:

Projekční matice je typicky perspektivní projekce. Projekční transformace konvertuje prostor viditelnosti do tvaru kostky. Protože blízká hranice prostoru viditelnosti je menší jak vzdálená, způsobí tento efekt zvětšení objektů, které jsou blízko u kamery.

Osvětlení

Povolení a zakázání osvětlení:

Všeobecně Direct3D aplikuje osvětlení na všechny vertexy, i na ty bez normálových vertexů. To je odlišné od chování od předchozích verzí Direct3D. Osvětlení lze pomocí parametru D3DRS_LIGHTING na hodnotu TRUE, nebo FALSE měnit.

Typy a zdroje barev světla:

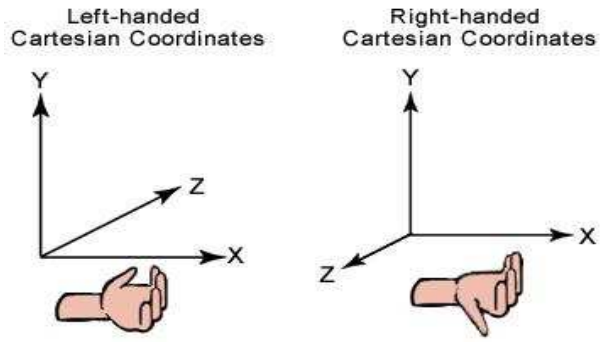
Světelné zdroje v Direct3D emitují difúzní, okolní a odrazové barvy jako odlišné, navzájem neslučitelné komponenty světla, které jsou nezávisle na sebe brané ve výpočtech okolo osvětlení. Taktéž vertexy, které systém renderuje, může použít difúzní, odrazové a emisivní barvy.

Efekt mlhy

Direct3D transformace a osvětlení mohou vytvořit efekt mlhy během osvětlování. Tento typ mlhy je obyčejně označovaný jako vertex mlha, protože informace o mlze jsou uložené v komponentě alfa daného vertexu.

Souřadné systémy

3D souřadnicový systém a geometrie typické 3D grafické aplikace používají dva typy kartézského souřadného systému: Pravotočivý a levotočivý.



Obr.15: Souřadné systémy pro 3D grafiku

Podle orientace os x a y určujeme zbývající osu z v případě pravotočivého systému pravidlem pravé ruky. Pokud máme počátek osy x v zápěstí a osa x ruku protíná(s rostoucí hodnotou na ose x jdeme podle prstů až k jejich konci), pak při nejkratším otočení ruky pro ztotožnění s osou y nám vztyčený palec ukazuje osu z. V případě levotočivého souřadného systému použijeme stejné pravidlo s levou rukou. Microsoft Direct3D používá levotočivý souřadný systém.

Pokud tvoříte aplikaci, která používá pravotočivý souřadný systém, musíte vytvořit dvě změny v datech Direct3D.

- Změňte pořadí vertexů trojúhelníka(z v0,v1,v2 na v0,v2,v1)
- Ve view matrix změňte znamínka členů _31, _32, _33 a _34

Na získání pravotočivého prostoru použijte funkce D3DXMatrixPerspectiveRH a D3DXMatrixOrthoRH na definování projekční transformace.

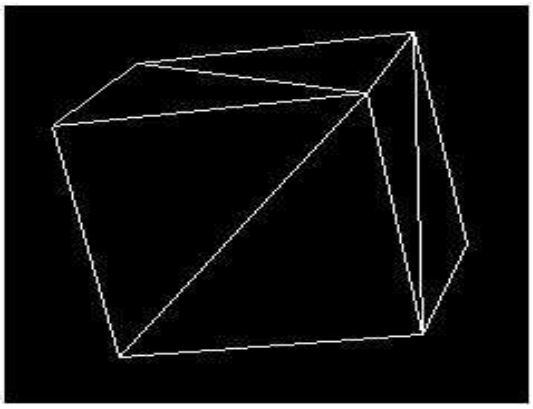
Nejčastější operace používané na objektech v prostoru jsou:

Posunutí,rotace,změna velikosti. Složení základních těchto transformací dostaneme vhodnou transformační matici, která není komutativní.(záleží na pořadí v kterém transformujeme nějaký objekt)

3D Primitiva

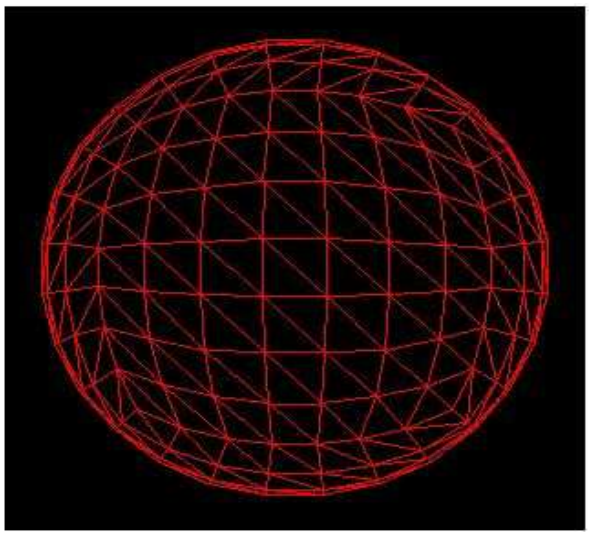
3D primitivem je soubor vertexů, které tvoří jednoduchou 3D entitu. Nejjednodušším primitivem je soubor bodů v prostoru (point list). Často bývají 3D primitiva mnohoúhelníky. Polygon je uzavřený 3D útvar s nejméně třemi vertexy. Nejjednodušší polygon je trojúhelník. Microsoft Direct3D používá trojúhelníky na tvorbu složitějších polygonů, protože všechny vertexy v trojúhelníku jsou zaručeně komplanární.(v jedné rovině) Trojúhelníky můžete kombinovat na složité a komplikované formy.

Následující ukázka ukazuje kostku. Na každou stěnu připadají dva trojúhelníky. Celá množina trojúhelníků formuje primitivum kostky. Na tento objekt pak můžeme aplikovat různé textury a materiály.



Obr.16: Kostka

Pomocí trojúhelníků je možné vytvořit primitiva, jejichž povrch připomíná křivou plochu. Následující obrázek ukazuje, jak může být povrch koule simulovaný trojúhelníky. Když aplikujeme materiál, vypadá po renderování jako křivá plocha. Nejlepší výsledek je při Gouraud stínování.



Obr.17: Koule

Pravidla rasterizace trojúhelníků:

Často se stává, že body specifikované pro vertexy nesednou přesně na pixely na výstupním zařízení(obrazovce). Na ošetření používá Direct3D rasterizaci trojúhelníků na zjištění, které pixely aplikovat na daný trojúhelník. Direct3D používá levo-horní vyplňovaní konvenci. Tu používá taktéž GDI a OpenGL. V Direct3D je rozhodující střed pixelu. Když se střed nachází v trojúhelníku je pixel jeho součástí. Středy pixelů jsou celočíselné souřadnice.

Módy stínování:

Použité stínování při renderování polygonu má velký vliv na jeho vzhled. Stínovací mód určuje intenzitu a barvu osvětlení v libovolném bodě na mnohoúhelníku.

Direct3D používá dva stínovací módy:

- Flat stínování
- Gouraudovo stínování

Poznámka:Existuje i Phongovo stínování, které je o dost náročnější, než Gouraudovo stínování.Nenachází však větší uplatnění, protože má skoro identický výsledný efekt jako Gouraudovo stínování.

Flat stínování

V tomto stínování Direct3D rendering pipeline renderuje polygon , přičemž ho zabarví celý barvou materiálu v jeho prvním vertexu. 3D objekty renderované s flat stínováním mají viditelné okraje mezi polygony.

Gouraudovo stínování

Když Direct3D renderuje polygony pomocí Gouraudova stínování,používá barvu v každém vertexu, normálový vektor a osvětlovací parametry. Interpoluje barvu přes celou plochu polygonu. Interpolace je lineární. Například, když vertex 1 má červenou složku barvy 0.8 a vertex 2 červenou složku barvy 0.4, potom použitím Gouraudova módu bude červená složka barvy středního bodu mezi těmito vertexy 0.6. Následující obrázek demonstruje Gouraudovo a flat stínování.

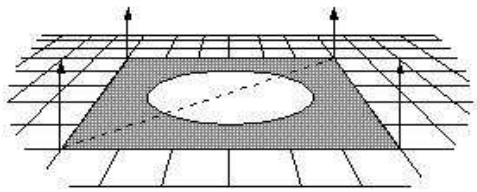
Obr.18: Gouraudovo stínování

Obr.19: Flat stínování

Porovnání stínovacích módů

Ve flat stínovacím módě je obrázek pyramidy zobrazený s ostrými hranami mezi spojenými trojúhelníky. V Gouraudově stínování jsou stínovací hodnoty interpolované přes hranu a výsledné zobrazení má zakřivenější povrch.

Gouraudovo stínování osvětlí plošky víc realisticky na rozdíl od flat stínování. Plošky ve flat módu mají uniformní barvu. Gouraud mód plošky osvětlí víc korektně. Tento efekt se znásobuje, pokud jsou body blízko u sebe. Gouraudovo stínování vyhladí ostré hrany mezi polygony, které sou viditelné ve flat módu, avšak může vracet pásy barvy, které nejsou hladce zobrazené přes polygony. Aplikace může tento problém korigovat zvětšením počtu polygonů, zvětšením rozlišení, nebo barevné hloubky. Gouraudovo stínování může zničit některé detaily. Jeden z příkladů je následující ilustrace, ve které světlo reflektoru je celé obsaženo na ploše polygonu.



Obr.20: Zobrazování detailů

V tomto případě Gouraudovo stínování, které interpoluje barvy vertexů nezachytí světlo reflektoru. Scéna bude renderovaná, jako by reflektor ani neexistoval.

Systém používá normály plošek ve flat stínovacím módu. V Gouraudově stínování Direct3D používá normály vertexů. Takže na ovládání osvětlení a texturových efektů se používají normály vertexů. Když aplikujeme Gouraudův mód na polygon, Direct3D použije normály vertexů na výpočet úhlu mezi zdrojem světla a povrchem. Dále přepočítává barvu a intenzitu pro každý vertex a interpoluje je pro každý bod přes celý povrch primitiva. Čím je větší úhel, tím je slabší osvětlení povrchu.

Interpolace trojúhelníků

Během renderování systém interpoluje základní charakteristiky vertexů daného trojúhelníka.

- Barva (color)
- Odraz (specular)
- Průhlednost (alpha)

Všechny tyto charakteristiky jsou modifikované v závislosti na módu stínování:

FLAT-Žádná interpolace. Povrch má barvu podle barvy prvního vertexu

GOURAUD-Lineární interpolace

Barva a odraz jsou ošetřované v závislosti na barevném modelu. V RGB interpoluje červenou, zelenou a modrou složku barvy. Průhlednost je posuzovaná odděleně, protože průhlednost může být implementována dvěmi různými cestami (texture blending, stippling).

MATICE a transformace

Matice se v Direct3D používají na definování světové (world), pohledové (view) a projekční (projection) transformace.

Matice

Pojem matice v Direct3D je čtvercová 4x4 matice (D3DMATRIX). Direct3DX knihovna poskytuje výhodný přístup k hodnotám v matici (C++). Místo odkazování na členy matice jménem je možné použít řádkový a sloupcový číselný index. Takže M(2,1) je třetí řádek a druhý sloupec.

3DTransformace

Direct3D používá matice na uskutečnění 3D transformací. Tato sekce je rozdělena do několika kroků:

- O 3D Transformacích
- Posunutí
- Otočení
- Změna měřítka
- Slučování matic

O 3D Transformacích

V aplikacích , které pracují s 3D grafikou, můžeme použít geometrické transformace na následující činnosti:

- Vyjádřit polohu objektu vzhledem k jinému
- Rotovat a měnit velikost objektu
- Měnit pozici pohledu (kamery), směr, perspektivu

Pomocí 4x4 matice můžeme transformovat libovolný bod do jiného bodu. V následujícím případě změníme polohu bodu (x,y,z) pomocí matice na (x´,y´,z´).

$$[x'\ y'\ z'1] = [x\ y\ z\ 1]\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} \tag{4.1}$$

Mezi nejpoužívanější transformace patří posunutí, rotace a změna velikosti. Matice je možno zkombinovat do jednoduché matice (násobení), takže najednou lze vykonat víc transformací. Matice, která mění vertexy podél každé osy je reprezentována následující maticí:

$$\begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

V C++ je matice v Direct3D deklarovaná jako dvourozměrné pole (D3DMATRIX).

Posunutí

Následující transformace posune bod (x,y,z) na nový bod (x´,y´,z´):

$$[x'\ y'\ z'1] = [x\ y\ z\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix} \tag{4.6}$$

Otočení

Transformace, které tu jsou popsané jsou pro levotočivý souřadný systém. Následující transformace rotuje bod (x,y,z) okolo osy x do nového bodu (x´,y´,z´).

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

Rotace kolem osy y:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Rotace kolem osy z:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

V těchto příkladech matic, je řecké písmeno theta úhel rotace v radiánech. Úhly jsou měřené ve směru hodinových ručiček, když se díváte podél osy v jejich směru. V C++ aplikacích se používají funkce D3DXMatrixRotationX , D3DXMatrixRotationY a D3DXMatrixRotationZ na vytvoření těchto matic.

Změna měřítka

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

Slučování matic

Velká výhoda při používání matic je , že je možno kombinovat efekty dvou, nebo více matic jednoduchým násobením. To znamená, že na rotaci a následné posunutí není potřeba použít matice dvě. Stačí vynásobit příslušnou matici rotace a posunutí, čímž dostaneme složenou matici, která v sobě obsahuje obě dvě transformace. Tento proces, sloučení matic (matrix concatenation), je možné zapsat jako následující vztah:

$$C = M_1 \times M_2 \times M_{n-1} \times M_n \tag{4.11}$$

V této rovnici je C složená matice vytvořená násobením postupně matic M_1 až po M_n , které představují částečné transformace.

Kopírování povrchů

Pojem blit je zkratka pro „bit block transfer“ (přesun bloků bitů), který představuje přesouvání dat bloků z jednoho místa v paměti na druhé. Blitting device driver interface (DDI) je v Direct3D 8.0 použitý jako primární mechanismus pro přesun velkých pixelů na bázi per-frame (počet framů za sekundu), mechanismus, který se skrývá za funkcí IDirect3DDevice8::Present. Transportace předlohy v blit operacích je uskutečněná funkcí IDirectDDevice::UpdateTexture. V Direct3D může být předloha také zkopírovaná použitím funkce IDirect3DDevice8::CopyRects, která kopíruje obdélníkové sady pixelů.

Page Fliping a Back Buffering

Page flipping je klíč k multimédiím, animacím a hrám. Software page flipping je analogický k animaci s blokem papíru. Na každém papíru změní umělec polohu objektu, takže, když rychle tyto stránky předhazujeme, kresba se začíná jakoby pohybovat.

Page flipping v programu je velmi podobné tomuto procesu. V Direct3D implementujete page flipping asi tak, že si připravíte sérii bufferů Direct3D, které jsou předem sestavené na zobrazení.

Buffer, který je aktuální, tj. je vidět na monitoru je front buffer, ostatní, které se mají zobrazit jsou back buffery. Aplikace zapisuje do back bufferů a potom jich přenese do front bufferu, takže se objeví na obrazovce. Zatímco systém zobrazuje obrázek, program opět zapisuje do back bufferu, takže obraz se nevykresluje přímo na obrazovku po částech (jako je to v klasickém GDI), ale najednou, přičemž nedochází k blikání. Tento proces vám umožňuje animovat obrázky rychle a efektivně. Direct3D ulehčuje page flipping tak, že definuje dvojitý buffering (back a front buffer).

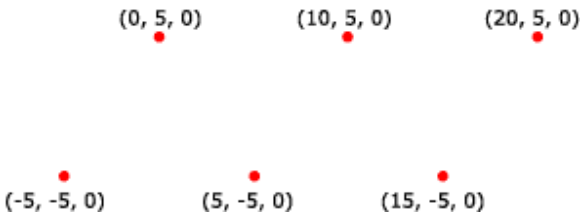
Typy primitiv podporovaných zařízením

Direct3D zařízení mohou vytvořit a manipulovat následujícími typy primitiv.

- Point Lists
- Line Lists
- Line Strips
- Triangle Lists
- Triangle Strips
- Triangle Fans

Point lists

Point lists je kolekce vertexů, které jsou renderované jako izolované body. Vaše aplikace je může použít v 3D scéně jako hvězdy, bodové čáry na povrchu polygonu apod.

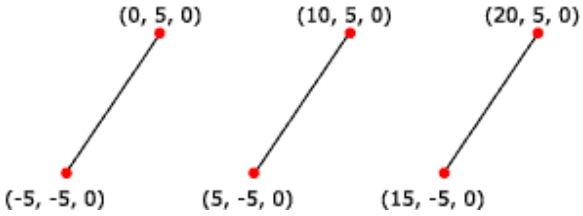


Obr.21: Renderovaný point list

Vaše aplikace může aplikovat materiály a textury na point list. Barvy materiálu nebo textury se objeví jen na nakreslených bodech.

Line Lists

Line list je seznam izolovaných přímých čar. Line listy jsou užitečné především pro takové úlohy, jako přidání sněhu s deštěm, nebo hustý déšť do 3D scény. Aplikace vytvoří line list vyplněním pole vertexů a počet vertexů musí být větší, nebo rovný dvěma.

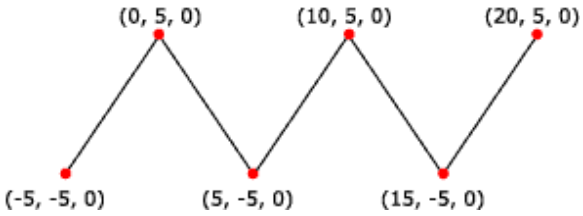


Obr.22: Renderovaný line list

Je možné aplikovat materiály a textury na line list. Barvy materiálu, nebo textury se objeví jen na nakreslených čarách.

Line Strips

Line strip je primitivum, které je složené z pospojovaných čárových segmentů. Je možné je použít pro vytvoření lomených čar a mnohoúhelníků.

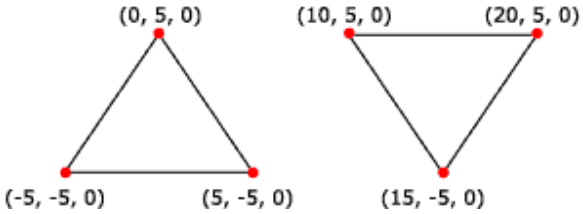


Obr.23: Line strip

Triangle Lists

Triangle list je seznam izolovaných trojúhelníků. Mohou, ale nemusí být blízko sebe. Triangle list musí mít nejméně tři vertexy. Počet vertexů musí být dělitelný třemi. Triangle listy je možné použít na vytvoření objektu, který je složen z disjunktních prvků. Následující

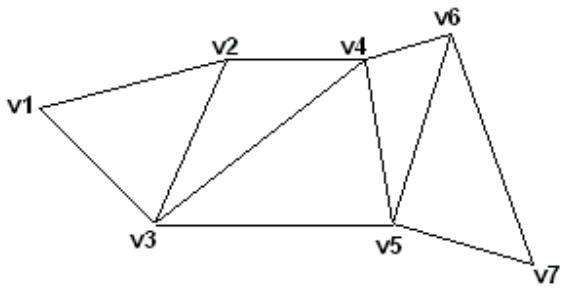
ilustrace ukazuje renderovaný triangle list.



Obr.24: Triangle list

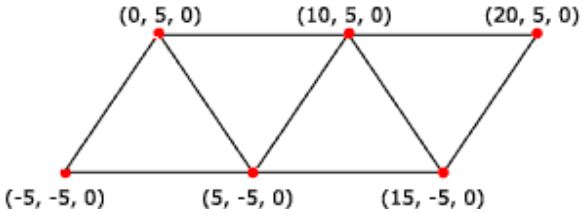
Triangle Strips

Triangle strip je série pospojovaných trojúhelníků.protože jsou pospojované, není potřeba opakovaně uvádět všechny tři vertexy pro každý trojúhelník. Například, stačí použít pouze sedm vertexů na definování následujícího pruhu.



Obr.25: Triangle Strips

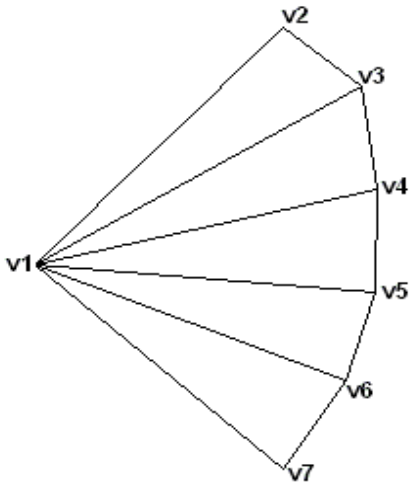
Systém používá vertexy v1,v2 a v3 na nakreslení prvního trojúhelníka.v2,v4 a v3 na nakreslení druhého trojúhelníka apod. Většina objektů v 3D scéně je složená z triangle strips. Triangle strips může být použité na specifikování komplexních objektů a přitom efektivně využívají paměť. Následující ilustrace ukazuje renderovaný triangle strip.



Obr.26: Renderovaný triangle strip

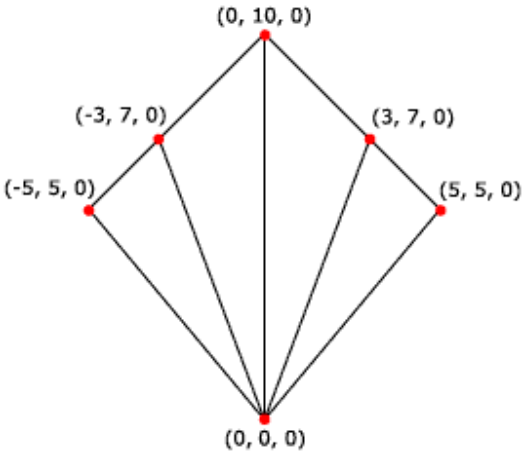
Triangle Fans

Triangle fan je podobný jako triangle strip, ovšem všechny trojúhelníky sdílejí první vertex(viz. Ilustrace).



Obr.27: Triangle fan

Systém používá vertexy v2,v3 a v1 na nakreslení prvního trojúhelníka, v3,v4 a v1 na nakreslení druhého trojúhelníka apod. Následující ilustrace ukazuje renderovaný triangle fan.



Obr. 28: Renderovaný triangle fan

4.4 Popis .NET framework

Ke změně stejné velikosti jako byla například na půdě Microsoft přechod z operačního systému MS-DOS k systému Windows, lze hovořit o změně v roce 2000, kdy byla uvolněna první beta verze .NET SDK společně s první beta verzí Visual Studia, dnes označovaného jako Visual Studio .NET.

Pod pojmem .NET můžeme dnes chápat několik věcí. Na jedné straně je snaha o novou generaci operačních systémů, všemožných typů aplikačních serverů a také snaha o vytváření nových vývojových nástrojů. Všechny uvedené produkty jsou závislé na jádře platformy nazvaném .NET framework.

Pozitivní změny platformy jsou:

- COMMON LANGUAGE RUNTIME, jenž se stává prostředím pro spouštění .NET aplikací
- OBJEKTOVĚ ORIENTOVANÉ KNIHOVNY dostupné ve všech programovacích jazycích
- SPOLEČNÁ JAZYKOVÁ SPECIFIKACE

.NET framework zcela výrazně mění pohled a praktickou realizaci vývoje programového vybavení.

O bolestech a strastech vývoje software je snad obeznámen každý, kdo se alespoň trochu pohybuje v softwarovém průmyslu. Svět software se zdá být značně složitý, nesourodý a nevyzpytatelný. Jen málokdo z nás se spoléhá pouze sám na sebe, jak tomu bylo v dobrých dobách CP/M nebo MS-DOSu. Dnes jsme odkázáni z důvodů znalostní náročnosti, lidských zdrojů a dynamiky vývoje na důsledky práce jiných programátorů a firem. Programové vybavení z počátku 90. let je charakteristické velice sporadickou schopností vzájemné komunikace. Jakákoli integrace aplikací v komplexnější řešení byla téměř nemožná. Pro moderní vývojové postupy je vytvořen zcela nová Run-time knihovna funkcí.

Množina nových služeb CLR

- Garbage-collection na systémové úrovni řídící životnost objektů
- Překladač z jazyka MSIL na nativní kód procesoru
- Typovou kontrolu založenou na přítomnosti Metadat obsahující všechny použité typy
- Deklarativní bezpečnostní model
- Podporu pro pokročilé ladění a profilování aplikací
- Integraci s existujícím COM modelem

MSIL (Microsoft intermediate language) je kód který vygeneruje kompilátor a je určen pro jednoduché přenášení kódu mezi pestrými hardwarovými platformami. Pro absolutní

volbu nad aplikacemi ponechal Microsoft jazyk C++, který může využívat neřízený kód při zřeknutí se výhod a služeb CLR.

Assembly

Assembly je logickou kolekcí, sestávající z jednoho, nebo více souborů .exe, .dll nebo .module souborů a zdrojů doplněných manifestem. Assembly pomáhá odstraňovat problémy s řízení verzí a s bezpečností. Podporuje bezpečnost přístupu ke kódu, obsahuje informace o „veřejných typech“, identitu, seznam souborů, exportované typy a zdroje a mnoho dalších např. bezpečnostních požadavků. Z C++ bylo převzato to nejlepší a proto se zde setkáváme s jmennými prostory (NAMESPACES). Přirovnat je můžeme k tomu, co velice dobře známe (adresářové systémy).

Společný typový systém

Realizace CTS je nezávislý typový systém pro jazykovou implementaci. Při vývoji software se již v C++ objevuje objektově orientované programování. Máme ovšem dvě skupiny oběktově orientovaných jazyků:

- Čistě objektově orientované jazyky
- Jazyky s implementovanými principy objektově orientovaného programování s neobjektovými vlastnostmi

Smalltalk a Eifel lze zařadit do první skupiny, jazyk C++ nebo Java do té druhé. Pod platformou .NET je vše realizováno objektově.

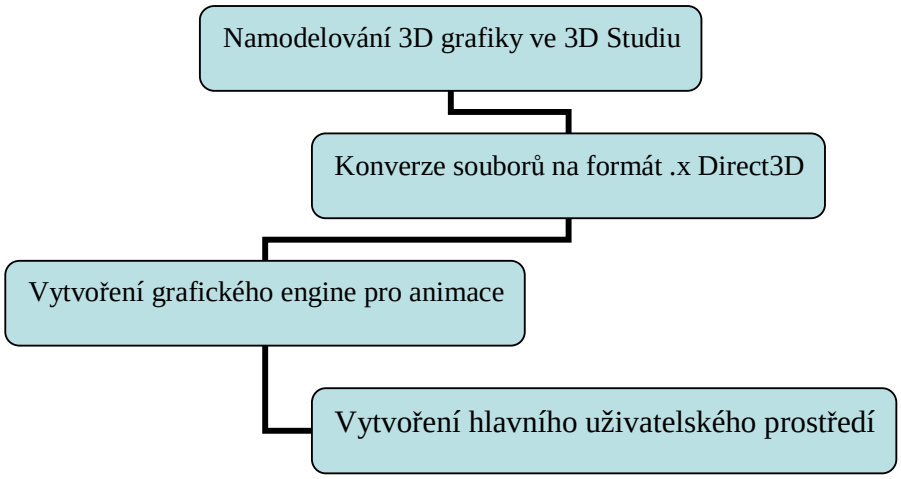
Bezpečnost .NET framework

Bezpečnost na této platformě je rozšířena na následující oblasti:

- Typová bezpečnost
- Identita kódu
- Přístupová bezpečnost kódu
- Povolení
- Deklarativní a imperativní bezpečnost
- Bezpečnost založená na rolích
- Kryptografické služby

5 Praktické vytváření vizualizace

5.1 Schéma hlavních fází tvorby programu



5.2 Popis všech animací

Pro vytváření animací jsem se snažil zachytit všechny možné stavy systému.

1. Potřeboval jsem vytvořit animaci větrné elektrárny, kde jsem chtěl naznačit vizualizaci vstupních veličin, aby uživatel zadal hodnotu síly větru a měl možnost pozorovat změny. Dále jsem chtěl zabezpečit vizualizaci při změně směru větru.

2. Chtěl jsem vytvořit animaci operátora č.1, který přímo ovládá a řídí výstup systému, na základě vstupu.

Jeho hlavní úkoly jsou:

- Ovládání el. sítě a rozvod proudu
- Přečerpávání vodní nádrže, pro udržování minimálního výkonu elektrárny

3. Nakonec jsem chtěl provést animaci operátora č.2, který má za úkol řešit havarijní stavy,(včetně oprav) a je op. č. 1 nadřazený.

Následující úkoly jsem vyřešil seznamem animací:

1. Animace větrného pole (vstupu)
2. Animace op. č.1-Distribuuje rozvod proudu při růstu síly větru
3. Animace výstupní veličiny-Růst vstupní síly větru (růst výroby proudu)
4. Animace op. č.1 -Další růst výkonu a distribuce elektřiny do sítě
5. Animace op. č.1-Snižuje distribuci proudu a pouští přečerpávání
6. Animace op.č.1 -Vypnutí el. při krizovém stavu
7. Animaci op.č.1 komunikujícího s op. č.2
8. Animaci op. č.2 komunikujícího s okolím

5.3 Namodelování objektů ve 3Dstudiu

Nejprve jsem si vytvořil scénu ve 3D studiu. Tvoří se jakási síť a na ní se nanáší textury. Vytvořil jsem si základní objekty, kterým se přiřazuje materiál a textury. Pro animaci člověka jsem použil kosti, pomocí kterých jsem změnil polohy operátora. Pro animaci se musely vytvořit framy „snímky“ pomocí kterých se počítá animace. Proto se pomocí kostí změnili polohy operátora a uložil jsem je jako jednotlivé scény.

5.4 Převedení formátů na x.formát

Poté se všechny vytvořené objekty převedou z formátu 3D studia na .x formát, s kterým pracuje Direct3D pro složité objekty. Jednoduché objekty se dají vytvářet přímo popisem vertexů ve zdrojovém programu. K tomuto účelu jsem použil program conv.exe.

5.5 Vytvoření grafického engine

Pro grafický engine jsem použil jazyk C++ a Direct3D 8.1. Pro animaci větrné elektrárny jsem použil změnu matic v čase a tím se vytváří otáčení listů kolem své osy a kolem osy sloupu. Pro animaci operátorů používám takzvaný tweening ,což je tvorba animace mezipolohováním z načtených klíčů a je potřeba tyto klíče načíst z .x souborů.

Vytvoření grafického engine:

Na úvod jsou v projektu přidružené soubory s koncovkou .h :

Na nadeklarování základních maker používaných v Direct3D jsem vytvořil ve Visual Studiu .NET soubor vetrnikD3D.h. Dále jsou zde definovány základní konstanty pro druh opakování animace, které opakují natáčení listů vrtule, aby se vytvořil dojem neustálého

natáčení rotoru.

Mezi další důležitá makra v tomto souboru jsou zde uvedena pro použití z-bufferu, světel, 3Dzařízení a renderování.

Dále je zde obsažena struktura pro práci s 3D soubory, kde je později třeba určit vertex buffer, index buffer, materiál 3D objektů, aktuální políčko v animaci a počet políček celkem. Zásobníky se deklarují z důvodů architektury DirectX, protože se uplatňuje takzvané přehazování (flipping) vykreslených objektů ve front a back bufferu.

V souboru stdafx.h, který je implicitně vytvořen Visual Studiem jsou přidružené hlavičkové soubory ,které jsou nezbytné pro chod aplikace.

D3dmath.h definuje základní proměnné a funkce pro soubor d3dmath.cpp.

V Resource.h jsou nadefinována různá další makra.

Vlastní důležitý kód pro základní nastavení aplikace je v souboru animace.cpp:

Na začátku tohoto souboru jsou přidružené hlavičkové soubory, které jsou výše uvedené. Makra pro velikost obrazovky jsou zde inicializována a je zde také nastavení pozice okna pro zobrazení animace.

Dále jsou zde definovány globální proměnné pro základní informace o oknu aplikace jako jsou: text titulku, handle okna a jméno třídy okna. Pro zobrazení okna je potřeba nastavit funkci winmain() s její smyčkou zpráv, grafickým akcelerátorem a hlavně zavolat v této funkci funkce pro vlastní nastavení a vykreslování scény.

D3dmath.cpp obsahuje matematické funkce na přepočítávání, přiřazování hodnot pro pozice kamery, dále přepočítává hodnoty pro matice pohledů a transformací. Nastavuje orientaci matice, vrací orientaci matice, přepočítává měřítko a řeší hlavní přepočítávání hodnot pro rotaci vrtulí k elektrárně. Přepočítá D3DXVECTOR3 a vrátí ho zpět hlavnímu souboru pro další zobrazení.

Soubor stdafx.cpp je implicitně vložen programovým prostředím a je v něm pouze přiřazen soubor stdafx.h.

Ta část projektu, která způsobuje vlastní operace pro 3D zobrazení je v souboru vetrnikD3D.cpp:

Na začátku tohoto souboru jsou po základním přidružení standardních souborů nainicializována 3 důležitá makra.

- Počet objektů ve 3D scéně
- Počet klíčů pro animaci
- Počet frame za sekundu

Klíče jsou důležité obrázky ve scéně, které jsou určující pro animaci. Mezi těmito klíči se pak propočítávají mezipolohy pro objekty ve 3D scéně.

Dále v souboru je kód pro načtení objektů ze souborů, které se předtím zvlášť překonvertovaly z formátu, který tvoří 3D grafický program pro kreslení 3D objektů

(.3ds nebo .max) na .x formát, který je textový a využívá ho Direct3D. Tato konverze se provádí pomocí programu CONV, který je volně stažitelný na internetu. Program CONV lze spouštět pouze pomocí textového režimu+použití parametrů a nemá grafické uživatelské prostředí. Tento program je nahrán na přiloženém CD k této diplomové práci.

Dále je v souboru inicializace hodnot pro pozici kamery a jsou i inicializována 2důležitá makra:

- Úhel za frame, který vykonává rotor
- Počet framů k natočení

Tyto hodnoty jsou důležité z důvodů počtu obrázků, kolik se jich má zobrazit mezi polohami daných úhlů. Tím lze měnit rychlost natáčení za větrem a přesnost zobrazení. Rotace je jednak kolem osy stojanu elektrárny pro natáčení podle směru větru a dále kolem osy rotoru pro vlastní vizualizaci otáčení listů vrtule podle síly větru.

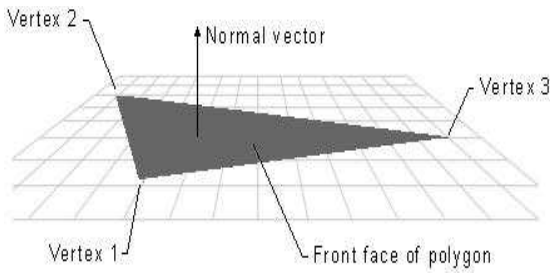
Dále je zde definice pole, které slouží k určení, jak se bude animace opakovat. Toto pole- D3DXVECTOR3 position_data[NUM_OBJECTS][MAX_KEYS] je zodpovědné za nastavení pozice pro každý klíč objektu.

Pro nastavení rotace kolem osy rotoru je zde makro RYCHLOST v rad/frame, čím menší hodnota se zde nastaví, tím pomalejší je rychlost otáčení.

Dále soubor obsahuje pole D3DXQUATERNION rotation_data[NUM_OBJECTS], což je zde pro rotaci listů vrtule. Je dána definicí osy polem (3x float). Počáteční orientace objektu je dána osami Y a Z polem D3DXVECTOR3 orientation_data[NUM_OBJECTS][2]. Další pole předdefinují měřítko a barvu objektů.

Definice funkce BOOL TVetrnik::Init() načte objekty ze souborů, přiřadí hodnoty barev, průhledností, osvětlení, matic a animace.

Funkce BOOL T3DFile::Load3DFile(LPSTR Filename) je zodpovědná za načtení sítě z 3D souborů , dále je zde nastaven index a vertex buffer a jsou zde dopočítány normály plošek všech 3D objektů.



Obr.29: Normálový vektor plošky

Pro tyto zásobníky se používá metoda zavírání a otevírání zásobníku (lock a unlock).

Dále jsou ze zásobníku materiálů extraktovány vlastnosti a nastavení ambient color pro materiál. Následně je potřeba vytvořit textury, které pokrývají povrch některých objektů.

Nakonec je pro 3D zařízení nastavena podle různých předdefinovaných maker průhlednost textur a mnoho dalších vlastností pro renderování scény.

Tato fce.: BOOL TVetrnik::Render() je hlavní pro vlastní renderování scény. Začíná voláním funkce pro 3D zařízení Beginscene(). Volá nadefinované matice kamer, matice projekce , volá zdroje dat apod. Ukončuje se pomocí Endscene().

BOOL TVetrnik::CalculateFrameRate() počítá framy.

BOOL TVetrnik::Move() je zodpovědná za pohyb pro natáčení podle větru.

Pro pohyb animace vpřed a zpět jsou v kódu poslední dvě funkce a je možné je podrobně prostudovat z příloh. Nakonec je do projektu ve složce Resources přidána ještě ikona Animace.ico a dále ještě textový soubor ReadMe.txt pro popis celého vytvořeného projektu.

Základní obecné schéma programu pro Direct3D:

Základní kostra programu v Direct3D pro základní načtení 3D objektu a jeho zobrazení (bez animace) popíšu následujícími řádky. Lze u něj vytvořit animaci pouze rotacím kamer.

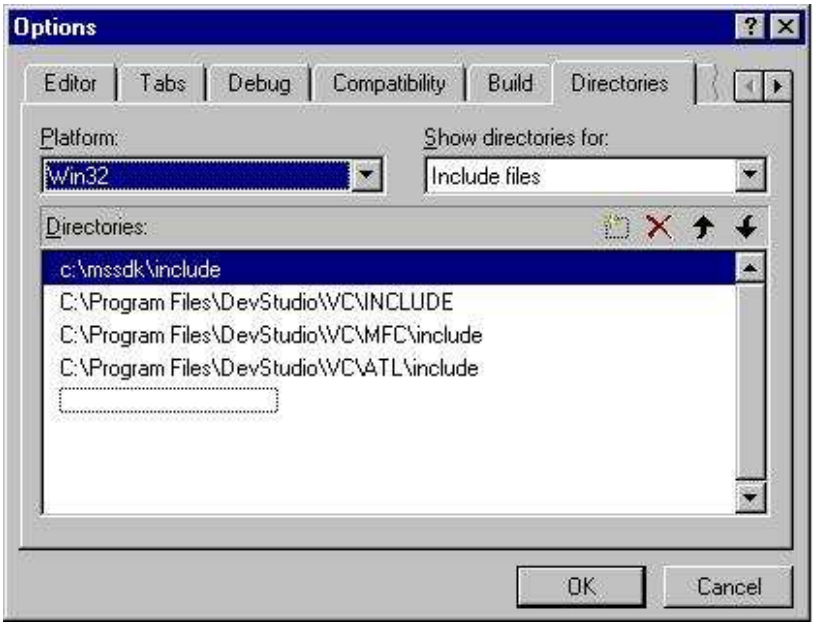
Na začátku je potřeba vytvořit D3Ddevice, což je objekt pro 3D zařízení a také 3D rendering zařízení. Pro použití 3D objektu ze souboru potřebujeme dále deklarovat objekt ze sítě pro náš systém, materiál, textury pro naši síť a nakonec počet materiálů pro síť.

- 1. funkce, která se objeví v programu nám musí inicializovat 3D zařízení , dále nastavit rozlišení obrazovky, nastavit zásobníky a mnoho dalších vlastností potřebných pro počáteční inicializaci. Její jméno je: InitD3D()
- Jako 2. funkce je potřeba nahrát 3D objekty ze souboru , dále jejich materiál a textury, je jí- InitGeometry().
- Funkce která je zodpovědná za vymazání z paměti všech nainicializovaných objektů a všech proměnných a struktur je cleanup().
- Ve funkci Setupmatrices() se nastaví všechny matice transformace pro zobrazení a pro jednoduchou animaci je možné nastavit rotaci kamery, což bude vést k vizuelní rotaci objektu.
- Poslední funkce render() je zodpovědná za nastavení Z-bufferu a dále volá předchozí nadefinované funkce pro vlastní vizualizaci.
- Nakonec je třeba napsat kód pro zobrazení okna pro aplikaci (ve fci. WinMain()) a zavolat výše nadefinovanou funkci Render.

Nastavení Visual Studia pro DirectX:

Pro přípravu vašeho vývojového prostředí (v mém případě Visual Studio .NET a Visual Studio 6.0) je potřeba tento nástroj připravit pro kompilaci vašeho kódu.

Přidejte do seznamu SDK/Include a SDK/Lib.



Obr.30: Nastavení Visual Studia 6.0

Pro chod celé mé aplikace i s hlavním dialogovým oknem je potřeba mít nainstalovaný .NET framework běhové prostředí, což sem nahrál na doprovodném CD a dále je potřeba mít nainstalovaný DirectX 8.1.

5.6 Vytvoření uživatelského programu, jeho popis a ovládání

Pro pouštění animací jsem vytvořil hlavní okno pomocí C# a .NET framework . Toto hlavní okno obsahuje možnosti volby uživatele volit vstupní parametry:

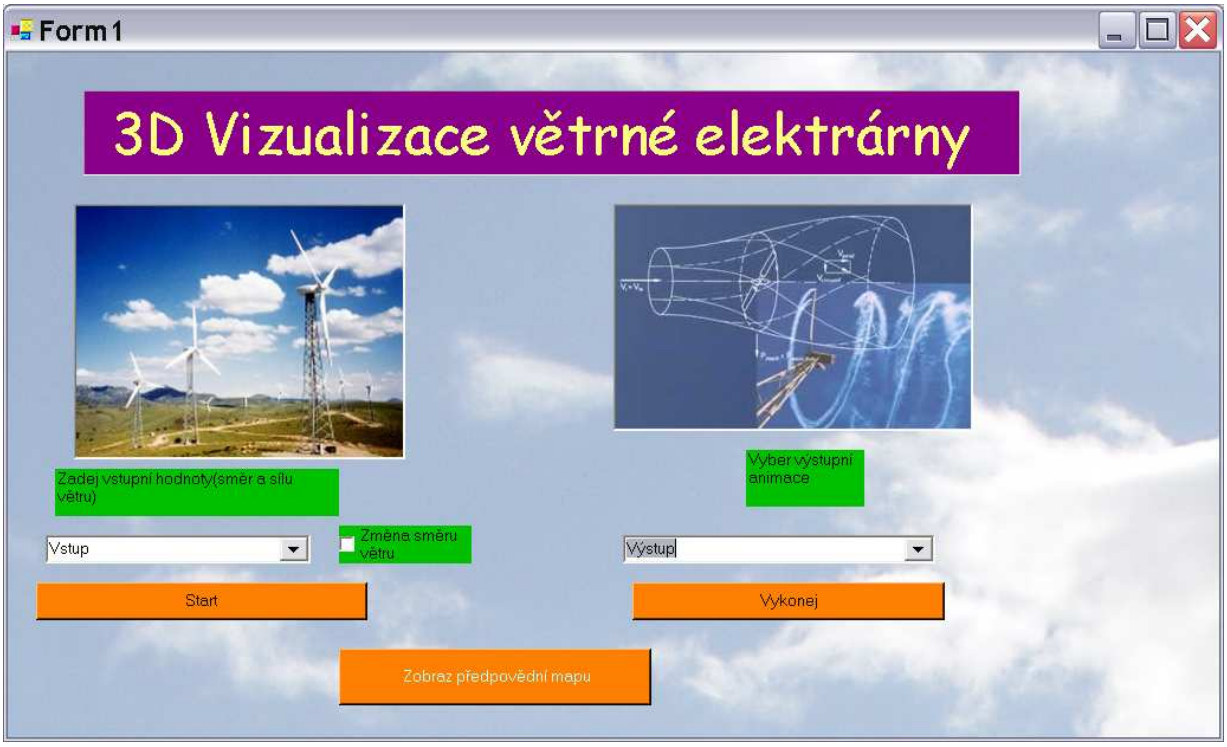
Sílu větru a jestli se bude měnit směr. Po odklepnutí tlačítka se zobrazí daný vstup.

Dále má možnost uživatel stisknout tlačítko pro zobrazení předpovědní mapy, která má ryze informativní charakter.

Pro výstupy zvolí uživatel pomocí nabídky výstupní animaci.

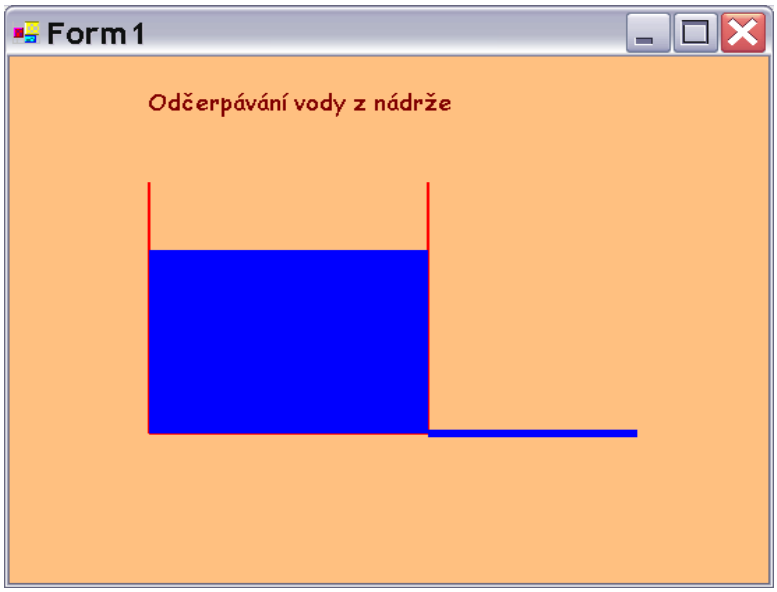
Po stisku tlačítek se zobrazí následující animace podle vybraných položek.

Na obrázcích jsou uvedeny jen základní animace.



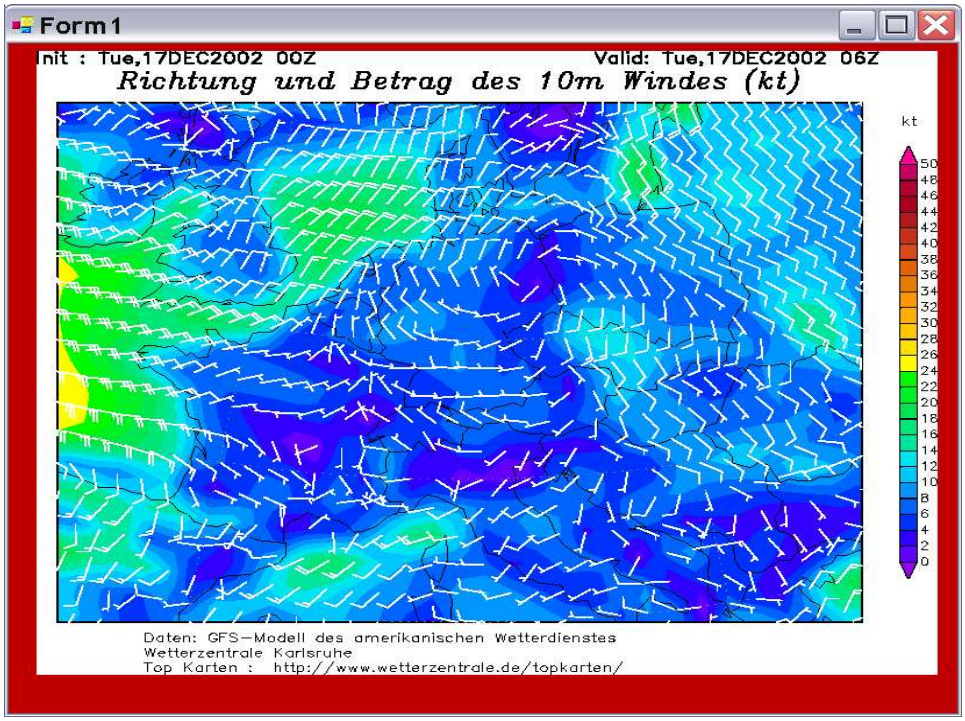
Obr.31: Hlavní okno programu

Přečerpávání vody v nádrži je vytvořeno pomocí fcí. GDI+ z frameworku:



Obr. 32: Znázornění přečerpávání

Zobrazení předpovědní mapy:

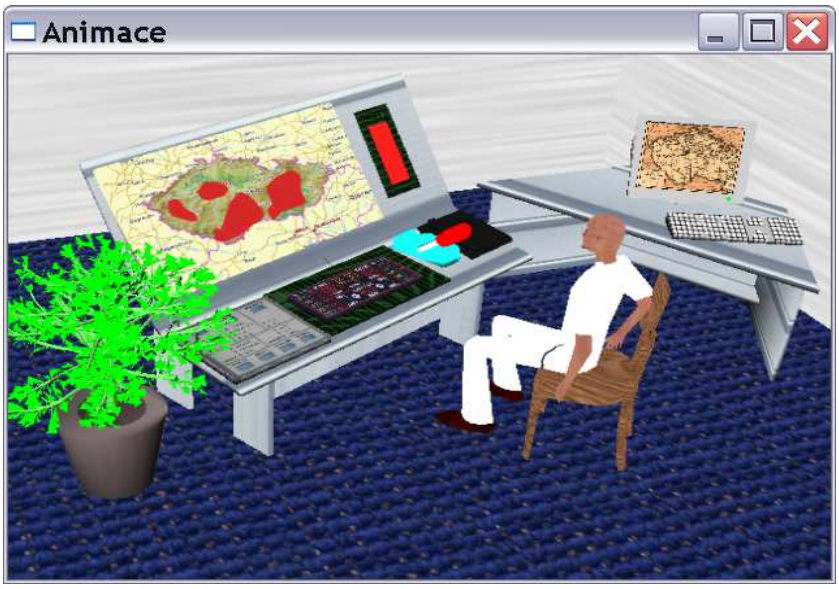


Obr. 33: Předpovědní mapa

Vizualizace podle uživatelem předdefinovaného vstupu elektrárny:

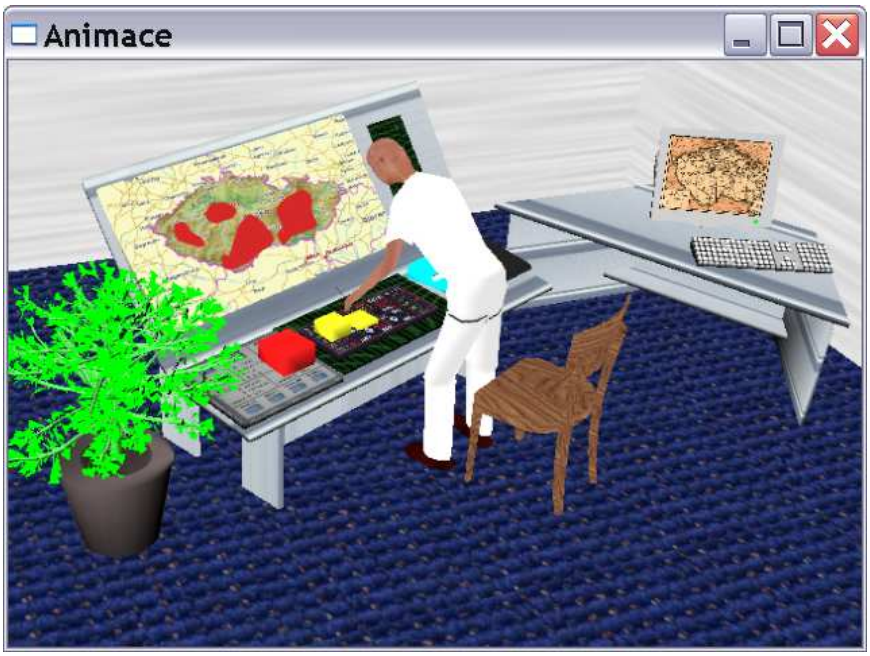
Obr.34: Animace větrné elektrárny

Animace výstupních veličin(výroby proudu na panelu):



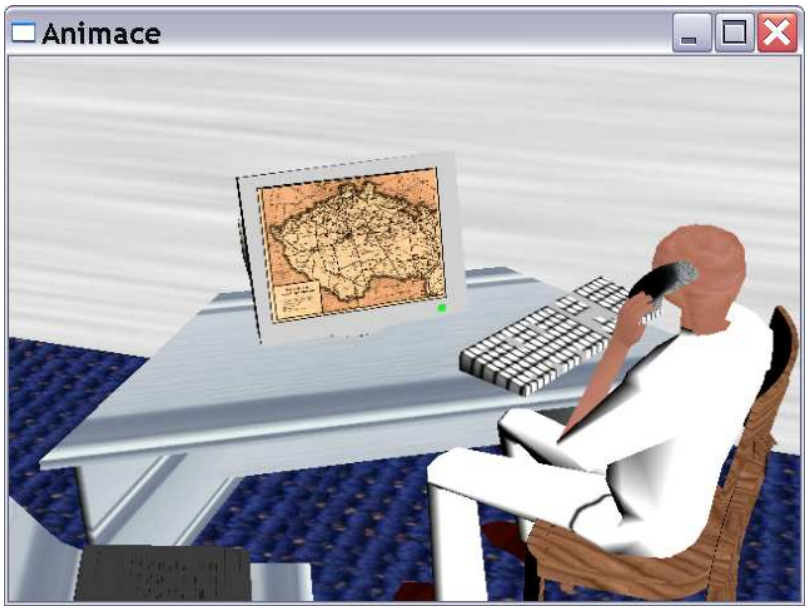
Obr. 35: Animace 1. operátora při monitorování systému

Animace práce operátora(Na obrázku je snižování výkonu a pouštění přečerpávání):



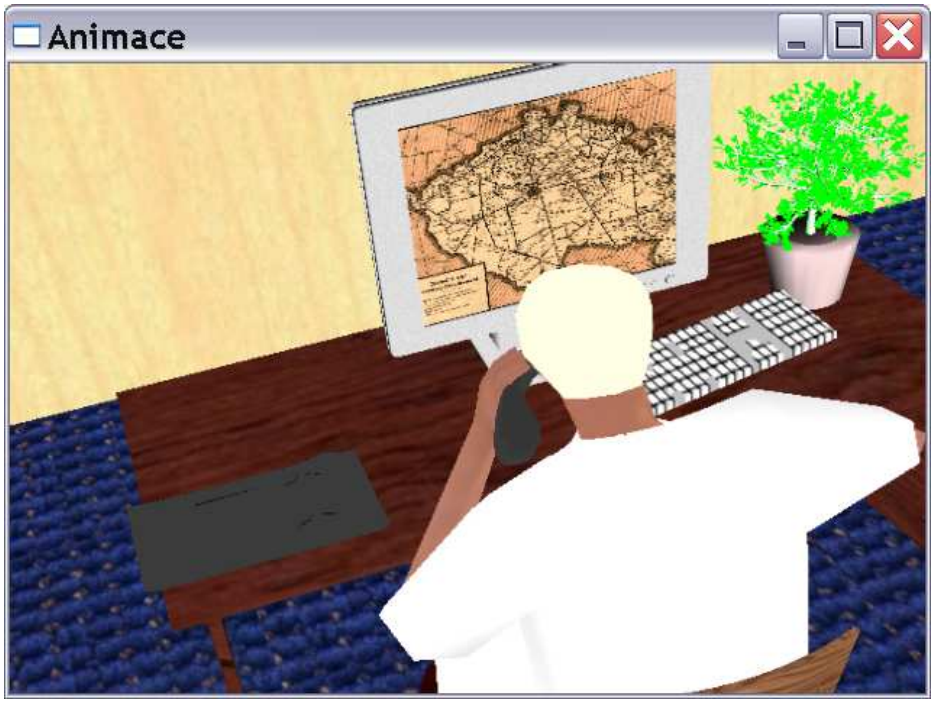
Obr.36: Animace 1. operátora při pracovní činnosti

Animace op. č.1 komunikujícího s okolím:



Obr.37: Operátor komunikující s op. č.2

Animace op. č.2 s okolím:



Obr.38: Operátor č.2 řídící nejdůležitější činnosti

6. Praktický význam pro zadávající firmu(závěr)

Na začátku mé práce jsem popsal možnosti využívání Zemských zdrojů energie. V mém případě se jednalo o větrnou energii. V další části jsem zde popsal možnosti vizualizací v technické praxi a mnou vybrané metody.

Poté jsem zde popsal mou používanou metodologii pro vytvoření animací v DirectX a naposledy jsem popsal vlastní vytvořené animace s respektováním všech stavů systému. Tato aplikace byla programována za použití DirectX 8.1 ve Visual Studiu .NET a jako grafický program bylo využito 3D Studio.

Moje diplomová práce byla vedena a konzultována se zadávající firmou Compic Praha, která se zabývá rozsáhlým softwarovým vývojem. Jedna z jejich animací vedla k zobrazování území při záplavách, což může vést k prevenci před vznikem škod.

Na moji práci je z mého pohledu nejdůležitější operátor č.1 pro ovládání, protože ho firma Compic bude používat místo jejich 2D operátora, kterého lze vidět na jejich webových stránkách.

Případnou další možností a její rozšíření by mohlo vést k jejímu propojení na rozsáhlé expertní systémy, které tato firma používá.

Seznam použité literatury:

[1] Petzold Charles: Programování ve Windows Win 32 API, Computer Press, Praha, 1999

[2] Herbert Schildt: Nauč se sám C++, SoftPress , Praha , 2001

[3] Eric Gunnerson : Začínáme programovat v C# , Computer Press, Praha , 2001

[4] Dalibor Kačmář : Programujeme .NET aplikace ve Visual Studiu .NET , Computer Press , Praha , 2001

[5] Petzold Charles: Programming Microsoft Windows with C# ,Microsoft press, Washington, 2002

[6] Výstavba větrných elektráren. http://www.proventi.cz/

[7] The industry's Foundation for High Performance Graphics. http://www.opengl.org/

[8] MSDN. http://msdn.microsoft.com/default.aspx

[9] The DirectX Experience. http://www.geocities.com/SiliconValley/Way/3390/about.html

Příloha

Přílohu předkládané diplomové práce tvoří zdrojové kódy animací v C++ a rozhraní Direct3D, včetně jejich 3D modelů v adresáři \Zdrojové kódy.

V tomto adresáři je kód v těchto souborech:

- Animace.cpp definuje počáteční nastavení zobrazení
- D3dmath.cpp definuje matematické funkce používané při výpočtech
- Stdafx.cpp obsahuje přidružené soubory
- VetrnikD3D.cpp je hlavní program pro renderování scény

V projektu jsou přidružené hlavičkové soubory:

Animace.h,Resource.h,d3dmath.h,stdafx.h,vetrnikD3D.h , ve kterých jsou nadefinována různá makra a proměnné používané v aplikaci.

Adresáře \WindowsApplication1 a \WindowsApplication3 obsahují zdrojové kódy v programovacím jazyku C# a používající prostředí .NET framework.

V adresáři \WindowsApplication2 se nachází hlavní dialogové okno pro spouštění všech animací.

Všechny výše uvedené zdrojové kódy je možné otevřít ve Visual Studiu .NET pomocí souborů s koncovkou *.sln.

Adresář \conv je program pro konverzi souborů 3D grafiky namodelované ve 3D studiu do .x formátu, který využívá Direct3D.

Dále je v příloze obsažen adresář \DirectX 8.1 Runtime for Windows 2000 DirectX, který je nezbytný pro chod animací.

Pro spouštění aplikace, která byla vytvořena v .NET je dále potřeba nainstalovat .NET framework, který se nachází v adresáři \DOT NET framework.

V adresáři \Hlavní spustitelný program je celá spustitelná aplikace, kterou lze spustit souborem WindowsApplication2.exe.

Celý text této práce je obsažen v adresáři \Diplomová práce text.

Prohlášení