Online modelovanie a prezeranie 3D objektov

(Milan Halabuk)

2010

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATEMATIKY, FYZIKY A INFORMATIKY

ŠTUDIJNÝ ODBOR 9.2.9 APLIKOVANÁ INFORMATIKA

Milan Halabuk

Online modelovanie a prezeranie 3D objektov

Bakalárska práca

Školiteľ: Mgr. David Běhal

Bratislava 2010

Poďakovanie

Chcem sa poďakovať svojmu školiteľovi Mgr. Davidovi Běhalovi za cenné rady a odbornú pomoc. Ďalej by som sa chcel poďakovať vedúcemu bakalárskeho seminára Mgr. Jurajovi Starinskému ako aj členom seminára za ich odborné pripomienky.

Abstrakt

Cieľom mojej práce je vytvoriť platformovo nezávislé prostredie na vytváranie 3D scén. Prostredie bude dostupné online a na jeho používanie bude potrebný webGL kompatibilný webový prehliadač. Práca bude mať dve hlavné časti. V prvej časti sa zameriam na vytvorenie modelovacieho nástroja, ktorý by mal byť prehľadný s intuitívnym ovládaním. Mal by byť vhodný pre ľudí, ktorý majú minimálne znalosti z počítačovej grafiky. Nemal by ich odradiť veľkým množstvom tlačidiel. K jeho prednostiam bude patriť to, že ho nebude treba inštalovať a taktiež bude zabezpečená multiplatformová kompatibilita. V druhej časti sa zameriam na tvorbu webového portálu, kde si budú môcť návštevníci prezerať vytvorené objekty.

Kľúčové slová

modelovanie 3D, online modely

Obsah

Zoznam obrázkov....6
1. Úvod....7
2. Analýza problematiky....8
2.1 Základné pojmy v renderovaní obrazu....8
2.1.1 Priebeh renderovania....8
2.1.1.1 Priestorové transformácie....10
2.1.1.2 Orezanie podľa zorného poľa(Clipping) 12
2.1.1.3 Odstránenie zadných strán(Back face culling)....14
2.1.1.4 Projekcia....15
2.1.1.5 Eliminácia skrytých povrchov, rasterizácia a tieňovanie....16
2.2 Základné princípy pri tvorbe súčasných webových aplikacií..19
2.2.1 WEB 2.0....19
2.2.2 Architektúra klient-server....20
2.2.2.1 Charakteristika klienta....21
2.2.2.2 Charakteristika servra....21
2.2.3 Princíp Model-View-Controller (MVC)....21
2.3 Návrhy riešenia....23
2.4 iné riešenia podobnej problematiky....24
3. Špecifikácia....26
3.1 Hlavné požiadavky....26
3.1.1 Požiadavky na modelársku a prezentačnú časť....26
3.1.2 Požiadavky na serverovú časť....27
3.2 Programovacie jazyky a platformy....27
3.2.1 PHP....27
3.2.2 HTML a HTML5....28
3.2.3 DOM(Document Object Model)....28
3.2.4 JavaScript....29
3.2.5 CSS (Cascading Style Sheets)....29
3.2.6 MySQL....29
3.3 Vstupné/Výstupné dáta....30
4. Návrh Riešenia....31
4.1 Návrh servra....31
4.1.1 Generovanie HTML a JavaScript kódu....31
4.1.2 Obsluha požiadaviek klienta....32
4.1.3 Autentifikácia užívateľa....33

4.2 Návrh Klienta.....34
4.2.1 Nástroj na vytváranie scén.....34
4.2.2 Nástroj na prezeranie scén.....35
5. Implementácia.....36
5.1 Implementácia Serverovej časti.....36

Zoznam obrázkov

Obrázok 2.0 – schéma procesu renderovania..... 8

Obrázok 2.1 – schéma renderovacej pipeline
Prevzaté z [RAG05].....9

Obrázok 2.2 -Objektové priestory
Prevzaté z [RAG05].....10

Obrázok 2.3 - Zorné pole kamery
Prevzaté z [AKE08]..... 11

Obrázok 2.4 – Pohľad z kamery
Prevzaté z [AKE08]..... 13

Obrázok 2.5 – Odstránenie zadných neviditeľných strán
Prevzaté z [SHA04].....14

Obrázok 2.6 – Perspektívna projekcia
Prevzaté z [OWE99]..... 15

Obrázok 2.7 – Prienik ihlanu a kocky.....16

Obrázok 2.8 – Klient-server architektúra..... 19

Obrázok 2.9 – Model View Controller.....20

Obrázok 2.10 – Základné vzťahy v MVC
Prevzaté z [MAR04].....22

1. Úvod

V dnešnej rýchle dobe sme svedkami veľkej informatizácie spoločnosti. Táto doba mení tvár sveta. Toto celé sa deje prostredníctvom nových technológií. Veľkú zásluhu na tomto jave má fenomén, s názvom internet. Internet sa ako všetko vyvíja, vylepšuje, čoraz viac sa na ňom objavujú nové webové služby, ktoré sme ešte pred pár rokmi považovali za výsostne lokálne (nainštalované na jednom konkrétnom počítači). Súvisí to najmä s vývojom nových štandardov v súlade s WEB 2.0 akými sú napríklad HTML5 alebo **webGL.** Tieto nové štandardy umožňujú vytvárať rozsiahle webové aplikácie, ktoré sa spúšťajú v prostredí internetového prehliadača. Otvárajú sa nám tým obrovské možnosti pre vývoj nových aplikácií, ktoré môžeme veľmi elegantne ponúknuť širokej skupine obyvateľstva.

V mojej bakalárskej práci sa zameriam hlavne na aplikačný interfejs javascriptu pre spracovávanie 3D počítačovej grafiky v prostredí prehliadača – webGL. Je to nový štandard, ktorý ešte nie je hromadne podporovaný, ale na základe dostupných informácií sa tento fakt má v budúcnosti zmeniť. Mojím hlavným cieľom bude vytvoriť webovú službu, ktorá bude využívať práve

webGL pre vykresľovanie a vytváranie 3D scén. Ďalej implementujem zdieľanie objektov prostredníctvom databázy na webovej stránke.

2. Analýza problematiky

V kapitole uvediem dôležité pojmy, zhrniem metódy a technológie potrebné pri riešení problematiky. Venovať sa budem základným princípom v počítačovej grafike, najmä renderovaniu. Spomeniem rôzne techniky pre zobrazovanie a reprezentáciu 3D objektov a na záver nejaké podobné technológie, ktoré sa zaoberajú zobrazovaním 3D obsahu na webe.

2.1 Základné pojmy v renderovaní obrazu

Renderovanie je jednou z kľúčových vecí v počítačovej 3D grafike. Pri tomto procese sa zobrazuje trojrozmerný objekt na dvojrozmernú plochu. Najčastejšie na monitor počítača. Základné atribúty, ktoré pri renderingu potrebujeme sú poloha pozorovateľa a zdrojov svetla. Naším cieľom je dosiahnutie čo najvyššej realistickosti obrazu.

2.1.1 Priebeh renderovania

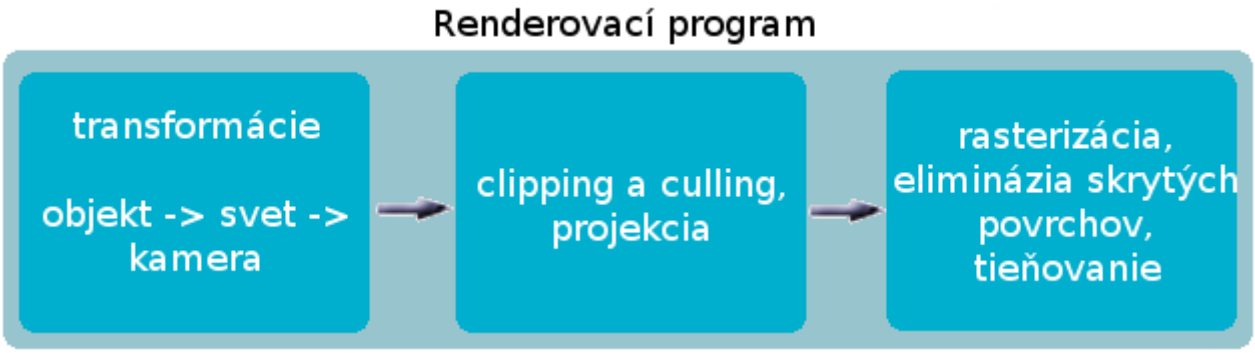
Pozrime sa bližšie na to ako renderovací program vykonáva transformáciu 3D sveta do výstupného obrázku. Celý tento proces je vykonávaný na základe matematických výpočtov. Je to akýsi protiklad k tomu akým spôsobom vykonávajú podobnú transformáciu videokamery alebo fotoaparáty. Na rozdiel od fotoaparátov renderovací program „nepozerá“ na scénu z diaľky ale využíva matematický model sveta, v ktorom vykonáva sadu precízne zadefinovaných úloh, ktoré taktiež nazývame renderovacia pipeline [RAG05]. Celý proces končí vygenerovaním realisticky vyzerajúcich obrázkov

scény.

Obrázok 2.0 – schéma procesu renderovania

V nasledujúcich riadkoch si zjednodušene ukážeme ako celý proces renderovania funguje. Jedná sa o komplexný proces, ktorý vykonáva dosť veľa komplikovaných matematických výpočtov.

Obrázok 2.1 znázorňuje klasickú renderovaciu pipeline. Dáta putujú v smere zľava doprava. Vstupom je matematický 3D model scény a výstupom je obrázok.



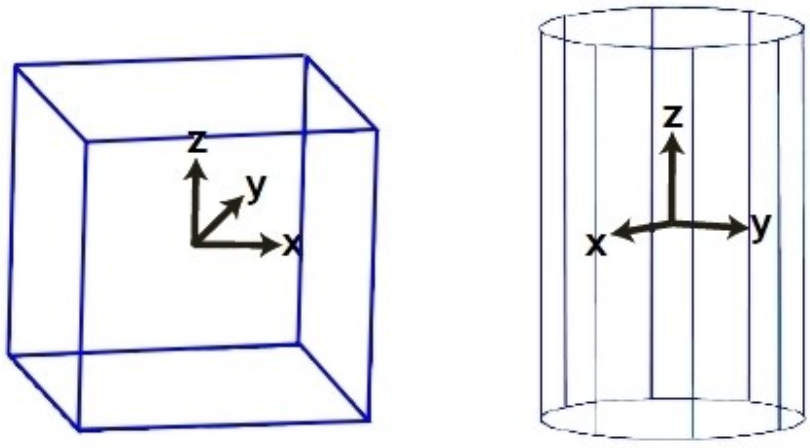
Obrázok 2.1 – schéma renderovacej pipeline

Model scény sa skladá z povrchov. Povrchy sa skladajú z materiálov, ktoré sú osvetľované zdrojmi svetla. Vo vymodelovanej scéne sa taktiež nachádza aj virtuálna kamera. Podľa jej polohy a uhlu natočenia renderovací program vytvára obrázky zo špecifickych uhlov pohľadu [RAG05].

2.1.1.1 Priestorové transformácie

Funkcie používané na zmenu veľkosti, polohy a natočenia objektov alebo kamery sa vo svete počítačovej grafiky nazývajú priestorové (geometrické) transformácie. [SHA04]

Prvým krokom v pipeline sú práve priestorové transformácie. Na obrázku 2.2 sú zobrazené 2 objekty – kocka a valec. Každý z týchto objektov je vymodelovaný nezávisle vo vlastnom objektovom priestore. Pre kocku sa začiatok súradnicovej sústavy nachádza v jej ťažisku a vzájomne kolmé osy X, Y a Z sú rovnobežné s hranami kocky. Kocku vieme modelovať aj použitím iných alternatívnych osí(napríklad počiatok súradnicovej sústavy stanovíme v jednom jej vrchole, osi X,Y sú navzájom kolmé a patria jednej strane, os Z je definovaná telesovou uhlopriečkou kocky). Valec býva zväčša modelovaný tak, že stred súradnicovej sústavy je v jeho strede. Os Z je rovnobežná s jeho výškou a osy X a Y sú rovnobežné s podstavou. Definícia povrchu pozostáva z parametrov, ktoré určujú tvar vzhľadom k základným osiam. Napríklad pre našu polygónovú kocku bude popis pozostávať z priestorových súradníc jej ôsmich vrcholov (Napr. [-1,-1,-1]) a každá z jej šiestich strán bude definovaná štyrmi vrcholmi (Napríklad vrcholy s indexmi 1,2,3 a 4).



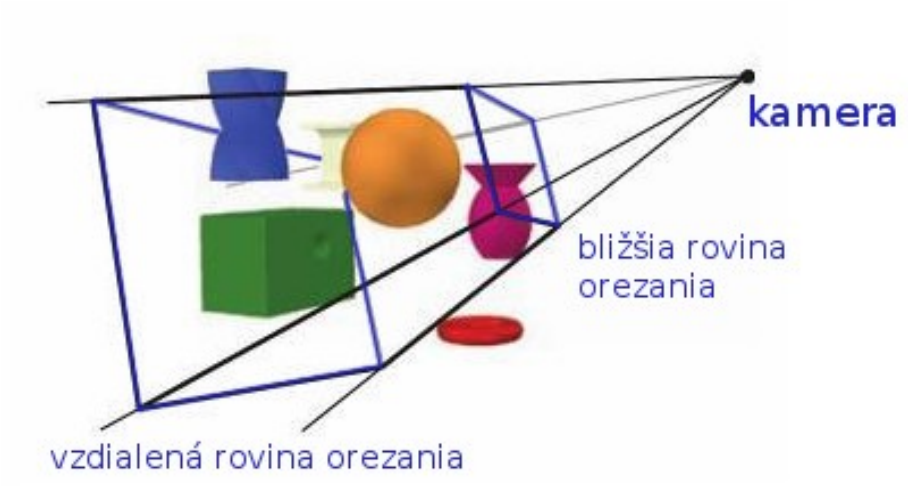
Obrázok 2.2 – Objektové priestory

Zjednodušene povedané, hrany predstavujú úsečky drôteného modelu, vrcholy sú miesta, kde sa stretávajú hrany a strany(povrchy) sú roviny

ohraničené vrcholmi a hranami [RAG05]. Podobne ako pre valec alebo kocku to platí aj pre ostatné objekty. Čiže každý objekt alebo model má svoj povrch definovaný vzhľadom k jeho vlastným osiam.

Dôležité je si uvedomiť, že každý objekt, ktorý ma byť umiestnený do globálneho súradnicového systému scény bude musieť prejsť kombináciou translácie, rotácie alebo zmenou veľkosti. Objekty sa pri vkladaní do scény umiestňujú podľa súradnicovej sústavy scény [SHA04]. Inými slovami je to ako keby kocka a valec boli umiestnení do stredu súradnicovej sústavy scény a ich osi by boli zhodné s osami scény. Potom sa s každým objektom vykoná translácia, rotácia alebo škálovanie tak aby bol výsledný stav podľa našich požiadaviek. Tento proces sa nazýva transformácia objektu do scény(object to world transformation) [SHA04]. Treba si tiež uvedomiť, že scéna obsahuje aj kameru, ktorá je tiež objektom. Kamera teda tiež musela prejsť procesom transformácie objektu do scény aby bola umiestnená na požadovanom mieste.

Pokiaľ sa chceme pozerať na vybrané objekty z pohľadu kamery, potrebujeme vykonať ďalšiu transformáciu. Zvyčajne presunieme počiatok súradnicovej sústavy do stredu kamery(os Z je rovnobežná so smerom natočenia kamery). Vznikne nám kamerový súradnicový systém [AKE08]. Pozície, natočenia a veľkosti objektov v scéne, ktorá obsahuje zdroje svetla a jednotlivé objekty môžu byť určené pomocou kamerovej súradnicovej sústavy. Čiže jednotlivé prvky scény podliehajú transformácii zo súradníc scény na kamerové súradnice.



Obrázok 2.3 – Zorné pole kamery

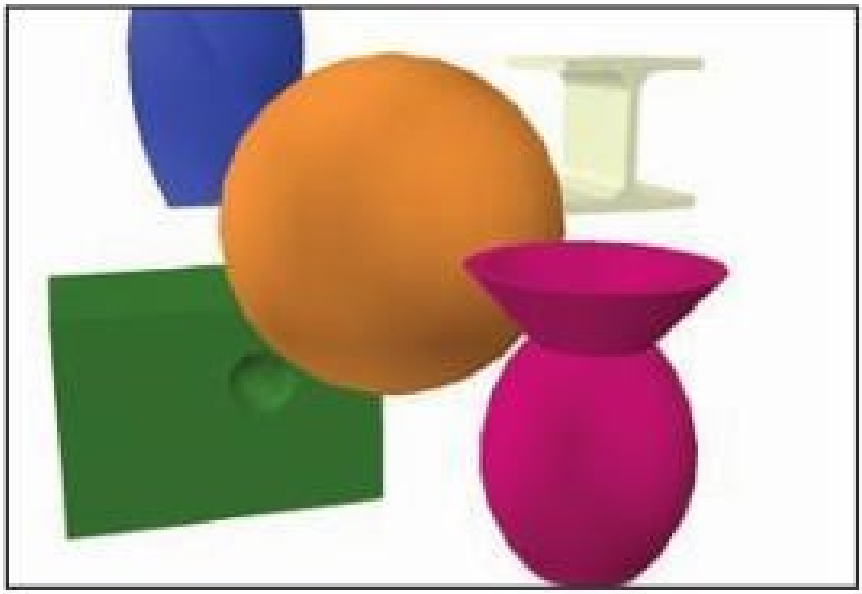
Pohľad na scénu cez kameru je zobrazený na obrázku 2.4. Presnejšie povedané, to čo kamera vidí je vo vnútri obdĺžnikového rámu. Je to kvôli tomu, že aj fyzické kamery alebo naše oči majú určité konečné zorné pole. Znamená to, že pohľad je obmedzený a sústredený na to čo je pred ním, naviac je rozšírený do strán a špecifické množstvo. Toto rozšírenie do strán sa nazýva šírka zorného poľa(FOV) [RAG05].

2.1.1.2 Orezanie podľa zorného poľa(Clipping)

Všimnime si obrázok 2.3, ktorý zobrazuje umiestnenie kamery v scéne. Môžeme na ňom vidieť útvar podobný pyramíde začínajúci pri kamere. Tento útvar sa nazýva objem zorného poľa(view volume) a predstavuje priestor, ktorý je pre kameru viditeľný. Objem a hĺbka zorného poľa sú obmedzené. Čiže kamera nemôže vidieť objekty, ktoré sú od nej ľubovoľne vzdialené. Vzdialená rovina orezania (tzv. far clipping plane), ktorá je rovnobežná s rovinou obrazu definuje maximálnu hĺbku zorného poľa kamery. Kamera taktiež nie je schopná

vidieť veľmi blízke povrchy objektov. Čiže blízka rovina orezania je umiestnená tesne pred kamerou a ohraničuje začiatok zorného poľa. Výsledkom je objem zorného poľa v tvare useknutej pyramídy definovaný práve týmito rovinami [AKE08]. Tento útvar je taktiež známy pod menom ihlan zorného poľa kamery (View frustum).

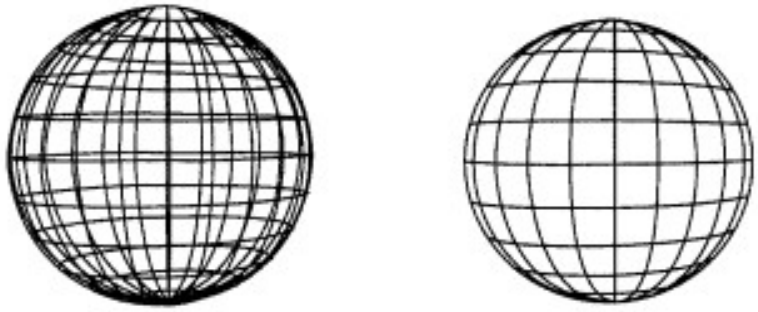
Takže teraz už máme stred súradnicovej sústavy presunutý do stredu kamery a všetky objekty umiestnené vzhľadom na tento stred. Môžeme začať spracovávať jednotlivé objekty scény tak aby to vyústilo do počítačového obrazu danej scény. Na obrázku 2.1 je táto sada operácií nazvaná clipping, culling a projekcia. Obrázok 2.4 zobrazuje výsledok operácie orezania(clipping). Na toto orezanie sa použilo šesť hraničných rovín ihlanu zorného poľa, ktoré vymedzujú objekty scény pre budúce spracovanie. Objekty ktoré kompletne ležia vo vnútri tohto ihlanu zostávajú nedotknuté. Naopak objekty, ktoré ležia mimo neho sú úplne zanedbané, pretože ich kamera nemôže vidieť. Ďalšiu skupinu tvoria objekty, ktoré pretína jedna alebo viac zo strán ihlanu zorného poľa. Tieto objekty, ktoré ležia čiastočne v zornom poli sú orezané. To znamená, že sú rozdelené na menšie časti(primitívy) podľa ktorých sa dá určiť, ktorá časť objektu je a ktorá nie je v zornom poli. Na obrázku 2.4 je vidieť ako je odrezaná vrchná časť modrého telesa. Objekty, ktoré nám zostanú po aplikácii orezania sa ďalej spracovávajú.



Obrázok 2.4 – Pohľad z kamery

2.1.1.3 Odstránenie zadných strán(Back face culling)

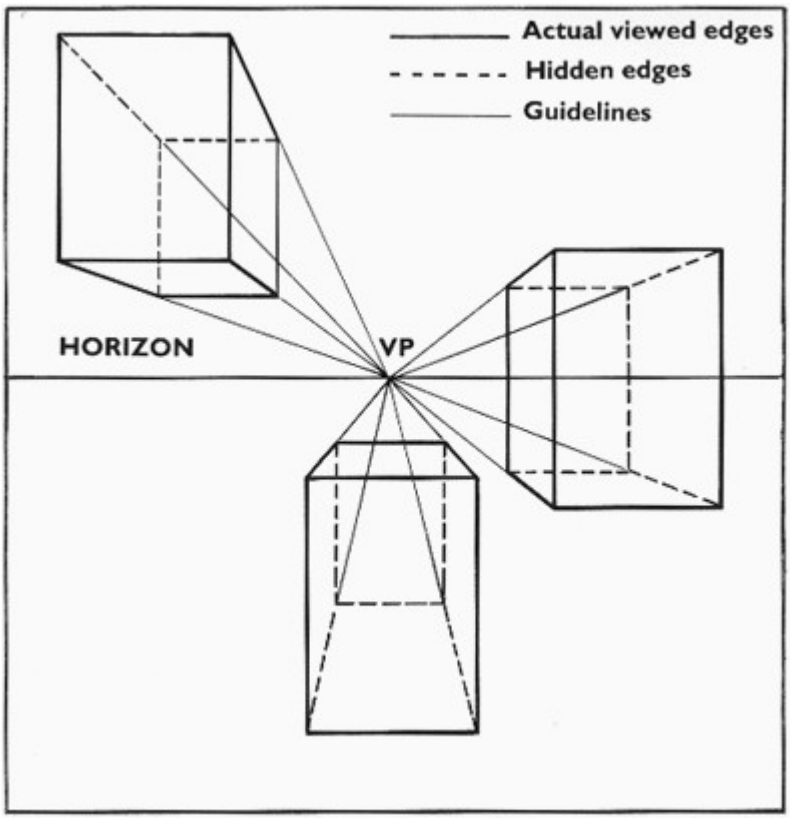
Renderovací program ďalej upravuje povrch objektov tak, že odstraňuje zadné strany, ktoré sú pre kameru neviditeľné(sú zakryté časťami iného povrchu) [WRI07]. Tento proces sa volá backface culling (odstránenie zadných neviditeľných strán). Základná myšlienka tohto procesu je taká, že každá strana má vonkajšiu a vnútornú plochu. Pre kameru sú viditeľné len tie strany, ktoré sú na ňu otočené vonkajšou plochou [RAG05]. Väčšina renderovacích programov dovoľuje nastaviť povrch(stranu) objektu ako obojstranný. Čiže viditeľný pre kameru z oboch strán. V takomto prípade renderovací program preskočí krok odstránenia pre kameru neviditeľného zadných strán a vyrenderuje všetky povrchy, ktoré sa nachádzajú v zornom poli.



Obrázok 2.5 – Odstránenie zadných neviditeľných strán

2.1.1.4 Projekcia

Nasledujúcim krokom v našej renderovacej pipeline je projekcia. Počas tohto procesu sa všetky povrchy, ktoré neboli v predchádzajúcich krokoch odstránené premietajú na rovinu. Vzniká tak výstupný obraz. Medzi najpoužívanejšie projekcie patrí perspektívna projekcia, ktorá vytvára dojem realistickosti. S perspektívnym videním sa stretávame aj v bežnom živote. Napríklad pri pohľade na dlhú rovnú cestu vidíme ako v diaľke pravá a ľavá krajnica splýva do jedného bodu. Obrázok 2.6 znázorňuje perspektívnu projekciu z pohľadu kamery. Predĺženia ľubovoľných rovnobežných hrán, ktoré nie sú zároveň rovnobežné po aplikovaní projekcie sa zbiehajú do jedného bodu (vanishing point) [OWE99].



Orbrázok 2.6 – Perspektívna projekcia

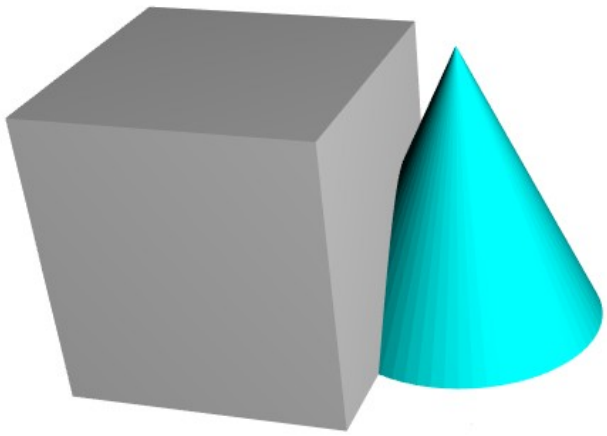
Po projekcii scény do 2D obrazu (roviny) sa z 3D objektov stanú 2D entity a spätná konverzia nie je možná. Ďalej sa na tieto novovzniknuté dvojrozmerné povrchy aplikuje rasterizovanie a tieňovanie.

2.1.1.5 Eliminácia skrytých povrchov, rasterizácia a tieňovanie

Po projekcii našich 2D povrchov do obrazovej roviny je potrebné vykonať elimináciu skrytých povrchov. Jedná sa vlastne o to, že povrchy, ktoré sú pred inými povrchmi v 3D scéne, budú vpredu aj v 2D obraze. Na zabezpečenie tohto stavu existuje niekoľko techník ale najpoužívanejšou je použitie takzvaného z-buffera (z predstavuje os, ktorá smeruje od kamery a podľa

ktorej sa meria relatívna hĺbka objektov v scéne) [RAG05].

Z-buffer si môžeme predstaviť ako kus pamäte, ktorej priestor je definovaný dvojrozmerným poľom a do každého prvku tohto poľa si renderovací program ukladá prechodné výsledky. Veľkosť tohto poľa závisí od rozlíšenia výstupného obrazu. Hodnoty, ktoré renderovací program ukladá do tohto poľa predstavujú porovnania hĺbok. Ešte predtým, ako sa čokoľvek zapíše do z-buffera sa jeho hodnoty nastavia na veľmi veľkú hodnotu, ktorá predstavuje nekonečnú vzdialenosť od kamery. Následne na to renderovací program začne prepisovať tieto hodnoty s reálnymi hĺbkami povrchov. Vždy prepíše vzdialenejší bližším (v prípade, že viac povrchov pripadá na ten istý prvok z-buffera). Predstavme si scénu z Obrázka 2.7. Ako prvý objekt spracujeme ihlan (kocku zatiaľ nie). Čiže všetky jeho viditeľné strany(povrchy) budú po spracovaní uložené v z-bufferi. Následne spracujeme kocku. Tie prvky z-bufferu, v ktorých boli uložené povrchy ihlanu a existujú nejaké časti kocky, ktoré sú bližšie sa počas tohto procesu prepíšu novými hodnotami. Keď sa dokončí výpočet hĺbok všetkých povrchov nasleduje odstránenie skrytých(neviditeľných) častí.



Obrázok 2.7 - Prienik ihlanu a kocky

Dôležité je si uvedomiť, že renderovací program môže spracovávať

objekty pre výpočet z-bufferu v ľubovoľnom poradí. Zisťovanie viditeľných strán má dva hlavné dôvody. Prvým z nich je, že renderovací program nemusí aplikovať tieňovanie na tie povrchy, ktoré sú skryté. A druhým je to, aby si výstupný obraz zachoval fyzikálnu korektnosť.

Okrem z-bufferu používa renderovací program ďalšie dvojrozmerné pole – raster. Raster tvorí základ pre výstupný obraz. Počas rasterizácie sú základné objekty(polygóny, body, čiary) transformované do príslušných pixelov [ROS06].

Posledným krokom v renderovacej pipeline je tieňovanie(shading). V tomto kroku sú pixle uložené v rastri zafarbené. Každý pixel môže obsahovať fragmenty nula, jedného alebo niekoľkých povrchov scény. Tieto povrchy majú nejakú farbu, na základe ktorej, renderovací program vypočítava výslednú farbu pixla. Zohľadňuje pritom polohu, intenzitu a farbu zdrojov svetla.

Po dokončení výpočtov tieňovania pošle renderovací program výstupný obrázok na obrazovku.

2.2 Základné princípy pri tvorbe súčasných webových aplikacií

V tejto časti opíšem niektoré základné pojmy zo sveta súčasných webových aplikácií. Zameriam sa najmä na moderné trendy akými sú WEB 2.0, architektúra klient-server a taktiež spomeniem základné princípy pri tvorbe OO PHP stránok pomocou MVC modelu.

2.2.1 WEB 2.0

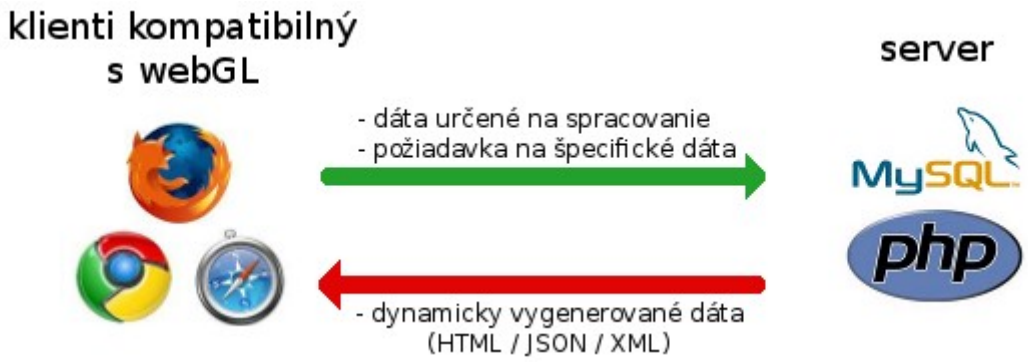
Tento termín ako prvý vyslovil Tim O'Reilly počas jednej konferencie v roku 2004. Za posledných 6 rokov sa toto slovo stalo vo svete World Wide Webu často používaným pojmom. Definovať čo je WEB 2.0 nie je ľahké, pretože neexistuje jednoznačná definícia. Na jeho opísanie sa používa niekoľko charakteristických čŕt, ktoré vystihujú jeho pravú podstatu [SHA08].

Jednou z hlavných čŕt je vytvorenie dojmu, že užívateľ je stredobodom pozornosti. Webová služba by mala byť navrhnutá tak, aby napĺňala všetky užívateľské potreby. Užívateľ by mal mať pozitívne pocity a zážitky (Rich user experience). Takéto webové služby by mali byť založené na ajaxe a mať jednoduchú obsluhu. Zvláštna pozornosť by sa mala klásť na dizajn. Užívatelia by mali mať možnosť si dizajn prispôsobovať vlastným potrebám. Ďalšou dôležitou charakteristikou WEB 2.0 webu je to, že užívatelia prispievajú k tvorbe obsahu. Mala by byť pri tom všetkom zachovaná platformová nezávislosť. Inými slovami by malo byť jedno z akého počítača, či operačného systému sa služba spúšťa. Webové stránky pred WEB 2.0 zvyčajne potrebovali

tým administrátorov. Moderné stránky sú automatizované a vyžadujú minimálne zásahy administrátora. Čím ďalej tým viac softvérových služieb je dostupných na internete formou webovej služby bez nutnosti inštalovať produkt v lokálnom počítači.

2.2.2 Architektúra klient-server

Táto architektúra predstavuje počítačovú sieť, v ktorej veľké množstvo klientov požaduje a následne prijíma dáta(služby) z centralizovaného servera [BRITA]. Klienti umožňujú užívateľom pristupovať k službám, ktoré poskytuje server. Taktiež zobrazujú výstup zo servera. Server čaká na podnety od klientov a až potom na ne zareaguje. Tento systém je efektívny vtedy, keď server a klienti vykonávajú rozdielne rutinné úlohy.



Obrázok 2.8 – Klient-server architektúra

2.2.2.1 Charakteristika klienta

- Inicializuje a odosiela požiadavky
- Čaká a prijíma odpovede
- Zvyčajne je v určitý čas pripojený na jeden alebo len zopár servrov
- Obsahuje grafické užívateľské rozhranie pomocou ktorého, ho užívatelia obsluhujú

2.2.2.2 Charakteristika servera

- Je pasívny
- Čaká alebo Prijíma požiadavky zaslané klientom, na ktoré následne odpovedá
- Naraz môže akceptovať žiadosti od väčšieho množstva klientov
- Zvyčajne neobsahuje žiadne užívateľské rozhranie a nie je teda priamo ovplyvňovaný užívateľmi

2.2.3 Princíp Model-View-Controller (MVC)

MVC predstavuje spôsob ako rozdeliť aplikáciu, alebo jej časť (aplikačný interfejs) do troch častí: model, view a controller [MAR04]. MVC bol pôvodne navrhnutý na vykonávanie operácií vstupu, spracovania a výstupu v grafickom užívateľskom rozhraní.

Obrázok 2.9 – Model View Controller

Model

- Model je objekt reprezentovaný dátami(napr. databáza) a funkciami(napr. funkcie pre prácu s databázou).
- Reaguje na žiadosti o informácie a vykonáva inštrukcie, ktoré editujú, mažú alebo pridávajú informácie.
- Slúži na zabezpečenie požiadaviek, ktoré stanovujú pravidlá pre prístup a aktualizáciu týchto údajov.

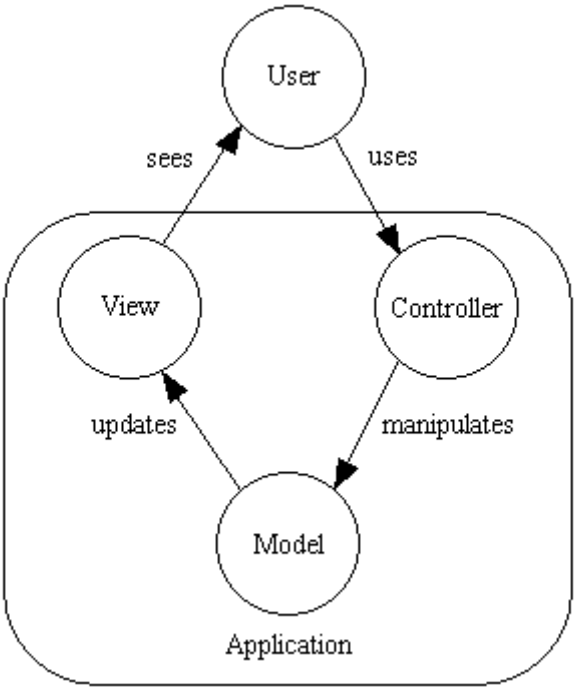
View

- Predstavuje určitú formu vizualizácie dát modelu.
- Vykresľuje na obrazovku špecifický obsah dát, ku ktorému pristupuje cez funkcie modelu.

Controller

- Mení informácie uložené v modeli, prijíma užívateľské požiadavky a podľa nich riadi MODEL a VIEW.

Model, View a Controller musia byť neustále prepojené. Aby sme to zabezpečili tak musia na seba navzájom odkazovať. Obrázok 2.10 ilustruje základné vzťahy v MVC.



Obrázok 2.10 – Základné vzťahy v MVC

2.3 Návrhy riešenia

V tejto podkapitole zhrniem riešenia, ktoré by som mohol použiť na riešenie problematiky zobrazovania 3D obsahu na internete. Budem sa sústrediť najmä na platformy kompatibilné so súčasnými webovými prehliadačmi. Ďalej budem klásť dôraz na možnosti, ktoré konkrétne riešenie ponúka a aké efektívne by bolo jeho použitie v praxi. Zhrniem klady a zápory jednotlivých platforiem a na záver napíšem prečo som sa rozhodol práve pre WebGL.

Na začiatku som zvažoval 2 riešenia. Prvým z nich bolo použitie JAVA appletu na báze gbVieweru [BEENE]. Toto riešenie je kompaktné, ľahko rozšíriteľné a nevyužíva žiadne neštandardné externé knižnice (napr. OpenGL). Taktiež je zabezpečená kompatibilita so všetkými bežne používanými

internetovými prehliadačmi. Medzi jeho hlavné nevýhody patrí absencia hardvérovej akcelerácie. Čiže použite tohto riešenia by bolo obmedzené len na zobrazovanie objektov s malým množstvom detailov. Po diskusii so školiteľom sme sa zhodli, že by bolo lepšie pohľadať platformu, ktorá umožňuje hardvérovú akceleráciu. Uvažoval som nad využitím technológie JOGL. Táto technológia využíva vývojovú(je vo vývoji) verziu JAVY, ktorá je naviazaná na aplikačný interfejs OpenGL a je navrhnutá na vykonávanie hardvérovo akcelerovanej 3D grafiky v JAVA programoch [JOGLA]. Dajú sa v nej teda tiež vytvárať applety, ktoré môžu byť potom umiestnené na web. Hlavnou nevýhodou je, že applet nevie priamo komunikovať s HTML DOMom ako to vie napríklad javascript. Tým pádom je ťažšie synchronizovať takúto aplikáciu so všetkými webovými službami, ktoré HTML DOM používajú.

Ďalšou technológiou, ktorá je vhodná pre tento účel má názov **WebGL**. Jedná sa o multiplatformový voľný(bez spoplatnenia) štandard určený pre 3D aplikačný interfejs založený na OpenGL ES 2.0 pracujúci cez CANVAS element (HTML5) ako súčasť DOMu(Document Object Model) [KHRON]. Zjednodušene povedané WebGL predstavuje rozšírenie javascriptu o funkcie podporujúce 3D počítačovú grafiku. Dôležitý je tiež fakt, že táto podpora je automatická, bez nutnosti doinštalovania akýchkoľvek doplnkov do operačného systému alebo prehliadača. Jedinou podmienkou pre spustenie je HTML5 kompatibilný prehliadač. Existuje ešte niekoľko možností ako zobrazovať 3D obsah na internete. Nebudem sa im všetkým venovať len spomeniem napríklad FLASH 3D, JAVA 3D alebo O3D.

2.4 iné riešenia podobnej problematiky

V tejto podkapitole rozoberiem iné riešenia podobnej problematiky. Svojím spôsobom nebudem mať veľa práce, lebo portálov, kde sa dá prezerať a

zároveň aj vytvárať 3D obsah zatiaľ veľa nie je. V nasledujúcich riadkoch sa pozriem na niektoré webové aplikácie, ktoré sa o toto snažia. Rozoberiem v skratke výhody a nevýhody týchto riešení.

Na úvod spomeniem web **insparia.com**. Je to stránka, ktorej hlavná služba je navrhnutá pomocou technológie JOGL. Z toho vyplývajú klady a zápory tohto projektu.

+ zabezpečená kompatibilita aj so staršími prehliadačmi

+ podpora hardvérovej akcelerácie

– odcudzenosť appletu od základnej stránky (spúšťa sa vo zvlášť okne)

– minimálna komunikácia s HTML DOMom

Ďalším zaujímavým projektom vo vývoji je **ogelo3d.free.fr**. Aj keď za posledných pár mesiacov dosť stagnuje. V podstate tam ide o vytvorenie modelovacieho prostredia na internete. Podľa ich slov to má byť podobné ako Blender.

+ použitie WebGL

+ podpora hardvérovej akcelerácie

+ jednoduchá synchronizácia s DOMom

– nutnosť použiť HTML5 kompatibilný prehliadač

3. Špecifikácia

Ako už bolo spomenuté cieľom mojej práce je vytvoriť webovú službu pre modelovanie jednoduchých 3D objektov v prostredí internetového prehliadača. Ďalšou dôležitou vecou je vytvorenie webstránky s databázou vytvorených objektov a s možnosťou ich prehliadania. Výsledný produkt môjho snaženia by mal byť jednoduchá, intuitívna, ľahko ovládateľná aplikácia, v ktorej sa budú dať modelovať 3D objekty.

3.1 Hlavné požiadavky

Medzi hlavné požiadavky moje aplikácie patrí platformová nezávislosť a dostupnosť pre širokú skupinu ľudí. Na zabezpečenie širokej dostupnosti bude musieť byť aplikácia umiestnená na webe formou interaktívnej stránky. Ďalej bude musieť byť zaradená do svetových internetových katalógov, aby sa o tejto službe ľudia dozvedeli. Pod pojmom dostupnosť stránky si nepredstavujem len to, či ju užívateľ objaví ale aj to ako prívetivo bude na neho pôsobiť. Aký dojem na ňom zanechá. Preto som sa rozhodol ju vyrobiť v súlade s WEB 2.0. Platformová nezávislosť je riešená použitím multiplatformových štandardov ako HTML5, PHP, JAVASCRIPT, CSS a MYSQL.

3.1.1 Požiadavky na modelársku a prezentačnú časť

Dôležitým prvkom bude jednoduchosť obsluhy. **Aplikácia bude určená pre bežných používateľov internetu, ktorý nemusia mať žiadne znalosti z poľa počítačovej grafiky**. Napriek tomu bude dôležité zabezpečiť, aby aplikácia bola silným nástrojom na modelovanie, v ktorom sa bude dať vytvoriť veľké množstvo objektov. Podobné požiadavky sú kladené aj na prezentačnú

časť aplikácie, ktorá musí mať málo tlačidiel a musí byť príjemné s ňou pracovať. Po technickej stránke bude nutné zabezpečiť rýchly beh aplikácie aj s väčším množstvom objektov v scéne prípadne s veľmi detailnými objektami.

3.1.2 Požiadavky na serverovú časť

Serverová časť bude plniť požiadavky zaslané užívateľom cez prehliadač. Bude sa využívať architektúra klient-server. Klient bude internetový prehliadač riadený užívateľom. Server bude zasielať vygenerovaný obsah klientovi a ten ho bude ďalej spracovávať. Dôležité bude zabezpečiť autentifikáciu užívateľov pomocou nejakého prihlasovacieho systému a tak isto zabezpečiť ukladanie užívateľských dát na server. Pod pojmom užívateľské dáta rozumiem vytvorené objekty, ich popisy, zaradenie do kategórií a podobne.

3.2 Programovacie jazyky a platformy

V tejto kapitole zhrniem programovacie jazyky, ktoré budem vo svojej práci využívať. Popíšem ich kľúčové vlastnosti poprípade spomeniem, prečo som sa pre danú technológiu rozhodol.

3.2.1 PHP

PHP (PHP:Hypertext Processor) je veľmi rozšíreným open source skriptovacím jazykom, ktorý je vhodný na vývoj webových aplikácií a môže byť vložený priamo do HTML kódu [PHP]. Jedná sa o programovací jazyk, ktorý beží na strane servera. V najbežnejšom prípade generuje HTML kód, ktorý je následne poslaný klientovi. Klient prijíma dáta vygenerované PHP skriptom bez toho aby vedel akým postupom boli vygenerované. Najlepšie na PHP je, že je

extrémne jednoduché pre začiatočníka, ale taktiež ponúka mnoho pokročilých vlastností pre profesionálneho programátora [PHP]. Taktiež umožňuje elegantné prepojenie s MySQL alebo PostgreSQL databázou.

3.2.2 HTML a HTML5

HTML (**H**yper **T**ext **M**arkup **L**anguage) je jazyk slúžiaci na vytváranie internetových stránok. Nie je programovacím jazykom ale znakovým (markup language) [W3S]. Tvoria ho špeciálny značky, takzvané **tagy**. Pomocou týchto tagov sa vytvára webová stránka. Tag je kľúčové slovo ohraničené ostrými zátvorkami (napr.
). Väčšina tagov je párových. Otvárací tag predstavuje začiatok bloku a uzatvárací predstavuje jeho koniec. Súbor tagov reprezentuje HTML dokument, ktorý je taktiež nazývaný aj internetovou stránkou.

HTML5 predstavuje najnovšiu verziu štandardu HTML. Implementuje v sebe technológie, ktoré boli doposiaľ doménou externých pluginov ako napríklad flash alebo java. HTML5 prináša niekoľko nových prvkov a atribútov, ktoré vystihujú potreby súčasných moderných stránok. Za zmienku stojí najmä podpora prehrávania videa pomocou tagu <video>. Veľmi dôležitou novinkou je podpora kresliacej plochy, ktorú definuje tag <canvas>. Spolu s tagom <canvas> súvisí aj podpora WebGL(viď. 2.3 Návrhy riešenia), ktoré pomocou neho vykresľuje 3D scénu na obrazovku.

3.2.3 DOM(Document Object Model)

DOM predstavuje aplikačný interfejs pre dokumenty typu HTML a XML. Poskytuje štrukturálnu reprezentáciu dokumentu a umožňuje upravovať jeho obsah alebo grafický výstup. Zjednodušene povedané spája webové stránky so skriptami alebo programovacím jazykom [MOZ1]. Najčastejšie slúži na

prepojenie HTML a JavaScriptu.

3.2.4 JavaScript

JavaScript je malý, objektovo orientovaný multiplatformový skriptovací jazyk [MOZ2], ktorý je primárne určený na dynamickú interakciu s HTML DOMom. Beží v internetovom prehliadači užívateľa, čiže je to skriptovací jazyk na strane klienta. Je zodpovedný za najväčšiu časť interaktívnych udalostí na web stránkach. Vkladá sa do HTML kódu stránok a len málo kedy je priamo viditeľný pre koncového užívateľa.

3.2.5 CSS (Cascading Style Sheets)

Kaskádové štýly predstavujú jednoduchý mechanizmus ako priať štýl(napr. typ písma, farbu, vzdialenosti rozostupov) do webového dokumentu [W3C]. Inými slovami CSS slúži na vizuálne formátovanie internetových dokumentov. Zabezpečujú oddelenie obsahu od vzhľadu. Je možné ich vložiť do externého súboru a tým zmenšiť veľkosť samotného HTML dokumentu.

3.2.6 MySQL

MySQL je najpopulárnejší otvorený SQL databázový systém [MYSQL] v súčasnosti vyvíjaný firmou Oracle. Má nasledovné základné charakteristiky.

- Relačný databázový systém

- Open source
- rýchly, spoľahlivý a ľahko sa používani
- pracuje v klient-server systémoch

3.3 Vstupné/Výstupné dáta

Vstupom do aplikácie bude súbor, ktorý užívateľ vloží pomocou grafického rozhrania. Tento súbor sa následne pošle na server pre ďalšie spracovanie. Konkrétne som si za vstup zvoli Microsoftový format .x, ktorý je primárne určený pre Direct 3D ale vie dobre poslúžiť aj v prípade WebGL. Do budúcna plánujem rozšíriť počet formátov, ktoré sa budú dať importovať. S takto importovaným súborom sa bude dať v aplikácii ďalej pracovať.

Primárnym výstupom bude súbor, ktorý sa ukladá na server. Bude obsahovať množinu objektov scény spolu s ich základnými údajmi. Medzi základné údaje patrí:

- Meno Objektu
- Typ Objektu
- Spôsob zobrazovania objektu(drôtený model alebo plné trojuholníky)
- Pozícia, Rotácia, Veľkosť a Farba objektu

Do budúcna plánujem rozšíriť aplikáciu o možnosť exportu scény do niektorých známych formátov (napr. WRLM).

4. Návrh Riešenia

V tejto kapitole načrtnem návrh riešenia danej problematiky. Budem vychádzať zo špecifikácie uvedenej v kapitole 3. Stručne popíšem navrhované riešenie pre serverovú ako aj pre klientovú časť. Na spustenie serverovej časti je potrebné mať nainštalovaný APACHE server spolu s PHP a MySQL. Na strane klienta je zas potrebné mať nainštalovaný webový prehliadač kompatibilný s HTML5 a WebGL (Napr. Firefox 3.7 alebo Chrome 5).

4.1 Návrh servera

Server je aplikácia, ktorá je riadená príkazmi z klienta a nemá žiadne užívateľské rozhranie. Túto aplikáciu nie je potrebné priamo ovládať. Slúži najmä na generovanie výstupných dát pre klienta ako aj na prácu so súbormi a taktiež na prepojenie klienta s databázou. Je to viacvláknová aplikácia, ktorá počúva na príkazy klienta na konkrétnom porte (v našom prípade je to port 80).

ZÁKLADNÉ ČRTY SERVRA

- **Nadviazanie spojenia** – spojí sa klient so servrom pomocou konkrétneho portu (Port 80)
- **Beh vlákna –** server vykonáva požadované spracovanie dát
- **Odoslanie výstupných dát klientovi –** odošle sa výsledok spracovania vo formáte JSON alebo HTML

4.1.1 Generovanie HTML a JavaScript kódu

Medzi základné funkcie servera patrí vytváranie výstupných HTML

dokumentov, ktoré obsahujú všetky dáta ako aj samotný zdrojový kód klienta. Pri tomto generovaní sa využíva PHP s MVC (Model-View-Controller) modelom. Webový portál bude rozdelený do troch hlavných častí.

- **Katalóg scén** – bude obsahovať jednotlivé scény uložené a utriedené do kategórií. Taktiež bude umožňovať jednoduché filtrovanie scén na základe ich patričnosti ku konkrétnej kategórii. Tieto Kategórie ako aj samotný obsah scén bude načítavaný z MySQL databázy.
- **Aplikácia na vytváranie scén** – z pohľadu servera bude táto časť obsahovať potrebné zdrojové kódy (Javascript a HTML) pre beh klientskej časti programu
- **Aplikácia na prezeranie scén** – server v tejto podsekcii podobne ako pri predošlej vygeneruje zdrojový kód aplikácie klienta, potrebný na jeho beh v internetovom prehliadači
- **Pomocník –** bude obsahovať stručný manuál k obsluhe aplikácie

4.1.2 Obsluha požiadaviek klienta

V tejto podkapitole popíšem požiadavky, ktoré môže zasielať klient na server. Komunikácia medzi serverom a klientom je realizovaná pomocou Javascriptu na klientskej strane a PHP na strane servera. Podobný typ komunikácie sa zvykne taktiež nazývať aj AJAX. Zameriam sa hlavne na popis spracovania týchto požiadaviek serverom.

Medzi základné požiadavky patrí **SAVE**. Ukladanie spracováva dáta zaslané klientom nasledovne. V prvej fáze sa pomocou metódy POST načíta dvojrozmerné pole dát (určuje jednotlivé **objekty a ich parametre**), **názov scény** a **id patričnej kategórie** (v prípade, že chceme len aktualizovať nejakú predošle vytvorenú scénu sa zašle aj **id scény**). Pomocou týchto údajov server následne pridá (prípadne aktualizuje) scénu do databázy. Na záver sa odošle klientovi správa o úspešnosti operácie ukladania.

Ďalšou nemenej dôležitou požiadavkou klienta je **LOAD**. Klient zašle pomocou metódy POST na server informáciu o požadovaných dátach. Konkrétnejšie táto informácia obsahuje id scény, ktorú požaduje. Server následne načíta z databázy meno scény, popis, definíciu jej objektov a id kategórie, do ktorej patrí. Tieto údaje spracuje, uloží do formy čitateľnej pre klienta a následne mu ich pošle.

Ďalšou požiadavkou je požiadavka na zaslanie definície konkrétnej primitívy, ktorá slúži ako základný stavebný kameň pri vytváraní 3D scén. Napríklad sa môže jednať o ihlan alebo valec. Server dostane na vstupe názov primitívy, ktorú požaduje klient. Následne načíta jej pole vrcholov, normál, pozície textúr a definície samotných strán. Dáta prevedie do formy čitateľnej pre klienta a nakoniec pošle.

4.1.3 Autentifikácia užívateľa

Webová služba bude verejne dostupná pre širokú skupinu užívateľom internetu. Z tohto dôvodu je nutné zabezpečiť autentifikáciu užívateľa, ktorý s aplikáciou pracuje. Pred uložením novej scény sa musí užívateľ najskôr prihlásiť poprípade zaregistrovať. Je to nutné z toho dôvodu aby server vedel kto vytvoril daný projekt. Taktiež to umožní užívateľovi vytvárať vlastnú databázu scén, ktoré môže editovať z hociktorého miesta na svete. Autentifikácia sa bude riešiť pomocou PHP a MySQL. Konkrétnejšie pri registrácii si užívateľ zadá vlastné prihlasovacie údaje a v prípade, že sú korektné sa uložia do databázy. Zadané heslo sa pred uložením zašifruje pomocou MD5. Takto zaregistrovanému užívateľovi sa stačí už len prihlásiť a môže využívať všetky funkcie aplikácie.

4.2 Návrh Klienta

Podľa špecifikácie sa pod pojmom klient chápe aplikácia vložená do HTML kódu, ktorej výkonná časť je naprogramovaná v JavaScripte. JavaScript umožňuje interakciu s HTML DOMom. Toto riešenie je ideálne pre použitie v prostredí internetu. Hlavným rozdielom oproti serverovej časti je prítomnosť užívateľského rozhrania. Pomocou neho užívateľ riadi celú aplikáciu.

ZÁKLADNÉ ČRTY KLIENTA

- **Obsahuje užívateľské rozhranie**
- **Prijíma pokyny od užívateľa**
- **Zasiela serveru dáta a požiadavky užívateľa**
- **Prijíma a zobrazuje požadované dáta**

4.2.1 Nástroj na vytváranie scén

Najdôležitejšou časťou klienta je nástroj, v ktorom užívateľ vytvára scény. Medzi najzákladnejšie funkcie patrí vkladanie elementárnych objektov s ktorými sa potom ďalej pracuje. Výsledná scéna vzniká na základe **rozmiestnenia**, **škálovania**, **rotácie** a **farby** daných objektov. Všetky tieto operácie uskutočňuje užívateľ intuitívne pomocou na to určených ovládacích prvkov. Ďalej užívateľské rozhranie obsahuje výpis všetkých vložených objektov, v ktorom je možnosť zvolený objekt odstrániť alebo duplikovať. Pokiaľ užívateľovi nestačia predvolené objekty, môže si **importovať** vlastné pomocou funkcie IMPORT OBJECT. Pred skončením vytvárania môže prihlásený užívateľ **scénu uložiť** a to dvomi spôsobmi. V prípade, že projekt bol už niekedy predtým uložený je možné ho aktualizovať alebo uložiť ako nový projekt. Pri ukladaní sa zadáva meno, popis a kategória do ktorej má byť scéna zaradená v katalógu. Po uložení je scéna zapísaná v databáze a užívateľ si ju môže

hocikedy znovu otvoriť pomocou funkcie otvoriť (open) a následne editovať.

4.2.2 Nástroj na prezeranie scén

Ďalšou časťou klientského rozhrania je nástroj na prezeranie scén vytvorených užívateľmi. Obsahuje jednoduché ovládanie, pomocou ktorého môže návštevník portálu otáčať a približovať zvolenú scénu. Taktiež môže pridávať komentáre, kde môže vyjadrovať svoje dojmy a pripomienky. Pri prezeraní nemusí byť prihlásený.

5. Implementácia

V tejto kapitole bakalárskej práce popíšem aplikáciu Meshmatrix, ktorá je výsledkom mojej práce. Táto aplikácia rieši problém vytvárania a zdieľania 3D scén na internete. Aplikácia pozostáva z .php a .js súborov, ktoré obsahujú jednotlivé triedy serverovej ako aj klientskej časti. Podobne ako v predchádzajúcej kapitole popíšem jednotlivé triedy, ich metódy a význam pre server a pre klienta.

5.1 Implementácia Serverovej časti

Ako som uviedol v návrhu riešenia, serverová časť využíva MVC model implementovaný v objektovo orientovanom PHP. Z tohto dôvodu sa jednotlivé triedy delia podla toho, z ktorej materskej triedy sú odvodené na 2 hlavné skupiny - **Controller** a **Viewer**. Na nasledujúcich stranách opíšem jednotlivé triedy, spôsob dedenia a inicializáciu jednotlivých funkcií.

Bibliografia

[RAG05] – **Ragvachary. 2005.** Rendering for Beginners Image Synthesis Using RenderMan. Focal Press, 2005.

[AKE08] – **Akenine-Möller, Haines, Hoffman. 2008.** Real-time rendering. A K Peters, 2008.

[SHA04] – **Shalini Govil-Pai. 2004.** Principles Of Computer Graphics. Springer, 2004.

[WRI07] – **Wright, Lipchak , Haemel. 2007.** OpenGL SuperBible 4th Edition. Addison-Wesley, 2007.

[OWE99] – **Scott Owen. 1999.** Perspective Viewing Projection. [Online][Dátum: 18. 05. 2010.] http://www.siggraph.org/education /materials/HyperGraph/viewing/view3d/perspect.htm

[ROS06] – **Randi J. Rost. 2006.** OpenGL® Shading Language, Second Edition. Addison Wesley Professional, 2006.

[SHA08] – **P. Sharma. 2008.** Core Characteristics of Web 2.0 Services. [Online][Dátum: 19. 05. 2010.] http://www.techpluto.com/web-20-services/

[BRITA] – **www.britannica.com.** client-server architecture.

[Online][Dátum: 19. 05. 2010.] http://www.britannica.com/ Ebchecked/ topic/1366374/client-server-architecture

[MAR04] – **Tony Marston. 2. 5. 2004.** The Model-View-Controller (MVC) Design Pattern for PHP. [Online] [Dátum: 19. 05. 2010.] http://www.tonymarston.net/php- mysql/model-view-controller.html

[BEENE] – **Gary Beene.** gbViewer Overview.

[Online] [Dátum: 20. 05. 2010.] http://www.garybeene.com/3d/3d-view.htm

[JOGLA] – **joglamp.org** [Online] [Dátum: 21. 05. 2010.] http://jogamp.org/jogl/www/

[KHRON] – **khronos.org** [Online] [Dátum: 21. 05. 2010.] http://www.khronos.org/webgl/

[PHP] – **php.net** [Online] [Dátum: 21. 05. 2010.] http://www.php.net/manual/en/intro-whatis.php

[W3S] – **www.w3schools.com** [Online] [Dátum: 21. 05. 2010.] http://www.w3schools.com/html/html_intro.asp

[MOZ1] – **developers.mozilla.org** [Online] [Dátum: 21. 05. 2010.]

https://developer.mozilla.org/en/DOM

[MOZ2] – **developers.mozilla.org** [Online] [Dátum: 21. 05. 2010.]
https://developer.mozilla.org/en/JavaScript

[W3C] – **w3.org** [Online] [Dátum: 22. 05. 2010.]
http://www.w3.org/Style/CSS/

[MYSQL] – **dev.mysql.com** [Online] [Dátum: 22. 05. 2010.]
http://dev.mysql.com/doc/refman/5.0/en/what-is-mysql.html