České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Roman Janovský**

Studijní program: Otevřená informatika
Obor: Softwarové systémy

Název tématu: **Víceuživatelský 3D editor na mobilní platformě**

Pokyny pro vypracování:

Analyzujte stávající mobilní aplikace umožnující vytváření a úpravu 3D geometrie.
Navrhněte a implementujte 3D editor pro mobilní platformu (primárně tablet se systémem ANDROID) umožňující kooperaci více uživatelů v rámci vytváření společné 3D scény v reálném čase. Předpokládejte, že každý uživatel ovládá vlastní zařízení. Editor bude umožňovat vytvářet geometrii, aplikovat materiály a textury. Vytvořte importní modul pro formát obj a exportní modul do formátu X3Dom. Vytvořte jednoduchý server pro sdílení exportovaných X3Dom scén.
Aplikaci otestujte s reálnými uživateli (3 kooperující uživatelé) vytvořením scény jejíž složitost budete konzultovat s vedoucím práce.

Seznam odborné literatury:

https://github.com/verse

Moderní počítačová grafika (2. vydání) - Jiří Žára, Bedřich Beneš, Jiří Sochor, Petr Felkel. Computer Press 2005.

Vedoucí: Ing. David Sedláček, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

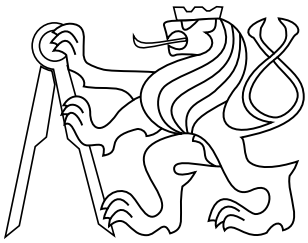
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26. 3. 2015

bakalářská práce

Víceuživatelský 3D editor na mobilní platformě

Janovský Roman



Květen 2015

Sedláček David Ing., Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačů

Poděkování

Chtěl bych v prvé řadě poděkovat svému vedoucímu práce Ph.D. Davidu Sedláčkovi za asertivní vedení, za poskytnutí vhodných podkladů využitých při této práci a za zapůjčení přístrojů na testování. Dále bych rád poděkoval svým participantům Martinu Janovskému, Bc. Věře Okruhlicové a Štefanu Okruhlicovi.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 21. 5. 2015 _____

Abstrakt

Cílem mé bakalářské práce je vytvořit 3D editor bežící na mobilní platformě Android, ve kterém by se mohlo několik uživatelů zároveň podílet, nezávisle na sobě, na tvorbě jedné scény. Navíc je potřeba zjistit, zda-li je takto kooperativní tvorba rozumná a efektivní.

Klíčová slova

3D; editor; víceuživatelský; android

Abstrakt

The goal of my bachelor thesis is to create 3D editor running on mobile platform, specifically on tablets with OS Android, in which several users could cooperate on creation of one 3D scene. Furthermore, it is necessary to find out, whether such cooperation is effective and meaningful.

Keywords

3D; editor; multiuser; android

Obsah

Zkratky

Použité pojmy:

GUI	Graphical User Interface
OS	Operating System
SDK	Software Development Kit
API	Application Programming Interface
XML	Extensible Markup Language
TCP	Transmission Control Protocol
GLSL	OpenGL Shader Language
$\vec{a}$	Značí vektor v prostoru
$\cdot$	Skalární součin dvou vektorů, např. $\vec{a} \cdot \vec{b} = c \in R$.
$*$	Součin dvou skalárů nebo skaláru a vektoru $\vec{a} * b = \vec{c} \in R^3$.
$\times$	Vektorový součin dvou vektorů $\vec{a} \times \vec{b} = \vec{c} \in R^3$.
MSFV	Make Solid Face Vertex
MEV	Make Edge Vertex
MEF	Make Edge Face

1 Úvod

3D scény je potřeba vytvářet postupně: umělci nakreslí svou představu, grafici co nejlépe převedou kresbu do 3D modelu a postupně ze všech modelů vytvoří scénu. Scéna se buď nachází na centrálním uložišti, nebo si autoři napíší vlastní editor, kam do scény přistupuje více lidí zároveň.

Vytvářením jedné scény a více modelů zároveň by se dalo předejít očividným chybám, které by se daly opravovat hned, co by se objevily. Moje aplikace by měla být právě takovým editorem, který umožňuje pracovat více lidem na jedné scéně v reálném čase.

Aplikace by měla fungovat na principu komunikace klient-server, kdy i server bude mobilní zařízení. Nebylo by potřeba nikde zakládat hromadné uložiště pro scény a tvůrcům by stačilo pouze propojit své přístroje a hned začít pracovat.

Nepředpokládám, že editor na mobilní zařízení může být, co se do funkcí a hardwarových nároků týče, tak rozsáhlý jako editory pro stolní počítače. Na to má vliv jednak nepříliš velká pracovní plocha (velikost samotného displeje) a hardwarové parametry jako velikost operační paměti, výkon procesoru a grafické karty. Jedná se především o prezentaci myšlenky spolupráce a vytvoření jednoduché aplikace na tvorbu a editaci geometrie.

Podle zadání bude editor založen na vytahování ploch. Tedy plocha se smaže, vrcholy duplikují a posunou a vytvoří se nová plocha. Staré a nové vrcholy se propojí hranami a mezi nimi se vytvoří další plochy. Aplikace bude doplněna o funkce pro vybírání geometrie, pro transformaci výběru, obarvování a texturování. Bude umožněno sdílení vytvořených scén a jejich prohlížení na stránkách, které poběží na školním serveru. Editor bude podporovat import scény ve formátu OBJ a export scény ve formátu X3D.

1.1 Stručný popis kapitol

1.1.1 Analýza

V analýze se zabývám rozborem problémů, které je potřeba řešit v OpenGL a na zařízeních s operačním systémem Android. Rozebírám v ní tři stávající aplikace pro tvorbu 3D scén a na tomto rozboru stavím svůj návrh aplikace.

V kapitole jsou dále shrnuty požadavky, které byly vytyčeny zadáním nebo které jsem považoval za užitečné. Požadavky jsou poté blíže vysvětlené ve scénářích případů užití.

1.1.2 Použité pojmy a Realizace

Nejprve vysvětluji důležité pojmy, které následně používám k vysvětlení algoritmů a datových struktur. Na to navazuji konkrétním přiblížením vlastní implementace a jak v nich výše uvedené algoritmy a datové struktury používám.

1.1.3 Testování s uživateli

V této kapitole vysvětluji jakým způsobem jsem svoji aplikaci testoval a jaké jsou výsledky tohoto testování. Součástí je také seznam použitých přístrojů a případné problémy s nimi.

1.1.4 Závěr

V závěru hodnotím dosažení vytyčených cílů a využitelnost aplikace. Také předkládám, jakým způsobem by bylo vhodné aplikaci dál směrovat.

2 Analýza

V této kapitole rozebírám chování existujících 3D editorů a jak ovlivnily návrh mého uživatelského rozhraní. Dále popisuji nezbytné náležitosti pro OS Android a OpenGL. Na konec kapitoly je zařazen seznam požadavků na aplikaci a přehled očekávaných případů užití.

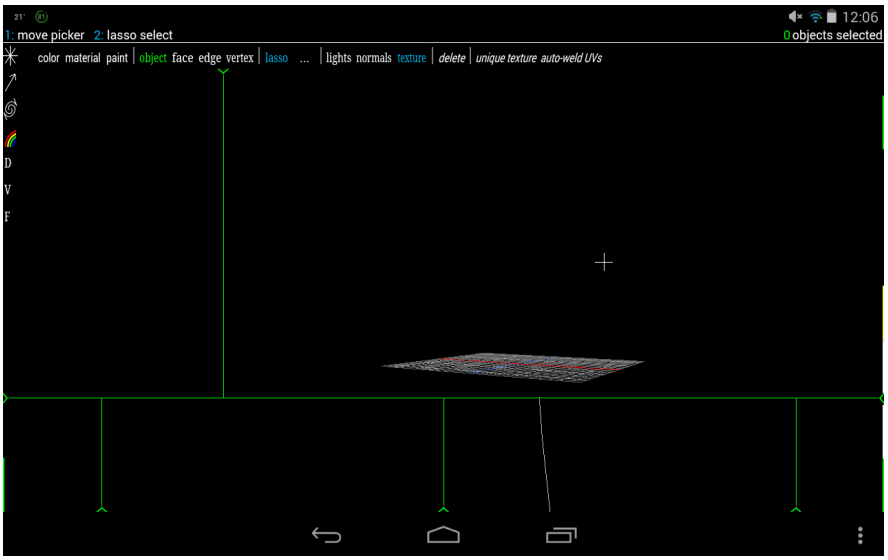
2.1 Stávající programy na vytváření 3D modelů

2.1.1 Spacedraw

Aplikace Spacedraw na Google Play Store od Scalisoft je profesionální aplikace na 3D modelování s podporou multi-touch zařízení.[1]

Spacedraw zvládá cokoli, co uživatel potřebuje pro 3D modelování. Bohužel nemá příliš dobře promyšlené ovládání. Plocha je rozdělena na více částí, do bloků ohraničených zelenými čarami, kde každá z nich umožňuje jiný způsob ovládání kamery. Přestože pro tyto části je možné nastavit jejich velikosti, rozbíjejí pracovní plochu a znepřehledňují práci.

Spacedraw poskytuje obsáhlou dokumentaci, ale pro potřeby mého vlastního editoru by bylo lepší intuitivní a jednoduché ovládání, přestože by neposkytovalo úplnou kontrolu nad funkcemi.

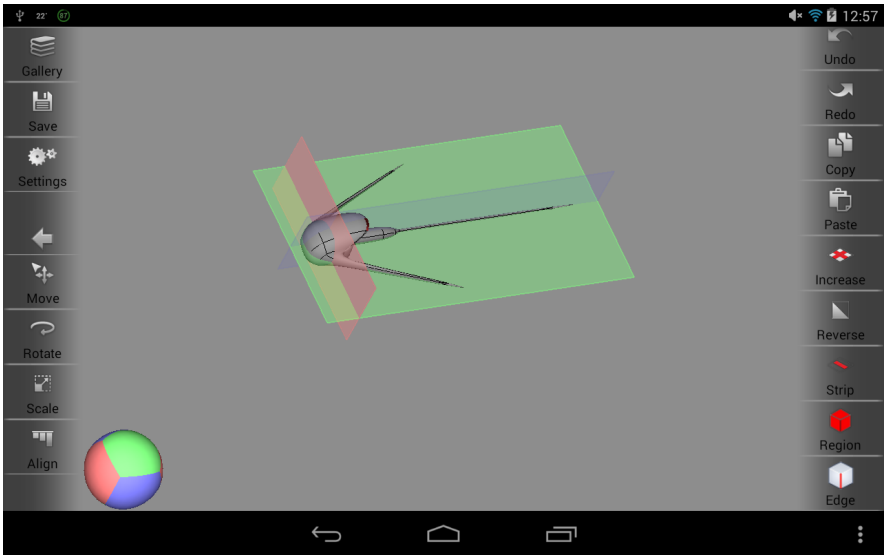


Obrázek 1 Spacedraw uživatelské rozhraní

2.1.2 SubDivFormer

Aplikace SubDivFormer na Google Play Store od ASCON je 3D editor založený na dělených plochách (*subdivision surfaces*).[2]

SubDivFormer má jednoduché intuitivní ovládání. Dva dotyky pro posouvání kamery a přibližování/oddalování a jeden dotyk na rotaci kamery kolem os X a Y. Nabízí základní funkce pro modelování jako jsou výběr, transformace výběru a extrude. Menu je po obou stranách a pokud je v něm více položek, než se vejde na obrazovku, tak se menu dá scrollovat. Jediné, co mi chybí, je jakákoli orientace v prostoru. Při posouvání ploch, hran nebo vrcholů není jasné, který směr je který.



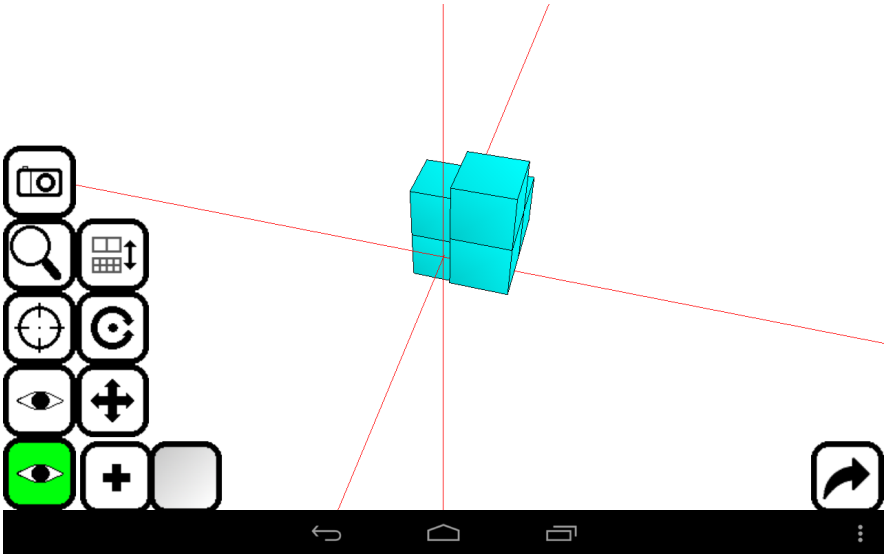
Obrázek 2 SubDivFormer uživatelské rozhraní

2.1.3 Qubism

Qubism na Google Play Store od Jonathana Quinna je aplikace na vytváření jednoduchých modelů pomocí krychlí.[3]

Každá krychle se dá pomocí předdefinovaných funkcí měnit na rozmanité tvary - kužele, válce, krychle se zkosenými hranami. Pomocí těchto tvarů se pak skládá model.

Funkce jsou schované v popup menu. Aktivní položka je zvýrazněna zeleně. Uživatelské rozhraní je jednoduché a uživatel se ho rychle naučí a pochopí. Práce díky tomu probíhá rychle.



Obrázek 3 SubDivFormer uživatelské rozhraní

2.1.4 Návrh rozhraní vlastní aplikace

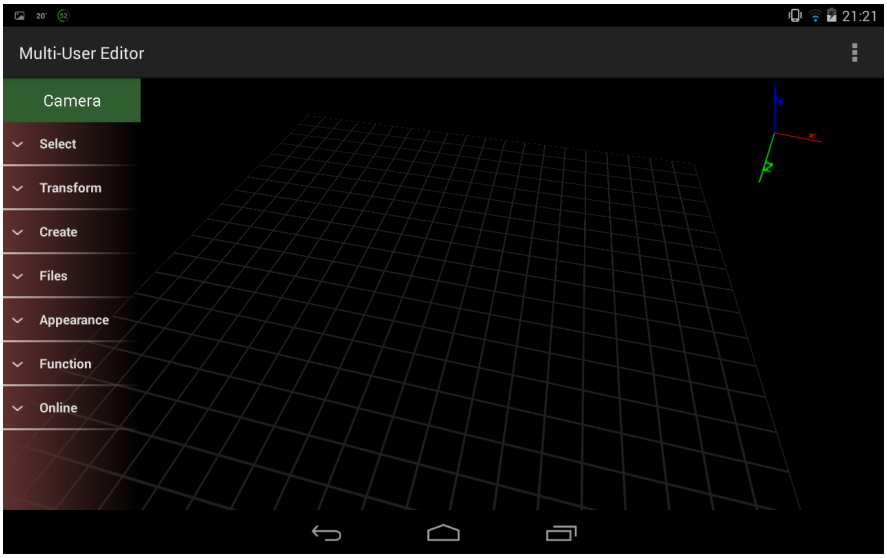
Vzhledem k vlastnímu vyzkoušení předchozích aplikací jsem se rozhodl vytvořit následující uživatelské rozhraní.

Aplikace při landscape orientaci bude mít jedno skrolovatelné menu na levé straně obrazovky. Menu bude obsahovat skupiny sdružující funkce s podobným chováním, tj. výběr různých komponent modelu nebo transformace výběru. Skupiny v menu se po rozkliknutí rozbalí a zobrazí samotné funkce. Při zvolení funkce se vypíše pro ověření, která funkce se používá.

V menu bude funkce kamera, která umožní uživateli kdykoli uzamknout nebo spustit ovládání kamery ve scéně. Když se pustí jakákoli funkce z menu, kamera se automaticky vypne. Tímto způsobem se předejde např. milnému výběru nebo pohnutí objektu při snaze otočit kamerou.

U aplikace SubDivFormer se mi špatně orientovalo v prostoru a při používání transformací jsem si nebyl jistý směrem os x, y, z. Proto ve středu scény bude čtvercový grid se čtverci velikosti 1x1 jednotka a v pravém horním rohu budou znázorněny směry os x, y, z pro lepší orientaci.

Protože považuji vytváření vlastních ploch pro ovládání, jako má aplikace Spacedraw, za nepříliš vhodné, rozhodl jsem se, že se veškeré akce budou ovládat pomocí základních gest. Pohyb dvou prstů bude pro posouvání kamery a zoom, jeden prst pro funkce a rotaci kamery. Každá funkce pro transformaci (rotace, posun, zvětšování/zmenšování) budou nezávislé na sobě a nebude možné provádět více než jednu v daný okamžik.



Obrázek 4 Stávající uživatelské rozhraní

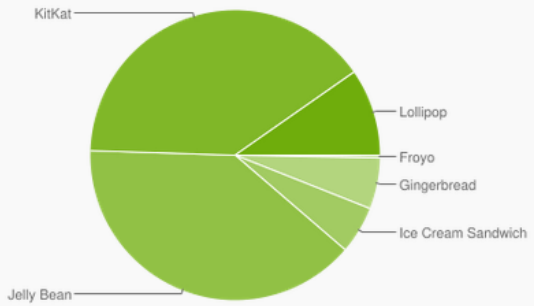
2.2 Požadavky na OS Android

OS Android pokrývá 79%[4] prodaných mobilních přístrojů, tj. tablet nebo smartphone. Z těch je potřeba vybrat API level tak, aby se pokrylo co největší množství přístrojů. Přestože se práce zaměřuje na tablety, rád bych vyzkoušel, bude-li možné modelovat v mé aplikaci i na smartphonu.

API level, neboli Aplication Programming Interface, je u OS Android vždy zpětně kompatibilní. Je tedy vhodné nastavit API level na co nejnižší možný. Pro svoji aplikaci jsem jako minimální zvolil API level 14. Android na tomto levelu poskytuje OpenGL ES 2.0 a umožňuje rozlišování dotyků rukou nebo stylem. Navíc představuje podporu pro Wi-Fi peer-to-peer s přístroji s certifikátem Wi-Fi Direct, kterým by se dala nahradit síťová komunikace, pokud by nebyl zrovna uživatel v místě s připojením.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

Data collected during a 7-day period ending on May 4, 2015.
Any versions with less than 0.1% distribution are not shown.



Obrázek 5 Podíl používaných verzí OS Android[5]

Se zvoleným API levelem 14 se pokryje 94% zařízení s OS Android, což činí 74% všech prodaných mobilních zařízení. API level 14 odpovídá systému 4.0.x Ice Cream Sandwich a vyšší.

2.3 Požadavky na OpenGL

Scéna v aplikaci je vykreslována pomocí OpenGL pro mobilní platformy: OpenGL ES verze 2.0. OpenGL ES 2.0 je dostupné od Android API levelu 8, tedy mnou požadovaný level 14 podporuje OpenGL ES 2.0 také.

Základem aplikací pro Android jsou *Activity*. *Activity* mají svůj vlastní životní cyklus. U operačních systémů pro stolní počítače se očekává stálý zdroj napájení, OS tedy udržují veškeré aplikace běžící. Ale OS Android je pro mobilní platformy s omezenou kapacitou baterie. Proto si každá aplikace, *Activity*, která přechází do pozadí, ukládá svůj stav do paměti, aby přestala odebírat energii a procesorový čas. Při tomto přechodu však dochází k vymazání OpenGL kontextu, např. GLSL shadery, a je ho poté znova potřeba vytvořit. Shader je také potřeba znovu vytvořit kdykoli dochází k otočení obrazu a přecházení mezi aktivitami v rámci aplikace.

2.4 Požadavky

2.4.1 Nefunkční požadavky

1. Aplikace bude naprogramovaná v jazyce Java.
2. Aplikace bude využívat OpenGL ES 2.0.
3. Aplikace bude pro Android 4.0.x Ice Cream Sandwich a vyšší.
4. Aplikace bude primárně pro landscape orientaci zařízení.
5. Aplikace bude cílena na tablety.
6. Aplikace bude umožňovat spolupráci po síti přes TCP sockety.
7. Struktura aplikace bude založená na modelu X3Dom.

2.4.2 Funkční požadavky

1. Bude vytvořena webová stránka umožňující sdílení scén.
2. Stránka pro sdílení scén bude obsahovat seznam nejnovějších příspěvků.
3. Sdílené scény se exportují ve formátu X3D.
4. Aplikace bude umět načítat modely ze souboru formátu OBJ.
5. Aplikace bude umožňovat obarvování a jednoduché texturování objektů.
6. Textury budou ve formátu JPG, GIF nebo PNG.
7. Bude možné měnit barvu jednotlivých ploch.
8. Bude možné vytvářet netrojúhelníkové 3D sítě.
9. Bude možné vybírat objekty, plochy a vrcholy 3D sítě.
10. Bude možné výběr posouvat, otáčet a zvětšovat/zmenšovat.

11. Uživatel se bude moci připojit k práci na cizím projektu.
12. Uživatel bude moci založit projekt pro kooperaci.
13. Aplikace bude nabízet sadu předdefinovaných tvarů.

2.5 Případy užití

Obecný přehled funkcí a jejich návaznosti je přiložen do příloh jako diagram případů užití.

2.5.1 Práce se soubory

Import souboru

1. Uživatel vybere položku Files v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Import file.
4. Systém nabídne uživateli výběr souboru.
5. Uživatel vybere soubor s formátem OBJ.
6. Systém zkontroluje, jestli umí načíst soubor.
   - 6.a Soubor se nedá načíst.
     i. Systém upozorní uživatele.
     ii. Uživatel vybere nový soubor nebo ukončí akci.
   - 6.b Soubor je možno načíst.
     i. Načtený objekt se přidá do scény.

Export souboru

1. Uživatel vybere položku Files v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Export file.
4. Uživatel vybere místo uložení.
5. Systém uloží soubor ve formátu X3D.

2.5.2 Vytváření geometrie

Vytvoř předdefinovaný objekt

1. Uživatel vybere položku Create v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Create "jméno objektu".
4. Systém vytvoří a přidá objekt do scény.

Extrude

1. Uživatel vybere položku Create v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Extrude.
4. Systém vytáhne všechny vybrané plochy ve směru jejich normály.

Nakreslení polygonu

1. Uživatel vybere položku Create v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Draw polygon.
4. Uživatel klikáním vytvoří polygon.

6.a Uživatel vytvoří objekt.

a) Systém zkontroluje, jestli polygon má alespoň 3 vrcholy.

a.i Polygon má 3 a více vrcholů.

i. Systém vytvoří polygon, pokud je platný[1].

ii. Systém upozorní uživatele, pokud polygon není platný.

a.ii Polygon má méně než 3 vrcholy.

i. Uživatel dostane možnost pokračovat na 6.b nebo 4.

6.b Uživatel ukončí kreslení polygonu.

2.5.3 Změna vzhledu

Textury

1. Uživatel vybere položku Appearance v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Texture.
4. Systém vybídne uživatele k výběru obrázku.
5. Uživatel vybere obrázek formátu GIF, JPG nebo PNG.

5.a Obrázek se podaří načíst.

i. Systém aplikuje texturu s UV souřadnicemi v ploše XY.

5.b Obrázek se nepodaří načíst.

i. Použije defaultní obrázek.

Materiál

1. Uživatel vybere položku Appearance v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Color.
4. Systém vybídne uživatele k výběru barvy a průhlednosti.
5. Uživatel vybere barvu.
6. Systém aplikuje barvu podle výběru.

Odstraň barvu

1. Uživatel vybere položku Appearance v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Remove face/vertex color.
4. Systém odstraní barvu plochy nebo vrcholu a nahradí ho barvou materiálu objektu.

[1] Polygon je platný, pokud je jednoduchý.

2.5.4 Výběr

Výběr

1. Uživatel vybere položku Select v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Select object/vertex/face.
4. Uživatel kliknutím na obrazovku vybere objekt.
   - 4.a Objekt vybrán.
     - a.a Vybrán mnou.
       - A. Vyřadí objekt z výběru a vrátí jeho původní barvu.
     - a.b Vybrán jiným uživatelem.
       - A. Objekt se nevybere.
   - 4.b Objekt ještě nevybrán.
     - i. Přidá objekt do výběru a změní jeho barvu pro vizualizaci.

Transformace

1. Uživatel vybere položku Transform v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Translate/Scale/Rotate.
4. Uživatel pohybem v daných osách změní polohu/rotaci výbraných objektů.

2.5.5 Funkce pro síťovou komunikaci

Sdílení scény

1. Uživatel vybere položku Online v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Share scene.
4. Systém vybídne uživatele k vyplnění jména scény.
5. Uživatel odešle soubor na server tlačítkem OK.
   - 5.a Soubor se podaří nahrát.
     - i. Server uloží scénu.
     - ii. Systém zobrazí odkaz na stránku se scénou.
   - 5.b Soubor se nepodaří nahrát.
     - i. Uživatel je informován o nezdaru.

Začít kooperaci

1. Uživatel vybere položku Online v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Start cooperation.
4. Systém spustí novou aktivitu s možností použít svůj přístroj jako server.
5. Uživatel spustí server.

Připojit se ke kooperaci

1. Uživatel vybere položku Online v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Join cooperation.
4. Systém spustí novou aktivitu s možností připojit se k serveru.
5. Potřeba zadat informace pro připojení.
   - 5.a Chce načíst kontakt.
     - i. Uživatel klikne na vybraný kontakt ze seznamu.

ii. Uživatel vyplní port kontaktu.

5.b Nechce načíst kontakt.

i. Uživatel vyplní pole pro adresu a port serveru.

6. Uživatel má možnost uložit přihlášení.

6.a Chce uložit informace.

i. Uživatel zaškrtne políčko Save contact.

ii. Uživatel vyplní jméno kontaktu.

iii. Po přihlášení je uživatel uložen do seznamu kontaktů.

6.b Nechce uložit informace.

i. Uživatel se přihlásí.

2.5.6 Doplňující funkce

Duplikování

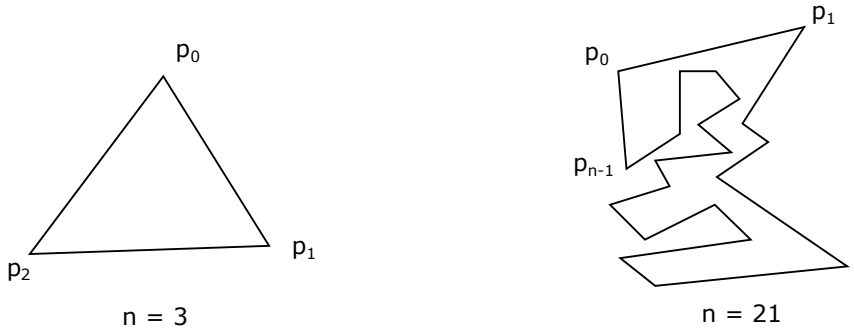
1. Uživatel vybere položku Function v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Duplicate.
4. Systém překopíruje souřadnice a normály objektu a vytvoří nový objekt s novým materiálem, ale stejnými vlastnostmi.

Delete

1. Uživatel vybere položku Function v menu.
2. Systém zobrazí nabídku akcí.
3. Uživatel vybere položku Delete object.
4. Systém vymaže ze scény všechna vybraná tělesa (celé objekty, nikoli pouze plochu nebo vrchol).

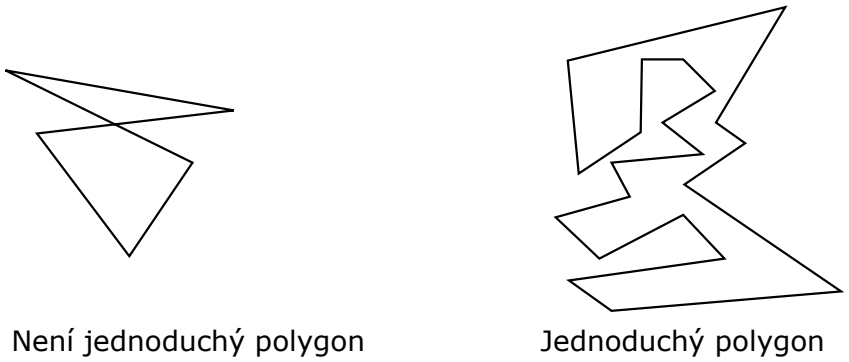
3 Použité pojmy

Polygon[6] Polygon neboli mnohoúhelník je uzavřený sled segmentů (p_i, p_{i+1}) pro $0 <= i <= n-1$ a (p_{n-1}, p_0), kde n >= 3. Polygon P je representován jeho vrcholy P $= (p_0, p_1, \ldots, p_{n-1})$.



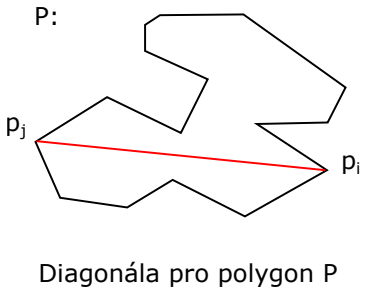
Obrázek 6 Příklady polygonu.[6]

Jednoduchý polygon[6] Jednoduchý polygon je polygon P, u kterého se žádné dvě po sobě jdoucí hrany neprotínají. Pro jednoduchý polygon je definovaný ohraničený vnitřek a ohraničený vnějšek, kde vnitřek je ohraničen hranami. Pokud se mluví o P, konvencí je zahrnovat vnitřek P.



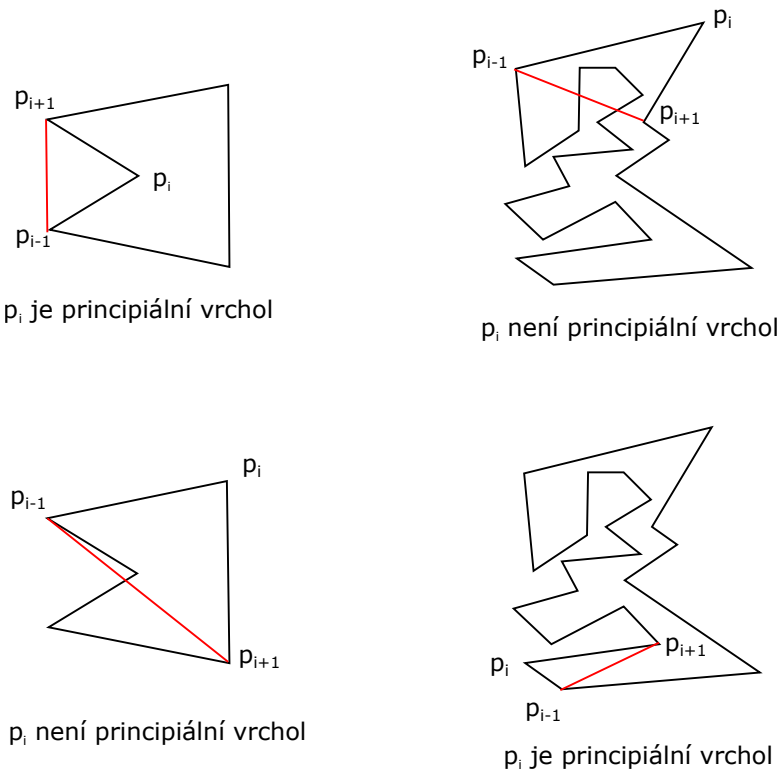
Obrázek 7 Neplatný polygon vlevo, jednoduchý polygon vpravo.[6]

Diagonála[6] Diagonála je úsečka spojující dva po sobě nenásledující vrcholy p_i a p_j, která celá leží v polygonu.



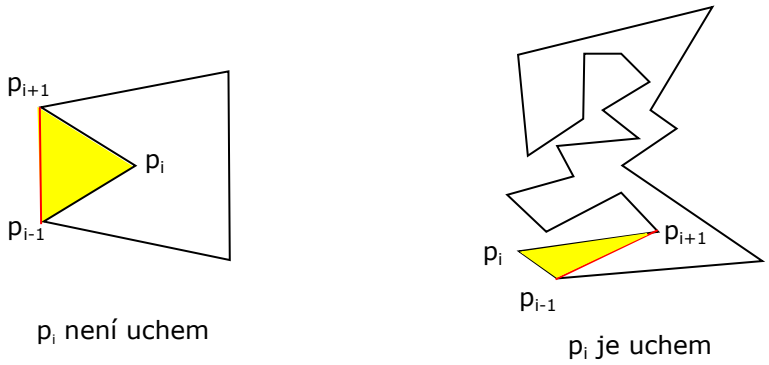
Obrázek 8 Diagonála pro polygon P.[6]

Principiální vrchol[6] Vrchol p_i se nazývá principiální, pokud diagonála (p_{i-1}, p_{i+1}) protíná hranici jednoduchého polygonu P pouze v p_{i-1} a p_{i+1}.



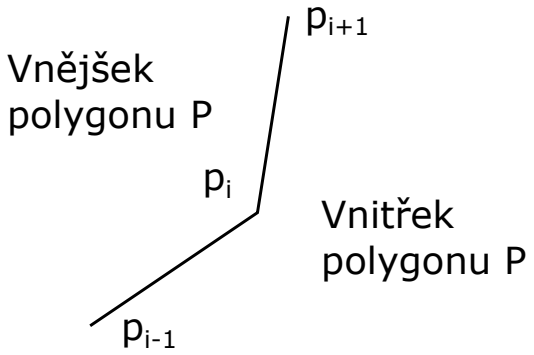
Obrázek 9 Příklady principiálních vrcholů.[6]

Ucho(ear)[6] Vrchol p_i jednoduchého polygonu P se nazývá ucho, pokud diagonála (p_{i-1}, p_{i+1}) celá leží v P. Dvě uši p_i a p_j se nepřekrývají, pokud vnitřek trojúhelníku (p_{i-1}, p_i, p_{i+1}) neprotíná vnitřek trojúhelníku (p_{j-1}, p_j, p_{j+1}).



Obrázek 10 Vrchol p_i v levém polygonu není uchem (diagonála (p_{i-1}, p_{i+1}) neleží v P), vrchol p_i je uchem.[6]

Konkávní vrchol[6] Vrchol p_i se nazývá konkávní, pokud se v něm jednoduchý polygon P stáčí vlevo, kdy vnitřek P je vpravo a p_i se prochází přes (p_{i-1}, p_{i+1}) po směru hodinových ručiček.



Obrázek 11 Vrchol p_i je konkávní.[6]

Orientovaná plocha Orientovaná plocha[7] rovnoběžníku je plocha určená svojí normálou. Od normální plochy se liší znaménkem, které je záporné, jestliže úhel z prvního do druhého vektoru tvořících rovnoběžník je po směru hodinových ručiček. Definice lze použít i na obecný mnohoúhelník.

Vektor a bod Rozlišujme bod a vektor v prostoru. Oba náleží R^3, ale vektor určuje směr od počátku souřadného systému a bod pouze polohu v souřadném systému. Rozdílem dvou bodů vzniká vektor.

Manifoldní objekt Objekt je manifoldní, tzv. varieta, jestli každý bod na jeho povrchu je obklopen malou plochou, která má topologii kruhu. Pro případ 3D polygonálních sítí to znamená, že každá hrana je obklopena právě dvěma plochami. Tedy objekt neobsahuje žádné díry.

4 Realizace

Tato kapitola se zabývá samotnou implementací a strukturou aplikace a obsahuje seznam a vysvětlení použitých algoritmů a datových struktur.

4.1 Použité algoritmy

4.1.1 Ořezávání uší (Ear Cut Algorithm)

Ořezávání uší[6] je algoritmus sloužící k triangulaci jednoduchých polygonů a je založen na tvrzení, na které přišel a dokázal Gary H. Meisters z University v Nebrasce v roce 1975[8]:

Two Ears teorém Kromě trojúhelníků má každý jednoduchý polygon alespoň dvě nepřekrývající se uši.

Algoritmus postupně vyhledává a ořezává uši polygonu P a vytváří z existujících vrcholů P nové trojúhelníky. Toho je potřeba pro vykreslování obecných polygonů, protože OpenGL požaduje buď trojúhelníkové, nebo čtvercové polygony.

Pseudokód funkce FindEar(P)

$i = 0$
ucho nenalezeno = true
while ucho nenalezeno
  if konvexní
    if trojúhelník tvořený p_{i-1}, p_i, p_{i+1} neobsahuje konkávní vrchol
      ucho nenalezeno = false
  if ucho nenalezeno
    $i = i + 1$
return p_i

Pseudokód funkce Triangulate(P, P', Normal)[9]

if vrcholy P jsou seřazené po směru hodinových ručiček
  seřaď vrcholy proti směru hodinových ručiček
while P má nenavštívené vrcholy
  $p_i = FindEar(P)$
  if p_i nalezen
    odstraň p_i z P a přidej trojúhelník (p_{i-1}, p_i, p_{i+1}) do P'
  else
    P neobsahuje další uši, ukonči algoritmus

V případě funkce *FindEar(P)* lze konkávní vrchol chápat tak, že žádný ze zbývajících vrcholů P neleží v trojúhelníku (p_{i-1}, p_i, p_{i+1}), vyjma vlastních vrcholů trojúhelníku (p_{i-1}, p_i, p_{i+1}).

Algoritmus od Johna Ratcliffa[9] pracuje ve 2D a testuje, ve které polorovině tvořené hranou trojúhelníku bod p_0 leží. Zjištění, ve které polorovině bod p_0 leží, je opět převedeno na test směru řazení vrcholů - po směru nebo proti směru hodinových ručiček.

Směr řazení se dá testovat výpočtem orientované plochy polygonu. Má-li plocha kladný obsah, pak je polygon řazen proti směru hodinových ručiček, v opačném případě po směru.

Aby se algoritmus mohl použít i ve 3D musí se všechny vrcholy zarovnat do jedné z rovin XY, XZ nebo XY, a poté vypustit příslušná souřadnice.

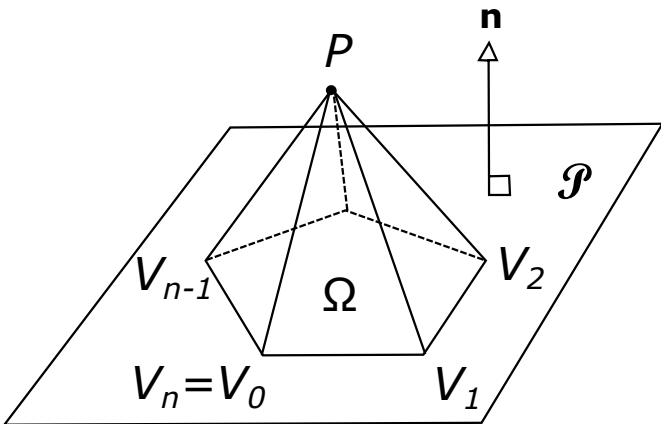
Ve své aplikaci využívám tento postup pouze pro seřazení vrcholů ve funkci *Triangulate(P,P',Normal)* a samotný test, jestli bod leží v trojúhelníku, jsem nahradil hledáním průsečíku trojúhelníku a přímky tvořené bodem p_0 se směrovým vektorem opačným k normále trojúhelníku.

4.1.2 Výpočet plochy rovinného polygonu[10]

Tento algoritmus vypočítá orientovanou plochu polygonu a tím se dá zjistit, je-li polygon řazen proti směru hodinových ručiček nebo po směru. Toho se využívá při triangulaci.

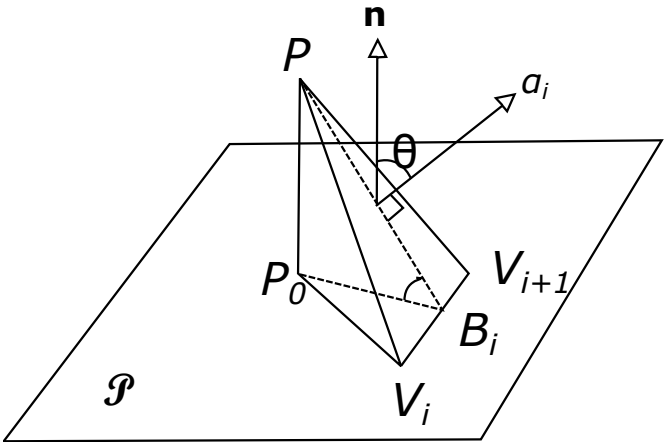
Definujme pro použití ve výpočtu plochy obecný rovinný polygon Ω jako množinu vrcholů $V_i = (x_i, y_i, z_i)$ pro $i = 0 \dots n$ s $V_n = V_0$, kde všechny jeho vrcholy leží na stejné rovině **P** s jednotkovým normálovým vektorem $\vec{n}$.

Mějme libovolný bod $P \in R^3$, S každou hranou $e_i = V_iV_{i+1} \in \Omega$ tvoří trojúhelník $\Delta_i = PV_iV_{i+1}$. Potřebujeme tedy vypočítat podstavu jehlanu tvořeném trojúhelníky Δ_i a polygonem Ω. Toho docílíme promítnutím trojúhelníkových stran tohoto jehlanu na rovinu **P** a sečtením orientovaných ploch trojúhelníků vzniklých projekcí.



Obrázek 12 Jehlan tvořený bodem P a polygonem Ω.[10]

Každému trojúhelníku $\Delta_i = PV_iV_{i+1}$ přiřadíme vektor $\vec{\alpha_i} = [(V_i - P) \times (V_{i+1} - P)]/2$, který je kolmý k Δ_i a jehož velikost je rovna ploše přislušného trojúhelníku. Promítneme bod P do bodu P_0 v rovině $\mathbf{P}$ a vytvoříme trojúhelník $T_i = P_0V_iV_{i+1}$. Poté vytvoříme úsečku P_0B_i z bodu P_0 do bodu $B_i \in e_i$ tak, že úsečka P_0B_i je kolmá na e_i.



Obrázek 13 Vlastnosti rozebíraných vektorů[10]

Jelikož úsečka PP_0 je také kolmá k e_i, body PP_0B_i definují rovinu kolmou k e_i, a tedy úsečka PB_i je kolmá k e_i. Velikost $|PB_i|$ je výška trojúhelníku Δ_i a velikost $|P_0B_i|$ je výška T_i. Dále úhel Θ jimi svýraný je roven úhlu mezi $\vec{n}$ a $\vec{\alpha_i}$. Z toho vychází:

$$\begin{gathered} S(T_i) = \tfrac{1}{2}|V_iV_{i+1}||P_0B_i| = \\ = \tfrac{1}{2}|V_iV_{i+1}||PB_i|\cos\Theta = \\ = S(\Delta_i)\cos\Theta = \\ = |\vec{n}||\vec{\alpha_i}|\cos\Theta = \\ = \vec{n}\cdot\alpha_i. \end{gathered}$$

Tato orientovaná plocha je kladná, pokud vrcholy T_i jsou seřazeny proti směru hodinových ručiček, když se podíváme na rovinu $\mathbf{P}$ z vrcholu $\vec{n}$. Součtem těchto orientovaných ploch získáme obsah polygonu Ω:

$$\begin{gathered} S(\Omega) = \textstyle\sum_{i=0}^{n-1} \vec{n}\cdot\vec{\alpha_i} = \\ = \frac{\vec{n}}{2}\cdot\textstyle\sum_{i=0}^{n-1}(PV_i \times PV_{i+1}). \end{gathered}$$

Dosazením $P = (0,0,0)$ dostaneme $PV_i = V_i$ a

$$S(\Omega) = \frac{\vec{n}}{2}\cdot\textstyle\sum_{i=0}^{n-1}(V_i \times V_{i+1}),$$

která je použita při implementaci algoritmu.

4.1.3 Vrhání paprsku

Vrhání paprsku je metoda běžně se používající pro test protnutí paprsku s objektem a při výpočtu osvětlovacího modelu. Jeho hlavní výhodou při hledání, jestli paprsek protíná objekt, je množství získaných informací. Například pro test protnutí lze použít vykreslení scény, kde každý objekt má unikátní barvu založenou na vybraném identifikátoru, a vybrání bodu, odkud se vrhá paprsek. Ale z toho se jednoduchým způsobem víc zjistit nedá. Zato při vrhání paprsku se automaticky spočítá průsečík a vzdálenost počátku paprsku a průsečíku. Paprsek nemusí být ani přímkou, ale i křivkou např. pro projektily. Moje aplikace využívá pouze hledání průsečíku při výběru geometrie a při vytváření nového polygonu.

Intersekce trojúhelníku a paprsku[11]

Pro potřeby OpenGL musí všechny polygony být buď čtverce, nebo trojúhelníky. Jelikož moje aplikace podporuje i obecné polygony, tak je potřeba, aby tato itersekce fungovala i pro ně. S využitím výše zmíněné triangulace můžu tuto intersekci testovat postupným procházením trojúhelníků vytvořených při triangulaci.

Každý bod $x \in R^3$ v trojúhelníku (v_0, v_1, v_2), kde $v_0, v_1, v_2 \in R^3$, může být definován jako konvexní kombinace jeho vrcholů, tedy:

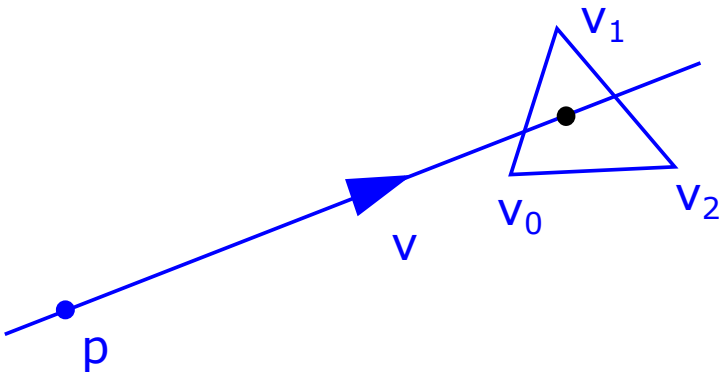
$$x = (1 - u - v) * v_0 + u * v_1 + v * v_2 \; pro \; u, v >= 0 \; a \; u + v <= 1.$$

Známe také parametrickou rovnici přímky:

$$x = o + d * \vec{l},$$

kde o je bod na přímce a $\vec{l}$ je směrový vektor přímky. Tedy pro bod, který leží zároveň na přímce i v trojúhelníku musí platit:

$$o + d * \vec{l} = (1 - u - v) * v_0 + u * v_1 + v * v_2.$$



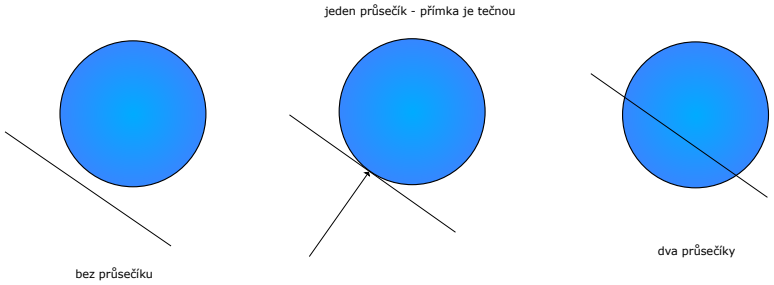
Obrázek 14 Průsečík přímky a trojúhelníku [11]

Můžeme tedy problém hledání průsečíku předefinovat na: existuje trojice (d, u, v) taková, že řeší rovnici výše a splňuje podmínky pro u, v? Je-li odpověď ano, pak takový průsečík existuje.

Intersekce koule a paprsku[12]

Jelikož bod jako takový nemá žádný objem a těžko by se pomocí prstů vybíral, přidal jsem mu obalovou kouli, která se chová při vybírání jako bod v prostoru.

Při hledání průsečíku koule a přímky mohou nastat tři případy. Přímka je tečnou koule a existuje pouze jeden průsečík. Přímka není tečnou koule, ale protíná ji. Pak existují právě dva průsečíky. Ve třetím případě neexistuje žádný průsečík.



Obrázek 15 Možné průsečíky přímky a koule.[12]

Definujme rovnici koule:

$$||x - c||^2 = r^2,$$

kde x je bod na kouli, c je střed koule a r její poloměr. Mějme rovnici přímky:

$$x = o + d * \vec{l},$$

kde d je vzdálenost podél přímky od počátečního bodu, $\vec{l}$ je jednotkový směrový vektor, o je počáteční bod přímky a x je bod na přímce.

Dosazením rovnice přímky do rovnice koule získáme:

$$||o + d * \vec{l}||^2 = r^2 \Leftrightarrow$$
$$\Leftrightarrow d^2(\vec{l} \cdot \vec{l}) + 2d * (\vec{l} \cdot (o - c)) + (o - c) \cdot (o - c) - r^2 = 0$$

Vyjádříme si rovnici jako kvadratickou formu:

$$ad^2 + bd + c = 0$$

$$a = \vec{l} \cdot \vec{l}$$

$$b = 2 * (\vec{l} \cdot (o - c))$$

$$c = (o - c) \cdot (o - c) - r^2$$

d získáme vypočítáním diskriminantu:

$$d = \frac{-(\vec{l} \cdot (o-c)) \pm \sqrt{(\vec{l} \cdot (o-c))^2 - (\vec{l} \cdot \vec{l}) * ((o-c) \cdot (o-c) - r^2)}}{(\vec{l} \cdot \vec{l})}.$$

Jelikož $\vec{l}$ je jednotkový vektor, pak jeho druhá mocnina velikosti, neboli $\vec{l} \cdot \vec{l}$, je rovna jedné. Můžeme tedy dále diskriminant zjednodušit:

$$d = -(\vec{l} \cdot (o - c)) \pm \sqrt{(\vec{l} \cdot (o - c))^2 - (\vec{l} \cdot \vec{l}) * ((o - c) \cdot (o - c) - r^2)}.$$

Je-li hodnota pod odmocninou $\sqrt{(\vec{l} \cdot (o - c))^2 - (\vec{l} \cdot \vec{l}) * ((o - c) \cdot (o - c) - r^2)}$ menší než nula, průsečík neexistuje. Jestliže je rovna nule, pak existuje právě jedno řešení a jeden průsečík a je-li větší než nula, pak má přímka a koule dva průsečíky.

Intersekce roviny a paprsku[13]

Při kreslení obecného polygonu vytváří aplikace body na jedné z rovin tvořené osami kartézských souřadnic. Proto je potřeba hledat průsečík paprsku a roviny pro vytvoření nového bodu.

Vyjádřeme si rovinu jako množinu bodů, pro které platí následující:

$$(x - p_0) \cdot \vec{n} = 0,$$

kde $\vec{n}$ je normálový vektor roviny a $x, p_0 \in R^3$ jsou body na rovině. Rovnice přímky je dána jako:

$$x = d * \vec{l} + o,$$

kde $d \in R$, $\vec{l}$ je směrový vektor přímky a o je bod ležící ná přímce. Substitucí do rovnice roviny získáme:

$$(d * \vec{l} + o - p_0) \cdot \vec{n} =$$

$$d * \vec{l} \cdot \vec{n} + (l_0 - p_0) \cdot \vec{n} = 0.$$

Vyřešením rovnice pro d získáme:

$$d = \frac{(p_0 - o) \cdot \vec{n}}{\vec{l} \cdot \vec{n}}.$$

Je-li $\vec{l} \cdot \vec{n} = 0$, pak přímka a rovina jsou rovnoběžné. Nastanou dva případy: pokud $(p_0 - o) \cdot \vec{n} = 0$, pak přímka leží v rovině, tedy každý bod přímky leží v rovině; jinak přímka neprotíná rovinu.

Platí-li $\vec{l} \cdot \vec{n} \neq 0$, pak je pouze jeden průsečík roviny a přímky. Hodnota d, a poté i průsečík, může být dopočítána zpětným dosazením.

4.1.4 Eulerovy operátory

Eulerovy operátory slouží na tvorbu manifoldních objektů. Zajišťují topologickou korektnost a platnost Eulerových rovností. Model vytvořený pouze pomocí těchto operací bude vždy uzavřené těleso. Pokud je model načten ze souboru a není manifoldní, pak se ani pomocí Eulerových operátorů manifoldním nestane. Využívám Eulerovy operátory na vytváření geometrie skládající se z okřídlených půlhran.

Eulerovy rovnosti

Eulerovy rovnosti pro obecné těleso s dírami:

$$F - E + V = 2(S - G) + R,$$

kde F je počet ploch tělesa, E je počet hran tělesa (půlhrana se svou párovou hranou se počítají jako jedna), V je počet vrcholů, S je počet těles, G je počet děr procházející skrz celé těleso, tzv. genus, a R je počet děr ve stěnách tělesa.

MSFV

MSFV, neboli Make Solid Face Vertex, je operace na založení tělesa v prostoru a vytvoření počáteční plochy. Na počáteční ploše se vytvoří jeden vrchol. V tomto stavu se model stále nedá vykreslit.

MEV

MEV, neboli Make Edge Vertex, vytvoří nový vrchol a hranu z počátečního vrcholu do nově vytvořeného. Pro každou okřídlenou půlhranu se přiřadí její půlhrana následující a její párová půlhrana.

MEF

MEF, neboli Make Edge Face, spojí poslední vrchol vytvářené plochy s prvním vrcholem a uzavře tak celou plochu. Nově vytvořená plocha se přídá do geometrie tělesa. Po této operaci je možné objekt vytvořit.

4.2 Použité datové struktury

4.2.1 Okřídlená půlhrana

Jednoduchým způsobem pro ukládání vrcholů, hran a ploch 3D geometrie je sdílený seznam vrcholů, hran a ploch, tzv. polygonální polévka. Bohužel tato struktura neobsahuje žádné dodatečné informace. Kdybychom potřebovali smazat jednu hranu a tak propojit dvě plochy, bylo by zapotřebí procházet celý seznam ploch v lineráním čase a kontrolovat, zda-li plocha obsahuje hledanou hranu. Se strukturou okřídlená hrana je však tento čas konstantní.

Okřídlená půlhrana obsahuje referenci na vrchol, ve kterém končí, na následující půlhranu, na plochu, které náleží, a párovou půlhranu.

Půlhrana je to proto, že reprezentuje jeden směr hrany. Tedy máme-li půlhranu e_1 z vrcholu v_1 do vrcholu v_2, jej párová půlhrana je hrana e_2 z vrcholu v_2 do vrcholu v_1 a její plocha sousedí s plochou hrany e_1.

Ve své implementaci využívám zpětné reference - patří-li vrchol hraně, pak ve vrcholu je reference na jednu z hran, které v ní začínají.

Nevýhodou okřídlené půlhrany je možnost ukládání pouze manifoldních objektů. Veškeré díry v objektu tak musí být uloženy jako existující plocha, která se však nevykresluje.

4.3 Implementace

4.3.1 Vrchol, půlhrana, plocha, geometrie

Podle funkčního požadavku č. 7 musí být umožněno obarvovat jednotlivé plochy tělesa a pro ostré hrany je potřeba, aby každý vrchol měl vlastní normálu. Tedy vrchol tělesa je pouze jeden, ale každá hrana/plocha od něj potřebuje jiné vlastnosti. Proto jsou v mém datovém modelu vrcholy vždy jiné instance, pouze sdílejí stejný identifikátor, pokud mají reprezentovat stejný vrchol tělesa. V přílohách je obsažen diagram tříd popisující vztahy mezi abstraktními třídami geometrie.

Abstraktní třída Vertex poskytuje rozhraní k práci s vrcholy. Definuje gettery a settery pro získání souřadnic ve 3D prostoru, normály, barvy, texturovacích souřadnic a hrany, která ve vrcholu začíná. Třída Vertex implementuje hledání průsečíku paprsku a koule.

Abstraktní třída Edge poskytuje rozhraní pro práci s hranami. Vlastní používaná třída EdgeHE implementuje datavou strukturu okřídlená půlhrana. Všechny třídy dědící z Edge musí implementovat funkce na získání koncového a počátečního vrcholu, plochy, s kterou hrana sousedí, výpočet délky hrany a funkci na rozdělení hrany na dvě.

Abstraktní třída Face representuje obecné vlastnosti ploch náležící 3D tělesu. Implementace rozšiřující Face musí poskytovat funkce na získání všech vrcholů obsažených na hranách plochy, získání všech hran plochy a odkaz na geometrii, které plocha náleží. Třída Face implementuje hledání průsečíků plochy nebo trojúhelníku s paprskem. Také umožňuje přiřazení barvy ploše - propaguje se do vrcholů. Jelikož se vykreslují plochy jako takové, musí oděděné třídy implementovat funkci na získání všech vrcholů na vykreslování. Pokud tedy plocha není trojúhelníková, musí si držet informaci o všech trojúhelnících, z nichž se skládá.

Abstraktní třída MeshGeometry definuje základní vlastnosti pro obecné 3D těleso. Těmi je seznam vrcholů, ploch a hran, ze kterých se těleso skládá. Slouží především jako kontejner pro výše uvedené plochy, hrany a vrcholy.

4.3.2 Výběr, transformace

Pokud vrchol, plocha, hrana nebo geometrie lze uvnitř scény vybrat, musí implementovat rozhraní Selectable. To nabízí funkce pro selekci jak při práci jednoho uživatele, tak při práci více uživatelů. Jestliže dovolují jakoukoli změnu polohy nebo barvy, musí implementovat rozhraní Transformable. To umožňuje změnu polohy, rotaci tělesa kolem svého středu nebo škálovaní od svého středu. Také nabízí funkci na změnu barvy a vrácení barvy, pokud předem byla změněna. Tato barva je odlišná od materiálu a dá se funkcí Remove color odstranit. Definice rozhranní Selectable a Transformable je v přílohách.

4.3.3 Graf scény

Jelikož podle funkčního požadavku č.3 má aplikace exportovat scénu do X3D, rozhodl jsem se, že strukturu objektů ve scéně přizpůsobím formátu X3D[14]. Taktéž jsem z něj převzal způsob vykreslování, počet aktivních světel ve scéně a osvětlovací model.

Samotný graf scény má stromovou strukturu. Obsahuje kořen třídy SceneNode, který dále ukládá svoje potomky. Mimo tuto stromovou strukturu fungují kamery. Graf scény si udržuje seznam všech definovaných kamer a referenci na aktivní (momentálně) používanou.

Mimo tyto základní objekty pro pohyb a vizualizaci scény obsahuje třídu Sandbox, která slouží k vykreslování uživatelského vstupu při kreslení geometrie. Třídy ViewCube a Grid jsou samostaně stojící objekty podobné struktuře třídy Mesh, ale zjednodušené, protože neumožňují transformace ani výběr, a vlastní jednoduchý shader. Jsou přidány pro zpřehlednění navigace ve scéně, kdy Grid tvoří čtvercovou síť se stranou délky 20 jednotek a rozdělením do čtverců se stranou délky 1. ViewCube je uchycena do pravého horního rohu a rotuje společně s kamerou, aby vždy ukazovala skutečnou orientaci os kartézské soustavy. Diagram tříd popisující vztahy v grafu scény je přiložen v přílohách.

4.3.4 Objekty ve scéně

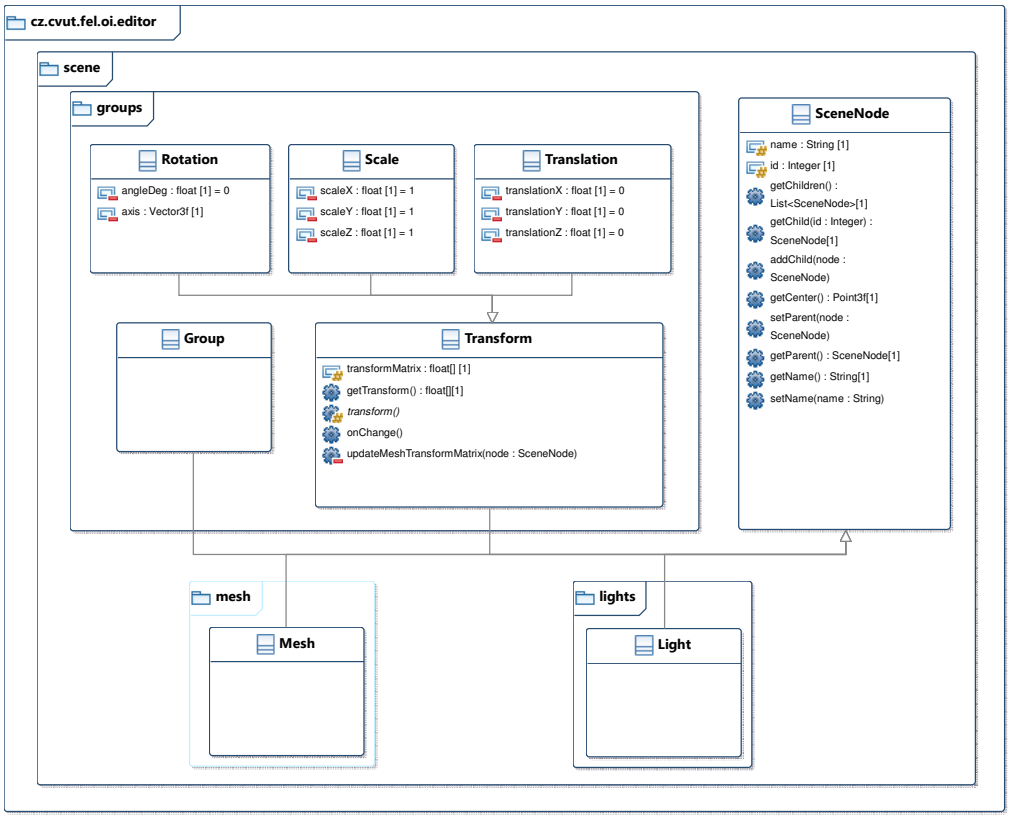
SceneNode

Třída SceneNode je základním objektem scény. Každý objekt ve scéně ji rozšiřuje. Každý objekt ve scéně má vlastní střed. Pokud je objektem například transformace, středem je průměr ze středů jejich potomků.

Group a Transform

Třída Group sjednocuje více objektů pod sebe, ale nepřidává jim žádnou novou informaci. Slouží k zpřehlednění scény. Narozdíl od Group, třída Transform objekty, které jsou uložené jako její potomci, přemisťuje, zvětšuje či zmenšuje a přesouvá podle konkrétně zvolené třídy: Rotation, Scale nebo Translation.

Moje implementace transformací se liší od transformací definovaných formátem X3D. Zatímco moje třída využívá postupné zapouzdřování transformacemi, podle normy X3D má uzel Transform tři atributy a těmi jsou práve translace, rotace a zvětšení.



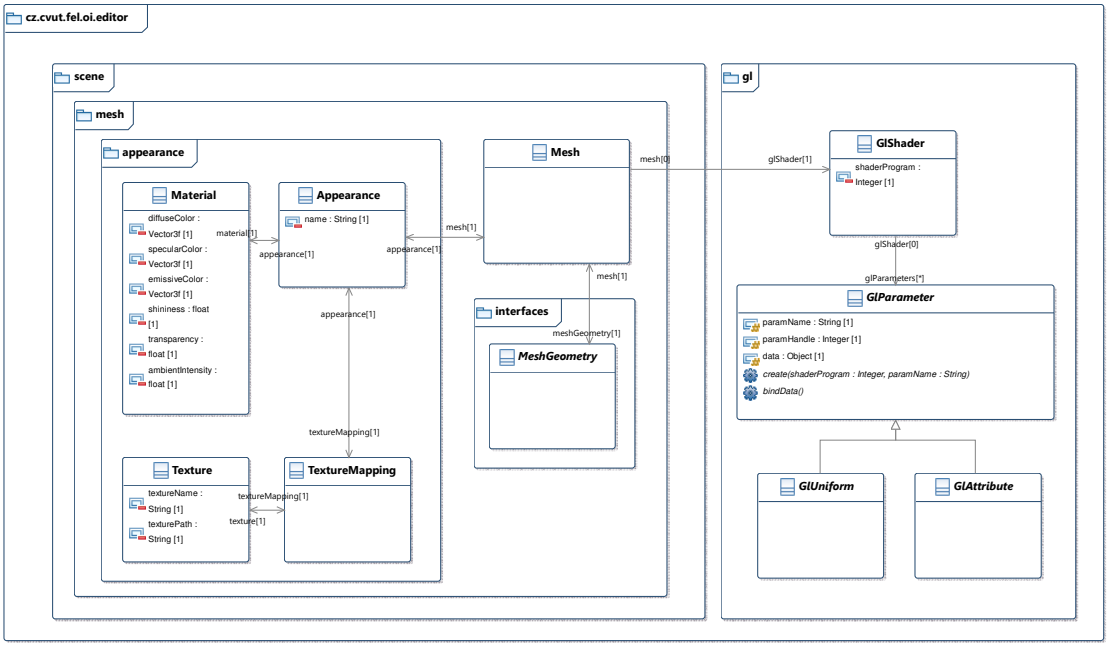
Obrázek 16 Diagram tříd objektů ve scéně.

Světla, kamera, akce

Světla se stejně jako v X3D přidávají přímo do scény a mají možnost globálního osvětlení scény nebo pouze v rámci svého rodičovského uzlu a jeho potomků. Kamera stojí mimo stromovou strukturu třídy SceneNode. Kamery jsou uložené ve vlastním seznamu v grafu scény. Jakou hierarchii světla mají je popsáno v diagramu tříd v přílohách.

Mesh

Třída Mesh tvoří jakousi superstrukturu pro vykreslování. Udržuje si aktuální transformační matici, vlastní shader, geometrii, třídu Points pro vykreslování extra vrcholů a Lines pro zvýraznění hran tělesa.



Obrázek 17 Diagram tříd pro Mesh.

Třída Mesh je podobná uzlu Shape v X3D. Potřebuje svůj definovaný vzhled ve třídě Appearance a geometrii třídy MeshGeometry. Appearance se automaticky přiřadí při vytvoření objektu a vytvoří si Material s defaultními hodnotami - průhlednost je nastavena na 0 - neprůhlednou, spekulární a emisivní barva jsou černé, difúzní barva má hodnoty RGB šedé $(0.8, 0.8, 0.8)$ a lesklost materiálu má hodnotu 0.2. Barva objektu, plochy nebo vrcholu lze nastavit pomocí funkce Color. Tato funkce využívá volně dostupnou knihovnu AmbilWarna[15].

Pro Appearance se také může přiřadit textura ve třídě Texture a mapování texturovacích souřadnic ve třídě TextureMapping.

Třída GlShader poskytuje rozhraní přístupu k GLSL fragment a vertex shaderu. Lze si u ní registrovat hodnotu, která je definovaná v GLSL shaderu a k němu přes GlParameter posílat data. GlParameter definuje způsob přístupu k shaderu. Od něj dědí třídy GlUniform, reprezentuje GLSL uniformy, a GlAttribute, reprezentuje GLSL atributy. Každý nový uniform nebo atribut potřebuje vlastní třídu, např. matice $\in R^{4,4}$ má GlMatrix4fUniform, která posílá data do shaderu.

GLSL shadery a osvětlovací model

Objekty ve scéně k vykreslování využívají Phongův osvětlovací model[16] podle normy X3D.

Vertex shader především přeposílá interpolované hodnoty atributů do fragment shaderu. Zároveň vypočítává natočení normál vrcholů vynásobením normálovou maticí a ukládá pozici vrcholů.

Fragment shader dostává na vstup dva číselné uniformy - jestli se bude využívat světlo a zda-li se využívá textura. Pokud se nevyužívá ani jedno, použije se barva vrcholů resp. materiálu. Pokud je použito pouze osvětlení, k barvě se dopočítá vliv světla na model.

Používá-li se textura, pak se používá i osvětlení. Její barva na místě texturovacích souřadnic se použije jako základní barva a vypočítá se k ní osvětlení.

Osvětlovací model potřebuje na vstupu materiál, světlo a základní barvu. Dovoluji až 8 aktivních světel ovlivňujících jeden model.

Rovnice osvětlení:

$$C_{out} = material.emissiveColor + \sum(\text{útlum}_i * spot_i * C_L * (ambient_i + diffuse_i + specular_i)),$$

kde $C_{out}, ambient_i, diffuse_i, specular_i, C_L, material.emissiveColor$ jsou barvy RGB spektra s hodnotami od 0 (barva není obsažena) po 1 (barva je plně obsažena).

C_{out} Výstupní osvětlená barva.

C_L Barva světla.

útlum_i $1.0/max(\text{světlo}_i.\text{útlum}.x + \text{světlo}_i.\text{útlum}.y * vzdálenost + \text{světlo}_i.\text{útlum}.z * vzdálenost * vzdálenost, 1.0)$. Útlum světla je zadán třemi hodnotami, které vyjadřují, jak rychle světlo klesá se vzdáleností od bodu.

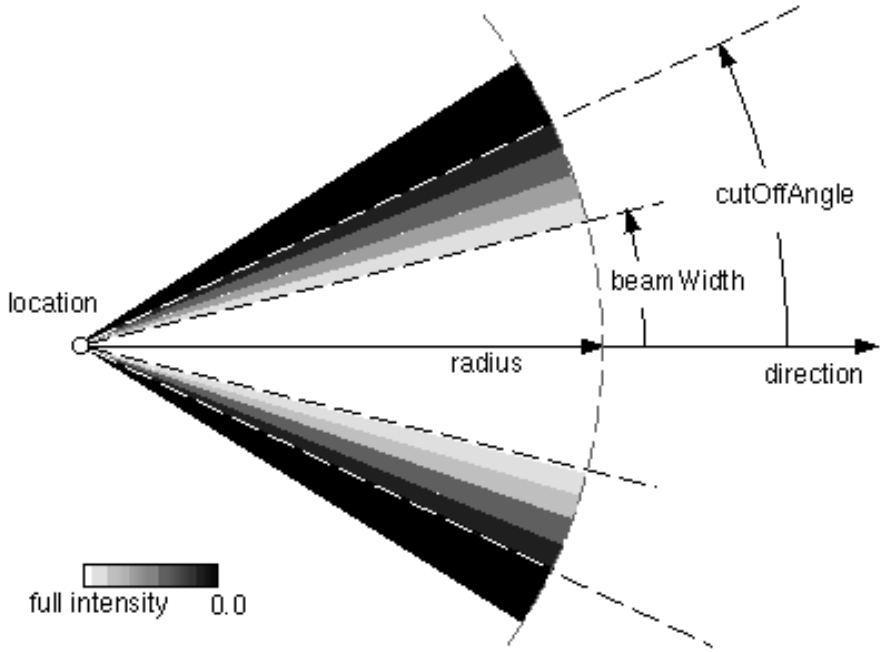
$spot_i$ Faktor reflektoru. Pro směrové a bodové světlo je roven 1. Pokud je $spotAngle >= cutOffAngle$, pak je 0. Pokud $spotAngle <= beamWidth$, pak je 1. Když je v intervalu $(beamWidth, cutOffAngle)$, pak se vypočítá $(spotAngle - cutOffAngle)/(beamWidth - cutOffAngle)$

$spotAngle$ $arcus\ cosinus(-L \cdot směr\ reflektoru)$

$\cdot$ Zde značí upravený skalární součin vektorů, který vrací skalární součin, pokud je větší než 0, jinak vrací 0.

$ambient_i$ $světlo_i.ambientIntensity * barva\ bodu * material.ambientIntensity$

$diffuse_i$ $světlo_i.intensity * barva\ bodu * (N \cdot L)$
$specular_i$ $světlo_i.intensity * material.specularColor * (N \cdot (L+v)/|(L+v)|)^{material.shininess * 128}$
N Jednotkový normálový vektor osvětlovaného bodu.
L Jednotkový vektor z osvětlovaného bodu ke středu světla. Pro směrové světlo je to $-směr\ světla$.
v Jednotkový vektor z osvětlovaného bodu k pozorovateli.



Obrázek 18 Popis reflektoru.

4.3.5 Instrukce

Ovládání scény v aplikaci se provádí v rámci volání vykreslovací funkce OpenGL ES onDrawFrame. Je to potřeba z toho důvodu, že mimo tuto funkci neexistuje platný OpenGL kontext a není tedy možné posílat data do shaderů. Většina instrukcí se tedy vytváří až když jsou dostupná potřebná data. Například při importování souboru, se nejdříve vytvoří celá geometrie s materiálem. Až poté se v instrukci přidají do třídy Mesh a do grafu scény.

Všechny instrukce dědí od abstraktní třídy Instruction a musejí implementovat funkci doAction, ve které je vlastní chování instrukce.

Instrukce se přidávají do seznamu ve třídě InstructionParser, odkud se berou při každém požadavku na vykreslení. Odebírání je omezené časem 15 milisekund, aby se zbytečně nezabíral čas pro vykreslování.

4.3.6 Controller

Na třídu Controller je využit návrhový vzor singleton. Je potřeba, aby byl pouze jeden a dalo se k němu přistupovat odkudkoli.

Controller zprostředkovává komunikaci mezi uživatelským rozhraním a sítí, mezi vykreslováním a datovou strukturou a předává instrukce. Jsou v něm uloženy veškeré reference - na graf scény, na třídy síťové komunikace, na spínání kamery a na zpracování instrukcí.

4.3.7 Síťová komunikace

Uživatel, který chce sdílet svoji scénu v menu zvolí položku Online a Start cooperation. Tím spustí novou aktivitu, která mu umožňuje spustit na svém zařízení server. Ostatní se k němu mohou připojit přes menu položku Online a funkci Join cooperation. Ti, co se chtějí připojit, musí vyplnit adresu a port serveru, který je vypsaný na zařízení, kde je spuštěný server.

Síťová komunikace je založena na posílání zpráv. Jedno z komunikujících zařízení se chová jako server. Server ukládá historii instrukcí a drží si aktuální identifikátory pro objekty ve scéně. Kdykoli je potřeba vytvořit nový objekt, klient naváže spojení se serverem a vyžádá si určitý počet identifikačních čísel. Z nich pak může vytvářet nové objekty. Spojení je navázáno prostřednictvím java tříd Socket a ServerSocket implementujících komunikaci využívající TCP protokol.

Přes síť se posílají instrukce, které si nesou nejdůležitější informace, a po přijetí serverem se přepošlou všem klientům, kromě toho od nějž přišly. Instrukce se pak vykoná na klientovi. Z tohoto důvodu všechny instrukce třídy Instruction a třída Mesh implementují rozhraní Externalizable. V metodách readExternal a writeExternal pak čtou a posílají nejdůležitější informace pro instrukci. Takto serializovaná instrukce se přidá do vlastní fronty ve třídě InstructionParser. Vykonávají se také během vykreslování scény.

4.3.8 Ovládání

Ovládání je závislé na jednotlivých funkcích. Přesto jsem se snažil, aby měly ovládání alespoň podobné.

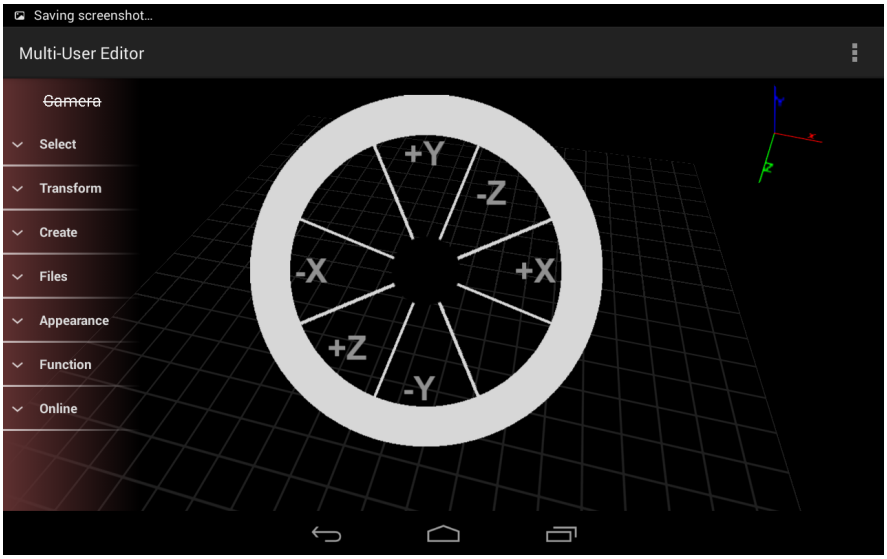
Perspektivní kamera umožňuje rotaci kolem os X a Y. Rotace závisí na směru pohybu prstu od bodu dotyku. Rotace kolem X je závislá na vertikální změně polohy a rotace kolem Y na horizontální. Rotace kamery se chová, jako by celá scéna byla uzavřena v kouli a pohybem prstu se koulela.

Translace se provádí pomocí dvou prstů a pohybuje kamerou v rovině XY. Tato rovina je brána vzhledem ke kameře. Pohyb v ose Z se provádí závisí na vzdálenosti obou prstů. Zoom interpoluje polohu kamery mezi původní polohou a centrem, kam je kamera zaměřená.

Položky v menu stačí vybrat dotykem a většina z nich je jednorázová a nevyžaduje další uživatelský vstup. Mezi ty, co potřebují další vstup patří selekce a transformace. Každá z nich má vlastní handler, který obdobně jako u kamery snímá pohyb prstu a dotyku.

Selekce požaduje pouze body dotyku, aby se tento bod mohl promítnout ze souřadnic obrazovky do souřadnic scény. Z tohoto bodu je vystřelen paprsek a hledá průsečíky ve scéně. Vybraný objekt se zvýrazní zelenou barvou (červenou je-li vybraný kooperujícím uživatelem).

Transformace vybraných objektů může být rotace, translace nebo škálování. Transformace jsou rozděleny na tři, aby nedocházelo k nechtěným transformacím. Každá z nich má podobné ovládání, pouze změněné osy, aby více odpovídaly skutečnosti.



Obrázek 19 Nápověda pro translaci.

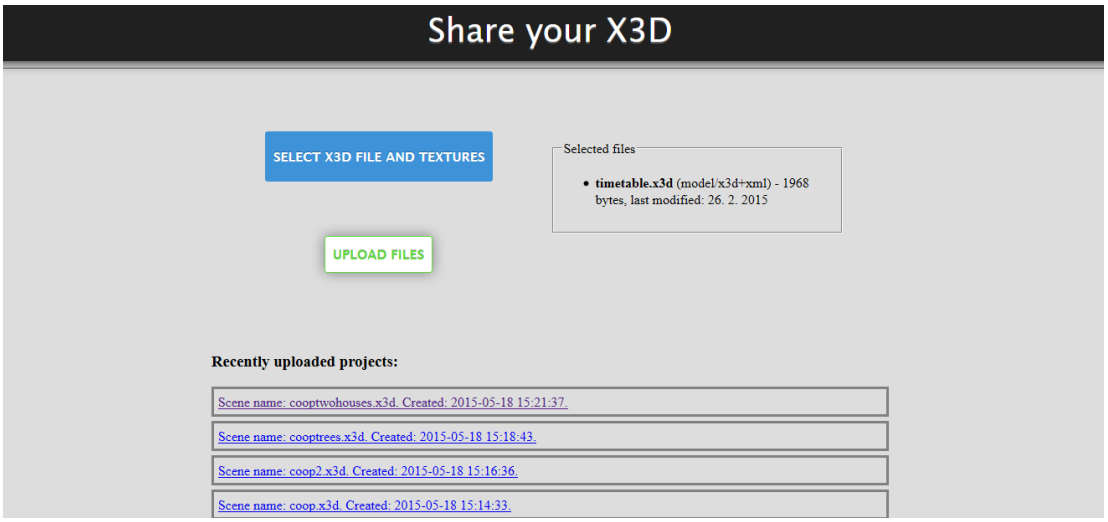
Je-li zapnuta jedna z transformací, pak se při dotyku obrazovky zobrazí nápověda, jak lze vidět v obrázku výše. V každém směru se bude funkce chovat jinak. Pokud uživatel vyjede prstem na +X nebo -X, pak se uzamkne osa a veškerý pohyb se přenáší na osu X ve světových souřadnicích. V případě, že se uživatel vrátí zpět do rozmezí 30 pixelů od bodu dotyku, může transformovat objekt podél jiné osy. Uzamknutí směru zabraňuje milnému pohybu v jiné ose.

4.3.9 Webová stránka pro prohlížení X3D

Internetová stránka pro sdílení X3D scén je dostupná na adrese `http://course-wa1.felk.cvut.cz/~janovrom/modeler/bp/www/index.php`, tedy na školním serveru ČVUT. Hlavní stránka umožňuje načíst jeden soubor X3D a k němu požadované textury.

Kontrola správnosti formátu obrázků a souboru s modelem se provadí na klientské straně přes parametr required u html *input* tagu a pomocí JavaScriptu. Na serverové straně ještě proběhne kontrola v PHP.

Pod formulářem k nahrávání souborů je seznam dvaceti naposledy sdílených scén. Z nich lze přejít na stránku se samotnou X3D scénou. Adresu scény je možné získat přímo po nahrání souborů, pokud se nahrály úspěšně.



Obrázek 20 Titulní strana webových stránek pro sdílení X3D scén.

5 Testování s uživateli

5.1 Participanti a jejich zadání

Pro vyzkoušení své aplikace jsem sehnal tři participanty. Dva z nich byli z technické školy a jeden z nich je studentem archeologie. Pouze jeden z nich někdy výrazněji pracoval s 3D editorem (konkrétně Autodesk Maya).

Aby si zvykli na ovládání a chování aplikace, nechal jsem každého z nich vypracovat jeden úkol ze tří připravených.

Prvním z úkolů bylo vytvořit jednoduché okno. Okenní tabulka měla být rozdělena křížem. Sklo mělo být oboustranně poloprůhledné. Scéna se měla vyexportovat do X3D.

Druhý úkol bylo vytvoření informační cedule na autobusové zastávce. Vytvořena měla být podle vzoru s tím, že všechny potřebné textury bylo možné najít ve složce programu. Scéna se poté měla nasdílet na webovou stránku (`http://course-wa1.felk.cvut.cz/~janovrom/modeler/bp/www/scene.php?name=18`).

a) Zadání **b)** Výsledek

Obrázek 21 Zadání a výsledek druhého úkolu.

Třetím úkolem bylo vytvořit terén velikosti 10 na 10 jednotek. Jednotky nebyly blíže specifikovány a uživatel se měl sám rozhoudnout, jak je použije.

Potom, co všichni splnili svůj úkol, museli společně při kooperaci po síti vytvořit jednoduchou scénu: dům se stromy. Výsledek jejich práce je možné vidět na `http://course-wa1.felk.cvut.cz/~janovrom/modeler/bp/www/scene.php?name=25`.

Obrázek 22 Scéna vymodelovaná při kooperaci

5.2 Vyhodnocení testování

Jediný úkol, který se zdál být neproveditelný, bylo modelování terénu. Jelikož aplikace nepodporuje dělení plochy na menší části, musel se terén modelovat z jednotlivých čtverců a to nebylo příliš pohodlné. Zde by se hodila funkce, která by spojovala vrcholy a hrany nebo rozdělovala polygon na menší části.

Poté, co byly vytvořeny scény, byli participanti požádáni, ať vyplní krátký dotazník. V něm jsem se ptal na to, jak hodnotí navigaci v aplikaci, jestli jim vyhovovalo ovládání kamery, zda-li by chtěli nějakou specifickou funkci a jak informováni byli o používané funkci. Dotazník je v přílohách.

Participanti nezávisle na sobě dospěli k závěru, že navigace v aplikaci není složitá a neměli problém najít v menu funkci, kterou potřebovali. Chyběl jim však seznam objektů ve scéně a na první pohled nevěděli, že se pohyb kamery dá vypnout a zapnout tlačítkem v menu.

U informování o používané funkci by uvítali, aby jim bylo sděleno, na kterou část geometrie - vrchol, plochu nebo objekt - lze funkci použít. Také by uvítali, pokud by se někde jméno aktivní funkce trvale zobrazovalo a neinformovalo se pouze o jejím použití či zapnutí.

Žádný z nich neměl problém s používáním kamery a rychle si na ni zvykli. Uvitali by však, kdyby se kamera zaměřila na vybranou geometrii a při použití funkce Unselect all se automaticky kamera spustila. Taky by používání kamery mělo nechat otevřenou stávající položku menu.

Jako další funkce by se podle nich hodily funkce na slučování a rozpojování vrcholů, dělení hran a vytváření nových hran v ploše spojením dvou vrcholů na hranách plochy.

Aplikace by také měla umožňovat zadání posunutí, rotace nebo zvětšení jako číselnou hodnotu a zobrazovat na vyžádání o kolik byl objekt zvětšen, posunut nebo otočen. Hodilo by se přepínání otočení gridu v aplikaci: aby byl nejen v rovině XZ, ale i v těch zbylých. Jeden participant chtěl možnost scénu uložit jako bitmapový obrázek a založit nebo načíst scénu. Všichni se shodli, že je potřeba tlačítko zpět.

Z jejich práce jsem dále usoudil, že by bylo vhodné, udělat transformace geometrie také v lokálních souřadnicích, umožnit měření geometrie a přidat upravování texturovacích souřadnic.

5.3 Použitá zařízení k testování

5.3.1 Tablet Lenovo Yoga 8

Rozlišení displeje 1280 x 800
Úhlopříčka 8"
Operační systém Google Android 4.4.2
Procesor
  Frekvence procesoru 1 200 [MHz]
  Model procesoru MediaTek MT8125
Paměť
  Velikost operační paměti 1 [GB]
  Typ paměti DDR2

5.3.2 Tablet Samsung Galaxy Tab 10.1 3G

Rozlišení displeje 1280 x 800
Úhlopříčka 10.1"
Operační systém Google Android 4.0.4
Procesor
  Frekvence procesoru 1 [GHz]
  Model procesoru Cortex-A9
  Chipset Nvidia Tegra 2 T20
Paměť
  Velikost operační paměti 1 [GB]
  Typ paměti DDR2

Samsung Galaxy Tab 10.1 měl problémy s GLSL shadery. Zařízení nedokázalo najít GLSL uniformy v shaderu pomocí funkce glGetUniform. I přes pevné nastavení manipulátorů na číselnou konstantu s hodnotou, kterou měl uniform na jiných zařízeních, se nedaly uniformy poslat do shaderu.

5.3.3 Smartphone Samsung I9505 Galaxy S4

Rozlišení displeje 1920 x 1080
Úhlopříčka 5"
Operační systém Google Android 5.0.1

Procesor
Frekvence procesoru 1.9 GHz [GHz]
Model procesoru Krait 300
Chipset Qualcomm APQ8064T Snapdragon 600
Paměť
Velikost operační paměti 2 [GB]

5.3.4 Asus MeMO Pad ME302C

Rozlišení displeje 1920 x 1200
Úhlopříčka 10.1"
Operační systém Google Android 4.3
Procesor
Frekvence procesoru 1.6 GHz [GHz]
Model procesoru Intel Atom Z2560
Paměť
Velikost operační paměti 2 [GB]

6 Závěr

Cílem mé práce bylo navrhnout a implementovat aplikaci, která by umožňovala kooperativní tvorbu jedné 3D scény. Cíle, které jsem pro ni vytyčil v rámci případů užití, založených na zadání, jsem všechny implementoval. Aplikace byla otestována a uživatelé dospěli k závěru, že se s aplikací dají vytvářet jednoduché objekty. Avšak vzhledem k mému pozorování a poznámkám participantů si myslím, že aplikace by potřebovala další rozšíření.

Tyto úpravy by především měly být rozšířením stávajících funkcí, které by měly umožňovat větší kontrolu nad úpravami geometrie.

Mezi tato rozšíření patří editace texturovacích souřadnic, číselné zadání transformací, umožnění transformací v lokálních souřadnicích objektu, informace o velikosti a vzdálenostech v geometrii. Dále by se aplikace měla rozšířit o pohyb v historii použitých funkcí - tedy funkci zpět. Vzhledem ke struktuře podobné X3D by šlo vytvořit editaci pomocí X3D uzlů.

Při spolupráci si při tvorbě nejprve spíše překáželi než pomáhali. Poté co se však domluvili, kdo bude pracovat na které části scény, neměli žádné další potíže. Důležitým faktorem také bylo, že všichni seděli poblíž sebe, a tedy v případě problémů mohli ostatní upozornit a požádat, ať se zaměří jinam. Tedy myšlenka kooperativní tvorby scény není chybná, ale bylo by potřeba, aby společně tvůrci komunikovali nebo pracovali podle předem stanovených scénářů.

Zjistil jsem také, že pokud uživatelé přemisťovali větší počet objektů zároveň, pak docházelo k drobným zpožděním odezvy síťové komunikace. Bylo by tedy dobré, kdyby se stávající posouvání řešilo jinak. Stávající aplikace posouvá objekt po desetinách jednotky délky a každé posunutí si vyžaduje vlastní instrukci, která se musí vykonat a poslat po síti. Tedy místo posílaní každého posunutí, by se čekalo, až uživatel zvedne prst a přestane geometrii přesouvat. Až po této události by se přeposlala transformace. V síťové komunikaci by se také hodila synchronizace scény na vyžádání. Tak by se mohli uživatelé připojit kdykoli během tvorby.

Co se smartphonů týče, aplikace na nich lze bez problémů používat, přesto malý displej znepřehledňuje a snižuje přesnot při práci. Vhodné by pro ně aplikace byla, pokud by uživatel používal stylus.

Literatura

[1] Scalisoft. *Spacedraw*. 20. březen 2015. URL: https://play.google.com/store/apps/details?id=com.scalisoft.spacedraw&hl=en (cit. 05. 05. 2015).

[2] ASCON. *SubDivFormer*. 21. duben 2015. URL: https://play.google.com/store/apps/details?id=com.ascon.subdivformer (cit. 05. 05. 2015).

[3] Jonathon Quinn. *Qubism*. 28. březen 2015. URL: https://play.google.com/store/apps/details?id=jquinn.qubism.android (cit. 05. 05. 2015).

[4] Neil Mawston. *Android Captures 79 % Global Smartphone OS Share in Q1 2015*. 30. duben 2015. URL: http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=10866 (cit. 05. 05. 2015).

[5] Android. *Dashboards*. 4. květen 2015. URL: https://developer.android.com/about/dashboards/index.html (cit. 05. 05. 2015).

[6] Ian Garton. *Ear Cutting for Simple Polygons*. 10. prosinec 1997. URL: http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/97/Ian/cutting_ears.html (cit. 06. 05. 2015).

[7] Wikipedia. *Determinant*. 4. květen 2015. URL: http://en.wikipedia.org/wiki/Determinant (cit. 06. 05. 2015).

[8] G.H. Meisters. "Polygons have ears". In: *American Mathematical Monthly* (1975), s. 648–651.

[9] John W. Ratcliff. *Efficient Polygon Triangulation*. 22. červenec 2000. URL: http://www.flipcode.com/archives/Efficient_Polygon_Triangulation.shtml (cit. 06. 05. 2015).

[10] Dan Sunday. *Area of Triangles and Polygons*. URL: http://geomalgorithms.com/a01-_area.html#3D%20Polygons (cit. 06. 05. 2015).

[11] Lighthouse3d. *Ray-Triangle Intersection*. 13. únor 2015. URL: http://www.lightouse3d/tutorials/maths/ray-triangle-intersection/ (cit. 08. 05. 2015).

[12] Wikipedia. *Line-sphere intersection*. 3. únor 2015. URL: http://en.wikipedia.org/wiki/Line-sphere_intersection (cit. 08. 05. 2015).

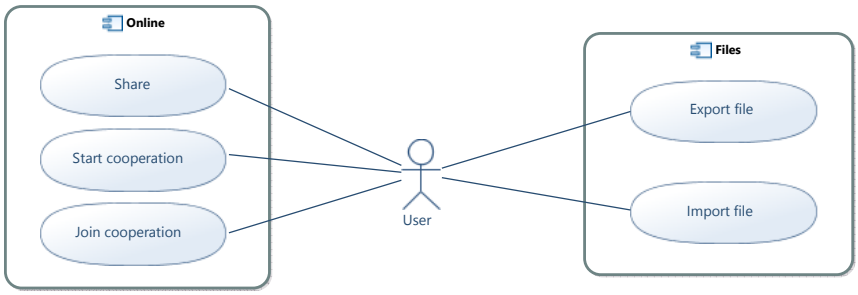
[13] Wikipedia. *Line-plane intersection*. URL: http://en.wikipedia.org/wiki/Line-plane_intersection (cit. 08. 05. 2015).

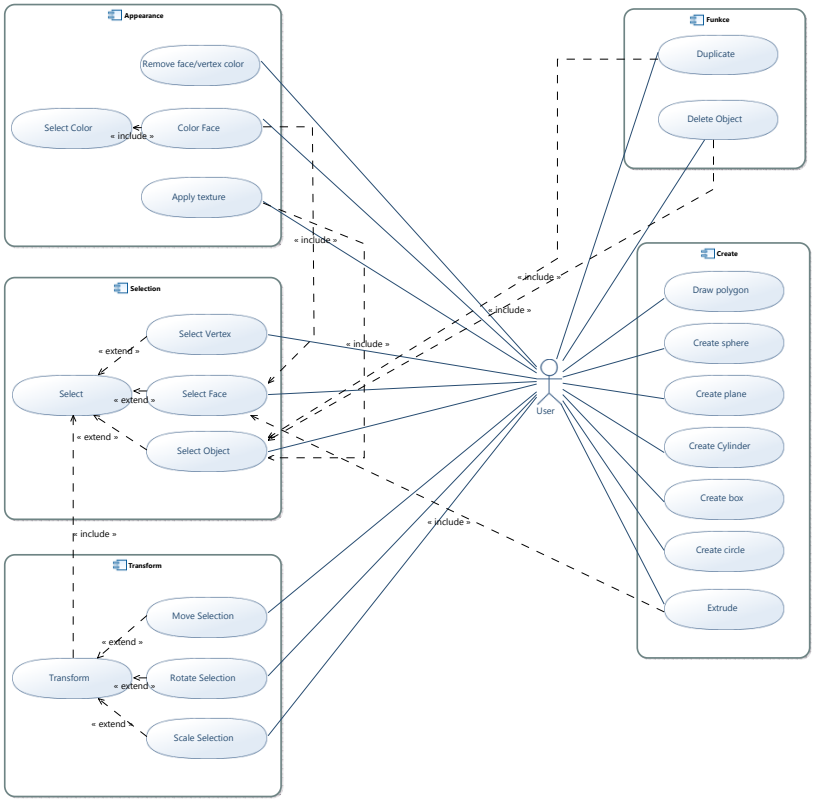
[14] Inc. Web3D Consortium. *Extensible 3D (X3D) ISO/IEX 19775-1:2008*. URL: http://www.web3d.org/documents/specifications/19775-1/V3.2/ (cit. 10. 05. 2015).

[15] yukuku. *Android Color Picker*. 6. květen 2015. URL: https://github.com/yukuku/ambilwarna (cit. 20. 05. 2015).

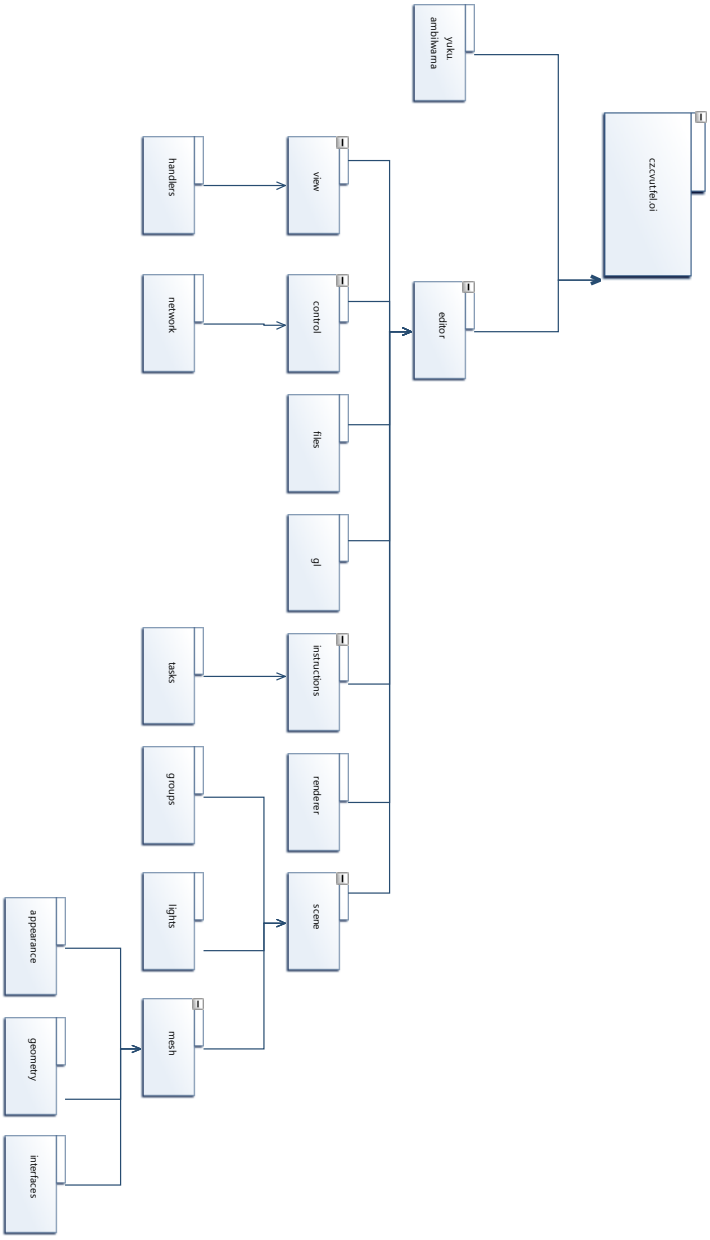
[16] Inc. Web3D Consortium. *Lighting component*. URL: http://www.web3d.org/documents/specifications/19775-1/V3.2/Part01/components/lighting.html (cit. 10. 05. 2015).

Příloha A - diagramy případů užití



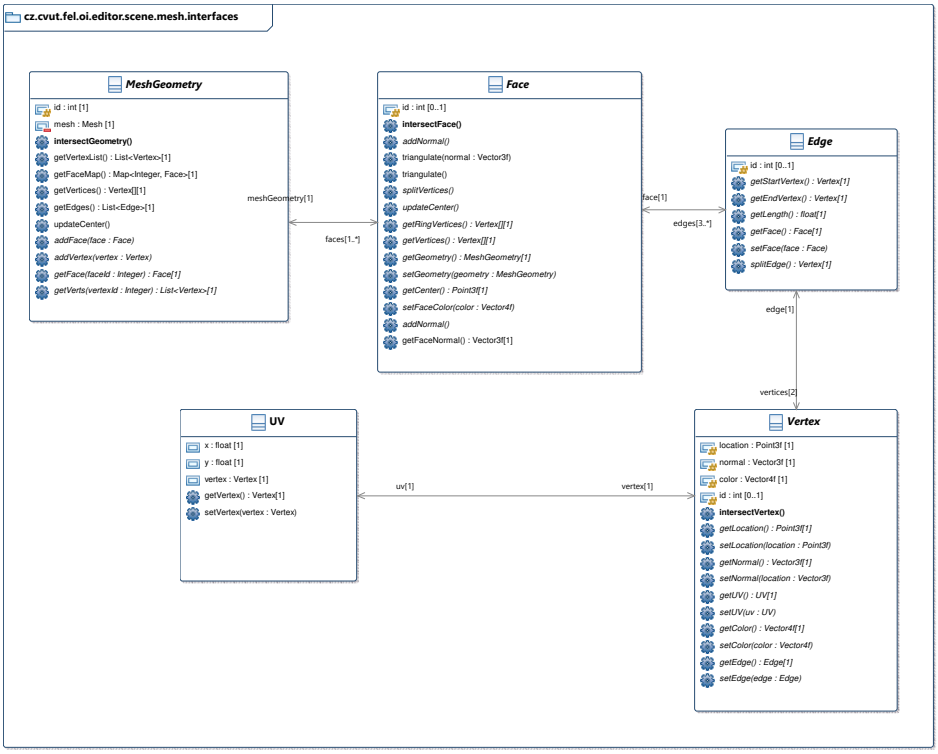


Příloha B - diagram balíčků

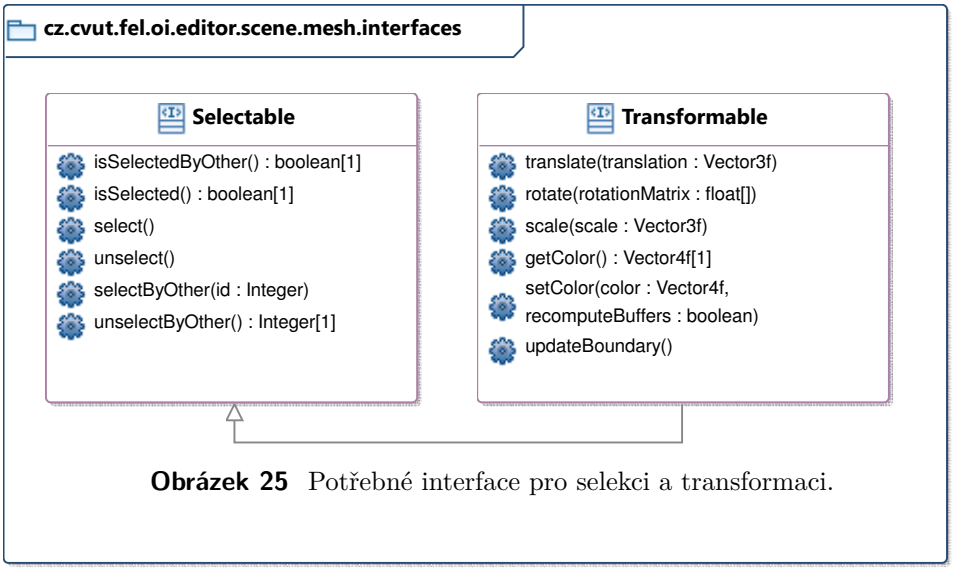


Obrázek 23 Diagram balíčků v aplikaci.

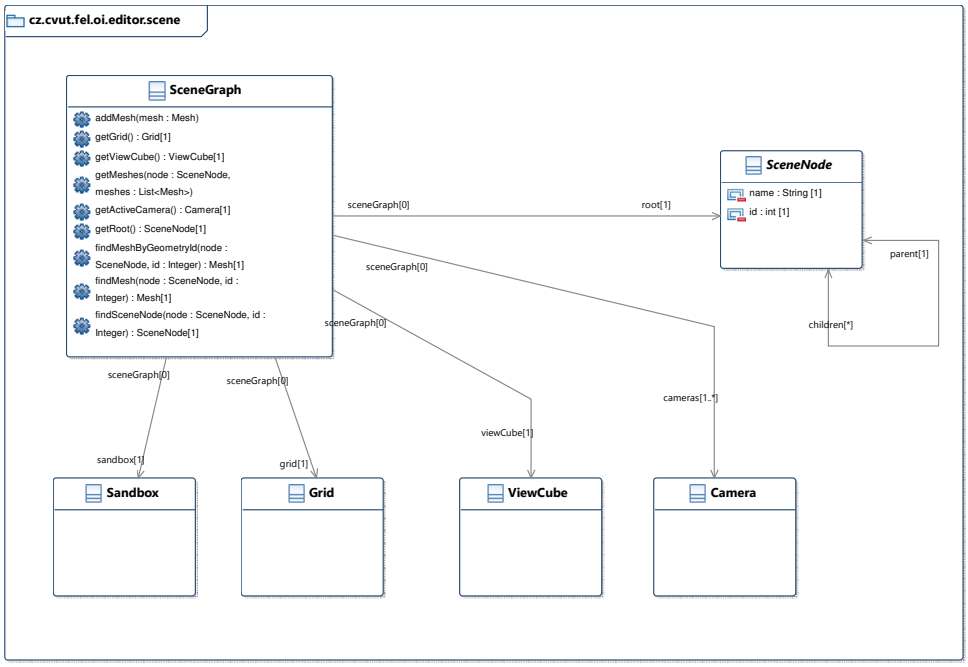
Příloha C - diagramy tříd



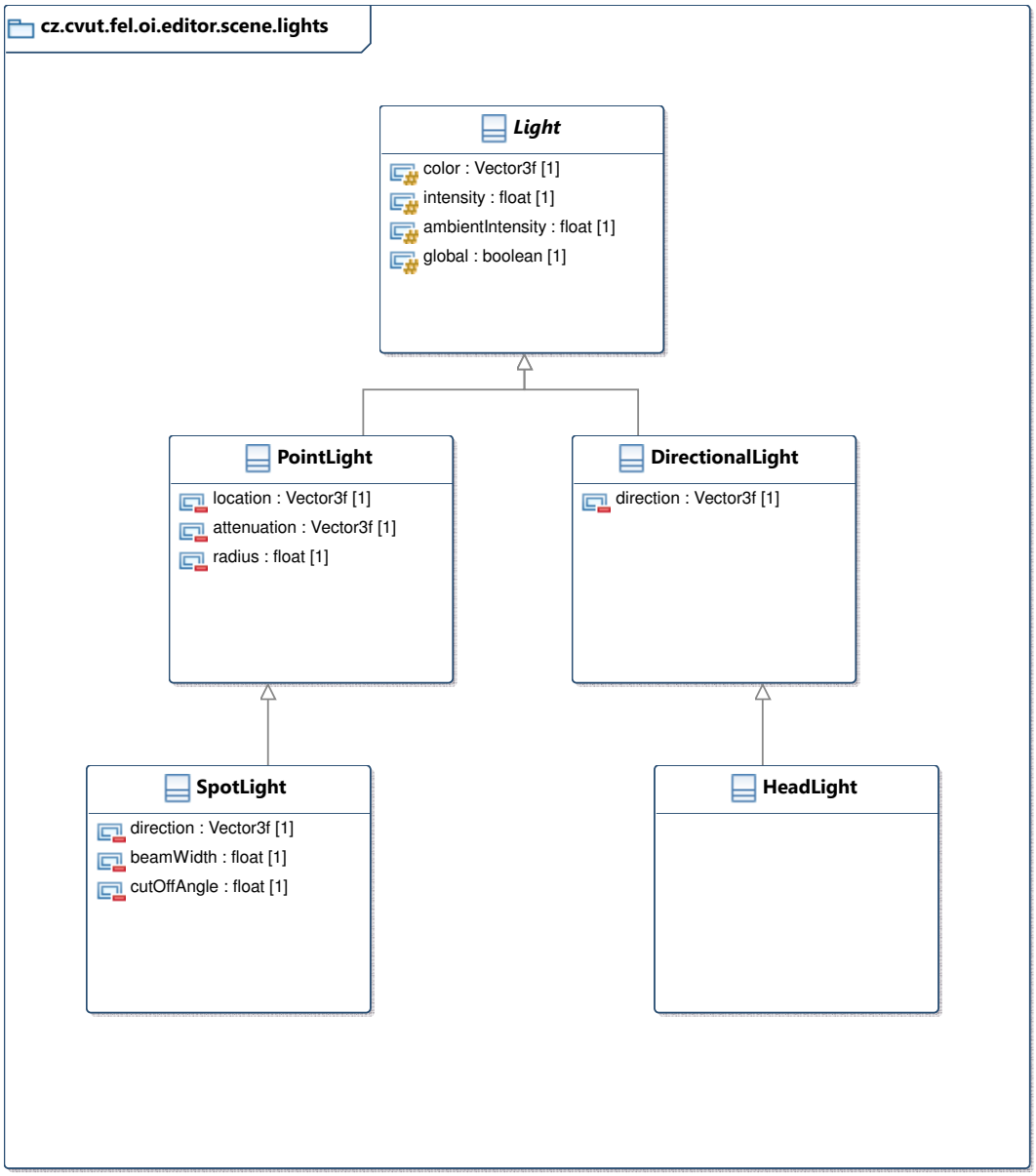
Obrázek 24 Diagram abstraktních tříd geometrie.



Obrázek 25 Potřebné interface pro selekci a transformaci.

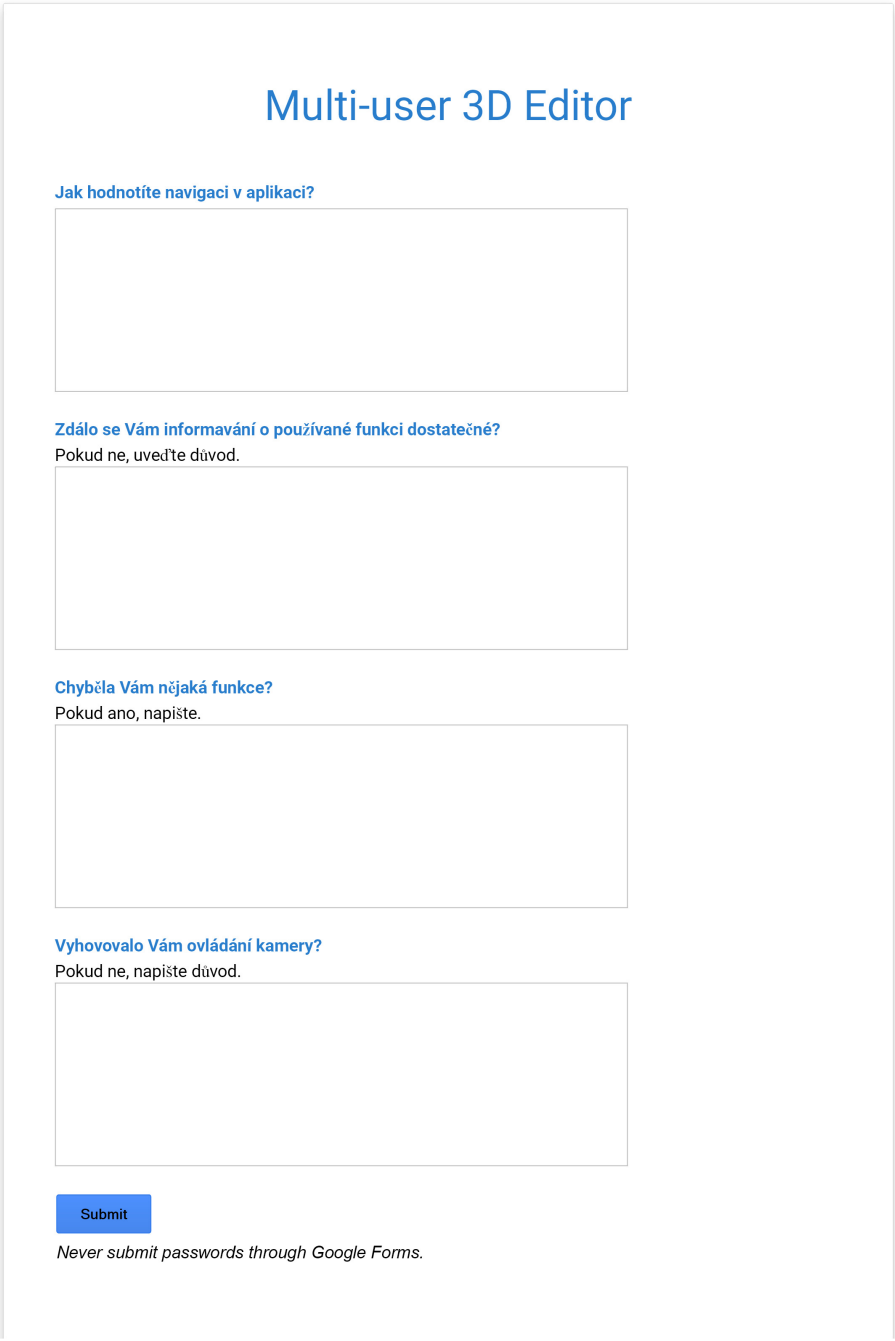


Obrázek 26 Diagram tříd grafu scény.



Obrázek 27 Diagram tříd znázorňující strukturu světel.

Příloha D - dotazník po testování



Obrázek 28 Dotazník pro participanty po skončení testu.

Příloha E - obsah přiloženého CD

Přiložené CD obsahuje zdrojové kódy mojí aplikace a všechny soubory, které potřebuje ke spuštění, tj. textury, shadery a XML preference. Ve složce bin je samotná spustitelná aplikace ve formátu APK. Dále je přiložena bakalářská práce ve formátu PDF včetně zdrojového kódu v LaTeXu. Ve složce test je umístěno PDF s dotazníkem po testování a dvě vytvořené a exportované scény do X3D.