VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

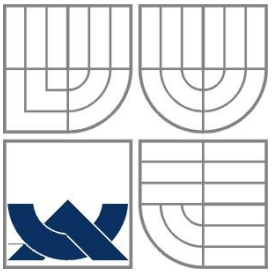
3D VIDEO BROWSER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

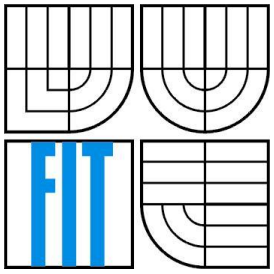
BC. VÍT MÍCHAL

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

3D VIDEO BROWSER

3D VIDEO BROWSER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. VÍT MÍCHAL
AUTHOR

VEDOUCÍ PRÁCE — ING. VÍTĚZSLAV BERAN
SUPERVISOR

BRNO 2009

Abstrakt

Cílem tohoto projektu je navrhnout a vytvořit aplikaci pro vizualizaci vzájemně propojených video dat. Vizualizace probíhá ve 3D prostoru a snaží se využít jeho výhody (např. vnímání hloubky). V tomto dokumentu je obsaženo nastudování různých kategorií prostorových uživatelských rozhraní. Dále obsahuje návrh tří možných uživatelských rozhraní a jejich ovládání. Je zde také popsána implementace a provedené testy uživatelského rozhraní. Aplikace jsou napsány v jazyce C++ s pomocí Open Inventor.

Klíčová slova

Uživatelská rozhraní ve 3D, virtuální realita, grafy, C++, OpenGL, Open Inventor, indexace video-souboru, XML, FFmpeg, testování použitelnosti

Abstract

The aim of this project is to design and create application for the visualization of interconnected video data. Visualization takes place in 3D space and seeks to exploit its advantages (such as depth perception). This document contains survey to different categories of spatial user interfaces. In addition, includes three possible designs of user interfaces and control. Implementation details and made usability tests are also described. Application is implemented in C++ using Open Inventor. The document includes evaluation of the results and made tests.

Keywords

Spatial user interfaces, virtual reality, graphs, C++, OpenGL, Open Inventor, , video-file indexation, XML, FFmpeg, usability test

Citace

Míchal Vít: 3D Video Browser. Brno, 2009, diplomová práce, FIT VUT v Brně.

3D VIDEO BROWSER

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Vít Míchal
26. květen 2009

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu práce Ing. Vítězslavu Beranovi za cenné rady a připomínky při řešení tohoto projektu.

Obsah

1 Úvod

1.1 Zadání a popis práce

Název mého diplomového projektu je „3D video browser“. Pod tímto termínem se skrývá balík aplikací pro prezentaci obrazových dat ve virtuálním 3D prostoru. Těmi obrazovými daty je míněná množina snímků (nebo také krátkých video-ukázek), které jsou navzájem propojeny vazbami. Například při hledání vzájemně podobných video-ukázek (tzv. šot) v jednom video-souboru vzniká takový typ dat. Aplikace dostane na vstup soubor s videozáznamem (ten, který byl zpracováván) a definici grafu v dalším souboru. Následně vygeneruje scénu a umístí do ní uživatele. Cílem je tedy zobrazení této sítě, tak abychom využili výhody 3D prostoru a umožnili uživateli dobrou orientaci v datech.

1.2 Struktura dokumentu

Tento dokument obsahuje studie teorie v oblasti prostorových uživatelských rozhraní a přehled několika aplikací využívajících 3D prostor pro vizualizaci dat. Jsou zde rozebrány jejich koncepty ovládání a zobrazování prostoru. V samostatné kapitole jsou popsány mnou navržené scény a možnosti interakce uživatele s okolím. Každý návrh se zaměřuje na co nejpřehlednější zobrazení různě rozsáhlých a složitých vstupních dat. Také se zde věnuji problematice získávání grafu dat ze souboru a způsobu jejich procházení. V další kapitole popisuji implementační záležitosti projektu, jako jsou použité knihovny, vnitřní representaci grafu, detekci komponent souvislosti, umisťování náhledů na povrch koule a také vysvětlení funkce simulační smyčky atd. V následující kapitole se seznámíme s provedenými testy a provedeme rozbor výsledků. V závěrečné kapitole jsem shrnul dosažené výsledky a nastínil možná vylepšení.

1.3 Návaznost na semestrální projekt

Tato práce navazuje na předchozí semestrální projekt, kde jsem studoval existující projekty, znázorňující informace ve 3D prostoru. Dále jsem navrhnul vzhled a způsob ovládání aplikace. Proto jsem použil a jen mírně upravil kapitoly Teorie a Návrh vlastního řešení.

1.4 Cíl práce

Výsledkem této práce je návrh, implementace a otestování uživatelského rozhraní, které čerpá z již existujících uživatelských rozhraní a inspiruje se jejich koncepty ovládání. Výsledný systém je použitelný jako samostatná aplikace nebo jako zobrazovací komponenta do jiných projektů.

2 Teorie

2.1 Grafická uživatelská rozhraní ve 3D

V literatuře se pod pojmem 3D uživatelská rozhraní objevuje široká škála rozhraní zobrazující a pracující s 3D objekty. Zatím se v praktických podmínkách neprosadilo pravé 3D rozhraní se všemi komponentami ve 3D prostoru. V současně době vznikají hybridní rozhraní kombinující prvky 2D a 3D zobrazení.

Na počátcích bylo 3D zobrazování doménou vědeckých pracovníků s přístupem k nákladným a výkonným superpočítačům. V dnešní době díky běžně dostupné podpoře akcelerovaného prostorového zobrazování narůstá potřeba přinést třetí rozměr do grafických rozhraní. Uživatelská základna se rozrostla ze skupiny vědců a odborníků na rozličnou společnost jak inženýrů, mediků, návrhářů, tak i běžných uživatelů počítačů. K těmto rozhraním se dostávají lidé s různými znalostmi počítačů a je nutné navrhovat je tak, aby ovládání rozhraní bylo přirozené a nebránilo jim v práci.

2.1.1 Vnímání hloubky

Jak sdělit pomocí dvou-dimenzionálního obrázku, kde se objekty nachází v prostoru? V reálném životě se o to postará náš mozek, který interpretuje obraz promítaný na sítnici. Ve virtuálním světě musíme prostor vykreslit tak, aby ho mozek pochopil jako tří dimenzionální. Lidskému mozku pomáhá k rozpoznání prostoru několik vodítek. Mezi ně patří: lineární perspektiva (sbíhání čar do jednoho bodu), menší velikost vzdálenějších předmětů, ubývání jasu vzdálenějších objektů, textura a také světla a stíny. Pokud budeme respektovat tyto principy, vytvoříme věrohodnou iluzi prostoru, kterou můžeme promítnout na 2D zobrazovací zařízení.

2.1.2 Způsoby interakce uživatele a 3D prostředí

Naším jediným účelem samozřejmě není pouhé vykreslení scény, ale také potřebujeme prostředí nějak ovládat a pohybovat se v něm. V uveřejněném článku [01] klasifikují interakci s prostorem do následujících skupin.

2.1.2.1 Navigace

V rozlehlých 3D prostředích je navigace nejčastější činnost, kterou uživatel provádí. Navigace by měla podpořit dojem přítomnosti v prostoru a umožnit účinný a efektivní pohyb mezi vzdálenými místy. Na druhou stranu by uživatel neměl být navigací příliš zaměstnáván, protože se potřebuje soustředit na své činnosti v prostoru.

Dále dělíme navigaci na části *cestování (travel)* a *hledání cesty (wayfinding)*. *Cestování* je pohybová komponenta navigace a zajišťuje přesun v prostoru. *Hledání cesty* je komponenta, která pohyb prostorem vymyslí a naplánuje.

Podle [01] můžeme obecně navigaci klasifikovat do tří kategorií:

- **Průzkum (Exploration)** – pohyb bez konkrétního cíle, pouhé zkoumání terénu.
- **Hledání (Searching)** – máme určitý cíl a snažíme se zjistit jeho umístění.
- **Manévrování (Maneuvering)** – pohyby na krátké vzdálenosti s vysokou přesností za účelem získání nejlepšího pohledu pro určitou úlohu.

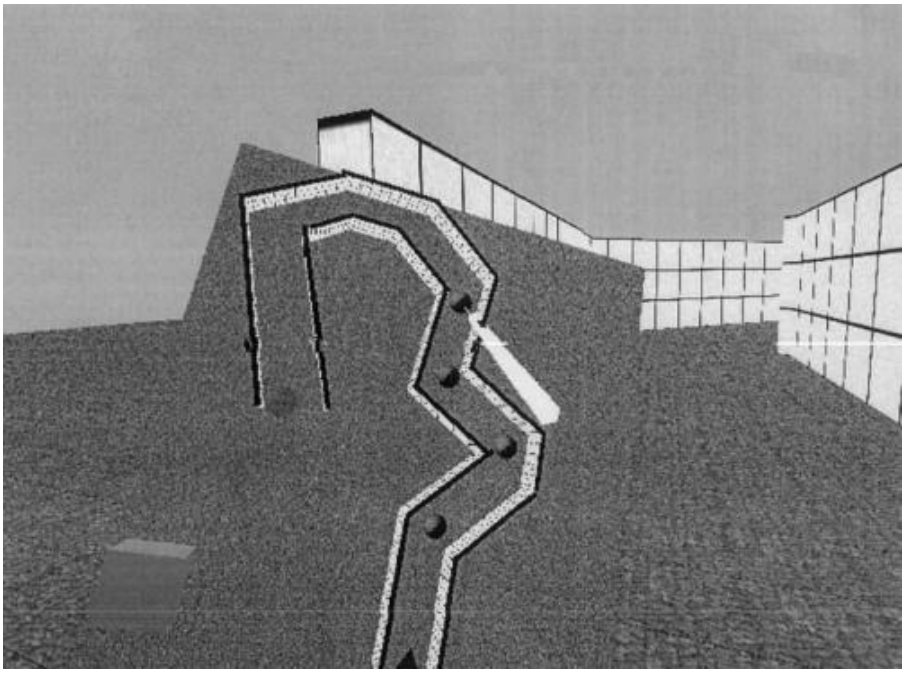
Většina způsobů *cestování* ve 3D prostoru spadá do následujících pěti kategorií (metafor):

- **Fyzický pohyb** – přirozený pohyb těla uživatele pomocí končetin; vyžaduje relativně hodně prostoru, zařízení na snímání pohybu, popř. šlapací pásy či „šlapací mlýn“. Tyto techniky jsou vhodné pro rozšířený pocitu přítomnosti uživatele v prostředí.

Obrázek 1: Virtusphere - zařízení pro snímání chůze (technika fyzického pohybu) [03]

- **Ruční manipulace se záběrem snímání** – při tomto způsobu navigace uživatel rukou provádí pohyb v prostoru. Jednoduše „chytí“ prostředí a přitáhne si ho k sobě. Touto technikou lze také rotovat okolo vybraného objektu v prostředí. Tento způsob může být efektivní, jednoduchý na osvojení, ale zároveň poněkud únavný.
- **Kormidlování** – neustálé určování směru pohybu. Nejčastěji se vyskytuje případ pohledově-orientovaného kormidlování, kdy směr orientace pohledu určuje směr cestování. Kormidlovací techniky jsou účinné a univerzální.

- **Cestování založené na cíli** – Uživatel vybere v prostoru cíl cesty a systém se postará o jeho přesun. Samotný přesun může mít podobu teleportování, kdy uživatel okamžitě skočí do jiné lokace. Nebo může být také prezentován jako postupný pohyb prostředím až k cíli. Z uživatelského pohledu se jedná o jednoduchou techniku.
- **Plánování cesty** – Uživatel určí cestu, kterou v prostředí poputuje a systém se postará o samotný přesun po dané cestě. Tím je jim umožněno řídit trajektorii pohybu a zároveň se věnovat svým úlohám. Uživatelé definují cestu např. umísťováním řídících bodů na mapě.



Obrázek 2: Technika plánování cesty - kuličky na obrázku jsou body, kudy prochází cesta [01]

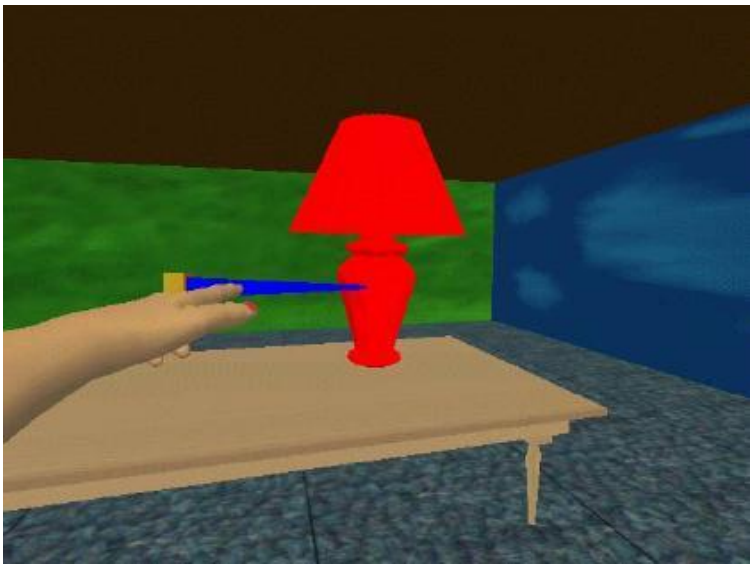
Vedle výběru typu navigace si návrhář uživatelského rozhraní musí ujasnit další otázky. Například tu, zda je cestování používáno jako činnost sama o sobě nebo, zda je součástí jiných úloh (Cestování založené na cíli umožní uživateli soustředit pozornost na vyšší cíle jeho práce). Další otázka se týká míry propracovanosti navigačních technik. Pokud použijeme některou sofistikovanější metodu, uživatelé budou více „vtáhnuti“ do prostředí, než kdybychom použili jednodušší techniku.
Druhou částí navigace je *hledání cesty*. Jedná se o proces, při kterém se hledá cesta prostředím s využitím informací o prostoru. Použití a získávání prostorových znalostí je ovlivněno mimo jiné také pohledem na prostředí (pohled z 1. osoby nebo pohled 3. osoby) a technikou cestování.

Uživatelé by měli získávat podporu od systému *hledání cesty*. Každý uživatel má odlišné orientační schopnosti, a pokud mu dáme přílišnou svobodu, muže se tím snížit jeho orientace

v prostoru. Tato podpora může být zaměřena buď na uživatele, nebo na prostředí. Do podpory zaměřené na uživatele spadá např. široké pole výhledu, ale také ozvučení scény. Podpora zaměřená na prostředí může být dělena na strukturální organizaci a vodítka. Do strukturální organizace patří jasně odlišitelné časti prostředí, které mohou být identifikovány a vztaženy k dalším částem prostředí. Vodítka jsou inspirována skutečným světem a patří mezi ně mapy, kompasy, ale také architektonické záležitosti jako osvětlení, barvy, textury a horizont. Do kategorie vodítek tedy spadá vše, podle čeho se orientujeme ve skutečném světě a je to přenesené do virtuálního prostředí.

2.1.2.2 Výběr a manipulace s objekty

Techniky pro manipulaci ve 3D prostoru by měli poskytovat alespoň jednu ze tří základních úloh: výběr, umísťování a rotaci objektu. Klasickým přístupem k návrhu manipulačních technik je zpřístupnit uživateli virtuální ruku (jakýsi 3D kursor), jehož pohyby korespondují s pohybem snímačem ruky (myši, nebo vstupním zařízení pro virtuální realitu).



Obrázek 3: Metoda selekce pomocí paprsku [04]

Výběr a manipulace jednoduše evokuje dotyk s objektem. Technika virtuální ruky je poměrně intuitivní, protože simuluje interakci ze skutečného světa. Vybrány mohou být pouze objekty, které jsou na dosah ruky.

Existuje několik metod výběru objektů, které obcházejí omezení délky ruky. Např. tzv. Go-Go technika rozšiřuje dosah uživatelovy ruky pomocí nelineárního mapování. Pokud uživatel překročí prahovou vzdálenost, mapování ruky do virtuálního prostoru přestane být lineární a ve výsledku délka paže roste. Další běžnou cestou k výběru a manipulaci s objekty v 3D prostředí je pomocí virtuálního paprsku (ray casting) vycházejícího z virtuální ruky. V okamžiku kdy paprsek protne nějaký objekt, lze s tímto předmětem manipulovat. Vzniklo několik variací na výběry pomocí

paprsku. Například metoda světelného kuželu (spotlight) využívá paprsek tvaru kužele, který se směrem od uživatele rozšiřuje. Předměty spadající do vnitřku paprsku jsou potom považovány za vybrané. Tato metoda vyžaduje další rozlišení objektů v případě, kdy do vnitřku spadá více předmětů. Pánové Forsberg, Herdon a Zeleznik navrhnuli kukátkovou (aperture) metodu. Jedná se opět o kuželový ukazatel, jehož směr je definován pozicí oka uživatele (popř. pozice kamery) a pozicí snímače ruky. Uživatel ovládá velikost výběru změnou vzdálenosti ruky od oka.

Všechny výše popsané techniky dávají uživatelům nástroje, které jim dovolují sahat a vybírat předměty ve virtuálním světě. Alternativní přístup by mohl být založen na principu změny velikosti okolního prostředí (popř. uživatele). Takto by si mohl pozorovatel zmenšit oblast okolního světa např. do velikosti dlaně a pracovat s ní jako s miniaturou. Tato technika byla použita v projektu 3DM immersive modeler (autoři Butterworth, Davidson, Hench).

Každá z uvedených metod selekce a manipulace má své výhody a nedostatky. Byla učiněna řada pokusů snažících se zkombinovat jejich vlastnosti tak, aby podávali nejlepší výsledky. Například projekt Virtual Tricorder (autoři Wloka a Greenfield) kombinuje výběr a manipulaci pomocí paprsku s navigačními technikami a ovládání úrovně detailu zobrazení (LOD) v jednom univerzálním nástroji.

Vývojářům se může zdát různorodost způsobu interakce ohromující. Obecně však existují určitá vodítka pro výběr té správné metody. Žádná technika není dokonalá, každá najde své uplatnění v určitých typech úloh a aplikací. Často nerealistické metody mají lepší výsledky než ostatní založené na věrné imitaci skutečného světa.

2.1.2.3 Řízení systému

Do kategorie řízení systému spadá situace, kdy chce uživatel zadat příkaz k vykonání. Vyvolání příkazu se provádí vybráním nějakého elementu z množiny elementů. V klasických desktopových aplikacích je volání příkazů běžnou a frekventovanou činností. Naneštěstí v prostorových uživatelských rozhraních často nemůžeme smysluplně využít běžná rozbalovací menu či příkazovou řádku. Je to dáno tím, že to, co celkem jednoduše funguje ve dvourozměrném prostoru, se přechodem do 3D prostoru stává neefektivní. Například stisknutí položky v menu plovoucí v prostoru je více obtížné než výběr z menu v ploše, pokud nemáte myš.

Řízení systému se dá rozdělit do čtyř skupin, jmenovitě: grafická menu (vizuální reprezentace příkazů), hlasové příkazy (menu procházená pomocí hlasu), interakce pomocí gest (množiny příkazů přístupné pomocí gest) a nástroje (virtuální objekty s konkrétní funkcí nebo stavem). Existují i kombinace předchozích čtyř přístupů.

Řízení systému je často součástí jiné univerzální úlohy interakce. Uživatel by měl zůstat soustředěn na cíl jeho činnosti a neměl by být rozptylován vedlejšími akcemi. Ideální je situace, kdy prostředí zůstává v jednom módu interakce. Jedním ze způsobů, jak toho dosáhnout, je umístění menu s řídícími prvky na určitou pozici relativně vůči hlavě (kameře) uživatele. Díky tomu můžeme použít 2D menu z klasických desktopových aplikací a zachováme tím prostorové vnímání scény. Pokud jsou

menu složitější, musí se nějak vyřešit problém s jeho přehledností. Ten se dá řešit optimalizováním struktury menu pomocí citlivosti na kontext nebo množinou nabídek a podnabídek. Návrháři by měli také myslet na správnou zpětnou vazbu směrem k uživateli, jakmile vybral nějakou nabídku nebo, když systém provádí některou činnost. Chyby v ovládání vzniklé nedostatečnou zpětnou vazbou jsou rušivé a zbytečné. I v této oblasti zůstává spoustu prostoru pro další výzkum, vyhodnocování a aplikování do praxe.

2.1.2.4 2D interakce v třídimensionálních prostředích

Pokud máme prostorové uživatelské rozhraní, neznamená to, že v něm nemůže být interakce probíhající ve dvou-dimensionálním prostoru. Ve skutečnosti 2D interakce může být pro určité úlohy výhodnější než její 3D varianta. Zvláště pokud nemáme k dispozici dotyková či hmatová zařízení, zpřístupní nám 2D interakce pocit zpětné vazby užitečné pro vytváření objektů, psaní atd. Pokud zauvažujeme o metodách výběru, dojdeme k závěru, že techniky výběru ve 2D jsou efektivnější. Kombinací výhod 2D a 3D interakcí můžeme vytvořit rozhraní, která jsou jednodušší k používání a intuitivnější. Rozhodnutí o tom, které akce části systému budou ve 2D a které ve 3D je důležitým momentem návrhu.

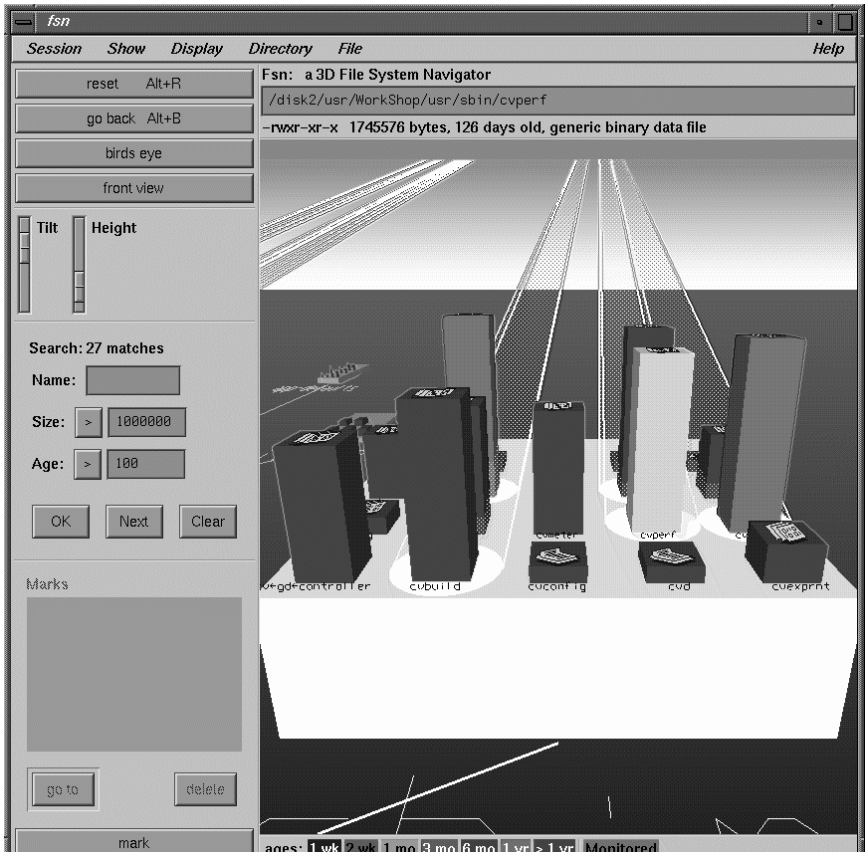
2.1.3 Aplikační domény prostorových grafických rozhraní

Nyní si představíme několik oblastí, ve kterých najdou podle [02] své uplatnění prostorová uživatelská rozhraní. Určitě by se našlo více využití než těch několik, které jsou tu vypsány, berme tento výčet jako orientační.

2.1.3.1 Vizualizace informací

Pomocí metafor jako jsou místnosti, domy, města můžeme začít využívat prostorových schopností, kterých jsme získali v reálném světě. Prostorové koncepty jsou nepochybně důležité v běžném životě a uživatelská rozhraní by z nich mohla čerpat.

File System Navigator (na obrázku č. 4) z dílny Silicon Graphics je prototyp, který si můžeme pamatovat z filmu Jurský park. Program zobrazuje souborový systém Unixu. Používá komponenty 2D rozhraní a zobrazuje je v prostoru. Adresáře jsou reprezentovány hierarchicky jako podstavce umístěné na ploše. Výška podstavce odpovídá množství obsahu uvnitř adresáře a barva o jeho datu vytvoření. Adresáře jsou spojeny čárami, po kterých je možné cestovat. Kliknutím na blok souborů zobrazí světelný kužel, který znamená označení souboru. Dvojité kliknutí otevře soubor k editaci.



Obrázek 4: File System Navigator - příklad vizualizace 2D konceptu ve 3D prostoru [5]

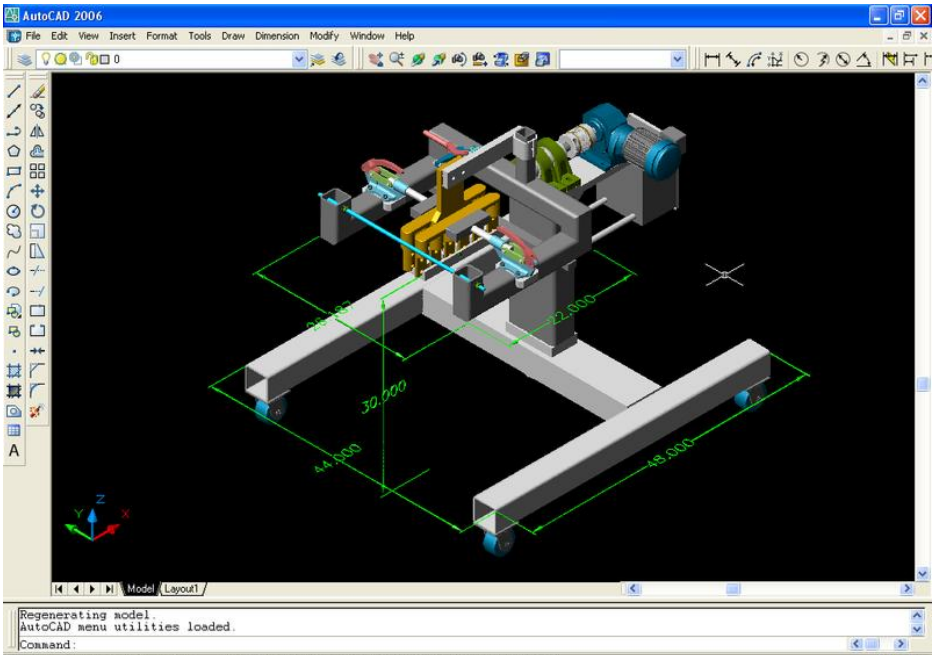
Otázkou zůstává, zda tento přístup pomáhá uživateli získat dobrou představu o obsahu disku, ale každopádně se jedná o zajímavý přístup k prezentaci stromové struktury.

2.1.3.2 Simulace

Simulátory létání či řízení mají často za cíl poskytnout umělé prostředí vypadající a reagující jako skutečné prostředí z reálného světa. V těchto 3D aplikacích je účelem vytvořit uživateli pohlcující a realistický zážitek. V této skupině nalezneme různé simulátory chemických, fyzikálních reakcí, aerodynamické tunely, letecké simulátory či simulátory dalších situací.

2.1.3.3 Počítačem podporovaný návrh

CAD systémy usnadňují inženýrům a architektům vytváření a manipulaci s 2D a 3D modely. Tyto systémy se zároveň starají o tvorbu a udržování výkresů, technické dokumentace a také mohou zobrazit náhled budoucího výrobku nebo budovy. Tím napomáhají vyhodnocení kvality produktů ještě dříve, než se dostanou do výroby.



Obrázek 5: Program na podporu návrh AutoCAD [06]

2.1.3.4 Počítačem podporovaná spolupráce

Počítačem podporovaná spolupráce (Computer Supported Cooperative Work - CSCW) je počítačová oblast zkoumající, jak lidé spolupracují s využitím počítačových technologií. Typické aplikace zahrnují elektronickou poštu, varovné a oznamovací systémy, videokonference, komunikační programy, hry pro více hráčů a sdílené aplikace v reálném čase (např. kolaborativní skicování nebo psaní). Virtuální prostředí mohou poskytnout přirozený způsob spolupráce pomocí počítačů.

Projekt DIVE (Distributed Interactive Virtual Environment) je toolkit pro budování multi-uživatelských distribuovaných aplikací. Projekt vytváří virtuální prostředí, ve kterém se setkávají spolupracovníci. Každý je reprezentován tzv. „avatarem“, který má individuální podobu, aby se účastníci navzájem rozlišili. Typickými prostředími jsou virtuální posluchárny, konferenční místnosti nebo třídy. Tímto napomáhají počítačové technologie zlepšit kooperaci lidí na velké vzdálenosti.

Obrázek 6: Virtuální školení v projektu DIVE [07]

2.2 Testování použitelnosti uživatelských rozhraní

Historie testování použitelnosti sahá až do osmdesátých let minulého století, kdy byly provedeny testy použitelnosti ve společnosti Xerox na jejich vyvíjené pracovní stanici Xerox Star.

Testování použitelnosti je technika ohodnocení softwarového produktu pomocí testů prováděných na uživatelích. Stává se nenahraditelnou praktikou, protože dává přímou odezvu, jakým způsobem uživatelé opravdu produkt používají. Sleduje se zejména schopnost plnit zamýšlený účel. Často se provádí na internetových aplikacích, počítačových rozhraních a zařízeních.

2.2.1 Cíle testování použitelnosti

Díky testování použitelnosti se nachází nové možnosti vylepšení softwaru z hlediska uživatele. Zpravidla se měří tyto čtyři aspekty: efektivita, přesnost, zapamatovatelnost a citová odezva.

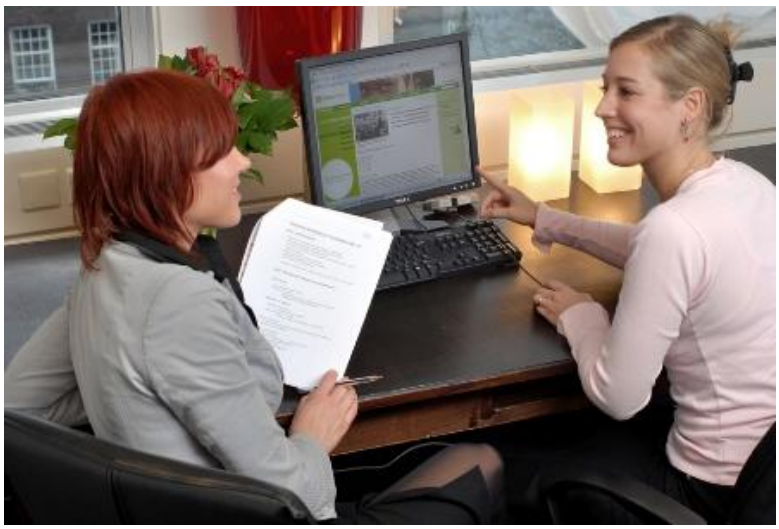
Výsledky prvního testu se obvykle používají jako výchozí pro sledování pokroku na produktu při dalších testech.

- **Efektivita** – kolik času a kolik kroků je potřeba udělat k dokončení základních úloh (například nákupu zboží, vytvoření nového účtu atp.).
- **Přesnost** – Kolik chyb lidé dělají? Jsou chyby zvratné či nezvratné? Dělali by je, kdyby měli správné informace?
- **Zapamatovatelnost** – Kolik si uživatel zapamatuje po delší době nepoužívání aplikace?
- **Citová odezva –** Jak se uživatel cítí, jakmile dokončí úlohu? Je spokojený nebo vystresovaný?

2.2.2 Metody

Připravení testů použitelnosti se skládá z vypracování scénáře použití, který představuje realistickou situaci, kdy testující osoba plní zadané úkoly v systému. Další člověk sleduje, jak si počíná a dělá si poznámky. Vedoucí testu se zpravidla ptá uživatele na doplňující otázky ohledně testu, aby lépe získal zpětnou vazbu. (např: „Bylo jednoduché najit tuto informaci?“ apod.)

- **Hallway testování** - do testu se přizve náhodná osoba (např. ta která jde náhodou chodbou – odtud název) a provede se s ní několik jednoduchých úkolů. Pro testy je nejlepší uzavřená místnost s počítačem.
- **Vzdálené testování** - vzdálené testování použitelnosti neboli nemoderované a asynchronní testování. Buď je testování prováděno ve stejném čase na odlišných místech anebo se testuje v různém čase a na různých místech. V prvním případě vedoucí testu sleduje testující osobu zprostředkovaně (kamera, hlasový hovor), může si dělat poznámky a klást doplňující otázky. V druhém případě musí být testovací scénář napsat tak, aby alespoň z části kompenzoval svou nepřítomnost.



Obrázek 7: Snímek z testování použitelnosti [13]

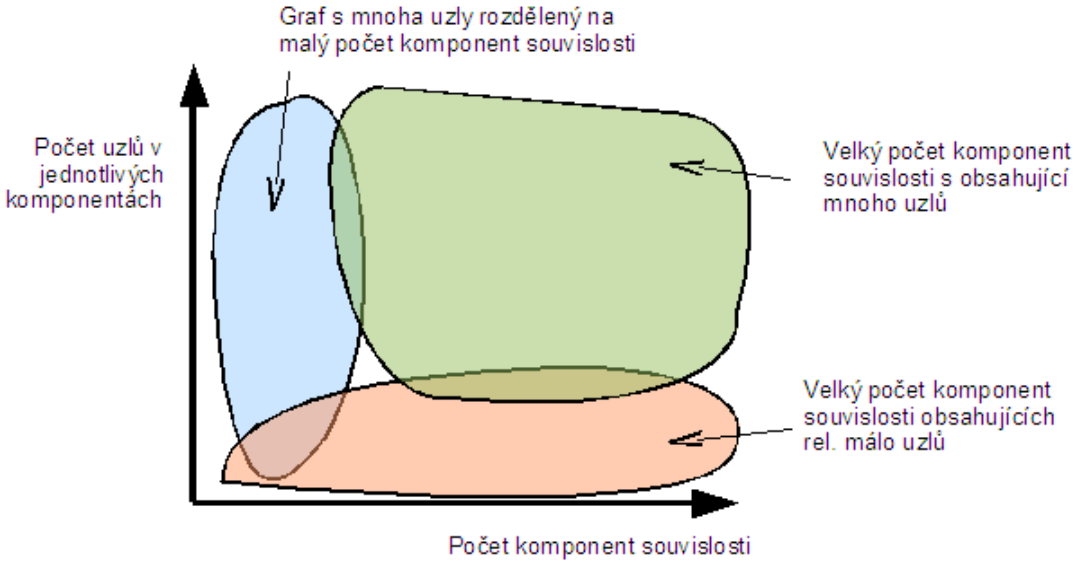
2.2.3 Vhodný počet testů

V raných devadesátých letech Jakob Nielsen, toho času výzkumník u společnosti SUN, zveřejnil koncept několika malých testů použitelnosti v každé části vývojového procesu. Jeho argument byl, že pokud dva nebo tři lidé mají potíže s navrženým rozhraním, nemá cenu si toto ověřovat na větším počtu lidí. Nejlepší výsledky dávají testy na skupině o přibližně 5 lidech.

3 Návrh vlastního řešení

V předchozí kapitole jsme se seznámili s různými druhy 3D uživatelských rozhraní a typy interakce s nimi. Nyní se pojďme zamyslet nad návrhem 3D Video Browseru. Účelem aplikace je vizualizace dat získaných rozpoznáváním podobných sekvencí ve videu. Vstupní data obsahují uzly a vazby mezi nimi. Uzly představují krátké video-sekvence (šoty) a hrany znamenají podobnost jednoho šotu s druhým. Jedná se tedy o neorientovaný graf, který může být různě členitý a souvislý či nesouvislý. Souvislý graf je takový, z jehož každého vrcholu existuje cesta do libovolného jiného vrcholu v grafu. Pokud je graf nesouvislý, je rozdělen alespoň na dvě souvislé části, které se nazývají komponentami souvislosti [08]. Graf na vstupu můžeme rozdělit podle počtu komponent a uzlů na čtyři skupiny:

- Malý počet komponent a velký počet uzlů v nich
- Velký počet komponent s mnoha obsaženými uzly
- Velký počet komponent s malým počtem uzlů
- Málo komponent s málo uzly (průnik horizontální a vertikální oblasti)



Obrázek 8: Znázornění různých variant složitosti grafu

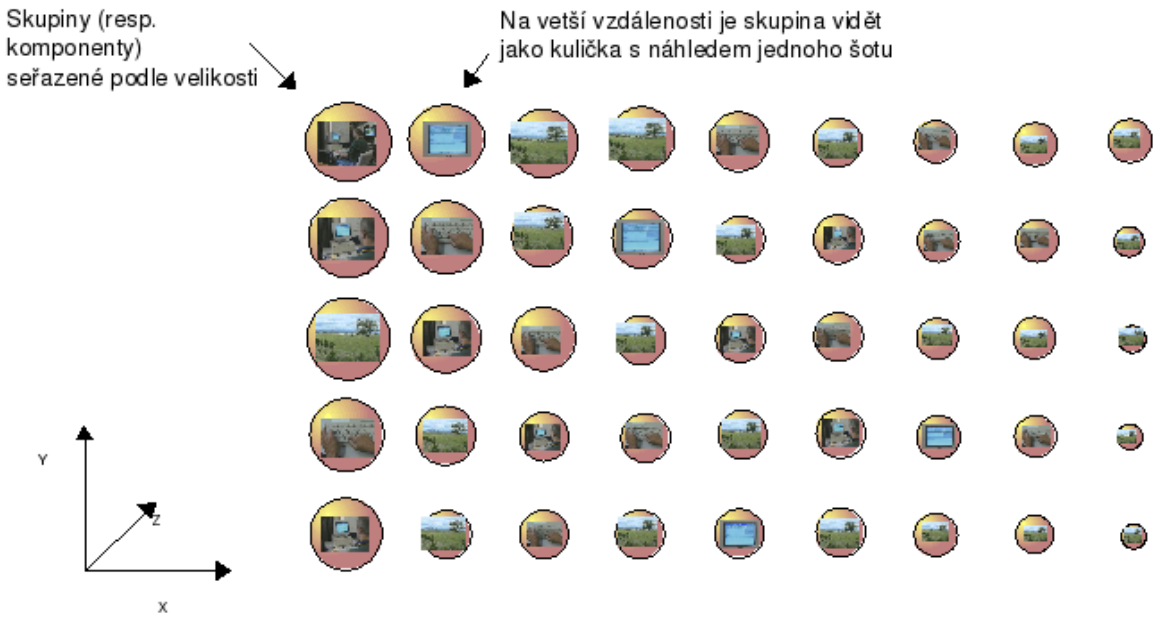
Na obrázku 8 vidíte tyto skupiny umístěné v grafu. Pro první tři typy grafu jsem se rozhodl navrhnout samostatné uživatelské rozhraní. První skupina grafů obsahuje malý počet komponent (řekněme 5 a méně), které jsou relativně složité, protože obsahují mnoho uzlů. Je zde tedy kladen větší důraz na pohodlné procházení grafu než na zobrazování komponent souvislosti (návrh uživatelského rozhraní č. 2). V druhé skupině máme mnoho komponent a mnoho uzlů v nich, měl

by být zde tedy promyšlenější způsob zobrazování a organizace komponent (návrh č. 3). Do třetí skupiny patří grafy s málo uzly v mnoha komponentách souvislosti (např. graf s mnoha izolovanými uzly) na jehož vizualizaci je specializován první návrh.

3.1 Návrhy zobrazení a ovládání

3.1.1 První koncept

Tento návrh upřednostňuje obecný přehled nad komponentami nad detailním zkoumáním jednotlivé komponenty. Je určený pro graf s větším množstvím komponent souvislostí, které nejsou příliš rozsáhlé. Proto jsou při prvním zobrazení scény komponenty zobrazeny v mřížce (obrázek 9) a seřazeny od největší (vlevo nahoře) po nejmenší (vpravo dole). Komponenta je z dálky zobrazena jako kulička (používám termín „planetka“) s náhledovým obrázkem šotu. Při přiblížení k planetce na menší vzdálenost její zobrazení přepne na detailnější pohled (obrázek 10).

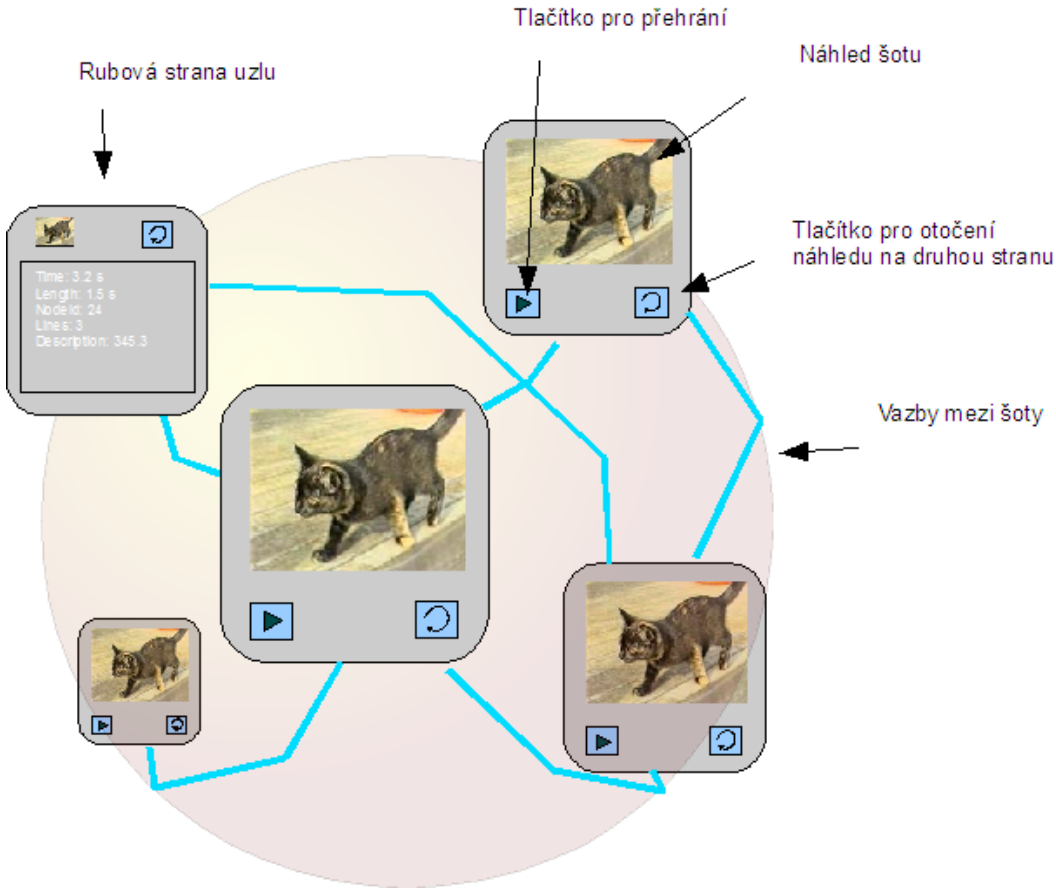


Obrázek 9: Pohled na přehledovou scénu

Podrobnější pohled na planetku zobrazuje jednotlivé uzly (šoty) propojené vazbami. Jednotlivé náhledy se samy automaticky nastavují lícovou stranou k pozorovateli. Každý náhled obsahuje obrázek šotu, tlačko pro přehrání video-sekvence a tlačítko pro otočení náhledu. Po stisku tlačítka otočení se náhled otočí na rubovou stranu a dále se natáčí k pozorovateli rubovou stranou, dokud jej

neotočí zpět. Na rubu náhledu jsou podrobnější informace o šotu vypsané pomocí textového okna, které lze i rolovat pokud se text nevejde naráz celý.

Planetku lze libovolně otáčet. Jednotlivé náhledy jsou umístěny na povrchu planetky.



Obrázek 10: Detail jedné komponenty souvislosti

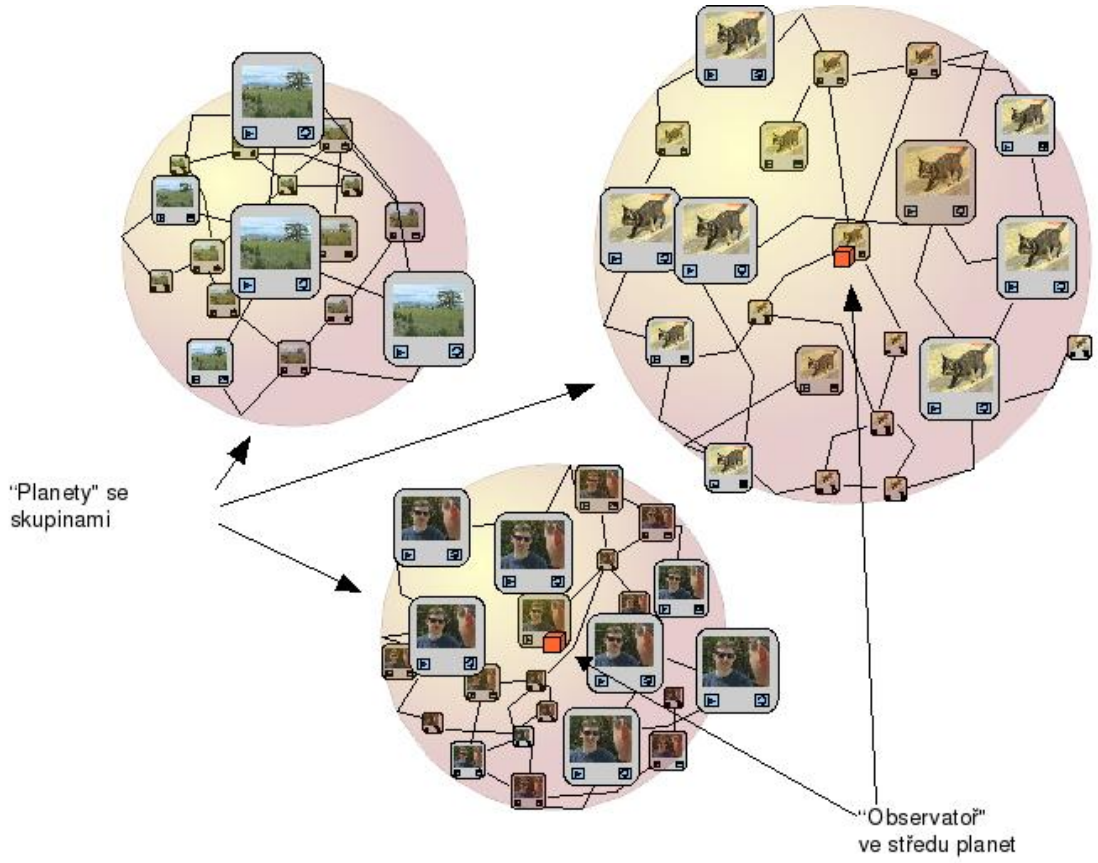
O pohyb v prostoru se stará systém, uživatel klikne na libovolnou planetku a je automaticky přesunut do blízkosti té planetky. Uživatel může libovolně otáčet pohledem, přibližovat a oddalovat.

Jako možné rozšíření tohoto pohledu vidím implementaci zobrazení více zdí (mřížek s planetami) za sebou, v případě, že by planetek bylo opravdu mnoho. Výsledná mříž by pak měla podobu kvádru a pravidelně rozmístěnými planetkami.

3.1.2 Druhý návrh

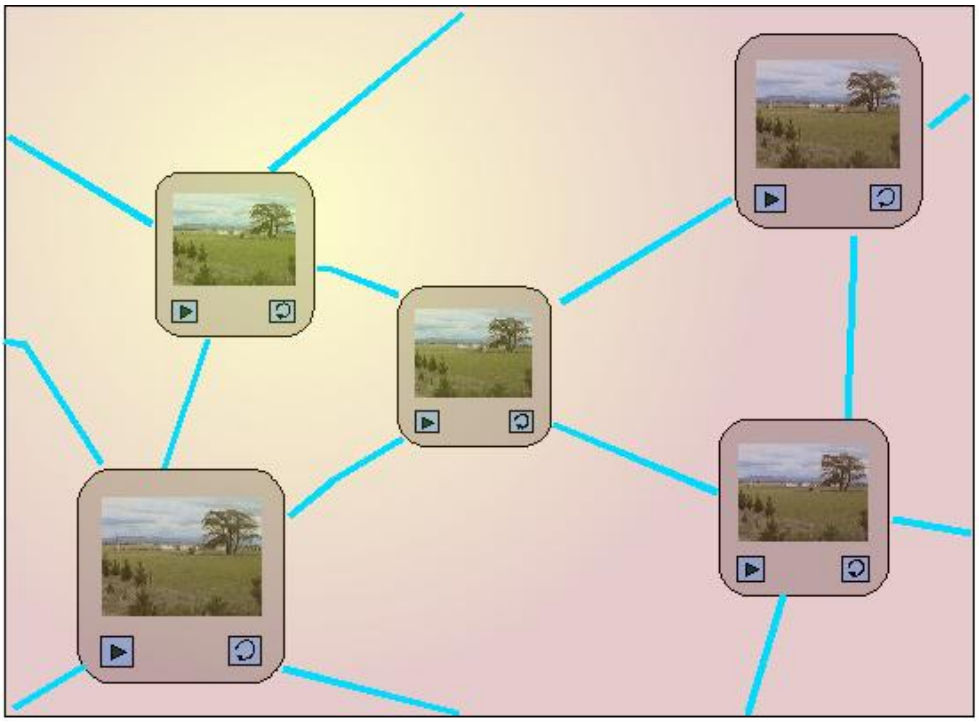
Tento koncept ovládání je specializován na graf, kde je málo komponent souvislosti, ale tyto komponenty jsou poměrně rozsáhlé. Na prvotním pohledu na scénu (obrázek 11) můžeme nalézt tři planety reprezentující komponenty souvislosti umístěné v prostoru nedaleko od sebe. Jako v předchozím návrhu jsou zde planety s náhledy šotů umístěnými na jejím povrchu. Novinkou v

těchto planetách je „observatoř“, to je objekt ve středu koule (planety). Pokud klikneme na observatoř, přesune se kamera na pozici tohoto středu a uživatel hledí na síť grafu „zevnitř“ (na obrázku 11). Toto umožní pohodlnější procházení grafu. Chytnutím myší za planetu a potažením je možné planetku otáčet a tím dále prozkoumávat graf.



Obrázek 11: Pohled na všechny "planety" (komponenty) naráz

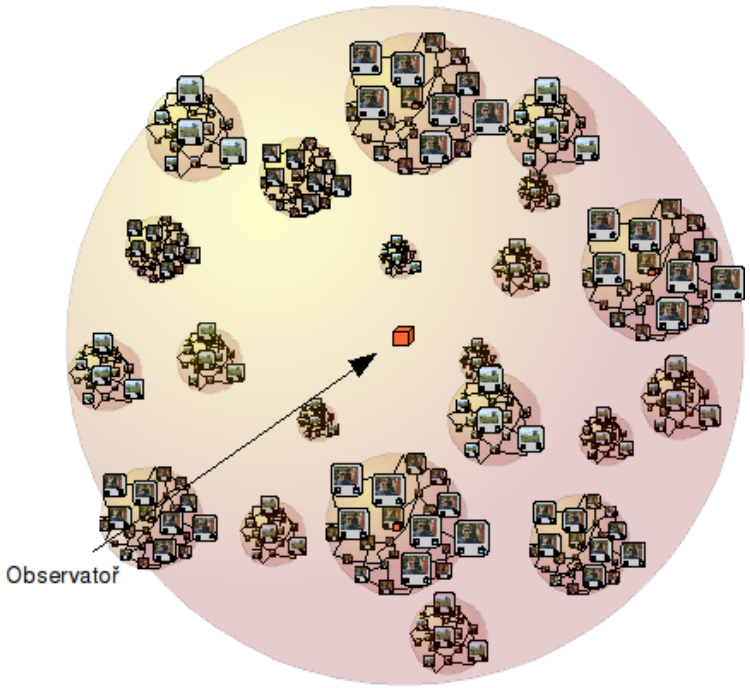
Když srovnáme tento návrh zobrazení s předchozím, zjistíme, že zde chybí mřížka a byla přidána observatoř. Mřížka planet byla odebrána proto, že počítáme s malým počtem komponent souvislosti grafu (tedy i planet) a tak je přehledné i jejich náhodné uspořádání v prostoru.



Obrázek 12: Pohled na graf z observatoře

Naopak observatoř byla přidána proto, že je zde očekáváno mnoho uzlů a vazeb. V observatoři se již uživatel nemusí starat o svoji polohu a může se plně soustředit na procházení grafu a zobrazování náhledů.

3.1.3 Třetí koncept

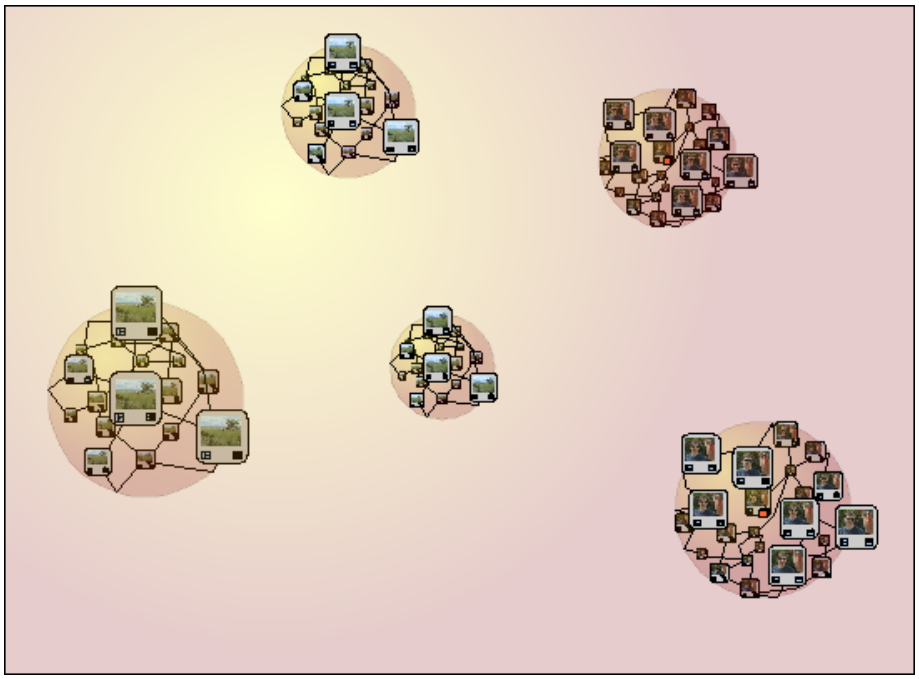


Obrázek 13: Třetí koncept - primární pohled

Třetí koncept ovládání je určen pro rozsáhlé grafy s mnoha komponentami souvislosti. Pro tuto situaci jsem se rozhodl použít metaforu planety jak pro zobrazení jednotlivých grafů, tak pro samotné komponenty souvislosti. Výsledkem je velká planeta, na jejímž povrchu je umístěno množství menších planetek (obrázek 13).

Uživatel může planetu otáčet tak, aby si mohl prohlédnout komponenty z různých stran a směrů. Uvnitř velké planety je observatoř se stejnou funkcí jako předchozím návrhu. Díky ní má uživatel pohled na scénu zevnitř.

Také jednotlivé malé planetky obsahují observatoře. Kliknutím na observatoř malé planetky je možné se do ní přesunout a pozorovat planetku zevnitř. Takto bude uživatel „přeskakovat“ z jedné planetky do druhé a bude zkoumat sítě šotů. Všechny přechody v prostoru jsou plynule animovány a ruční otáčení má svoji setrvačnost.



Obrázek 14: Pohled z observatoře ve středu velké planety

3.2 Formát vstupního souboru s popisem grafu

Aplikace načítá graf šotů a podobností z textového souboru na disku. Jako formát uložení dat jsem zvolil XML pro jeho snadnou čitelnost pouhým okem a také kvůli množství již existujících knihoven usnadňujících jeho načítání.

Vstupní soubor se skládá z otevíracího elementu *graph* s nepovinným atributem *name*, kam je možné vepsat název grafu. Také obsahuje nepovinný atribut *file* s názvem video-souboru, na kterém byl vstupní graf vygenerován.

Následuje definice uzlů a hran. Element *node* má atribut *nodeid* jednoznačně identifikující uzel v rámci grafu. Atribut *startframe* definuje, od jakého rámce ve video-sekvenci začíná šot. Délka šotu je určena atributem *length*. Tyto atributy jsou povinné. Hrany reprezentují elementy *line*. Atribut *start* obsahuje identifikátor *nodeid* jednoho s uzlů, ke kterému je hrana napojena, a atribut *end* obsahuje identifikátor druhého napojeného uzlu. Uzlové elementy obsahují popisek, který se zobrazí na zadní straně šotu. Definici končí uzavírací element *graph*.

Následuje příklad vstupního souboru:

```
<?xml version="1.0" encoding="UTF-8"?>
<graph name="Jméno grafu" file="sequence.avi" >
  <!-- UZLY -->
  <node nodeid="1" startframe="320" length="324" desc="prvni-uzel" />
  <node nodeid="2" startframe="567" length="432" />
  <node nodeid="3" startframe="866" length="456" />
  <node nodeid="4" startframe="234" length="768" desc="treti-uzel" />

  <!-- HRANY -->
  <line start="2" end="4" />
  <line start="3" end="4" description="nepovinný popisek" />
  <line start="2" end="3" />
  <line start="1" end="4" />
  <line start="3" end="1" />
</graph>
```

4 Realizace návrhu řešení

V této kapitole si projdeme podrobnosti vypracovaného řešení. Seznámíme se s knihovnami použitými při řešení. Vysvětlíme si, jak je vytvářena zobrazovaná scéna a jak je řešen pohyb v prostoru. Také bude zmíněno získávání snímků a přehrávání šotů.

4.1 Použité prostředky

Výsledkem projektu jsou komponenty napsané v C++ a využívající OpenGL pro zobrazování ve 3D. Tyto komponenty využívá demonstrační aplikace. Rozhodl jsem se používat následující knihovny pro řešení projektu:

- **Coin3D** knihovna slouží pro tvorbu aplikací ve 3D prostoru pomocí OpenGL. Používá Open Inventor API, které organizuje scénu do stromové struktury objektů a usnadňuje tak řadu činností v OpenGL. Tato knihovna je multiplatformní a je šířena zdarma.
- **Qt** knihovna je určena pro vytváření klasického uživatelského rozhraní. Původně měla její funkci zastat knihovna WxWidgets, ale Qt lépe spolupracuje s Coin3D. Je také distribuována zdarma a pro různé platformy.
- **Libavcodec** knihovna z balíku programů s názvem FFmpeg. Jedná se o open-source řešení pro kompresi a dekompresi řady multimediálních formátů. Je psána v jazyce C a je součástí mnoha existujících softwarových projektů.
- **TinyXML** je jednoduchá knihovna pro načítání a vytváření XML souborů. Pomocí ní se bude provádět načítání vstupních souborů s grafy.

Aplikace je primárně vyvíjena pro prostředí MS Windows, ale díky použitým knihovnám není problém je přesunout na prostředí Mac nebo Linux.

4.2 Reprezentace grafu

Třída *VBGraph* se stará o správu uzlů a vazeb mezi mini. Struktura *VBNode* reprezentuje uzel a struktura *VBLine* představuje vazbu mezi dvěma uzly. Třída *VBGraph* udržuje dynamická pole uzlů a hran.

Každý uzel je definován jednoznačným identifikátorem, počátečním snímkem, počtem snímků v daném shotu, volitelným popiskem a také číslem komponenty souvislosti. První čtyři složky mohou být načítány z XML popisu v souboru nebo náhodně generovány. Číslo komponenty vyjadřuje příslušnost ke komponentě souvislosti a musí se zjistit analýzou grafu.

Hrany mají ukazatele na první uzel, druhý uzel, popisek a také identifikátor komponenty souvislosti. Jedná se o neorientovaný graf, takže nezaleží na pořadí odkazovaných uzlů. Jenom je potřeba dbát na to, aby nevznikaly duplicitní hrany s obráceným pořadím koncových uzlů.

4.2.1 Načítání dat z XML souboru

Jak jsem již napsal, pro import dat ze souboru používám knihovnu TinyXML. Její použití je velmi snadné. Nejprve nalezneme kořenový element <*graph*>. Načteme z něj do třídy *Graph* atributy *file* a *name*. Dále postupujeme k definici uzlu. Procedura načte jednotlivé uzlové elementy a z jejich atributů zkopíruje hodnoty do vektoru uzlů. Stejný postup je u načítání hran. V případě chybného nebo neúplného souboru procedura vyhodí výjimku.

4.2.2 Generování náhodného grafu

Generování náhodného grafu potřebuje ke své funkci tři vstupy: maximální počet uzlů, maximální počet hran a vstupní video soubor. Postupně se vytvářejí jednotlivé uzly a jejich hodnoty. Hodnota počátečního snímku je samozřejmě vybrána s ohledem na počet snímků ve vstupním souboru. Taktéž délka šotu musí být v možnostech souboru. Následuje propojení vzniklých uzlů náhodným počtem vazeb. Před vložením náhodné vazby musíme ověřit, zda již v grafu není přítomna.

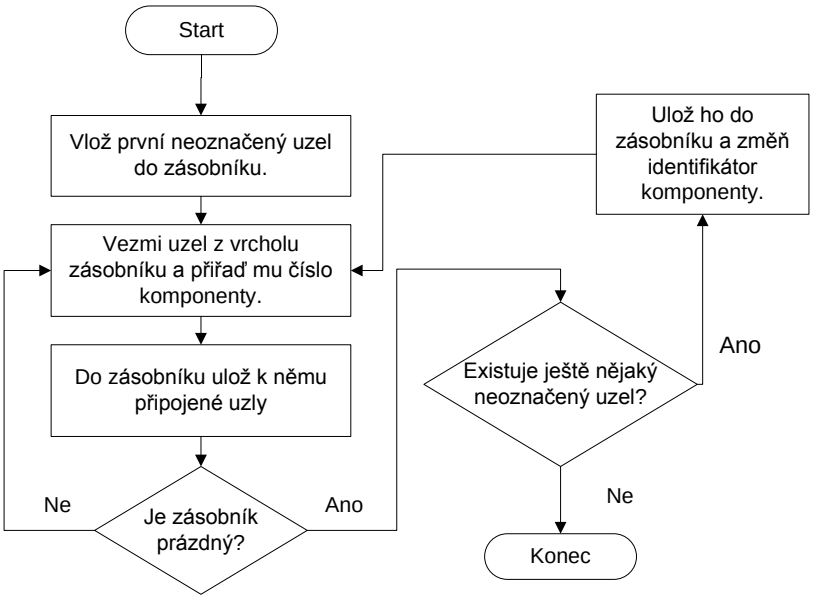
Volbou parametrů maximální počet uzlů a maximální počet hran můžeme volit přibližný počet komponent souvislosti. Pokud zvolíme větší počet uzlů než hran, vzniknou nám nejméně dvě komponenty v závislosti na rozdílu počtu hran a uzlů. Při poměru uzel-hrana 2:1 vzniká zpravidla jedna velká, několik menších a řada jednočlenných komponent souvislosti.

4.2.3 Zjištění komponent souvislosti

Pokud získáváme podobu grafu ze souboru, musíme prozkoumat jeho souvislost. Každý uzel má u sebe identifikátor komponenty. Při načtení ze souboru je tento identifikátor nastavený na -1.

Nejprve vezmeme libovolný neoznačený uzel a vložíme jej do zásobníku. V dalším kroku jej zase vybereme a přiřadíme mu identifikátor první komponenty. Do zásobníku vložíme k němu připojené neoznačené uzly. A opět vezmeme vršek zásobníku, označíme jej a připojené uzly vložíme na vrch zásobníku. Cyklus opakujeme, dokud není zásobník prázdný. Vyprázdnění zásobníku znamená, že byly nalezeny všechny uzly v jedné komponentě. Pokud ještě v grafu existuje nějaký neoznačený uzel, vložíme ho do zásobníku a inkrementujeme identifikátor komponenty. Toto je prováděno tak dlouho dokud, nejsou všechny uzly označené.

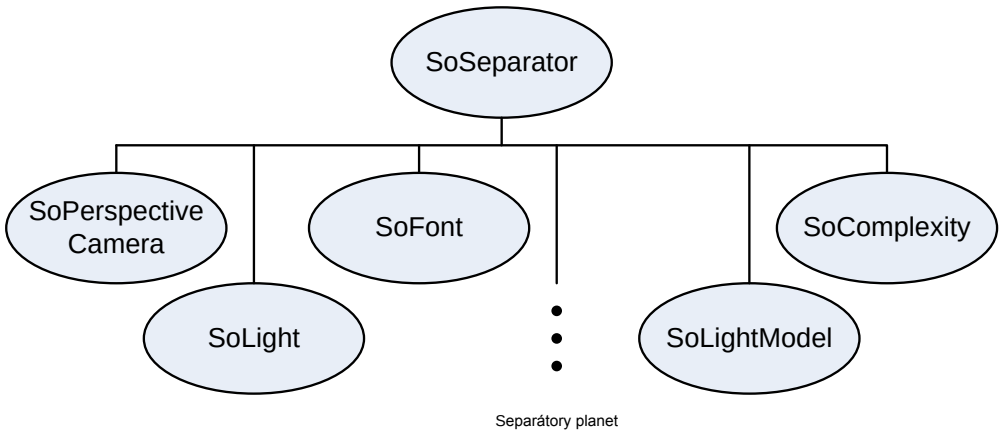
Identifikátory komponent doplníme také do hran podle komponent připojených uzlů.



Obrázek 15: Vývojový diagram nalezení komponent souvislosti

4.3 Zobrazování scény

V této kapitole se pojďme zaměřit na samotné vykreslování scény řešené pomocí Open Inventor API. Jedná se o objektové rozšíření OpenGL knihovny, kterou používá k zobrazování. Toto rozhraní používá svůj vlastní graf scény. Uzly grafu mohou°být různého typu, od geometrie těles, přes definice materiálů, textury, světla až po transformační uzly. Speciální typem uzlu je uzel *SoGroup*, který dokáže sdružovat další uzly a tak vytvářet stromovitý graf scény. Každé zanoření do nižších pater stromu má stejný efekt jako uložení transformačních matic do zásobníku.



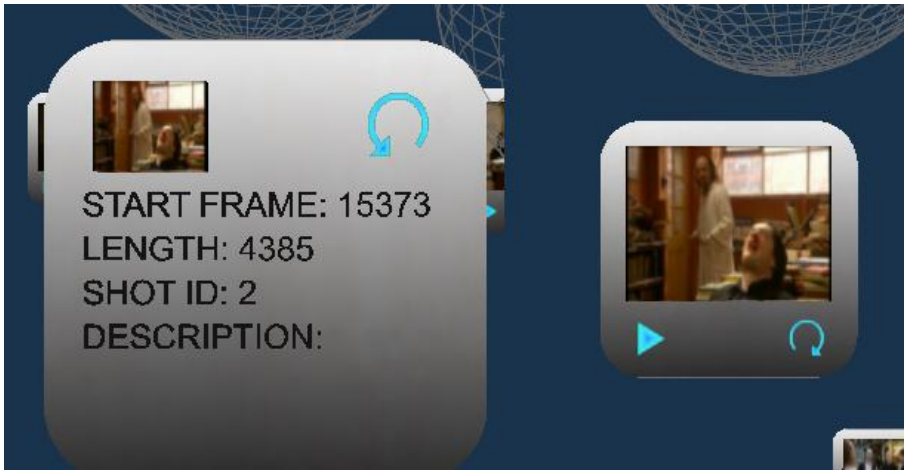
Obrázek 16: Schéma kořenového grafu scény v Open Inventor

Inicializace scény probíhá vytvořením kořenového uzlu, nastavením osvětlovacího modelu, definováním parametrů kamery, kvality zobrazování textur a nastavení písma. Kořenovému uzlu přidáváme postupně další potomky podle potřeby.

Abych zvýšil rychlost vykreslování, použil jsem pro planety a pro modely šotů dvojí reprezentaci. Jednu detailní, takovou jak je navržená výše, a druhou jednoduchou, zobrazovanou čistě jako obdélník s náhledem videa. O přepínání mezi jednoduchým a složitějším zobrazením se stará uzel *SoLevelOfDetail*. Tento uzel má jako potomky modely s rozdílnou složitostí a podle plochy, kterou zabírají na plátně, přepíná zobrazení vždy na jeden z nich.

4.3.1 Model šotu a jeho zobrazování

Šot je zobrazen jako panel, na kterém je několik tlačítek, plátno a popisky. Na přední straně panelu je plátno s tlačítky pro přehrání a otočení. Na zadní straně nalezneme malý náhled plátna, opět tlačítko na otočení a popisky.



Obrázek 17: Ukázka výsledné přední a zadní strany šotu

Model hlavního panelu a tlačítek je načítán ze souboru a můžeme ho nalézt v adresáři *models*. Na tyto modely jsou dále aplikovány textury, které se nachází v adresáři *maps*. Na plátnu je vidět náhled videa, který je získán z video-souboru a nanesen jako textura.

Text na zadní straně je model vzniklý protažením profilu písma do prostoru. Jedná se tedy o sadu 3D objektů umístěných na panel. Bylo nutné ošetřit zarovnávání textu tak, aby se vešel na desku šotu. Protože tento objekt je relativně bohatý na množství trojúhelníků, je přidán do scény pouze, pokud je panel otočený na rubovou stranu.

Abychom zjistili, zda uživatel kliknul na některé z tlačítek, použijeme třídu *SoRayPickAction*, která vyšle pomyslný paprsek z pozice kamery skrz bod na průmětně, kde uživatel stiskl tlačítko myši. Open Inventor prohledá graf scény a vrátí seznam uzlů scény, které představují cestu od kořenového uzlu, jenž protíná paprsek. Následně prohledáme seznam uzlů, a pokud, najdeme příslušné tlačítko a spustíme přiřazenou akci.

Animované otáčení je řešeno pomocí uzlu *SoEngine*, který umožňuje dynamicky měnit některé vlastnosti ostatních uzlů. V našem případě je jedná o uzel rotace, který je připojený na jednu z dvou

instancí uzlu *OneShotEngine*. Jedna instance obstarává rotaci na zadní stranu a druhá na přední stranu.

Přehrávání šotu probíhá jak na přední straně, tak na druhé straně díky tomu, že obě plátna sdílí stejnou texturu.

Další vlastností panelu šotu je, že se neustále natáčí tak, aby ho uživatel viděl vždy z přední strany. Toto je dosaženo propojením hodnoty jiného uzlu rotace a orientace kamery ve scéně. Pokaždé, když se změní orientace kamery, změní se i natočení šotu.

4.3.2 Zobrazování komponenty souvislosti

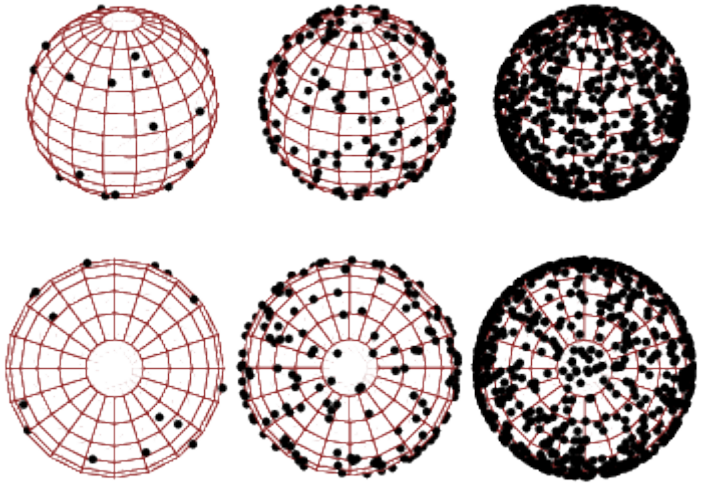
Komponenta souvislosti může obsahovat jeden nebo více navzájem propojených šotů. V případě jednočlenné komponenty je zobrazen pouze jeden šot v jejím středu. Rozsáhlejší podgrafy vidíme jako řadu šotů rozmístěných na povrchu drátěné koule. Jednotlivé vazby jsou znázorněny modrými čárkovanými úsečkami. Uvnitř planety se nachází observatoř, kterou symbolizuje žlutá krychle.
Šoty jsou na povrch koule umisťovány náhodně podle rovnic 1 až 3 převzatých z [9]. Na vstupu jsou dvě náhodně vybraná čísla x_1 a x_2 v intervalu (-1, 1). Čísla zároveň musí splňovat podmínku, že součet jejich druhých mocnin je menší než 1. Je použito rovnoměrné náhodné rozdělení.

$$x = 2x_1\sqrt{1 - x_1^2 - x_2^2} \tag{1}$$

$$y = 2x_2\sqrt{1 - x_1^2 - x_2^2} \tag{2}$$

$$z = 1 - 2(x_1^2 + x_2^2) \tag{3}$$

Výsledkem je náhodná souřadnice bodu na povrchu jednotkové koule. Pro naše účely musíme ještě vynásobit všechny souřadnice poloměrem planety.



Obrázek 18: Rozmístění bodů při počtu 100, 1000 a 5000 počátečních bodů [09]

Ještě předtím, než se generované souřadnice použijí, je potřeba ověřit, zda navrhované umístění není příliš blízko jinému šotu. Projdeme tedy souřadnice již vytvořených šotů, a pokud je blíže než jistá

mez, opakujeme generování náhodného bodu. Poloměr planety je kvůli přehlednosti nastaven tak, že je lineárně závislý na počtu šotů, aby nedošlo k „přehuštění“ planety.

4.3.3 Rozložení planet v prostoru

Málokdy se vyskytuje zobrazovaná scéna s jednou komponentou souvislosti. Proto je potřeba je organizovat do různých struktur. O těchto strukturách si povíme v následujících odstavcích.

4.3.3.1 Zobrazení do mřížky

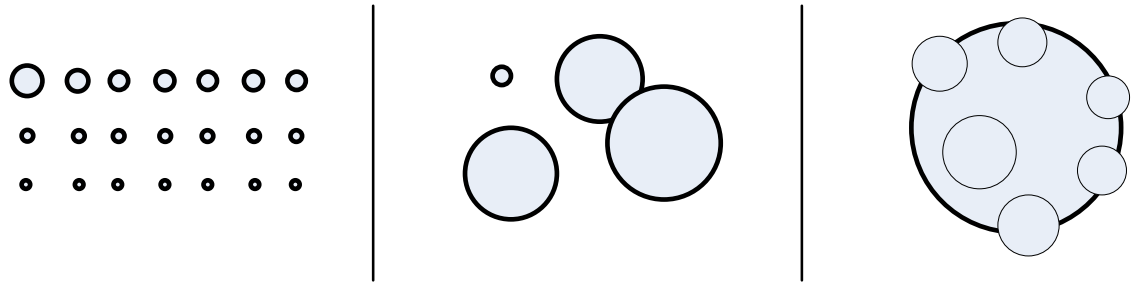
V tomto rozložení jsou planety seřazeny podle velikosti a poté zobrazeny do tabulky. Tabulka má fixní počet sloupců a proměnný počet řádků podle počtu komponent souvislosti. Je potřeba ověřovat, zda do sebe planety nezasahují a volit tak rozestupy podle největší z nich. Zobrazení do mřížky je vhodné pro velký počet malých komponent souvislosti. Třída *VBGridLayout* implementuje toto uspořádání.

4.3.3.2 Nahodilé zobrazení

Toto je velmi prosté zobrazení. Komponenty souvislosti jsou rozmístěny náhodně v prostoru. Testujeme, zda jedna nezasahuje do druhé. Toto zobrazení je vhodné zejména pro malý počet rozsáhlých komponent souvislosti. Tento druh uspořádání vytváří třída *VBModestLayout*.

4.3.3.3 Zobrazení v planetě

Pod tímto rozložením si můžeme představit jednotlivé planety nanesené na povrch jedné velké planety. Tato velká planeta má také svou observatoř. Lze vytvořit pomocí třídy *VBPlanetLayout*. Zobrazení v planetě je vhodné pro velký počet středně velkých planet.



Obrázek 19: Srovnání tří možností zobrazení: mřížkové (vlevo), nahodilé (uprostřed), planetové (vpravo)

4.3.4 Simulační smyčka

Animace objektů můžeme v Open Inventor API zajistit dvěma způsoby, buď pomocí *engines* a nebo definováním tzv. *callback* funkcí. *Engines* se hodí pro jednoduché animace bez možnosti velkých modifikací (např. otáčení, rotace a posouvání).

Definování *callback* funkce získáme větší svobodu v implementaci avšak za cenu většího objemu kódu. Tímto způsobem realizujeme přehrávání obsahu videa, pohyb kamery v prostředí (uživatelský nebo automatický) a také zobrazování počítadla snímků za vteřinu.

Přehrávání obsahu se spustí po kliknutí na příslušné tlačítko, čímž se nastaví globální ukazatel na texturu přehrávaného šotu. Poté se v pravidelných intervalech textura obnovuje na nové a nové snímky až do vyčerpání délky šotu. Přehrávat lze pouze jeden šot naráz. Při spuštění ukázky dalšího šotu se přehrávání předcházejícího šotu zastaví.

Simulační smyčku také využívá třída *VBCameraPoint*, která umožňuje uživateli volný pohyb v prostoru. Uživatel se pohybuje v prostředí jak pomocí myši a klávesnice, tak pomocí automatického přesouvání do blízkosti vybraného šotu. V třídě *VBCameraPoint* je uložena aktuální pozice, rychlost a směr pohybu. Tyto položky se mění v závislosti na čase a aktivitě uživatele. Pokud uživatel spustí cestování k určenému šotu, nastaví se vektor cesty, rychlost a příznak cestování. Při každém následujícím průběhu smyčkou je změněna poloha podle rychlosti a směrů. Dále se hlídá vzdálenost od šotu a v případě jejího dosažení se postupně snižuje rychlost tak, aby se kamera zastavila v přiměřené vzdálenosti od šotu. Cestování lze přerušit stiskem libovolné klávesy pro pohyb.

Další variantou cestování je cestování do observatoří. V tomto případě se kamera přesouvá přímo na pozici observatoře. Zde je potřeba ošetřit skrytí modelu observatoře, aby nebránila ve výhledu. Po jejím opuštění se model zase objeví.

Poslední funkcí, kterou simulační smyčka podporuje, je zobrazování počtu snímků za sekundu. Při každém průchodu se zvýší počet snímků o jeden a zároveň se sleduje čas, od kterého se počítání provádí. Pokud je čas větší než jedna sekunda, zobrazí se počet snímků a počítání běží od počátku.

4.4 Získání obrazových dat z video souboru

Pro práci s video soubory jsem se rozhodl použít knihovnu *libavcodec* z balíku *FFmpeg*. Podporuje řadu multimediálních formátů a kodeků. Je velmi výkonná a relativně jednoduchá na používání.

Proces získání požadovaného snímku ze souboru spočívá v otevření souboru, získání informací o datových proudech, zjištění použitého kodeku v daném proudu a následném čtení jednotlivých snímků. Pro celou scénu máme otevřený jeden soubor a při vytváření náhledů se přesuneme na dané místo snímku a získáme obraz. Tento obraz snímku se zkopíruje do uzlu textury plátna. Použitá textura je společná pro přední plátno i pro zadní náhled. Díky tomu přehrávání probíhá na obou stranách desky šotu.

5 Testy a jejich vyhodnocení

V této kapitole si uvedeme provedené testy a jejich výsledky. Nejprve jsem provedl měření rychlosti vykreslování a velikosti použité paměti. Následně jsem pomocí testů na několika dobrovolnících sledoval jejich schopnost pracovat s prostředím. Na následujících řádcích si provedeme rozbor jednotlivých testů.

5.1 Výkonnostní test

Měření výkonu probíhalo na 4 počítačových sestavách. V tabulce č. 1 vidíte podrobnější přehled naměřených hodnot. Na každé ze čtyř hardwarových konfigurací jsem zjistil hodnotu počtu snímků za vteřinu pomocí vestavěného počítadla. Hodnota zabrané operační paměti byla zjištěna pohledem na příslušnou kolonku ve správci úloh ve Windows.

Tabulka 1: Tabulka naměřených hodnot na různých konfiguracích a různě složitých scénách

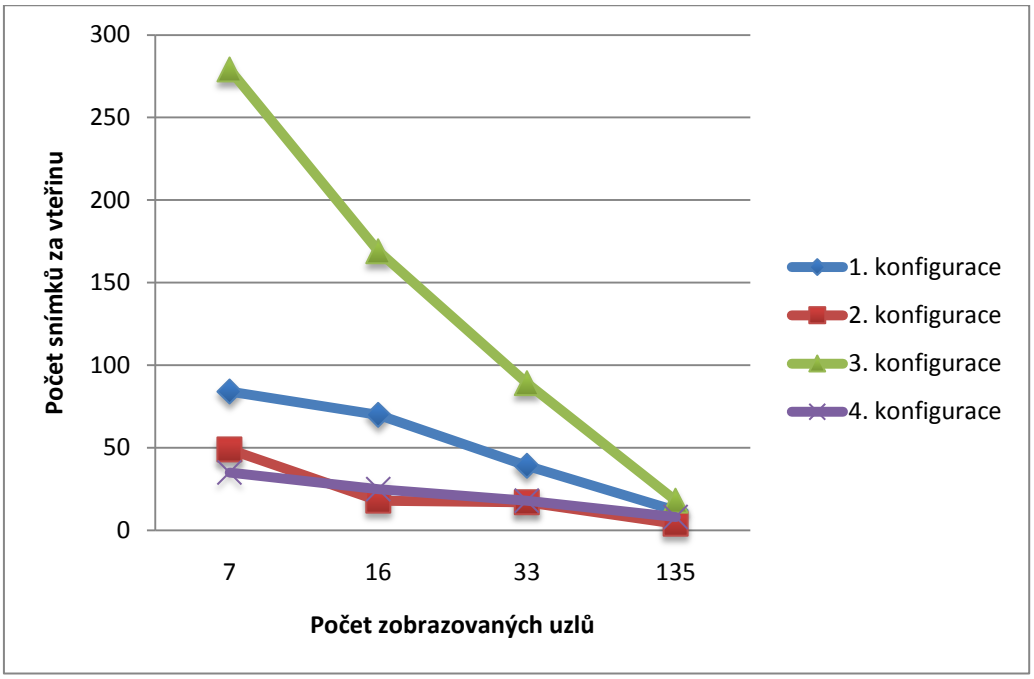
Počet šotů ve scéně	FPS	MEM	FPS	MEM	FPS	MEM	FPS	MEM
7	84	72	49	51	279	54	35	60
16	70	83	18	70	169	65	25	72
33	39	92	17	105	89	89	18	95
135	12	173	4	176	18	176	8	174
	1.konfgurace *Core Solo 1.8Ghz, 1,5 GB RAM, Intel 945 Express*		***2. konfigurace*** *Celeron M 1.2 Ghz, 225 MB RAM, int. Graf. karta intel*		***3. konfigurace*** *AMD Athlon 3000+, 1.8 GHz, 2.5 GB RAM, NVidia GeForce8200*		***4. konfigurace*** *Intel Core 2 Duo 1.5 GHz, 1GB RAM, Int. Graf. karta Intel 965 Express*	

Na výsledcích měření je vidět, že výkon klesá s rostoucí složitostí grafu, což je celkem logické. Na jedné konfiguraci se již při 30 zobrazovaných uzlech snížil výkon na 17 snímků/sek. a při 135 uzlech ve scéně se pohyboval okolo 4 snímků/sek, což je již silně za hranicí použitelnosti.

Velmi slabé hodnoty počítače č. 4 jsou překvapivé. Jedná se o průměrně výkonný stroj, ale změřené hodnoty snímků za vteřinu jsou podprůměrně. Lépe si vedly všechny ostatní sestavy, i když byly tabulkově slabší. Po krátkém zkoumání jsem zjistil, že na počítači nejsou správně nainstalované ovladače ke grafické kartě. Systém proto nepoužil hardwarové urychlení a vykreslování probíhalo softwarovou cestou přes procesor.

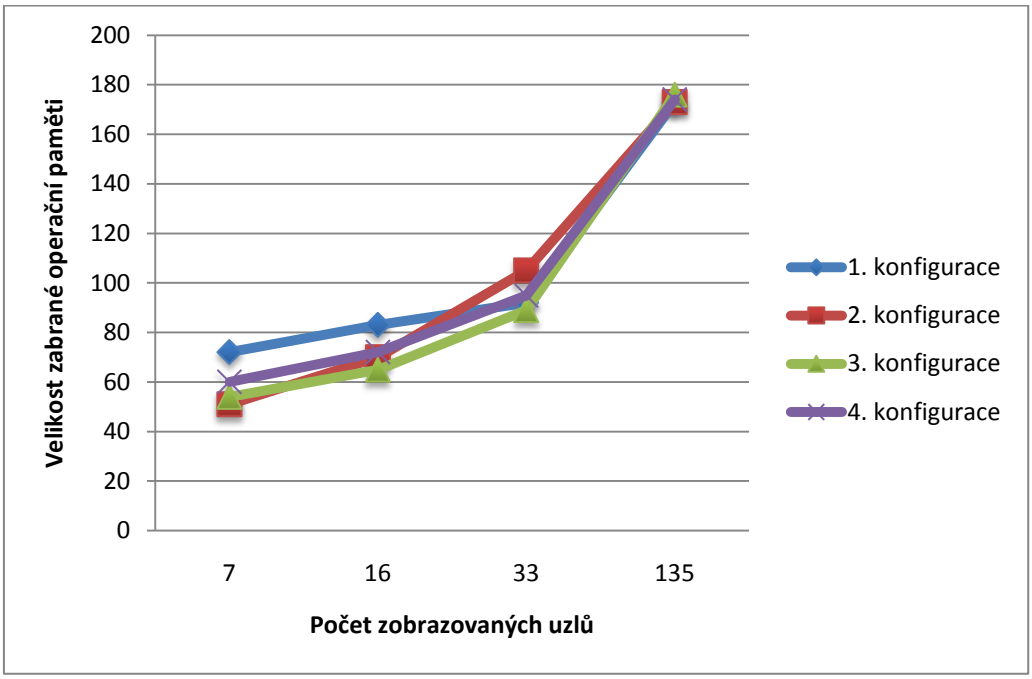
Měření u třetí konfigurace zkomplikovala synchronizace s obnovovací frekvencí monitoru. Při prvním měření se hodnoty vždy (až na nejnáročnější scénu) zastavily na 60 snímcích za sekundu. Po vypnutí vertikální synchronizace jsme již mohli získat skutečné hodnoty počtu snímků za vteřinu. Výsledky na sestavě č. 3 jsou bezkonkurenčně nejlepší. Může za to grafická karta NVIDIA GeForce

8200. Jako jediná v testu není integrovaná na základní desce. Pro scénu s deseti uzly se počet snímků pohyboval okolo 250, což je skoro 4krát více než druhý nejlepší výsledek.



Obrázek 20: Závislost rychlosti obnovování scény v závislosti na počtu zobrazovaných šotů

Z hlediska paměťové náročnosti si všechny sestavy vedly přibližně stejně. Z naměřených hodnot vyplívá, že testovací aplikace jako taková si v paměti vyhradí 50 až 60 MB a každý další šot (uzel) toto číslo zvýší přibližně o 1MB.



Obrázek 21: Závislost spotřebované operační paměti v MB

Z testů výkonnosti vyplývá, že aplikace je co do počtu snímků za sekundu nejpoužitelnější ve scénách do 30 uzlů. Ve větších scénách již není chod a ovládání tak plynulé. Paměťové nároky jsou celkem uspokojivé, nicméně prostor k úsporám paměti by se zřejmě ještě našel.

5.2 Testy použitelnosti

Po testech výkonu jsem provedl několik měření použitelnost. Testovací seance probíhala vždy s jedním člověkem sedícím za mým počítačem. Testující osoba plnila jednoduché úkoly ze seznamu a já jsem dohlížel, jak jednotlivé úkoly plní. Zároveň jsem stopoval čas, který potřeboval pro splnění každého z úkolů. Zapisoval jsem si, jaké používá prostředky (myš, klávesnici, automatické přibližování).

5.2.1 Skupina testujících lidí

Používání uživatelského rozhraní jsem testoval na 4 lidech z mého okolí. Všechno to jsou muži ve věku 23 až 27 let. Mají technické vzdělání a považují se za zkušené v práci s počítačem. Předpokládám, že podobné charakteristiky má i cílová skupina lidí, pro které je určen produkt mé práce.

5.2.2 Plněné úkoly

Nyní se podíváme podrobněji, jaké úkoly jsem si připravil pro dobrovolné testery. Nejprve jsem je posadil ke svému počítači, otevřel jsem jim adresář s aplikací a také jsem jim předložil níže uvedený seznam úkolů s úvodním odstavcem. Jazyk a oslovení v dotazníku byl zvolen, tak aby odpovídal charakteristice testujícího dobrovolníka. Figuranti se s uživatelským prostředím nikdy předtím nesetkali. Někdy bylo potřeba vysvětlit některý termín (např. šot, komponenta souvislosti apod.), aby dobře pochopili úlohu a mohli začít pracovat. Poznámky psané kurzívou jsou doplňující otázky, na které jsem hledal odpověď, zatímco dobrovolníci plnili jednotlivé úkoly.

Ahoj, vítej u mého testu použitelnosti. V následujících minutách se seznámíš s aplikací 3D Video Browser a sám si zkusíš některé triviální úlohy. Zajímá mě tvoje potýkání s aplikací a možnosti jejího vylepšení.

Nejprve si otevři adresář 3dvb-test a spusť program 3DVB.exe.

1. Otevři soubor test1.xml, vyber Grid layout a rozhlédni se po scéně, skus se otočit o 360 stupňů dokola.

Sledujeme, jak dlouho uživateli trvá, než zjistí jak se otočit. Sahá nejprve po myši nebo po klávesnici? Vypadá tato činnost jednoduše?

2. Zkus se co nejjednodušeji dostat do blízkosti nejvzdálenějšího šotu.

Využije možnost automatického cestování nebo se vydá po vlastní ose?

3. Dokážeš se podívat, co je na druhé straně šotu?

Pokusí se manuálně přejít na druhou stranu nebo už si všiml, že se automaticky otáčí a využije příslušné tlačítko?

4. Jaký je popisek šotu?

5. Zkus přehrát a zastavit přehráváni.

6. Nyní si otevři soubor test2.xml a vyber Modest layout. Vyber si největší planetu a zkus se dostat do jejího středu.

Používá klávesnici nebo myš?

7. Spočítej počet vazeb mezi šoty.
Sledujeme čas a zacházení s rotací. Zda vyjíždí z observatoře apod.

8. Nyní přejdi do libovolné sousední observatoře.

Použije automatické cestování nebo pojede ručně?

9. Otevři další soubor test3.xml a u něj vyber Planet layout. Navštiv observatoř ve středu zobrazení a spočítej počet komponent souvislosti.

10. Pokus se najít snímek, na kterém je záběr na kostru.

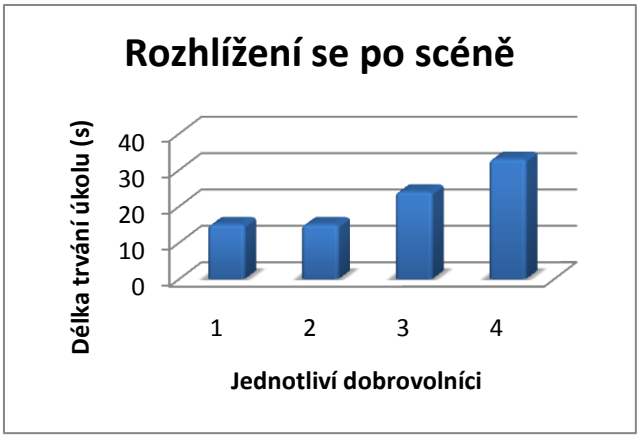
5.2.3 Výsledky a vyhodnocení

Nyní se pojďme podívat, jaké informace se nám podařilo získat při sledování figurantů při práci s mnou navrženým uživatelským rozhraním.

5.2.3.1 Úkol č. 1 – Rozhlížení se po scéně

V první úloze jsem sledoval, jak se subjekt postaví k prostředí, které nikdy neviděl. Někteří automaticky maximalizovali okno, aby viděli ze scény co nejvíce. Všichni použili nejdříve myš, aby změnili úhel pohledu na scénu. Po té zkoumají, zda fungují kurzorové klávesy a co umožňují. Průměrná doba rozhlížení byla cca 20s.

Návrh na zlepšení: Okno aplikace rovnou maximalizovat nebo roztáhnout na celou obrazovku.



Obrázek 22: Graf délky vypracování 1. úlohy

5.2.3.2 Úkol č. 2 – Cestování k vybranému šotu

Cílem této úlohy bylo zjistit, zda použijí k přiblížení automatické cestování, aniž bych jim o této možnosti řekl. To znamená, že museli experimentovat, aby se o takové vlastnosti dozvěděli. Všichni automaticky začali nejdříve používat klávesnici. Jeden na použití myši přišel sám a zbytek po nápovědě.

Průměrná doba cestování 8 vteřin.

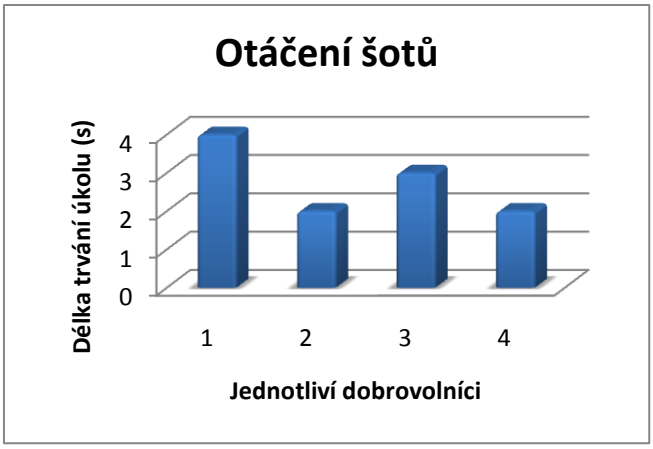
Návrh na zlepšení: Vložit do uživ. rozhraní efekt, který by po najetí myši na šot, názorně (ikona. piktogram, text v bublině) sdělil uživateli možnost přiblížení kliknutím myši.

Obrázek 23: Graf délky vypracování 2. úlohy

5.2.3.3 Úkol č. 3 – Otáčení šotů

Tímto úkolem jsem sledoval, zda uživatelé pochopí, že šoty se k nim automaticky natáčí čelem a není tedy možné vidět jejich druhou stranu jinak než kliknutím na příslušné tlačítko. Výsledek byl pozitivní, všichni bez výjimky použili tlačítko.

Průměrná doba plnění úkolu 3 sekundy.

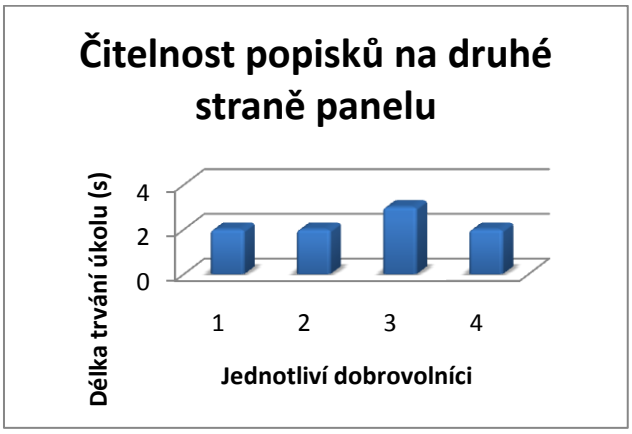


Obrázek 24: Graf délky vypracování 3. úlohy

5.2.3.4 Úkol č. 4 – Čitelnost popisků na druhé straně panelu šotu

Tato úloha měla ověřit čitelnost a přehlednost popisků na druhé straně šotu. Zde nenastaly žádné komplikace a všichni se dokázali orientovat v popiscích. Po krátkém vysvětlení chápali i význam jednotlivých položek.

Průměr délky plnění úkolu: 2s.



Obrázek 25: Graf délky vypracování 4. úlohy

5.2.3.5 Úkol č. 5 – Přehrávání video ukázek

Opět jedna z jednodušších úloh. Účastnici měli za úkol spustit přehrávání video ukázky. Vše proběhlo hladce a nezaznamenal jsem žádné potíže.

Průměrná doba splnění úkolu: 1s

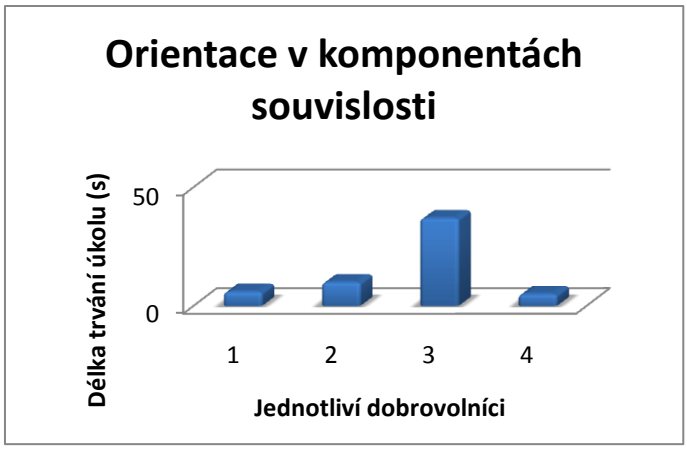
Obrázek 26: Graf délky vypracování 5. úlohy

5.2.3.6 Úkol č. 6 – Orientace v komponentách souvislosti (alias v planetách)

Úkol číslo šest byl navržen k otestování konceptu observatoře. Zajímalo mně, zda uživatelé přijdou na to, že střed planety je aktivní a lze do něj přejít automaticky. Ve výsledku dva lidé použili automatické cestování, protože je napadlo, že ona žlutá kostka (observatoř) bude aktivní podobně jako šoty. Zbylí dva se nejprve posouvali ručně pomocí klávesnice a po nápovědě použili observatoř.

Průměrná doba řešení: 14,5s

Návrh na zlepšení:Informovat uživatele o vlastnostech observatoře (podobně jako u šotů).



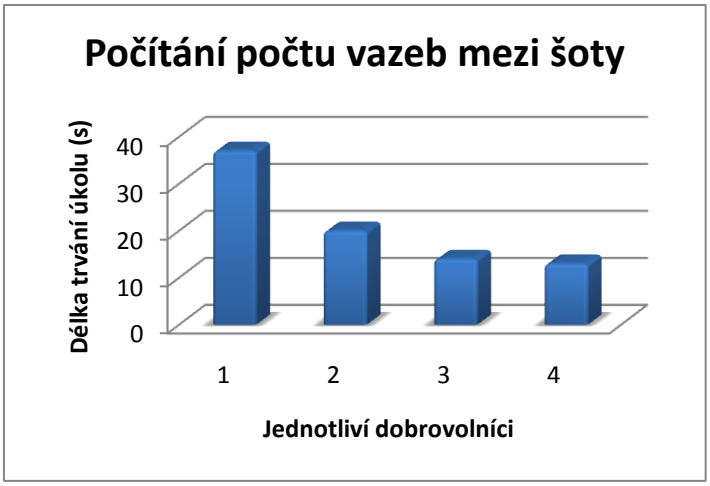
Obrázek 27: Graf délky vypracování 6. úlohy

5.2.3.7 Úkol č. 7 – Počítání počtu vazeb mezi šoty

Počítání vazeb mezi uzly slouží jako modelový příklad situace, kdy uživatel potřebuje přehled nad jednotlivými šoty v planetě. Při počítání uzlů nejprve musí najít uzly a pak spočítat jednotlivé vazby mezi nimi. Při tomto úkolu se všichni dobrovolníci rozhodli opustit observatoř a odstoupit od planety dostatečně daleko, aby viděli všechny uzly naráz. Počítání se tak obešlo bez neustálého manipulování s myší a s klávesnicí. V tomto ohledu se observatoř příliš neosvědčila jako užitečný koncept.

Průměrná doba počítání byla 21 sekund.

Návrh na zlepšení: Buď ještě více rozšířit šířku záběru při vstupu do observatoře, nebo doplnit pohled z observatoře o dodatečné zobrazování údajů (např. počet připojených uzlů apod.)

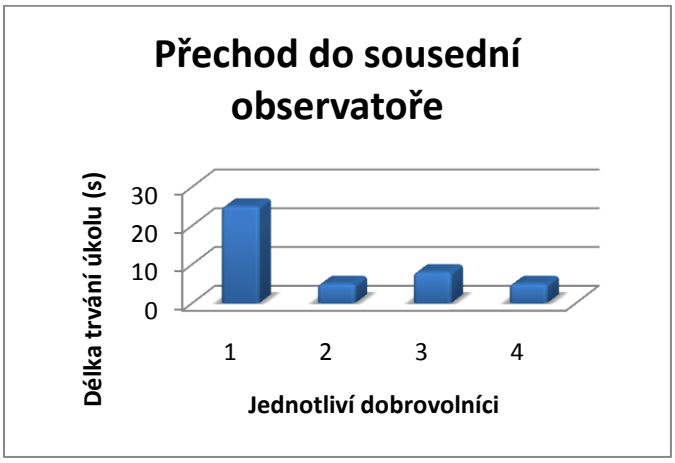


Obrázek 28: Graf délky vypracování 7. úlohy

5.2.3.8 Úkol č. 8 – Přechod do sousední observatoře

Osmá úloha testuje, zda uživatel použije automatické cestování mezi observatořemi nebo jestli pojede ručně pomocí kurzorových kláves. V podstatě všichni použili automatické cestování. To znamená, že si tuto funkci zapamatovali a rozhodli se ji používat.

Průměrná doba přechodu do sousední observatoře: 10s



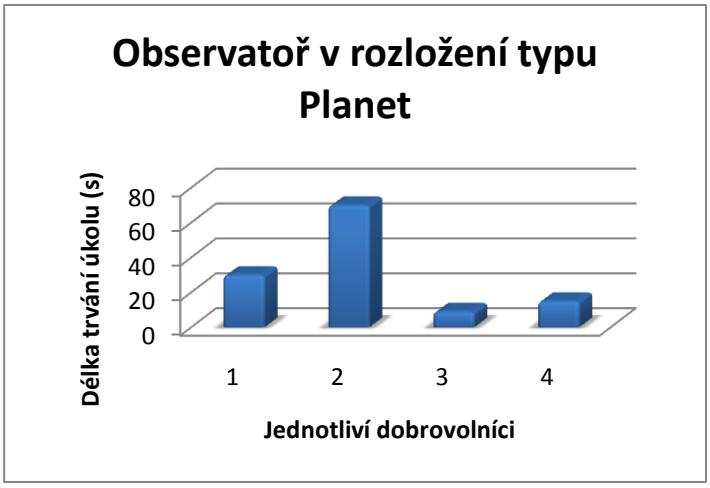
Obrázek 29: Graf délky vypracování 8. úlohy

5.2.3.9 Úkol č. 9 - Observatoř v rozložení typu Planet

Figuranti měli za úkol otevřít scénu v Planet rozložení a spočítat počet komponent souvislosti. Sledoval jsem, zda je jim nějak nápomocná observatoř ve středu zobrazení. Ani zde se neprojevila observatoř nijak zvlášť užitečná. Většina z dobrovolníků observatoř nakonec opustila a počítala komponenty z přehledné vzdálenosti od zobrazení.

Průměrná délka počítání byla 31 sekund.

Návrh na vylepšení rozhraní: Podobně jako u úkolu č. 7 zvětšit šířku záběru a doplnit zobrazení dodatečných informací.



Obrázek 30 Graf délky vypracování 9. úlohy

5.2.3.10 Úkol č. 10 – Hledání snímku s konkrétním obrázkem

Poslední činností bylo hledání konkrétního šotu v celém grafu. Figuranti hledali šot, na němž jsou zobrazeny kosti. Zajímalo mně, zda využijí observatoře nebo jiný prostředek manipulace s prostorem. Ve výsledku se raději spolehli na vlastní prostorovou představivost a hledali pomocí rotace myší a pohybem přes kurzorové klávesy.

Dobrovolníci nalezli daný šot v průměru za 69 vteřin.

Obrázek 31 Graf délky vypracování 10. úlohy

Výsledky testů ukázaly, že aplikace je vcelku ovladatelná a figuranti se s ní lehce seznámili. Nicméně stále existují prostory pro zlepšení. Zejména koncept observatoře se neukázal tak účinný při orientaci v prostoru. Z výkonového hlediska je na tom uživatelské rozhraní průměrně. Odezva systému je nejlepší při grafech do velikosti 30 uzlů.

6 Závěr

V tomto dokumentu je obsaženo studium základních principů 3D uživatelských rozhraní. Z uvedených zdrojů jsem vybral několik typů ovládání a interakce v prostorových prostředích. Byly zde nastíněny oblasti, ve kterých najdou prostorová uživatelská rozhraní svá uplatnění. Ve stručnosti jsem se také věnoval pojmu testování použitelnosti a jeho metodám.

Dále jsem provedl analýzu možných vstupních dat a na jejím základě jsem navrhl tři koncepty uživatelských rozhraní, které je prezentují. Každý z konceptů je specializován na určitou podobu vstupního grafu. Součástí návrhu je definice podoby vstupního textového souboru s popisem grafu.

V části věnované implementaci jsem se snažil postihnout všechny oblasti vývoje aplikace. Věnoval jsem se problematice vnitřní reprezentace grafu v programu, zjišťování komponent souvislosti, načítání definice grafu ze souboru apod. Dále jsem popsal použití Open Inventor API pro vytváření zobrazované scény a simulační smyčky. V závěru kapitoly je zmíněno získávání obrazových dat z video-souborů.

Výsledné rozhraní jsem otestoval jak z hlediska výkonu, tak z hlediska kvality uživatelského rozhraní. Využil jsem volného času několika dobrovolníků a pomocí jednoduchých testů jsem zjišťoval, schopnost prostředí dostát jejich požadavkům.

Výsledkem mé práce je implementace prostorového uživatelského rozhraní sloužící k prezentaci navzájem propojených video-ukázek. Prostředí může nabídnout tři druhy uspořádání uzlů ve scéně. Je možné ho použít také jako samostatnou zobrazovací komponentu v jiných aplikacích. Aplikace najde uplatnění všude tam, kde je potřeba vizualizovat síť k sobě navzájem připojených uzlů představující úseky videa. Jako další možnosti rozšíření bych si dokázal představit zobrazování více interaktivních nápomocných informací uživateli (např. mapa, více statistických údajů a navigačních údajů).

Literatura

[01] Bowman, D., A. , Kruijff, E., LaViola, J., J., Poupyrev, I., Presence, 2001, February, An Introduction to 3-D User Interface Design, str. 96, Dokument dostupný na URL http://people.cs.vt.edu/~bowman/papers/3dui_presence.pdf (květen 2009)

[02] BenHajji, F., Dybner, E., 3D Graphical User Interfaces, Department of Compuetr and Systems Sciences Stockholm University and The Royal Institute of Technology, 1999, Dokument dostupný na URL http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C97E41BE16ADBE3024826A3CE9C34F36?doi=10.1.1.103.5038&rep=rep1&type=pdf (květen 2009)

[03] Kuntz, S., A VR Geek Blog, 2008, ,Virtusphere review, Dokument dostupný na URL http://cb.nowan.net/blog/2008/05/22/virtusphere-review/ (květen 2009)

[04] Bowman, D., Evaluation of Virtual Grabbing/Manipulation Techniques, Dokument je dostupný na URL http://people.cs.vt.edu/~bowman/grab.html (květen 2009)

[05] Nielsen J., Coping with Information Overload, Dokument je dostupný na URL http://www.useit.com/papers/information_overload/ (květen 2009)

[06] Kepoh, AutoCAD Tutorial & Design Collection, Dokument je dostupný na URL http://autocad2u.blogspot.com/2006/08/what-is-autocad.html (květen 2009)

[07] Brdička, B., Víceuživatelské virtuální prostředí, Karlova Univerzita Pedagogická fakulta, 1999 Dokument je dostupný na URL http://it.pedf.cuni.cz/~bobr/MUVE/ (květen 2009)

[08] Mareš, M., Matoušek, D., Škoda, P., Grafy, Dokument je dostupný na URL http://ksp.mff.cuni.cz/tasks/18/cook2.html (květen 2009)

[09] Weisstein, Eric W., Sphere Point Picking, Dokument je dostupný na URL http://mathworld.wolfram.com/SpherePointPicking.html (květen 2009)

[10] Usability.gov, Learn About Usability Testing, Dokument je dostupný na URL http://www.usability.gov/refine/learnusa.html (květen 2009)

[11] Open Inventor Architecture Group, Open Inventor Mentor, Dokument je dostupný na URL http://www-evasion.imag.fr/~Francois.Faure/doc/inventorMentor/sgi_html/index.html (květen 2009)

[12] Open Inventor Architecture Group, Open Inventor C++ Reference Manual, Dokument je dostupný na URL http://techpubs.sgi.com/library/manuals/0000/860-0108-001/pdf/860-0108-001.pdf (květen 2009)

[13] Nicoline van Elten, Effective usability testing for Intranet, Dokument je dostupný na URL http://www.jungleminds.com/publications/articles/effective_usability_testing_for_intranet /(květen 2009)

Seznam Příloh

Příloha 1

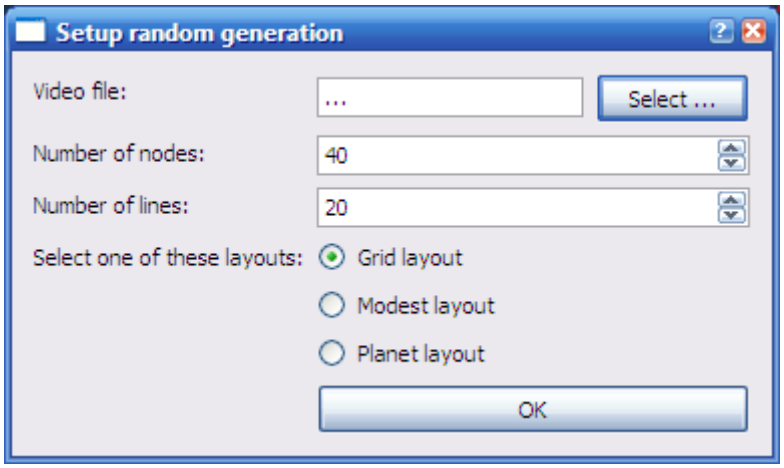
Uživatelský manuál ukázkové aplikace

Hlavní okno ukázkové aplikace se skládá z pruhu nabídek a zobrazovacího kontejneru. V pruhu nabídek nalezneme položky „Open“, „Generate random graph“ a „Exit“.

Otevření souboru a generování náhodného grafu

Volbou „Open“ otevřeme popis grafu v XML souboru. V popisu souboru je uvedena relativní cesta k video souboru.

Volbou „Generate random generation“ otevřeme dialogové okno, ve kterém vybereme soubor s videem, na kterém chceme vygenerovat graf. V tomto okně můžeme také nastavit maximum počtu uzlů a maximum počtu hran výsledného grafu. A nakonec si vybereme jednu z možností zobrazení scény.



Ovládání a pohyb v prostředí

Otáčení kamery v prostředí probíhá pomocí myši. Tažením myši se stisknutým jejím levým tlačítkem pootáčíme pohledem do prostředí.

Pomocí kurzorových kláves se můžeme pohybovat dopředu, dozadu a lze se otáček doleva a doprava.

Kliknutím na model šotu se přiblížíme do jeho blízkosti. Stisknutí modrých tlačítek na panelu lze spustit přehrávání ukázky, nebo otočit šot na druhou stranu. Na druhé straně se nachází popisky, malý náhled plátna a tlačítko pro otočení.

V každé planetě se nachází observatoř (viditelná jako žlutá kostka), do které se dá vstoupit a pozorovat tak planetu zevnitř. Pokud se uchýlíte do observatoře, aplikace vám rozšíří úhel záběru a máte tak, lepší přehled. Observatoř lze opustit šipkami dopředu a dozadu, vybráním nějakého šotu nebo jiné observatoře.

V planetovém rozložení je hlavní planeta obarvena červeně a vedlejší jsou ponechány v bílé. Observatoř hlavní planety má načervenalou barvu na rozdíl od ostatních, ale jinak je její chování stejné jako ostatních observatoří.

Příloha 2

Postup použití v jiném projektu

Potřebné knihovny a jazyky

Na přiloženém nosiči v adresáři „potrebne doplnky“ lze nalézt:

- balík FFmpeg zkompilovaný pro Windows platformu.
- Qt knihovnu pro uživatelská rozhraní
- Balík MinGW s kompilátory
- Coin3d, což je implementace Open Inventor API
- Knihovna SoQt – pro „propojení“ Qt a Coin3d
- Knihovna Simage – rozšiřující knihovnu Coin3d o načítání různých obrazových formátů ze souboru.

Postup vložení do existujícího projektu

- Do zdrojového kódu přidejte vkládání hlavičkového souboru „VBvideobrowser.h“.
- Přidejte do vašeho formuláře instanci komponenty VBSoQtRenderArea asi nějak takto:

```
renderWidget = new VBSoQtRenderArea(MainWindowClass);
renderWidget->setObjectName(QString::fromUtf8("newRenderArea"));
try {
      renderWidget->generateRandom("videosoubor.avi", 40, 10
,LAYOUT_MODEST_ID);
}
catch(runtime_error &e){
      cout << e.what() << endl;
}
MainWindowClass->setCentralWidget(renderWidget);
```

Volání metody *generateRandom* je obaleno blokem pro zpracování výjimek, kvůli možné chybě při zpracování video souboru. Pro načítání grafu z XML souboru použijte metodu: `renderWidget->loadFromFile("graf.xml", "/cesta/ksouboru/", LAYOUT_MODEST_ID);`

Parametry této metody jsou: název souboru, cesta k souboru a identifikátor kýženého rozložení. Jako identifikátory rozložení je možné použít `LAYOUT_MODEST_ID`, `LAYOUT_GRID_ID` nebo `LAYOUT_PLANET_ID`.

Postup kompilace

- Otevřete soubor *_DVB_qt.pro* a upravte cesty ke knihovnám tak, aby odpovídaly Vaším podmínkám.
- Zadáním příkazu *qmake* vytvořte soubory Makefile.
- Příkazem *make –f Makefile.Release* spustíte kompilaci projektu. Spustitelný soubor naleznete v podadresáři *Release.*

Příloha 3

Zápisky z testů použitelnosti

V této příloze jsou uvedeny moje zápisky z testovacích seancí, v tabulkách je uveden úkol, jeho přibližná doba provedení a poznámky, které jsem si zapsal při sledování figuranta.

Figurant Petr

Úkol č.:	Čas (s):	Poznámky:
1	15	bez váhání, použil klávesnice a myši
2	7	váhání mezi myší a klávesnicí, nepřesné ovládání pomocí klávesnice
3	4	bez váhání, použil tlačítko na otočení
4	2	ok
5	2	bez zaváhání
6	6	myší do observatoře
7	37	opouští observatoř, počítá ručně
8	25	nejprve klávesnice a pak objeví automat.
9	30	opouští observatoř, počítá ručně
10	23	bez observatoře

Figurant Josef

Úkol č.:	Čas (s):	Poznámky:
1	15	maximalizuje, používá myš a po nápovědě i klávesy
2	2	klávesnice, po nápovědě i myš
3	2	tlačítko bez váhání
4	2	ok
5	1	bez zaváhání
6	10	odvodil si, že to bude podobně jako s šoty
7	20	klávesnice
8	5	bez zaváhání
9	70	stěžuje si na pomalost - negativní emoce

10	10	nevyhovuje mu ovládání

Figurant František

Úkol č.:	Čas (s):	Poznámky:
1	24	myš, klávesnice, intuitivně
2	5	automatické přibližování - po nápovědě
3	3	bez váhání, tlačítka
4	3	ok
5	1	bez váhání
6	37	nejprve klávesy a poté, automatické cestování
7	14	klávesnice
8	8	automatické cestování
9	9	z observatoře, rotace myší
10	26	opouští observatoř

Figurant Pavel

Úkol č.:	Čas (s):	Poznámky:
1	33	maximalizuje okno, otáčí se myší
2	21	automatické cestování
3	2	tlačítka, bez váhání
4	2	ok
5	1	ok
6	5	ručně se přiblížil a observatoř
7	13	bez observatoře, klávesnice
8	5	automatický pohyb
9	15	automatické cestování
10	220	klávesnice a myš, chaotické otáčení