Jihočeská Univerzita v Českých Budějovicích
Přírodovědecká Fakulta

Hudební Vizualizér pro Virtuální Realitu

Bakalářská Práce

Vypracoval: Matěj Simota
Vedoucí Práce: Ing. Václav Novák CSc.
České Budějovice 2017

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: **Matěj Simota**
(jméno, příjmení, tituly)

Obor – zaměření studia: .....Aplikovaná informatika .....

Katedra/ústav, kde bude práce vypracovávána: .....UAI PrF JU .....

Školitel: ....Ing. Václav Novák, CSc.....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF:
.....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce:

Hudební vizualizér pro virtuální realitu

Cíle práce:

Cílem práce je vytvoření systému využívající virtuální reality, jež bude v přímé vazbě na poslouchanou hudbu. Touto cestou lze dosáhnou podstatné zvýšení intenzity prožitku či relaxace. V neposlední řadě očekáváme přínos i pro automatizaci. Student využije platformu macOS s iOS s použitím 3D editoru Cinema4D a v iOS-SceneKit. Pro 3D animací je nutno použít formát Collada firmy Sony. Seznámí se se základními rysy Hudební vizualizace tak, aby je mohl uplatnit v návrhu systému.

Prakticky:

1. Vytvoří funkční vizualizér. To je softwarový produkt umožňující vyzkoušet a otestovat virtuální realitu navrhnutou v teoretické části práce
2. Za použití plug-inu MoGraph pro vytvoření základových animací a použití vhodného plug-inu pro klíčové snímkování animací, vytvoří vhodné grafické prostředí.
3. Zajistí získání dat pro ovládání animací ze zvukových stop
4. Implementuje ovládání pomocí akcelerometru
5. Umožní vhodné rozmístění kamer ve scéně pro dosažení 3D efektu
6. Umožní pro deformace obrazu pro použití ve virtuální realitě s přibližovacími čočkami
7. Předpokládá se i uživatelská interakce
8. Vyhodnotí výsledky podle navržených testovacích scénářů.

Financování práce: .....

Vedoucí práce: .....podpis: .....

U externích vedoucích fakultní garant práce: .....podpis: .....

Garant oboru bak. studia, pokud je obor zajišťován jinou katedrou/ústavem, než ze které je školitel (nepožaduje se u oboru biologie): .....podpis: .....

Vedoucí katedry/ústavu, kde bude práce vypracovávána: .....podpis: .....

Případný souhlas vedoucího ústavu AV: .....podpis: .....

V Českých Budějovicích dne 4. 1. 2017 Podpis studenta: .....

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě – v úpravě vzniklé vypuštěním vyznačených částí archivovaných pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 17.dubna 2017 Matěj Simota

Poděkování

Rád bych poděkoval panu Ing. Václavu Novákovi CSc. Za průběžné konzultace a rady ohledně mojí bakalářské práce. Také za výuku předmětu Programování pro iPad/iPhone, které mě nasměrovalo k tomuto tématu.

Simota, M., 2017: Hudební vizualizér pro virtuální realitu [Music visualiser for virtual reality. Bc. Thesis, in Czech] – 33p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Bakalářská práce „Hudební vizualizér pro virtuální realitu“ se zabývá teoretickým základem k vývoji mobilní aplikace a jejím samotným vývojem. V teoretickém základě popisuje definici a technologie pro interpretaci virtuální reality. Dále pojednání o software a hardware použitém k realizaci aplikace. Před samotným návrhem aplikace také zkoumání principů zvukového signálu a hudebních vizualizací. V praktické části je popsán postup vytvoření 3D prostředí a jeho přenos do prostředí frameworku SceneKit. Následně principy, které byly použity pro naprogramování reakce této 3D scény na spouštěnou hudbu, ovládání pomocí pohybu hlavy uživatele, jednotlivé komponenty aplikace, testování a optimalizace.

Abstract

Thesis „Music visualiser for virtual reality“ deals with theoretical basis of developing mobile app and the actual development. The theoretical basis describes the definition and interpretation technology for virtual reality. Further discussion of software and hardware used to implement the application. Also examining the principles of sound signal and music visualization and then designing the app. The practical part describes how to create 3D environment and how to transfer it into the environment SceneKit framework. Consequently, the principles of which were used for programming the response of the 3D scene to playing music via head movement of the user, testing and optimization.

Keywords

Apple, iOS, app, music, visualiser, swift, xcode, cinema, 4D, 3D, virtual, reality

Obsah

1. Úvod

Virtuální realita je v dnešní době nový směr technologie. Její využití je široké, nicméně největší potenciál má zatím v herním průmyslu. Objevování nových způsobů, jak tuto technologii využít je určitě velmi důležité. Jedním z nich by se mohl stát i poslech hudby v kombinaci s hudebním vizualizérem.

Cena headsetů interpretujících virtuální realitu uživateli je velmi vysoká. Existuje však cesta, kterou lze zpřístupnit VR téměž každému. Tou je použití mobilního headsetu. Chytré telefony lze použít pro interpretaci virtuální reality, pokud si k nim uživatel zakoupí cenově dostupný headset. Technologie SceneKit umožňuje vytvoření 3D prostředí aplikace pro operační systém iOS. Jazyk Swift obsahuje prostředky pro vytvoření aplikace simulující virtuální realitu. Vzhledem k povaze samotné aplikace je důležité, aby si ji mohl každý vyzkoušet, a tak volba mobilní virtuální reality je vhodná právě pro tuto práci.

Práce obsahuje teoretický základ, který zahrnuje pojednání o virtuální realitě, současných prostředcích pro její interpretaci a předpoklady pro vývoj aplikace pro mobilní zařízení s operačním systémem iOS. Dále obsahuje dokumentaci k vývoji a realizaci samotné aplikace hudebního vizualizéru.

2. Cíle Práce

Cílem práce je vytvoření systému využívající virtuální reality, jež bude v přímé vazbě na poslouchanou hudbu. Touto cestou lze dosáhnout podstatné zvýšení intenzity prožitku či relaxace. V neposlední řadě očekáváme přínos i pro automatizaci. Student využije platformu macOS a iOS s použitím 3D editoru Cinema 4D a v iOS – SceneKit. Pro 3D animaci je nutno použít formát Collada firmy Sony. Seznámí se se základními rysy hudební vizualizace tak, aby je mohl uplatnit v návrhu systému.

3. Virtuální Realita

3.1 Definice a Využití

Virtuální realita ve své podstatě představuje alternativní svět, který se více či méně může podobat tomu skutečnému nebo může být zcela fiktivní. Je to tedy přímo ten nehmatatelný smyšlený svět, do kterého se pomocí prostředků pro interpretaci virtuální reality snaží uživatel vstoupit. Tento pojem je v dnešní době spojován právě s těmito prostředky, tedy ve většině případů s audiovizuálními brýlemi nebo jinými komplexnějšími systémy. I když se tedy může zdát, že virtuální realita je nový, dosud moc neobjevený směr technologie, není tomu tak. Alternativní reality existují od doby, kdy vznikla první počítačová, konzolová nebo jiná hra, která umožňuje uživateli interakci s okolním prostředím a různými způsoby na tuto interakci reaguje a poskytuje uživateli zpětnou vazbu. Prostředky umožňující zobrazení a interakci uživatele v alternativní realitě mohou být tedy i monitor, klávesnice a myš, případně gamepad.

I přesto že pojem virtuální realita není úplně přesným označením soudobých technologií, v této práci bude pro zachování jednoduchosti a srozumitelnosti nadále tento pojem a jeho zkratka „VR“ také používán ve spojení s obecným označením interpretačních prostředků, kterými se zabývá následující kapitola.

Již v padesátých letech 19. století Morton Leonard Heilig vytvořil hru, která by se dala považovat za virtuální realitu, podobnou té dnešní. Šlo o hrací automat, kde hráč řídil motorku, cítil na obličeji vítr, vibrace, a dokonce i vůně simulující prostředí města. Určit kde je hranice toho, co se dá považovat za virtuální realitu a co už ne, je velmi těžký úkol a záleží spíše na subjektivním názoru.

V ideálním případě by do systému interpretující virtuální realitu patřily prostředky pro simulaci vizuálního, sluchového, hmatového i čichového dojmu. Celý proces vývoje technologií zabývajících se tímto tématem se snaží docílit toho, aby měl uživatel pocit, že se fyzicky přenesl do jiné, alternativní reality. Realizace zařízení, které by něco takového umožňovalo se ještě před pár lety zdála neuskutečnitelná. Nyní vývoj různých produktů pro simulaci lidských vjemů nabral na rychlosti. To, co se tehdy zdálo nemožným, je teď o dost blíže ke skutečnosti.

Obr.1 Virtuální realita v 19. století

Dnes stojíme u zrodu tohoto odvětví a teprve objevujeme jeho potenciál a různé možnosti využití, které nabývají širokých rozměrů. Jedním z těch méně známých je vzdělávání, případně trénink. Tímto způsobem využívá virtuální realitu především armáda, která pomocí audiovizuálních brýlí a speciálně upraveného vybavení simuluje prostředí boje, pilotování letadla nebo dokonce seskok padákem. Výhody takového tréninku jsou zřejmé. Jedná se především o bezpečnost a odstínění následků spojených s chybami trénovaných vojáků, které by mohli nastat ve skutečném světě. Dalším zajímavým způsobem, jak uplatnit VR v komerční sféře je architektura a návrhy interiérů. Zákazník si pomocí brýlí, se kterými se může v počítačovém návrhu rozhlížet, lépe představí, jak bude výsledný interiér vypadat ve skutečnosti. Stejně tak VR technologii využívají velké společnosti prodávající nábytek a umožňují svým zákazníkům pohled na jejich produkty v alternativní realitě z pohodlí domova. Ovšem to, co bezpochyby tlačí vývoj dopředu, je právě zábavní průmysl, hlavně počítačové a konzolové hry a v neposlední řadě i filmy točené speciálními kamerami.

Virtuální realita (VR) (nebo virtuální prostředí) je technologie umožňující uživateli interagovat se simulovaným prostředím. Technologie virtuální reality vytvářejí iluzi skutečného světa (např. při výcviku boje, pilotování, lékařství), nebo fiktivního světa

počítačových her. Jde o vytváření vizuálního, sluchového, hmatového či jiného zážitku budícího subjektivní dojem skutečnosti pomocí zobrazovacího zařízení počítače, speciální audiovizuální helmy, brýlí atd., popř. oblečení snímajícího pohyb a stimulujícího hmat nebo jiné vjemy vyvolávající techniky.[1]

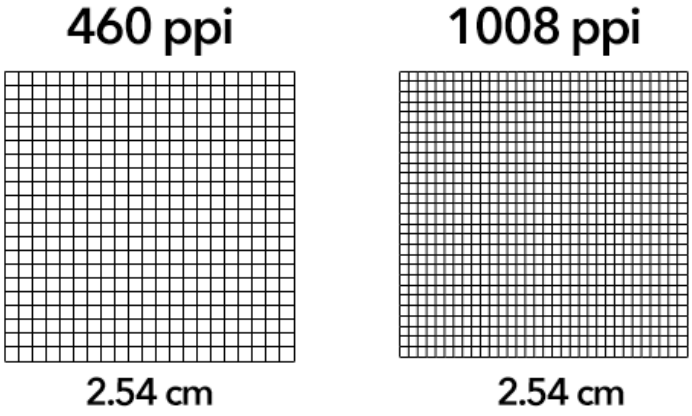
3.2 Současné Prostředky pro Interpretaci a Interakci VR

3.2.1 Smyslové Vjemy

V současné době několik velkých společností vyvíjí a prodávají speciální audiovizuální headset pro interpretaci virtuální reality. Vzhledem k tomu že sluchem, a hlavně zrakem přijímá člověk nejvíce okolních vjemů, nese tento produkt velmi důležitou roli. Vizuální a sluchová část interpretace se tedy stává na poli VR jakýmsi standardem.

Jednotlivé headsety různých výrobců se liší svými technickými parametry a také cenou. Majoritními výrobci jsou Sony, HTC a Oculus, zakoupený společností Facebook Inc, ještě před prvním vydáním brýlí Rift. HTC a Oculus prodávají brýle pro platformu PC, zatímco Sony pouze pro svou konzoli PlayStation. To může být důvodem menšího rozlišení zobrazovacích panelů, vzhledem k absenci konkurence. Tento fakt se ovšem projevuje také výrazně nižší cenou oproti ostatním produktům.

Výše zmiňovaný headset se tedy obecně skládá ze dvou obrazovek a sluchátek. Tyto obrazovky mají velmi vysoké hodnoty PPI. Je to logické, pokud se člověk dívá na monitor, který je položen na stole před ním, je pro něj velmi těžké rozeznat rozdíl, když jemnost



Obr.2 Orientační zobrazení PPI

zobrazovacího panelu přesáhne hranici 500 PPI. U VR headsetu je to úplně jinak. Obrazovku má uživatel velmi blízko obličeje a zobrazení obsáhne celé jeho zorné pole. Ideální obrazovka by musela mít přibližně 1000 PPI, pro ten nejlepší zážitek z VR. Pro srovnání, Oculus Rift, stejně tak jako HTC Vive mají hodnotu PPI zhruba 460. U další generace headsetu by se tato hodnota mohla zvětšit více než dvojnásobně. Společnost Sharp vydala zprávu, že pracuje na OLED obrazovkách určených pro VR, které dosahují hodnoty PPI až 1008.

Každá z obrazovek zobrazuje virtuální svět z trochu jiného úhlu a tím dosahuje 3D efektu. Na stejném principu fungují i 3D televize, kdy uživatel každým okem vidí také jiný obraz, díky speciálním polarizovaným brýlím. Člověk rozeznává vzdálenost objektů právě díky tomu, že má dvě oči a objekty vidí každým z jiného úhlu, tuto informaci mozek vyhodnotí a vytváří představu trojrozměrného prostoru. Dokazuje to jednoduchý pokus, kdy člověk se zavřeným jedním okem se snaží procházet místnost, ve které nikdy dříve nebyl a má problémy s naražením do nábytku a nejistotou vzdálenosti zdí.

Jedním z věcí, které jsou nezbytnou součástí audiovizuálního headsetu, jsou externí motion trackery. Odpověď na otázku, proč není pohyb snímán akcelerometrem a kompasem, je celkem jednoduchá. A to chybovost, akcelerometry sice fungují dobře, ale nejsou dokonalé. Externí zařízení zajišťující snímaní pohybu, jsou mnohem přesnější a ani po dlouhé době používání nemůže dojít k výpočetní chybě nebo posunu nulového bodu. Headset má na sobě mnoho bodů, které emitují infra záření, motion tracker umístěný na stole, v případě Oculus Rift, nebo v rozích místnosti, v případě HTC Vive, toto záření snímají a vyhodnocují pozici a náklon hlavy uživatele. Hýbáním hlavy neovlivňuje uživatel pouze obraz, ale také zvuk ve sluchátkách pro docílení opravdového dojmu 3D zvuku.

Obr.3 HTC Vive, Oculus Rift, PlayStation VR

Zatím bohužel neexistují rukavice, které by simulovali dotýkaní se různých objektů v alternativní realitě, i když by to jistě představovalo obrovský pokrok v celém odvětví virtuální reality. Takzvané Data Gloves, neboli datové rukavice, umožňují pouze snímání pohybů ruky. Jsou tedy vstupním, nikoliv výstupním zařízením. Využívají je zejména grafici pro vytváření animací, na poli VR se spíše stále používají herní ovladače, které mají alespoň vibrace, které sice nejsou žádnou převratnou novinkou, i přesto se jedná o hmatovou odezvu. Zajímavým projektem se zdá být také oblek s názvem Teslasuit, který slibuje haptickou odezvu po celém těle uživatele. Jeho pořizovací cena se odhaduje na přibližně 40.000,- Kč. Datum zahájení prodeje není zatím známý.

3.2.2 Pohyb ve Virtuálním Světě

Vyřešení pohybu uživatele ve virtuální realitě, tak aby měl dojem, že na daném místě opravdu je, je problém, který asi nebude nikdy úplně vyřešen. Existuje několik možností, jak docílit alespoň částečného řešení, přičemž záleží na způsobu a prostoru, v jakém se bude člověk pohybovat. Možností je několik.

V současné době nejlepší dojem může zažít člověk, který v alternativní realitě jezdí pouze autem. Ve virtuálním prostoru se pohybuje pouze za pomocí volantu a pedálů, ve skutečném světě tedy sedí v sedadle a je stále na stejném místě. Vyskytuje se tu však problém. Tím je setrvačná síla, která na skutečného řidiče působí při prudkém zrychlení, zabrzdění

Obr.4 Cyberith Virtualizer

i v zatáčkách. I tento aspekt lze nasimulovat pomocí některého ze závodních simulátorů. V případě simulátoru od značky 4DOF lze dosáhnout odezvy o síle přetížení až 2G.

V případě volného pohybu, tedy chození a běhání, existuje zařízení, díky kterému může uživatel běhat do všech směrů, a přesto zůstat na místě. Toto zařízení vyvíjí společnost Cyberith. Člověk, který se chce pomocí Cyberithu přenést do alternativní reality, si musí nasadit speciální boty, které mají minimální tření k podložce, na které běhá a v podstatě po ní nohama klouže. Tato deska pak snímá jeho pohyb. Okolo pasu má připevněný kruh, který zase snímá pokrčení nebo naopak vzpřímení. I v tomto případě je zde mnoho nevyřešených aspektů, které se liší od skutečného světa. Při naražení do zdi, může uživatel běžet stále dál, v ruce má stále stejnou zbraň, pokud si sedne do auta musí vyměnit zbraň za gamepad a tak dále.

Všechny zmíněná zařízení jsou jednak velmi drahá a jednak velmi prostorově náročná. Běžný uživatel si tedy bude muset, při hraní her ve VR, stále vystačit s gamepadem, tak jako tomu bylo doposud. Má to tu výhodu, že gamepad je svým způsobem univerzální, tak jak se vyvíjel v historii pro hraní všech různých žánrů her. Jednotlivé společnosti prodávající audiovizuální helmy, dodávají ke svým zařízením také své vlastní speciální gamepady. Tím nejzajímavějším se zda být gamepad společnosti Oculus, který se snaží docílit dojmu že virtuální ruce jsou vaše skutečné. Kromě toho dovoluje uživateli interakci s objekty a také rozpoznává gesta ruky. Například mávání, zvedaní palce a podobně.

3.2.3 Headsety pro chytré telefony

V předchozích kapitolách zmiňované headsety, které jsou kompletním zařízením pro virtuální realitu nejsou jedinou možností, jak obraz interpretovat uživateli. Téměř každý z nás nosí stále u sebe zařízení, které má veškeré prvky a funkce, které jsou třeba pro sestrojení podobného headsetu. Tím zařízením je chytrý mobilní telefon. Má obrazovku, která u dražších modelů dosahuje relativně dobré hodnoty PPI, akcelerometr a kompas. Na trhu existují headsety, do kterých se mobilní telefon vloží a audiovizuální helma je hotová.

Tyto headsety pro mobilní telefony lze rozdělit do dvou základních kategorií. Jedny z nich jsou papírové a druhé z umělé hmoty. Papírové se samozřejmě vyznačují výrazně nižší cenou, nicméně nejsou tak komfortní. Dražší zařízení poskytují větší pohodlí, nemusí se

u obličeje držet rukama a některé z nich jsou dodávány s bezdrátovým dálkovým ovládáním.

Jedním z těchto produktů je i headset VR Box 2.0, který bude použit v praktické části této práce, spolu s mobilním zařízením iPhone 6 od společnosti Apple. S přihlédnutím k faktu, že tato práce byla vytvořena pro zkoumání nového směru využití virtuální reality, je mobilní headset vhodnější variantou. Může si ho dovolit každý, alespoň ten papírový a aplikaci si vyzkoušet.

3.2.4 Omnidirectional Camera

Omnidirectional camera je záznamové zařízení, které zachycuje obraz všude okolo sebe, tedy jeho zorný úhel je 360°. To lze využít pro natáčení filmů pro virtuální realitu. Při následné správné transformaci obrazu a příslušným softwarem, se lze díky audiovizuální helmě ve snímku rozhlížet okolo sebe.

I když zatím VR headset nemá doma každý, na sociálních sítích jako je Facebook nebo na serveru Youtube se již objevují videa natočená touto technologií. Je možné se v nich rozhlížet i pomocí myši. Nejčastěji je využívají velké společnosti k natáčení svých reklamních kampaní.

3.3 Kompatibilita s MacOS a iOS

Zmíněné audiovizuální helmy nejsou zatím kompatibilní s operačním systémem macOS. Výrobci Oculus Rift uvádějí, že žádný z nabízených notebooků značky Apple, nesplňuje doporučené požadavky pro VR. Týká se to zejména grafických karet, které jsou v počítačích zabudovány. MacBooky jsou všeobecně známy jako pracovní stanice. Podpora titulů na herním trhu také není velká a virtuální realita zatím nemá v jiných odvětvích tak velké využití.

Změnu by mohl přinést cloud systém od společnosti nVidia, který slibuje vysoký výkon ze vzdálených počítačů a následné přenášení obrazu k uživateli. Pokud by tato technologie uspěla, mohli by se i uživatelé macOS dočkat zážitků z her ve virtuální realitě. Stálým předpokladem je budoucí podpora macOS ze strany výrobců VR brýlí.

Jediná možnost pro uživatele zařízení Apple tedy zatím stále zůstává použití mobilních headsetů s jejich mobilními telefony.

With PC gamers enjoying the likes of the HTC Vive and Oculus Rift VR headsets, many Mac users are left wondering "Can I use VR on a Mac?" and simply put, probably not. The founder of Oculus Palmer Luckey has said on the record several times that the Rift is "not going to work on any MacBook that exists or is known to exist in the near future" because of one simple reason – graphics cards. During an IGN Live interview at E3 2015, Luckey remarked "People have said, 'Why don't you support Macs? So many people have Macs.' It's true. A lot of people have Apple hardware, especially in the laptop space. But the GPUs in those, they're not even close to what we're pushing for our recommended spec." [2]

4. Cinema 4D a SceneKit

Cinema 4D je software pro vytváření 3D grafiky. Obsahuje v sobě jak nástroje pro modelování, materiály a nasvětlení tak i svůj vlastní render engine. V této práci bude tento software použit zejména pro modelování a animaci objektů, které budou následně použity v aplikaci pro hudební vizualizaci. Na rozdíl od konkurenčního softwaru je Cinema 4D uživatelsky přívětivá. V tom se trochu podobá programovacímu jazyk Swift a jeho technologii SceneKit.

SceneKit je API, vytvořené společností Apple, které umožňuje relativně jednoduché vytváření 3D her pro iOS a macOS. Pro iOS je dostupný od roku 2014. Obsahuje v sobě vlastní physics engine, částicové systémy a 3D objekty vytvořené v externích programech do něj lze importovat ve formátu Collada, kterému se věnuje jedna z následujících podkapitol. Nástroje v XCode usnadňují práci s tímto frameworkem.. Nabízí editaci scén, vkládání objektů, a dokonce i editaci materiálů.

SceneKit combines a high-performance rendering engine with a descriptive API for import, manipulation, and rendering of 3D assets. Unlike lower-level APIs such as Metal and OpenGL that require you to implement in precise detail the rendering algorithms that display a scene, SceneKit only requires descriptions of your scene's contents and the actions or animations you want it to perform.[3]

4.1 Scene Graph

Scene graph je pojem, který má Cinema 4D a SceneKit společné, trochu se od sebe ale liší. V Cinema 4D se dá za scene graph považovat stromová hiearchie, která je zobrazena v sekci správce objektů. Můžeme zde vidět všechny objekty, světla, kamery a další elementy, které se nacházejí ve scéně, včetně jejich vlastností. SceneKit takovou hiearchii nemá zobrazenou v žádném správci, ale při programování umístění objektů do scény vlastně takovou hiearchii programátor vytváří pomocí objektů a nodů.

4.2 Kompatibilita Prvků Cinema 4D a SceneKit

Přenášení prvků scén ze softwaru Cinema 4D do SceneKit je velmi jednoduché, existuje ale několik omezení. Při přenášení geometrie, tedy samotného modelu, není žádný problém. Je třeba dávat pozor na počet polygonů objektu. Pokud grafik chce vyrendrovat obrázek v Cinemě, nevadí mu, když čas renderingu bude například 1 minuta. U některých realistických scén může trvat rendering jednoho snímku i několik hodin. V případě SceneKit se pohybujeme v interaktivním prostředí. Rendering tedy neprovádí renderovací engine ale grafická karta v počítači nebo telefonu. Výpočet by měl dosahovat výkonu okolo 60 snímků za vteřinu. Optimalizací počtu polygonů objektu lze výpočet dramaticky zrychlit. Dalším prvkem, který lze importovat jsou kamery a světla. V předchozích verzích SceneKitu, byl programátor omezen na pouhé tři typy světla. V aktuální verzi jsou již všechny druhy světel, včetně IES nebo ambient. Import materiálů je v aktuální verzi také bez problémů a nabízí všechny kanály, které nabízí 3D software. Je tedy na vývojáři, jestli bude materiály editovat v Cinema 4D nebo v SceneKitu.

Problém může nastat u přenášení animací. Cinema 4D je uživatelsky velmi přívětivá. Společnost Maxon se snaží uživateli zpříjemnit práci ve všech směrech. K animaci objektů zde existuje mnoho nástrojů, které dramaticky usnadňují práci a generují animace na základě nastavených atributů. Jakýkoliv deformátor nebo generátor animací SceneKit nepodporuje. Před importem scény s animovaným objektem musí vývojář použít takzvaný Baker. Baker realizuje rozklad složité animace na jednotlivé klíčové snímky a zapisuje pozici, rotaci a velikost každého prvku do jednotlivých klíčových snímků. Pro 3D software to může být komplikace, která snižuje výkon. Pro účel přenesení animace do SceneKit je to ovšem ideální řešení.

4.3 Formát Collada

Collada je formát vytvořený společností Sony jako oficiální formát pro digitální grafiku konzole PlayStation3 a PSP. Jedná se o formát s otevřeným XML schématem, čitelným v jakémkoliv textovém editoru a s koncovkou .dae. Formát Collada podporuje přes tři desítky programů pro tvorbu nebo interpretaci 3D grafiky, včetně Cinema 4D a Xcode.

Nyní je majetkem neziskového konsorcia Kronos Group, která sdílí se společností Sony autorská práva. V roce 2012 se stala ISO standardem pro 3D vizualizaci a industriální data (ISO/PAS 17506:2012).

5. Hudební Vizualizace

5.1 Zkoumání Hudebních Vizualizací a Jejích Principů

Základním principem hudební vizualizace je rozdělení zvukového signálu na sekce. Lidský sluch je schopen slyšet zvuk o frekvenci 20 Hz – 20 000 Hz. Tento rozsah se rozdělí na několik částí a zvuková stopa je vnímána každou touto částí zvlášť.

Čím vyšší frekvence, tím vyšší tón. Basové tóny se pohybují od 20 Hz do 160 Hz, přičemž se dále dělí na hluboké, střední a vyšší. Střední tóny jsou v rozmezí 160 Hz – 1280 Hz a výškové tóny od 1280 Hz do 20480 Hz. Jádro hudebního vizualizéru následně rozpoznává intenzitu signálu v příslušném rozmezí a nějakým způsobem ho interpretuje uživateli na obraz.

Nejjednodušší hudební vizualizací je zobrazení sloupců jednotlivých sekcí. U takové vizualizace se mění výška sloupce, podle intenzity signálu v příslušné sekci. Tato vizualizace nejlépe odpovídá hrané hudbě. Existuje spousta jiných druhů, které zobrazují abstraktní obrazce, které se mění v závislosti na hudbě. Nicméně u takového druhu vizualizace záleží spíše na představivosti uživatele, jestli se hraná skladba opravdu odráží v těchto pohybujících se obrazcích.

V praktické části této práce bude vytvořena vizualizace, podobná sloupcům, které zobrazují intenzitu daného signálu. Tuto vizualizace bude přenesena do 3D prostředí a následně interpretována uživateli ve virtuální realitě.

"Music visualization" can be defined, in contrast to previous existing pre-generated music plus visualization combinations (as for example music videos), by its characteristic as being real-time generated. Another possible distinction is seen by some in the ability of some music visualization systems (such as Geiss' MilkDrop) to create different visualizations for each song

or audio every time the program is run, in contrast to other forms of music visualization (such as music videos or a laser lighting display) which always show the same visualization[4]

5.2 Přehled dostupných hudebních vizualizací pro VR v AppStore

V AppStore existuje celkem 5 aplikací, které umožňují zobrazení hudebního vizualizéru ve virtuální realitě. Všechny až na jednu jsou dostupné zdarma a všechny jsou postavené na abstraktním prostředí nebo obrazcích. Všechny budou postupně stručně rozebrány, zejména z jejich technické stránky a možností nastavení.

5.2.1 White Cliffs VR Music Visualizer

White Cliffs nabízí možnost vizualizace v režimu VR nebo normálním zobrazení, ovládaném také pomocí akcelerometru. Neposkytuje možnost přehrání vlastní hudby, ani přes mikrofon. Scéna zobrazuje jednoduché abstraktní obrazce.

5.2.2 Matrix Music Visualiser VR

Matrix umožňuje přehrávání vlastní hudby v zařízení a vstup přes mikrofon. Stejně jako předchozí vizualizer, lze ho také spustit ve VR režimu nebo v klasickém zobrazení. Scéna zobrazuje psychedelické abstraktní prostředí.

5.2.3 SpectrumVR

Spectrum nabízí pro přehrávání na výběr celkem 5 skladeb, přehrávání vlastní hudby chybí, stejně tak jako vstup přes mikrofon. Ani v tomto případě nechybí výběr mezi VR a klasickým zobrazením. U klasického zobrazení se v prostředí uživatel pohybuje prostřednictvím dotykové obrazovky. Prostředí u této vizualizace je relativně pravidelné a na hudbu reaguje zejména změnou barev obrazců.

5.2.4 VR Music

VR Music, na rozdíl od ostatních aplikací nenabízí výběr mezi skladbami ale mezi 9 hudebními žánry. Vlastní hudbu nepodporuje, vstup přes mikrofon také ne. VR a klasické zobrazení nechybí. Prostředí je jednoduché, abstraktní a po určitém intervalu se změní na jiné.

5.2.5 Visual Music

Visual Music nabízí nejširší možnosti nastavení. Jako vstup lze zvolit vlastní hudbu v zařízení, mikrofon, a dokonce hudbu ze serveru SoundCloud. VR a klasické zobrazení je samozřejmostí. Na výběr jsou také celkem tří abstraktní prostředí, které překvapivě dobře reagují na hranou hudbu.

Obr. 5 Přehled vlastnosti aplikací

6. Návrh Aplikace

Vytvořená aplikace bude určena pro operační systém iOS a bude kompatibilní s mobilním zařízením iPhone 6 a lepším. V ideálním případě v kombinaci s mobilním headsetem pro virtuální realitu.

Při průzkumu dostupných aplikací pro zobrazení hudebního vizualizéru ve VR, bylo zjištěno, že pouze některé z nich podporují spuštění vlastní hudby a vstupu přes integrovaný mikrofon. Předpokládá se, že tyto funkce budou do aplikace implementovány.

Bylo také zjištěno, že prostředí veškerých dostupných aplikací jsou abstraktní obrazce nebo scény. V tom se bude aplikace lišit. Bude zobrazovat sloupce, které zobrazují intenzitu jednotlivých frekvencí, přenesených do 3D prostředí. Obohaceno bude také o stroboskop, který bude ovlivňovat jedno z hlavních světel scény.

Veškeré 3D modely a animace budou vytvořeny v softwaru Cinema 4D. Materiály, nasvětlení, kamery a další aspekty budou vytvořeny přímo v prostředí XCode. Tím bude zajištěn lepší přístup k atributům, které bude určitým způsobem ovlivňovat vstup.

6.1 Návrh Scény pro Vizualizaci

Scéna se bude skládat ze sloupců, které budou rozmístěny na tři strany. Před sebou bude mít pozorovatel sloupce umístěné do tvaru písmene T. Po levé a pravé straně budou sloupce celkem ve třech řádcích nad sebou. Tímto rozložením nevznikne situace, kdy by se dva sloupce mohli spolu střetnout. Každý z nich má dostatečný prostor pro svoji animaci.

Podlaha a pozadí scény budou černé a budou vytvářet dojem tmy. Animované sloupce budou tmavě šedé, osvětlené pouze ambientním světlem. Při spuštění jejich animace se změní vlastnost svítivost jejich materiálu na aktuální barvu v závislosti na čase. Ve scéně existuje ještě jedno světlo, a to u zadní stěny (stroboskopu). Toto světlo se rozsvěcuje a zhasíná v závislosti na zvuku a na celou scénu vrhá měkké stíny.

7. Použité Prostředky

7.1 Použitý Hardware a Jeho Specifikace

7.1.1 iPhone 6

iPhone 6 je celkem devátým modelem smarthphonu, který vydala společnost Apple. Jeho prodej odstartoval na podzim roku 2014. Tento model je vhodný pro použití v mobilním headsetu zejména kvůli obrazovce. Ta je vetší než u předchozích modelů, nicméně stejně velká jako u aktuálního modelu iPhone 7 (pokud není brán zřetel na verze plus). To samé platí i pro hodnotu PPI, které dosahuje 326 a dodnes se nezvýšila.

Procesor Apple A8 disponuje 64-bitovou architekturou. Ukrývá v sobě 2 jádra, každé z nich taktované na 1.4 GhZ. Aplikaci využívající virtuální realitu by měl hladce zvládnout při 60 FPS.

Obr.6 iPhone 6 Space Gray

7.1.2 VR Box

Tento headset je vhodný pro použití s téměř jakýmkoliv chytrým telefonem. Zásuvka, která se zboku vytáhne, má nastavitelný úchytový systém pro různé velikosti zařízení. Popruhy, díky kterým drží headset na hlavě, jsou samozřejmě také nastavitelné. Čočky, které jsou před

obrazovkou telefonu, lze posunout odpředu i dozadu. Tím lze do určité míry vyrovnat dioptrické vady pozorovatele. Headset je ale dostatečně velký i pro použití s dioptrickými brýlemi. Čočky může uživatel také posouvat dál nebo blíž k sobě, každý si tedy může nastavit co mu nejvíce vyhovuje. Pro aplikace, které využívají zadní kameru, lze tento headset také použít. Klapka, která na levé straně stíní okolní světlo je sundavací.

Obr.7 VR Box

7.2 XCode

Xcode je IDE společnosti Apple, které obsahuje balíček profesionálních vývojářských nástrojů pro vývoj softwarových aplikací na platformy iOS a Mac OS. Nejnovější dostupná oficiální verze je 8.1, ta umožňuje vyvíjet aplikace pro nejnovější verze operačních systémů Apple iOS 10 a Mac OS Sierra. Apple ho nabízí volně ke stažení z Mac App Store, ale pouze pro operační systémy OS X. [5]

XCode podporuje celou řadu programovacích jazyků. Hlavními jazyky pro vývoj aplikací pro macOS a iOS je Swift a také starší Objective-C, který se stále v aplikacích používá. Zajímavostí je, že tyto dva jazyky lze kombinovat ve stejném projektu. Společnost Apple udržuje aktuální a přehlednou dokumentaci pro oba z nich.

Pro testování nabízí programovací prostředí XCode simulátor pro iOS zařízení. V tomto simulátoru lze zvolit jakékoliv zařízení od iPhonu až po Apple Watch. Uživatel může také využít možnost otestovat svou aplikaci přímo na svém zařízení, stačí když ho připojí

a aplikaci si do telefonu nebo tabletu nainstaluje. Od září 2016, kdy proběhla konference WWDC 2016, lze využít tuto funkci i bez předplaceného developerského účtu.

Interface builder je nástroj, který maximálně usnadňuje vytváření uživatelského rozhraní. Díky jeho funkcím stačí vytvořit jeden interface pro všechny velikosti obrazovek a pro landscape i portrait mode. Při správném nastavení se uživatelské rozhraní chová responzivně a upravovat ho ručně pro různé typy obrazovek je spíše výjimečnou záležitostí.

7.3 Programovací Jazyk Swift 3

Jazyk Swift je poměrně nový. Apple ho představil v roce 2014 a je určen pro programovaní aplikací výhradně pro platformu macOS a iOS. Od svého vzniku je na vzestupu, co se týče počtu programátorů a velmi se podobá jazyku C# nebo Java.

Swift je z větší části obdobou Objective-C za využití moderních konceptů a syntaxe. Při jeho představení byl jednoduše představen jako "Objective-C bez C" Swift na rozdíl od Objective-C nevyužívá pointery, v případě potřeby je však možné je využít. Dále byl nahrazen Smalltalkový způsob volání metod za tečkovou notaci a jmenné prostory, což je běžné v ostatních C-like jazycích jako je například Java nebo C#. Swift přináší pojmenované parametry a zachovává klíčové vlastnosti Objective-C, často při zjednodušení syntaxe.[6]

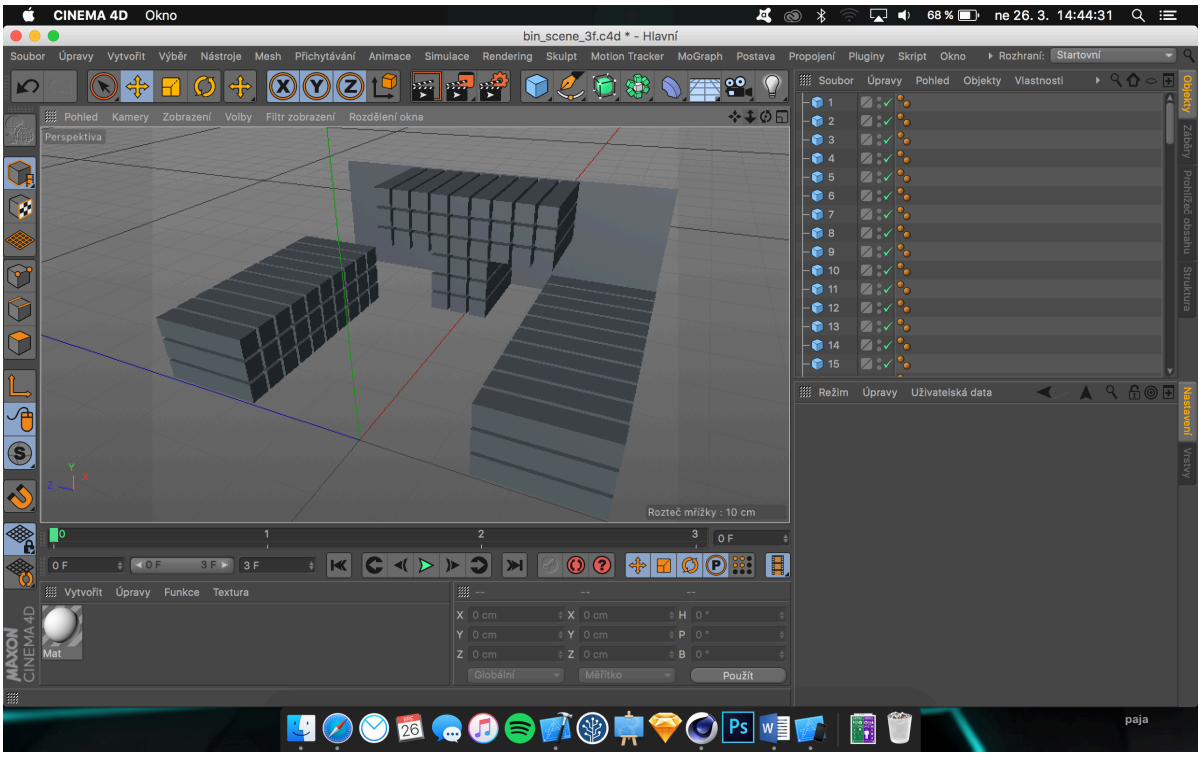
Při představení verze Swift 3.0 v roce 2016 začal Apple prosazovat myšlenku, že každý člověk se může naučit kódovat a vytvářet aplikace. Natočil několik videí na toto téma a neustálý vývoj tohoto jazyku směřuje ke zjednodušování syntaxe pro lepší čitelnost kódu a jednodušší vytváření složitých struktur a konstrukcí.

8. Grafická Část

8.1 Vytvoření scény a animací pomocí softwaru Cinema 4D

Připravený 3D soubor se scénou vizualizéru obsahuje celkem 96 kvádrů, které budou reagovat na spuštěnou hudbu. Za prostřední stěnou je situována plocha, která představuje stroboskop. Podlaha bude následně vytvořena v prostředí SceneKit. Pokud by byla vytvořena v Cinema 4D, nemohla by být přenesena jako nekonečná plocha. Světla, materiály a kamery budou vytvořeny také ve SceneKitu, z důvodu lepší manipulace v kódu.

Sobory s těmito 3D objekty budou existovat celkem tři. Každý z nich bude obsahovat animace všech objektů. První soubor bude obsahovat animaci pouze do jedné třetiny celkové dráhy krychle, druhý do dvou třetin a třetí celou animaci. Na tyto animace se při spuštění aplikace vytvoří reference a následně budou spouštěny podle intenzity signálu na příslušnou krychli.



Obr.8 Scéna v editoru Cinema 4D

8.3 Procházení barevného spektra RGB

Krychle, na které bude aplikována animace, se zbarví do aktuální barvy. Tato barva bude pro všechny stejná a postupně bude procházet celé barevné spektrum. Pro implementaci bude potřeba aby se v pravidelném časovém intervalu měnily uložené hodnoty RGB, podle předem zjištěných pravidel.

Při procházení barevného spektra v grafickém programu byly zjištěny pravidla změny. Na obrázku je vidět šest záchytných bodů, které budou stačit pro vytvoření přechodu celého spektra.

Obt.9 Záchytné body pro procházení barevného spektra

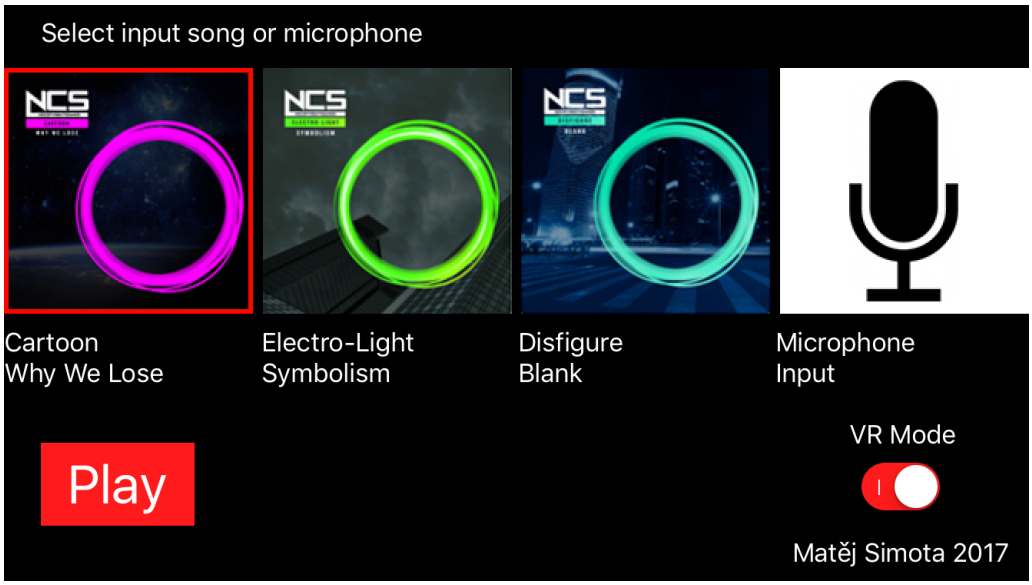
Při implementaci je použita konstrukce switch, která se provádí v krátkém časovém intervalu.

```
switch (red,green,blue) {
        case (_,_,_) where (red == 255 && blue < 255 && green ==
0):
            blue += 1
        case (_,_,_) where (blue == 255 && red > 0 && green == 0):
            red -= 1
        case (_,_,_) where (blue == 255 && red == 0 && green <
255):
            green += 1
        case (_,_,_) where (green == 255 && blue > 0 && red == 0):
            blue -= 1
        case (_,_,_) where (green == 255 && blue == 0 && red <
255):
            red += 1
        case (_,_,_) where (red == 255 && green > 0 && blue == 0):
            green -= 1
        default:
            print("default rgb")}
```

9. Vývoj Aplikace

9.1 Vstupní UI, Před Vložením Zařízení do Brýlí

Při spuštění aplikace má uživatel na výběr několik možností nastavení. Zejména si může vybrat celkem ze tří skladeb, které si může přehrát. Alternativou je vstup přes mikrofon. Tímto způsobem aplikace umožňuje si přehrát vlastní hudbu z externího zdroje. Ve spodní části obrazovky lze ještě zapnout, popřípadě vypnout, VR režim. V případě vypnutí této funkce bude namísto dvou pohledů pro VR brýle, zobrazen pouze jeden. Ten bude přes celou obrazovku a uživatel ho bude moci ovládat pomocí dotyku. Posledním prvkem je tlačítko „play“, které spustí samotnou vizualizaci.

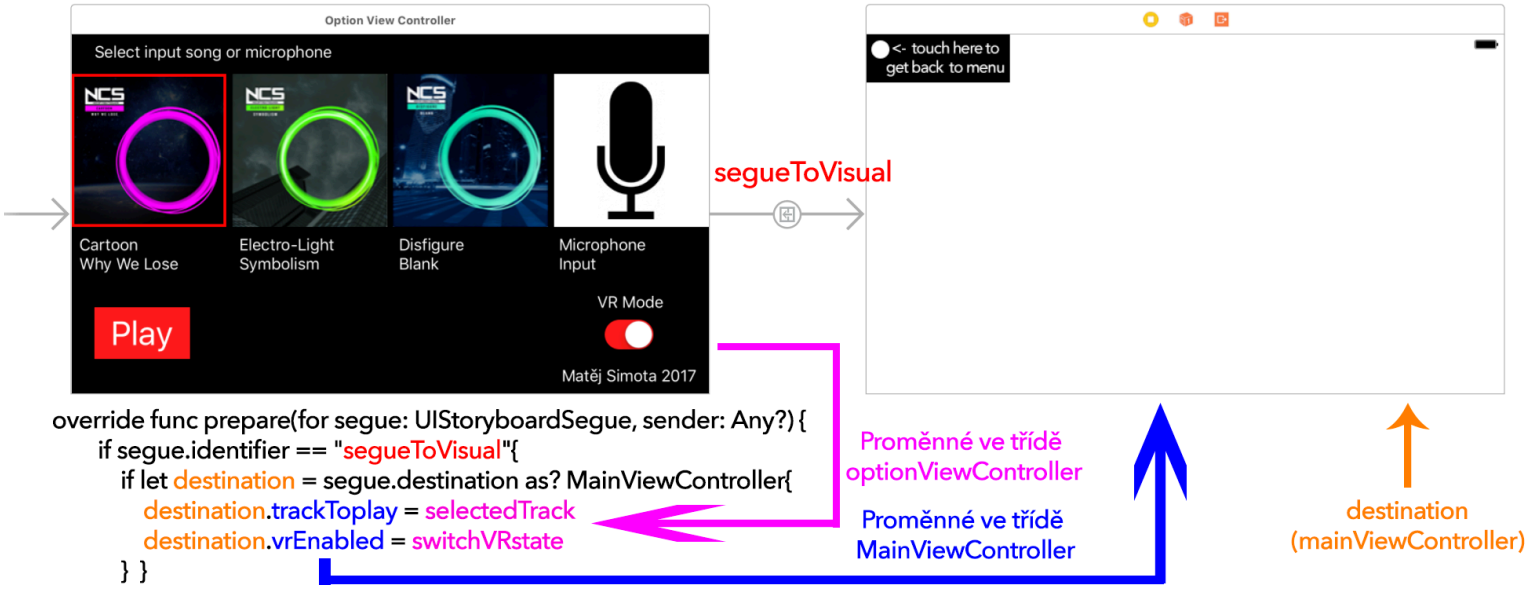


Obr.10 Vstupní UI

9.2 Předání parametrů nastavení

Na úvodní obrazovce uživatel vybírá, jakou zvukovou stopu chce přehrát a jestli chce aplikaci spustit ve VR režimu. Tyto informace se předají jako dva parametry do třídy, kde se bude odehrávat vizualizace. V této třídě při načítání dojde k příslušnému nastavení scény. Toto předání proběhne za pomoci metody prepare, která se volá před každým přechodem mezi jednotlivými obrazovkami.

Třída SpectralViewController má připravené pro toto nastavení dvě proměnné, trackToPlay a vrEnabled, které se nastavují nastaví před vykonáním segue (přechod mezi jednotlivými obrazovkami), z aktuálních hodnot třídy optionViewController.



Obr.11 Schéma přenosu parametrů mezi třídami

9.2 Získání Referencí na Objekty a Animace v 3D Souborech

Jak již bylo řečeno, 3D soubory se scénou existují celkem tři. První z nich je použit pro zobrazení, získání referencí na objekty a animace největšího rozsahu. Zbylé dvě slouží pouze k importu animací středního a nízkého rozsahu.

Objekty jsou uloženy do pole objektů. Jako identifikátor je použit jejich název. Animované kvádry jsou pojmenovány čísly vzestupně tak, jak jdou po sobě od nejnižších frekvencí po nejvyšší. Nicméně SceneKit si objekty, které jsou pojmenovány číselnými znaky, přejmenuje podle svého vlastního systému. Na začátek názvu se vloží lomítko a za něj číslo. Toto číslo představuje nikoliv název, ale pořadí objektu v object manageru. V Cinema 4D je tedy nutné scénu připravit tak, aby byly objekty seřazeny vzestupně podle jejich názvů.

Animace jsou identifikovány stejným způsobem. Pro každý ze souborů je vytvořeno vlastní pole. Při snímání frekvence zvuku je poté animace převzata a aplikována z pole odpovídající danému rozsahu. Animace jsou také automaticky pojmenovány. A to ve formátu

název scény, pomlčka a číslo objektu. Pro jejich identifikaci tedy bude využito jednoduché možnosti interpolace stringů v jazyce Swift a proměnné cyklu.

```
func ziskatReference(){

        let scene2f = SCNScene(named:
"../3d.scnassets/bin_scene_2f.dae")
        let scene3f = SCNScene(named:
"../3d.scnassets/bin_scene_3f.dae")
        backPlain = scene.rootNode.childNode(withName: "back",
recursively: true)!
        for i in 0...95 {
            rBiny.append(scene.rootNode.childNode(withName:
"_\(i+1)", recursively: true)!)

            rAnimace3f.append(rBiny[i].animation(forKey:
rBiny[i].animationKeys[0])!)
            //nastaveni animace
            rAnimace3f[i].repeatCount = 1
            rAnimace3f[i].autoreverses = true
            rAnimace3f[i].isRemovedOnCompletion = true
            rAnimace3f[i].fadeInDuration = fade
            rAnimace2f.append((scene2f?.rootNode.childNode(wit
hName: "_\(i+1)", recursively: true)?.animation(forKey:
"bin_scene_2f-\(i+1)"))!)
            //nastaveni animace
            rAnimace2f[i].repeatCount = 1
            rAnimace2f[i].autoreverses = true
            rAnimace2f[i].isRemovedOnCompletion = true
            rAnimace2f[i].fadeInDuration = fade
            rAnimace1f.append((scene3f?.rootNode.childNode(wit
hName: "_\(i+1)", recursively: true)?.animation(forKey:
"bin_scene_3f-\(i+1)"))!)
            //nastaveni animace
            rAnimace1f[i].repeatCount = 1
            rAnimace1f[i].autoreverses = true
            rAnimace1f[i].isRemovedOnCompletion = true
            rAnimace1f[i].fadeInDuration = fade
            rBiny[i].removeAllAnimations()
        }
    }
```

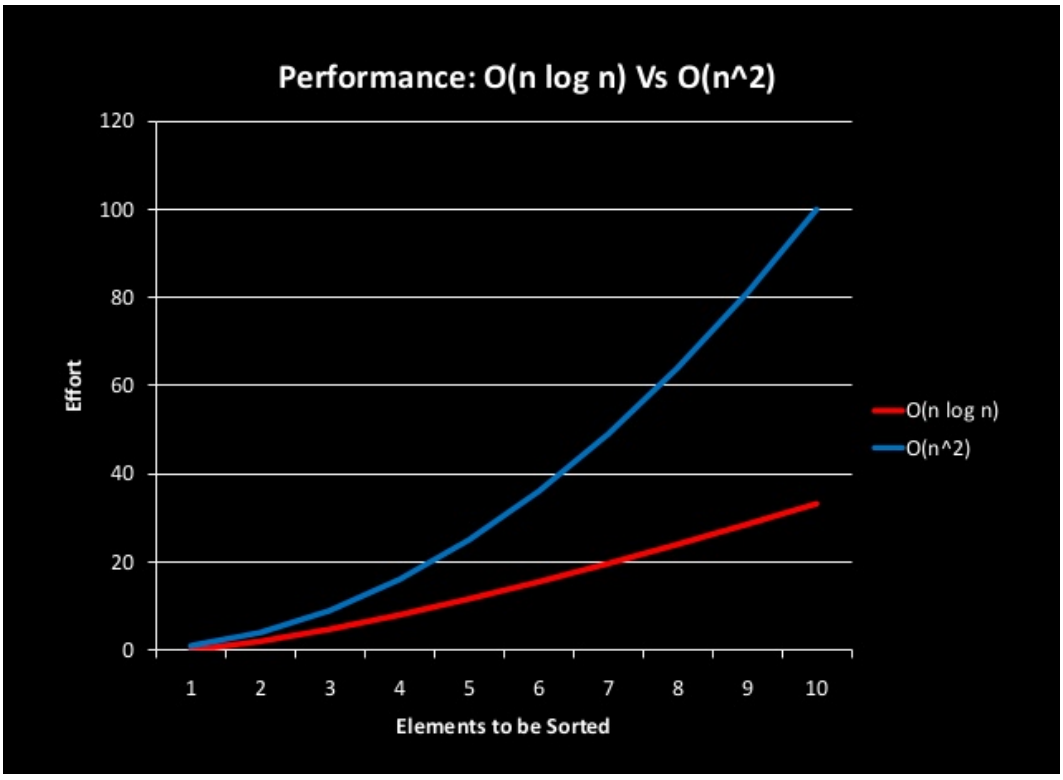
9.3 Získání Dat pro Ovládání Animací ze Zvukových Stop

9.3.1 FFT Transformace

FFT je zkratka z anglického "Fast Fourier Transform", rychlá Fourierova transformace. Fourierova transformace je důležitý matematický nástroj se závratně mnoha praktickými aplikacemi. Například zní-li ve zvukovém záznamu mnoho různých tónů současně, lze s pomocí Fourierovy transformace zjistit frekvence všech přítomných tónů. [7]

Fourierova transformace umožňuje rozložit křivku signálu na jednotlivé frekvence. V případě aplikace hudebního vizualizéru postačí, pokud se zvukový signál rozdělí na 96 pásem (počet krychlí ve scéně). U každého pásma bude sledována normalizovaná hodnota intenzity v decibelech a následně podle této intenzity bude přidána animace.

Algoritmus pro spočtení FFT je efektivní způsob, jak spočítat diskrétní Fourierovy transformace (DFT). Zatímco algoritmus pro spočtení DFT má složitost O(N^2), FFT disponuje složitostí pouze O(N log N).



Obr.12 Srovnání složitosti algoritmů

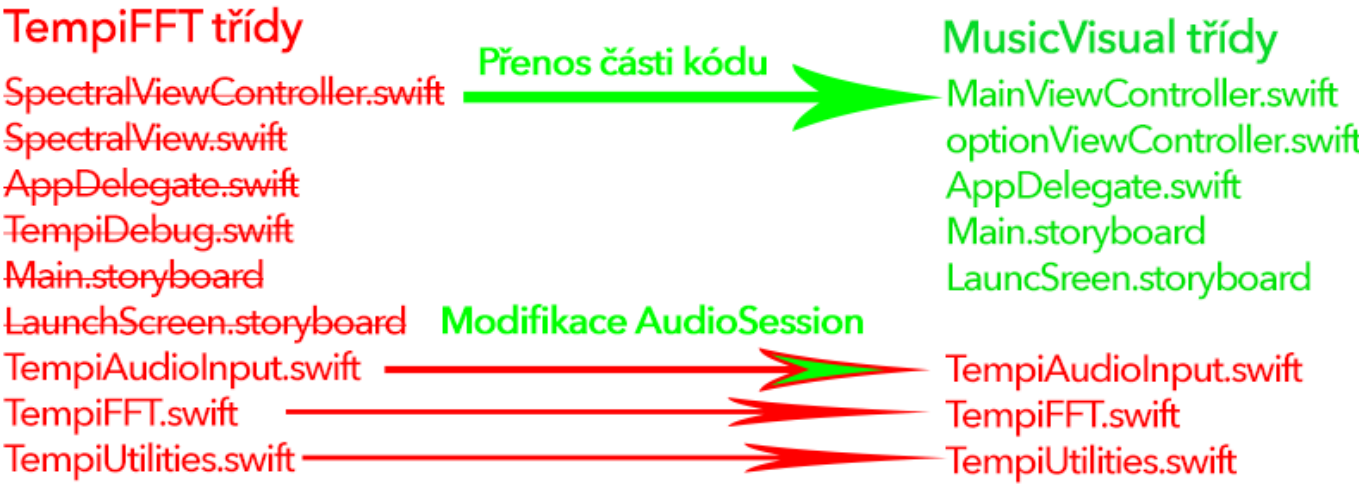
9.3.2 TempiFFT

TempiFFT demonstrates how to input audio via AVFoundation for recording or processing and implements an FFT to display a real-time spectrum plot of incoming audio. [8]

Komponent získávání dat ze zvukového vstupu bude převzat od vývojáře jménem John Scalo. Ten jako jediný vytvořil aplikaci, která umožňuje FFT transformaci a využívá přitom pouze jazyk Swift, nikoliv Objective-C. Svou aplikaci TempiFFT označil licencí public domain. Nevztahují se na něj tedy žádná autorská práva.

Aplikace TempiFFT zachytává zvuk z integrovaného mikrofonu a následně v reálném čase zobrazuje graf frekvencí na obrazovku. Převzetím a modifikací části této aplikace bude docíleno dodávání požadovaných dat z právě přehrávané zvukové stopy.

Modifikace aplikace spočívala zejména v úplném zrušení pohledu, který zobrazoval spektrum a přesunutí určitých bloků kódu do hlavní třídy nové aplikace. Pohled byl nahrazen 3D scénou. Mírnou modifikací prošla také audioSession, která v původním stavu neumožňovala přehrání hudby a zároveň její snímaní pro následné zpracování. Dvě celé třídy byly přesunuty a využity pro zpracování signálu. Těmi jsou TempiAudioInput a TempiFFT. TempiAudioInput zajišťuje zvukový vstup, povolení uživatele pro přístup k integrovanému mikrofonu, nastavení audioSession a audioUnit. Třída TempiFFT realizuje samotnou transformaci zvukového signálu. Další převzatou třídou je TempiUtilities, která pouze zajišťuje pokračování aplikace až po vykonání dané sekce kódu.



Obr.13 Schéma přenosu tříd do aplikace

9.4 VR - Ovládání Pohledu Pomocí Accelerometru

9.4.1 Core Motion

Core Motion Framework umožňuje aplikaci přístup k datům z gyroskopu a kompasu, integrovaných v zařízení. Pomocí proměnné deviceMotionUpdateInterval lze určit, jak často se budou gyroskopická data generovat (v sekundách). Pro účel simulace virtuální reality je vhodné data obnovovat šedesátkrát za vteřinu.

K ovládání kamery ve virtuální realitě poslouží třída CMAttitude, která poskytuje informace o horizontálním a vertikálním náklonu a směrové orientaci.

9.4.2 Implementace

```
motionManager = CMMotionManager()
        motionManager?.deviceMotionUpdateInterval = 1.0 / 60.0
        motionManager?.startDeviceMotionUpdates(using:
CMAttitudeReferenceFrame.xArbitraryZVertical)
```

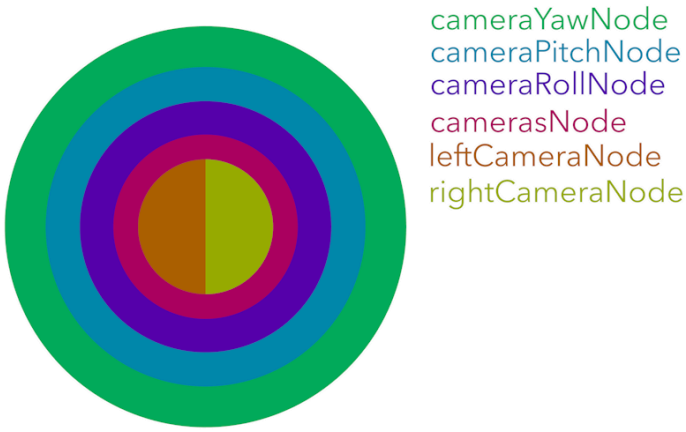
V první řadě je potřeba deklarovat a nastavit motionManager.. Funkce startDeviceMotionUpdates vyvolá činnost motion manageru, jeho parametr je nastavený na xArbitraryZVertical, tedy osa Z bude v tomto případě vertikální a zbylé dvě osy jsou neurčené, protože kamera se bude otáčet. Pro korektní počáteční stav pohledu je třeba otočit kameru o 90 stupňů po ose x. Pokud by k tomuto posunu nedošlo, uživatel by při pohledu před sebe, ve virtuální realitě koukal do země.

```
func renderer(_ aRenderer: SCNSceneRenderer, updateAtTime
time: TimeInterval)
    {
        if let mm = motionManager, let motion =
mm.deviceMotion {
            let currentAttitude = motion.attitude

 cameraRollNode!.eulerAngles.x = Float(currentAttitude.roll)
 cameraPitchNode!.eulerAngles.z = Float(currentAttitude.pitch)
 cameraYawNode!.eulerAngles.y = Float(currentAttitude.yaw)
        }
    }
```

Funkce renderer aplikuje získaná data z Core Motion na předem vytvořené tři nody Každý z nich představuje rotaci okolo jedné osy kartézské soustavy souřadnic. Tyto nody jsou následně hierarchicky vloženy do sebe. Jako poslední je v této hierarchii camerasNode, ve kterém jsou obě kamery vykreslující výsledný obraz.

Pokud jsou nody vloženy do sebe a nadřazený node nějakým způsobem změní svou pozici nebo rotaci, ovlivní to i veškeré nody v hierarchii pod ním.



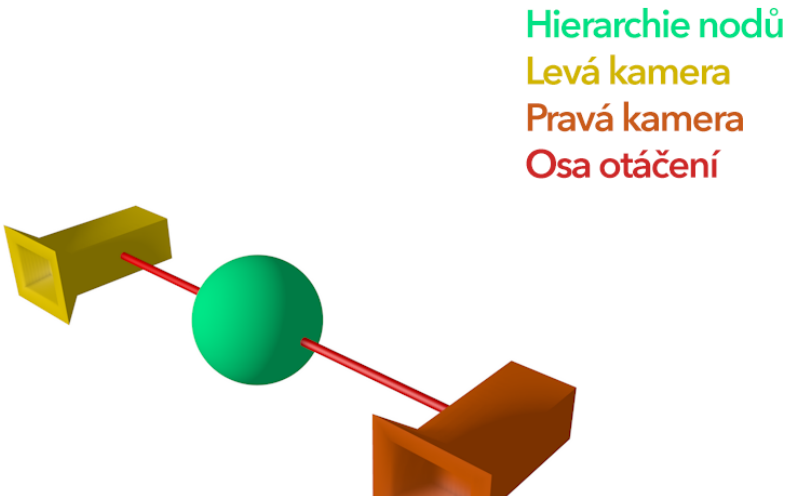
Obr.14 Hierarchie Nodů pro ovládání kamery

9.5 Rozmístění Kamer ve Scéně Pro Dosažení 3D Efektu

3D efektu lze dosáhnout rozmístěním kamer podobně, jako jsou rozmístěny lidské oči. Tedy vedle sebe. Díky tomu, že v aplikaci je jak levá, tak pravá kamera umístěná v hierarchii nodů až úplně na konci, stačí pozici kamer posunout a tím vytvořit osu otáčení. V případě, že pozorovatel nakloní hlavu, nakloní se také obraz a když se bude otáčet nebo jinak pohybovat, kamery se budou chovat korektně, tak jak je tomu ve skutečnosti.

Tímto nastavením je docíleno realistického dojmu z pohybu hlavou. Pozorovatel uvidí každým okem trochu jiný obraz, mozek si tyto obrazy spojí a vytvoří dojem opravdového 3D prostoru.

Každá z kamer musí mít svůj vlastní SCNView. Obrazovka zařízení bude rozdělena na dvě poloviny. Na levé straně bude SCNView zobrazující pohled levé kamery a na pravé straně pohled pravé kamery.



Obr.15 Rozmístění kamer a systém jejich pohybu

9.6 Uživatelská interakce

Z hudební vizualizace se provede přechod zpět do menu, pokud přehrávaná zvuková stopa skončí. V případě, že uživatel zvolí vstup přes mikrofon, aplikace se nikdy zpět do nastavení samovolně nevrátí. Je potřeba implementovat prvek, který tuto funkci bude umožňovat. Vzhledem k tomu, že hudební vizualizace je zobrazena na celém display zařízení, nelze umístit trvalé tlačítko kamkoliv do pohledu. Tento problém je vyřešen jednoduchým tlačítkem, které do několika sekund zmizí, nicméně implementovaný TapRecognizer zůstane.

Vizuální podoba tlačítka tedy nese pouze informativní charakter, kde se má uživatel dotknout obrazovky v případě, že se bude chtít vrátit do menu. Nastavení intervalu kdy obrázek tlačítka již nebude vůbec vidět, je nastaven na dobu přibližně stejnou jako trvá vložení zařízení do VR Boxu. Zmizení tlačítka bude zajištěno timerem, který bude měnit alpha kanál obrázku tlačítka.

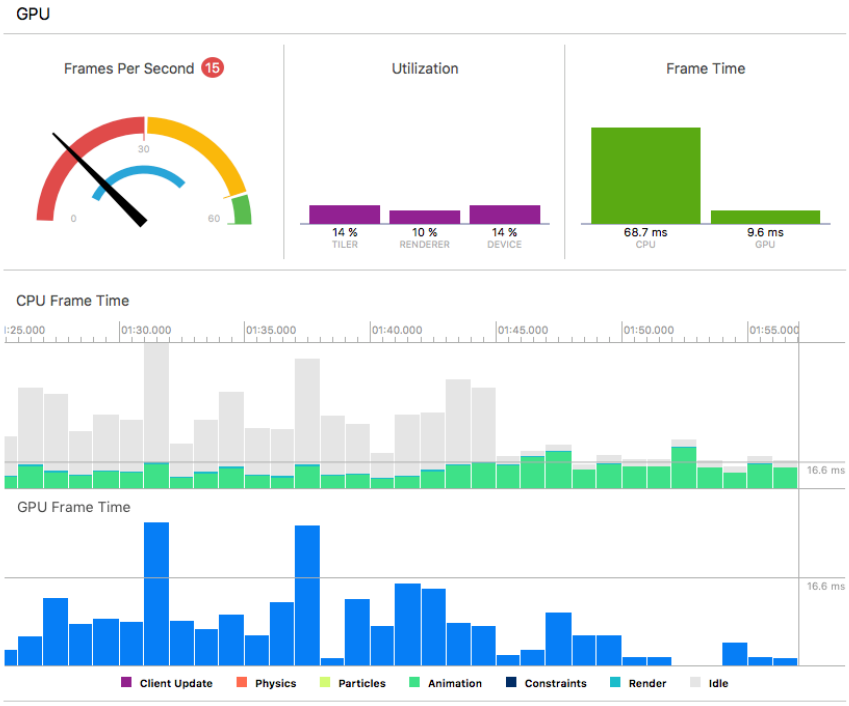
```
timer2 = Timer.scheduledTimer(timeInterval: 0.1, target: self,
selector: #selector(self.alphaToZero), userInfo: nil, repeats:
true)

func alphaToZero(){
        if(backToMenu.alpha > 0.02){
        backToMenu.alpha -= 0.01
        }
```

9.9 Testování

Pro testování je použit iPhone 6 připojený k XCode. Aplikace se na něm přes vývojové prostředí spustí a nechá se připojený. XCode nabízí velké množství nástroju pro testování, v případě této aplikace postačí sledovat hodnotu FPS a CPU Frame Time.

Z výsledků tetování aplikace vyplývá, že v některých částech přehrávaných skladeb dosahuje FPS nepřijatelně nízké hodnoty. V částech, kde je aktivních většina frekvencí a tím pádem i animací, dosahuje FPS hodnot okolo 10 – 20 snímků za vteřinu.



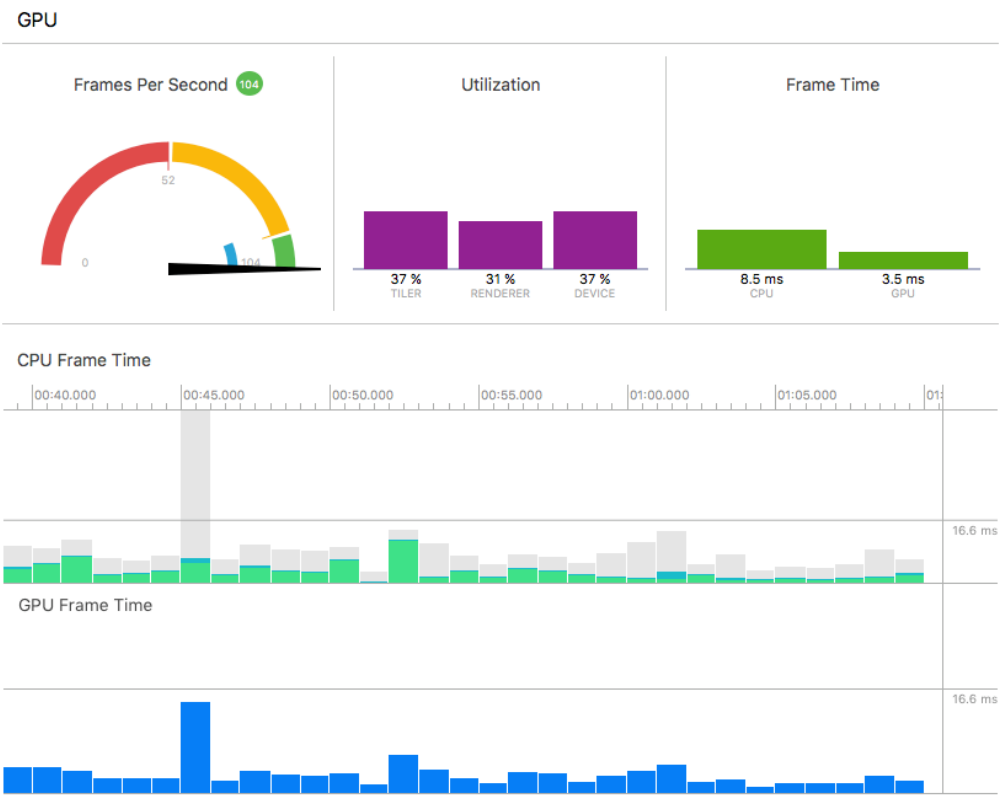
Obr.16 FPS před optimalizací

V aplikaci nejsou použity složité světelné systémy ani materiály, jedinou příčinnou může být pouze přehlcení scény animacemi. Je to patrné i z grafu na obrázku, kde animace z drtivé většiny zabírá CPU Frame Time. Při sledování vizualizéru je vidět, že animací je tam zbytečně mnoho. Do aplikace tedy bude zavedena optimalizační proměnná animationControl. Tato proměnná ponese hodnotu maximálního počtu aktivních animací ve scéně. Vizualizér má celkem 96 krychlí, řekněme že bude stačit, když bude aktivní jedna třetina z nich. animationControl bude nastaven na hodnotu 30 a aplikace bude otestována znovu. Pře každým přidáním animace se zkontroluje, jestli počet animací ve scéně (hodnota proměnné

animationControl) nepřesahuje hodnoty 30. Pokud ano, animace bude zahozena. Aktualizace proměnné se vykonává v následujícím bloku kódu.

```
self.animationControl = 0
            self.scene.rootNode.enumerateChildNodes({ (child,
stop) in          //vykoná následující block pro všechny
childNodes scény animace
                if child.animationKeys.count > 0 {
                    self.animationControl += 1
                }
```

Po optimalizaci dosahuje FPS vysokých hodnot, v průměru kolem 60 – 70 snímků za vteřinu a v méně náročných místech až 110. Animace v případě této aplikace dramaticky ovlivňují výkon. Po snížení jejích počtu vizualizér neztratil na kvalitě, spíše naopak. Výsledná scéna dává uživateli větší dojem, že animace odpovídají hrané hudbě.



Obr.17 FPS po optimalizaci

10. Závěr

Výsledná aplikace splňuje vytyčené cíle. Na vstupní obrazovce má uživatel možnost zvolit si jednu ze tří obsažených skladeb, nebo si může pustit vlastní hudbu prostřednictvím vstupu přes mikrofon. Existuje zde i možnost zobrazení bez VR headsetu a ovládání kamery pomocí dotykové obrazovky.

Při následném spuštění aplikace se uživatel ocitá ve scéně mezi krychlemi, které reagují na různé frekvence spuštěné hudby a zároveň se rozsvěcují. Uživatel má možnost se vrátit zpět do menu dotykem levého horního rohu obrazovky. Tato možnost je mu představena při každém spuštění samotné scény mizící nápovědou v odpovídajícím rohu.

Animace byly vytvořeny v softwaru Cinema 4D a úspěšně přeneseny do prostředí SceneKit. Ve scéně se uživatel pohybuje pomocí zabudovaného gyroskopu a kompasu a tím aplikace simuluje virtuální realitu.

Realizace celé aplikace proběhla díky vytvoření grafického prostředí, použití dostupných frameworků pro simulaci virtuální reality a převzetí kódu pro Fourierovu transformaci, který byl modifikován pro potřeby aplikace. Pokud by software proběhl dalším vývojem a nabízel více scén pro vizualizaci, stačilo by pouze pojmenovat objekty ve scéně stejným systémem a vytvořit k nim příslušné animace.

11. Požitá literatura

[1] Virtuální Realita. *Wikipedia: the free encyclopedia* [online]. Česká Republika, 2016 [cit. 2017-04-05]. Dostupné z: https://cs.wikipedia.org/wiki/Virtuáln%C3%AD_realita

[2] How to use VR on a Mac: Can you play VR games on OS X? *MacWorld* [online]. Velká Británie, 2017 [cit. 2017-04-05]. Dostupné z: http://www.macworld.co.uk/how-to/mac/how-use-vr-on-mac-can-you-play-vr-games-on-os-x-macos-2017-3640213/

[3] SceneKit. *Apple* [online]. California, 2017 [cit. 2017-04-05]. Dostupné z: https://developer.apple.com/reference/scenekit

[4] Music Visualisation. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA), 2017 [cit. 2017-04-05]. Dostupné z: https://en.wikipedia.org/wiki/Music_visualization

[5] XCode. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-25]. Dostupné z: https://cs.wikipedia.org/wiki/Xcode

[6] Swift (programovací jazyk). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-25]. Dostupné z: https://cs.wikipedia.org/wiki/Swift_(programovac%C3%AD_jazyk)

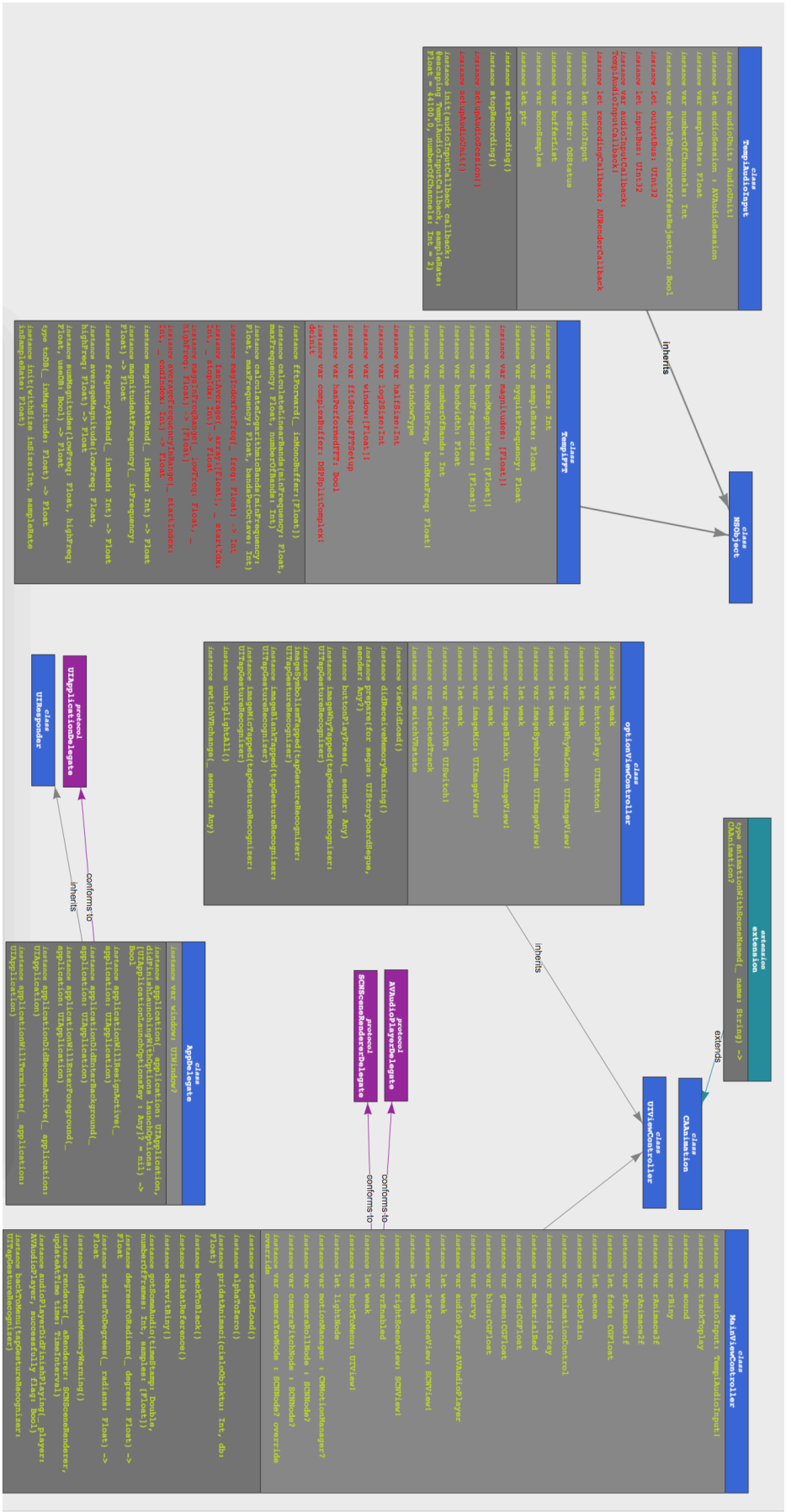
[7] FFT Spectra. *Fft-spectra.sourceforge* [online]. Česká Republika, 2015 [cit. 2017-04-05]. Dostupné z: http://fft-spectra.sourceforge.net/index-cz.html

[8] TempiFFT. *GitHub* [online]. San Francisco, 2016 [cit. 2017-04-05]. Dostupné z: https://github.com/jscalo/tempi-fft

12. Seznam příloh

Příloha A - UML
Příloha B – CD s projektem XCode (zdrojové kódy)

Příloha A – UML aplikace



Příloha B – CD s projektem XCode (zdrojové kódy)