VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

OBRAZOVÁ A VIDEO INTERPRETACE ZVUKOVÝCH DAT

AUDIO DATA INTERPRETATION IN IMAGE AND VIDEO

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Jiří Dobiášovský

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Kamil Říha, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Bc. Jiří Dobiášovský ***ID:*** 195781
Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Obrazová a video interpretace zvukových dat

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte možnosti vizualizace zvuku na základě jeho parametrického popisu. Navrhněte a implementujte vytvoření syntetického obrazu/objektu resp. syntetické videosekvence, kde budou parametry obrazu/videa měněny na základě časově proměnných parametrů zvuku (hlasitost, spektrální složení, tempo…).
Možné nástroje pro implementaci: PureData, OpenCV, OpenGL a MS Visual C++.
V rámci diplomové práce je požadováno naprogramování kompletního vizualizačního řetězce obsahujícího měření parametrů (tempo, hlasitost apod.) a jejich pokročilá vizualizace ve videosekvenci.

DOPORUČENÁ LITERATURA:

[1] GONZALEZ, R. C.; WOODS R. E.: Digital Image Processing, Prentice Hall, New Jersey, 2002.
[2] BRADSKI, G.; KAEHLER A.: Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, Inc. USA 2008, ISBN: 978-0-596-51613-0.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 19.5.2023

Vedoucí práce: doc. Ing. Kamil Říha, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

Abstrakt

Obsahem práce je zpracování problematiky obrazové a video interpretace zvukových dat. V teoretické části je nejprve zpracováno téma zvukového signálu, jeho vlastnosti a reprezentace zvukového signálu v časové a frekvenční oblasti. Dále je objasněno jeho digitální zpracování a jsou popsány jednotlivé komunikační protokoly, pomocí kterých je možné přenášet zvuková data nebo komunikovat s audio a video zařízeními. Na základě poznatků z teoretické části je zvolen vhodný systém pro 3D generativní vizualizaci zvukového signálu na základě MIDI dat nebo jeho obálky. Je vytvořen systém pro zpracování zvukových dat v DAW a jejich odeslání po lokální síti pomocí vhodného protokolu. V prostředí určeném k vizualizaci je vytvořen program, který odeslané signály přijímá a vhodně třídí. Je popsán proces tvorby vizuálních efektů, způsob jejich využití při vizualizaci zvukového signálu a vzájemná komunikace mezi oběma programy je demonstrována na vhodných příkladech a pilotním projektu. Systém, který je výsledkem této práce lze použít pro 3D generativní vizualizaci hudebních dat v reálném čase.

Klíčová slova

Audio, video, vizualizace, Ableton, Unreal Engine

Abstract

The content of the thesis is the elaboration of the problem of image and video interpretation of sound data. In the theoretical part, the topic of the sound signal, its properties and the representation of the sound signal in the time and frequency domain are treated first. Furthermore, its digital processing is described and individual communication protocols with which it is possible to transmit audio data or communicate with audio and video devices are described too. Based on the findings from the theoretical part, a suitable system is chosen for 3D generative visualization of the sound signal based on MIDI data or its envelope. A system is created for processing sound data in the DAW and sending it over the local network using a suitable protocol. In the environment chosen for visualization is created a program that receives the sent signals and sorts them appropriately. The process of creating visual effects is described, the method of their use in the visualization of the audio signal and mutual communication between the two programs is demonstrated using suitable examples and pilot project. The system resulting from this work can be used for real-time 3D generative visualization of music data.

Keywords

Audio, video, visualization, Ableton, Unreal Engine

Bibliografická citace

DOBIÁŠOVSKÝ, Jiří. Obrazová a video interpretace zvukových dat [online]. Brno, 2023 [cit. 2023-05-18]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/151161. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Kamil Říha.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Jiří Dobiášovský*
VUT ID studenta:	*195781*
Typ práce:	*Diplomová práce*
Akademický rok:	*2022/23*
Téma závěrečné práce:	*Obrazová a video interpretace zvukových dat*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucího závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 19. května 2023 ----------

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc Ing. Kamilu Říhovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: 19. května 2023 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

Úvod

Obrazovou a video interpretaci zvukových dat lze v kontextu této práce chápat jako proces interpretace zvuku, respektive hudby, za pomoci obrazu nebo videosekvence. Vizualizace hudby mapuje průběh nahrávky na základě hudebních dat pomocí vhodných nástrojů a může sloužit jako nástroj pro rozšíření posluchačského zážitku, součást umělecké instalace nebo ji lze použít pro zpřístupnění hudebního zážitku sluchově handicapovaným posluchačům.

V rámci diplomové práce byl navržen a vytvořen systém pro vizualizaci hudby s ohledem na tyto konkrétní cíle:

1. Popsat zvukový signál a jeho charakteristiky, digitální zpracování zvukového signálu a způsoby digitální komunikace mezi hudebními softwary a zařízeními.
2. Stanovit konkrétní řešení a oblast hudebního průmyslu, pro kterou bude vizualizaci možné využít
3. Vybrat vhodný software a způsob vizualizace
4. Realizovat vizualizaci a její funkčnost demonstrovat na vhodných ukázkách

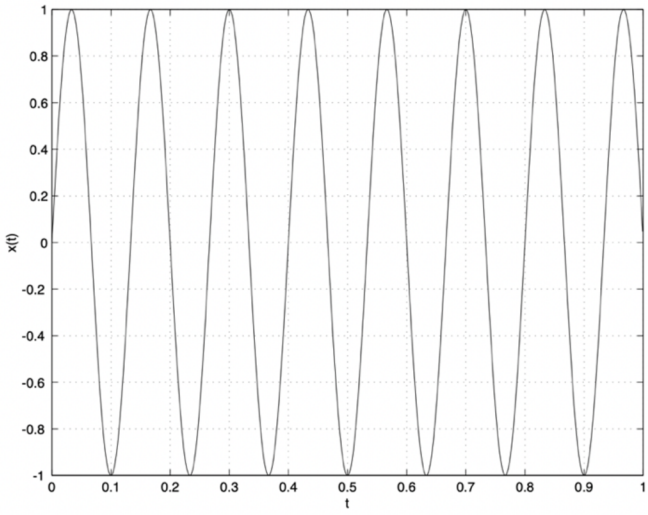
V rámci diplomové práce byl vytvořen systém pro vizualizaci hudby způsobem, který nenabízí běžně dostupný komerční software. Byl navržen nástroj pro vzájemnou komunikaci mezi hudebním softwarem pro producenty a DJe a prostředím pro tvorbu vizuální scény. Jde o systém, pomocí kterého je možné zaznamenávat hudební data a ta v reálném čase sdílet po lokální síti s dalším programem. Ten tato data zaznamenává a vhodně třídí. Na jejich základě poté upravuje libovolné parametry vizuální scény, popřípadě na ní spouští události. Hlavní odlišnost od běžné dostupných systémů je, že se jedná o 3D generativní vizualizaci tvořenou živě, nikoli o vizualizaci za pomoci předpřipravených statických videosekvencí, což dělá vizuální zážitek pokaždé unikátním. Další odlišností je použití MIDI dat.

1. ZVUKOVÝ SIGNÁL

Zvukem nazýváme podélné mechanické vlnění v látkovém prostředí, kdy molekuly kmitají kolem svých rovnovážných poloh. V důsledku působení zvukové vlny dochází k zhušťování a ředění molekul v určitých místech vlny. Jinými slovy dochází ke změnám tlaku podél směru šíření. Zvukový signál je reprezentace takového vlnění, typicky jako měnící se hladina elektrického napětí pro analogové signály nebo jako řada binárních čísel pro digitální zvukový signál. Základní zvuková vlna může být popsána pomocí tří základních parametrů – amplitudy, frekvence a fáze. Amplitudou lze rozumět sílu vlny, která určuje, jak moc se stlačuje a ředí vzduch. Frekvence je rychlost, kterou vlna rozkmitává molekuly z rovnovážných poloh. Fáze popisuje polohu vlny v jejím cyklu. Základní forma zvukového signálu je harmonická vlna o frekvenci 20 Hz až 20 kHz, což odpovídá přibližným hranicím lidského sluchu. Libovolný zvukový signál lze vnímat jako součet harmonických vlnění o různé amplitudě, frekvenci a fázi, který může být obohacen o šumovou a ruchovou složku. Zvukové signály mohou vznikat přímo nebo mohou být zaznamenány elektroakustickým převodníkem, kterým může být například mikrofon. Zaznamenané nebo uměle syntetizované zvukové signály lze poté zpětně reprodukovat pomocí reproduktoru nebo sluchátek. Zvuk zaznamenaný elektroakustickým převodníkem, nebo zvuk analogově syntetizovaný lze pomocí AD převodníku převést na signál digitální. Digitální zpracování audio signálů nabízí oproti analogovému zpracování řadu výhod, mezi které patří větší preciznost, flexibilita nebo odolnost vůči šumu. Nicméně analogové zpracování zvuku má stále své výhody a v některých aplikacích je nenahraditelné. Převod digitálního zvukového signálu zpět na analogový se provádí pomocí DA převodníku. [1][2]

1.1 Reprezentace zvukového signálu v časové oblasti

Základní a nejpřímočařejší reprezentace zvukového signálu je v časové oblasti, a to jako průběh amplitudy signálu v závislosti na čase (Obr. 1.1). Na časovém průběhu signálu lze snadno pozorovat změny v jeho dynamice. Podle změn dynamiky signálu lze poté odhadnout charakter signálu. Zvukovému signálu budou náležet spíše periodicky opakující se změny, hlukové signály naopak bývají nahodilé. Mezi další parametry, které lze odhadnout z časového průběhu signál patří také počet harmonických složek, přičemž na tvar časového průběhu signálu má vliv jejich počet, amplituda, fáze a přítomnost šumové složky. [1]



Obr. 1.1 Časový průběh harmonického signálu [3]

1.1.1 Časová obálka signálu

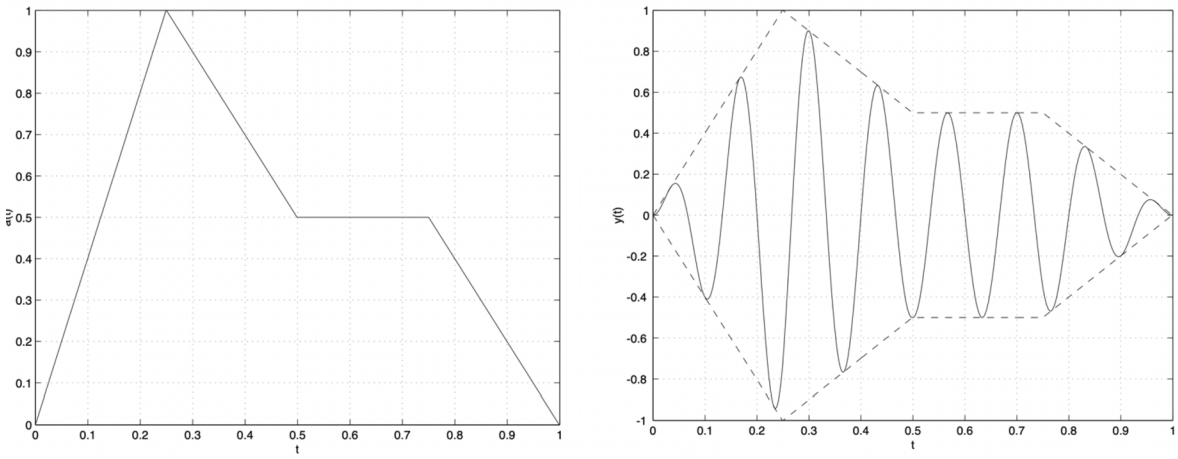
Časová obálka signálu popisuje, jakým způsobem se mění dynamika signálu v závislosti na čase. Hudební zvukový signál lze rozdělit na dílčí elementy, kterými jsou například jednotlivé tóny nebo údery bicích nástrojů. Jejich časový průběh se standardně popisuje pomocí ADSR obálky. Její tvar je popsán následujícími parametry:

- Attack time (A) – čas, ve kterém amplituda dosáhne svého maxima
- Decay time (D) – čas, ve kterém amplituda signálu poklesne na udržovací úroveň danou parametrem *Sustain level*
- Sustain level (S) –úroveň, na kterou signál poklesne po uplynutí *Decay time*, a kterou si signál udržuje po dobu znění
- Release time (R) – doba potřebná pro pokles energie signálu z S na nulu

Časovou obálku signálu můžeme chápat jako funkci *A(t)*, která moduluje amplitudu signálu podle následujícího vzorce. [1]

$$f(t) = A(t) \cdot g(t) \tag{1.1}$$

Kde $A(t)$ reprezentuje modulační funkci a $g(t)$ nemodulovaný signál.



Obr. 1.2 ADSR obálka signálu a výsledný modulovaný signál [3]

1.1.2 Analýza přechodových jevů:

Zobrazení v časové oblasti umožňuje analýzu přechodových jevů, což jsou náhlé změny nebo krátkodobé události v audio signálu. Přechodové jevy mohou zahrnovat perkusivní zvuky, údery nebo jiné rychlé útoky. Ty lze vizuálně identifikovat jako ostré hroty nebo náhlé změny v průběhu signálu.

1.1.3 Efekty založené na čase:

Reprezentace v časové oblasti je užitečná pro pochopení časově založených efektů, jako je zpoždění, ozvěna a dozvuk. Tyto efekty zavádějí změny v časové oblasti, což vede k opakování nebo odrazům v průběhu vlny.

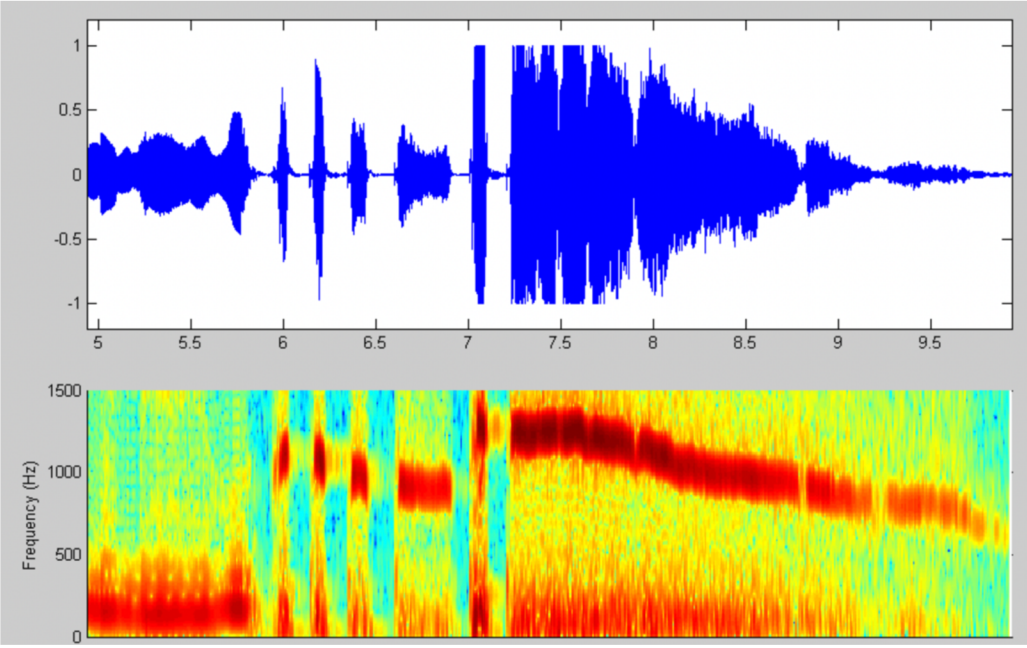
1.1.4 Úpravy a manipulace:

V časové oblasti je možné provádět přesné úpravy a manipulaci se zvukovými signály. Výběrem konkrétních částí průběhu je možné provádět operace, jako je stříhání, kopírování nebo vkládání jiných částí zvukové stopy, popřípadě použití různých zvukových efektů.

Reprezentace časové oblasti sice poskytuje cenné informace o zvukovém signálu, ale má omezení, pokud jde o analýzu frekvenčního obsahu a složitých harmonických struktur. Pro podrobnou frekvenční analýzu se běžně používají techniky založené na Fourierově transformaci, jako je reprezentace frekvenční oblasti (např. spektrogramy), ve spojení s analýzou časové oblasti.

1.2 Reprezentace zvuku ve frekvenční oblasti

Reprezentace zvuku ve frekvenční oblasti zkoumá jednotlivé složky signálu a jejich uspořádání. Uspořádání frekvenčních složek se nazývá frekvenční struktura a její zápis se označuje jako frekvenční spektrum. Pokud se spektrum zvukového signálu mění v závislosti na čase, lze charakter zvuku popsat pomocí spektrogramu. [1]



Obr. 1.3 Zvukový signál a jeho spektrogram

1.3 Digitální zpracování zvukového signálu

Digitální systémy pracují s diskrétními signály. Diskrétním signálem rozumíme signál, který se na rozdíl od analogového nemění souvisle s časem. Signál, jehož hodnota se mění v izolovaných okamžicích nazýváme vzorkovaný. Signál, který v určitém okamžiku nabývá pouze jedné z konečného počtu hodnot nazýváme kvantovaný. V praxi užívaný digitální signál je jejich kombinací. Digitální signál lze tedy popsat jako konečnou posloupnost celých čísel z určitého intervalu. [4]

1.3.1 Fourierova transformace

Fourierova transformace je matematická integrální transformace, která převádí digitální signál mezi časově a frekvenčně závislým vyjádřením pomocí harmonických signálů. [4]

1.3.2 Diskrétní Fourierova transformace

Kmitočtová analýza diskrétních signálů se běžně provádí na počítačích nebo mikroprocesorech a pro výpočet Fourierovy transformace je potřeba znalost matematického vyjádření signálu nebo spektra. Při zpracovávání naměřeného signálu nebo spektra ve formě vzorků je tedy zapotřebí použít numerickou metodu známou jako diskrétní Fourierova transformace (DFT), která tento problém řeší. DFT našla své uplatnění především po rozvoji výpočetní techniky a v dnešní době jsou procesory realizující DFT součástí řady přístrojů. Výpočet DFT obsahuje velké množství operací a je výpočetně velmi náročný. Obecně lze říci, že DFT je metoda, která podle rovnice 1.2 konvertuje sekvenci N komplexních čísel $x_0, x_1, \dots, x_{N-1}$ na novou sekvenci N komplexních čísel. [4]

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi k n}{N}} \qquad \text{Pro } 0 \leq k \leq N-1 \qquad (1.2)$$

Kde x_i jsou považovány za hodnoty funkce nebo signálu ve stejně vzdálených časech $t = 0,1, \dots, N-1$.

Výstup X_k je komplexní číslo, které kóduje amplitudu a fázi sinusové vlny s frekvencí $\frac{k}{N}$ cyklů za časovou jednotku. Výsledkem výpočtu X_k je nalezení koeficientů aproximace signálu lineární kombinací takových vln. Protože každá vlna má celočíselný počet cyklů za N časových jednotek, bude aproximace periodická s periodou N.

1.3.3 Rychlá Fourierova transformace

V dnešní době se k výpočtu DFT používá algoritmus FFT (*Fast Fourier Transform*), který vyžaduje zlomek nutných matematických operací v porovnání s DFT a který je nyní nejrozšířenějším způsobem výpočtu Fourierovy transformace. Rozdíly v rychlosti výpočtu mohou být obrovské, obzvláště pro větší objemy dat. Algoritmus FFT se běžně používá ve všech rozšířených komerčních matematických programech, kterými jsou například Maple nebo Matlab. [4]

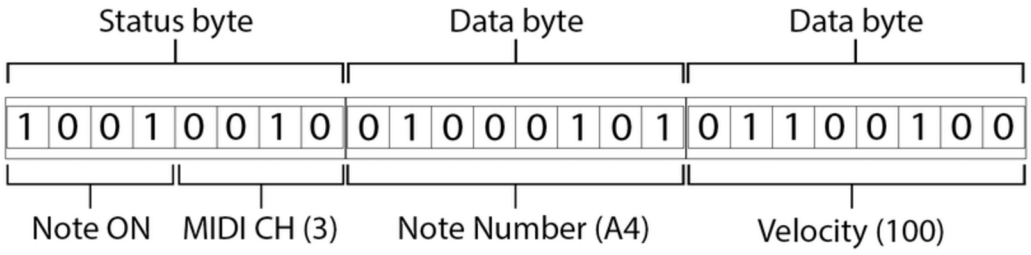
1.4 MIDI protokol

Musical instrument digital interface (zkráceně MIDI) je univerzální komunikační protokol mezi řídícím zařízením, a podřízeným zařízením. Tento komunikační protokol slouží primárně pro výměnu dat a synchronizaci mezi elektronickými hudebními nástroji. MIDI protokol jednotně a časově přesně kóduje audio data a události, kterými můžou být například stisknutí klávesy, změna polohy potenciometru nebo změna hlasitosti. V praxi MIDI protokol funguje tak, že jedno hlavní zařízení (*master*) dává příkazy, a ostatní zařízení (*slave*) tyto příkazy vykonávají.

Protokol také obsahuje příkazy, které mohou uživatelé protokolu aplikovat dle své vlastní vůle. Díky tomu lze MIDI protokol použít například k synchronizaci osvětlení nebo k ovládání jiných zařízení. [5][6]

1.4.1 MIDI zpráva

MIDI zpráva se skládá z několika bytů. První stavový byte popisuje typ zprávy. Následuje datový byte, který přenáší konkrétní informaci. V případě, že zpráva obsahuje mnoho dat, lze využít více datových bytů. Každá MIDI zpráva se tudíž skládá z právě jednoho stavového bytu a jednoho či více bytů datových (Obr 1.4). V jednom MIDI bytu je pro kódování přenos dat k dispozici 7 bitů. Každý z nich může pro určitý parametr nabývat 128 různých hodnot. V případě, že je 128 hodnot málo, je možné využít dva byty, a tudíž moci použít až 16384 různých hodnot pro jeden parametr. [5]



Obr. 1.4 Struktura MIDI zprávy [7]

1.5 OSC protokol

Open Sound Control (OSC) je protokol pro komunikaci mezi počítači, zvukovými syntezátory a dalšími multimediálními zařízeními, který je optimalizován pro moderní síťové technologie. Poprvé byl vyvinut v centru nové hudby a zvukových technologií CNMAT na univerzitě v Berkley. Jeho původním účelem bylo sdílet data, která zahrnují například noty, sekvence, gesta a mnoho dalších parametrů mezi hudebními nástroji. OSC ve své nejzákladnější podobě je protokol zasílání zpráv typu peer-to-peer. Jedna věc, která jej odlišuje od ostatních je to, že je open-ended a dynamický. Běžně používané protokoly jako například DMX nebo MIDI, jsou výborné v tom, k čemu jsou určeny. Nelze s nimi však odeslat současně číselnou hodnotu i textový řetězec, protože na rozdíl od OSC zpráv mají pevně danou velikost. Zpráva OSC se skládá ze dvou hlavních složek. Z adresy a následně z dat samotných. Adresa je velmi podobná například URL, kde jsou jednotlivé složky a podsložky odděleny lomítkem. U dat nehraje rozdíl jejich množství a ani to, jestli jsou typem textový řetězec nebo číselná hodnota. Další výhodou tohoto protokolu je možnost využití dat v rozlišení pouze 64 bit u libovolného datového typu – na rozdíl od 128 hodnot při použití MIDI.

Jednotka přenesená pomocí OSC se nazývá paket. Jedná se o souvislý blok binárních dat obsahující nejen data samotná, ale i informaci o velikosti paketu. Paket může být buď samostatný příkaz nebo zpráva která, je zpracována serverem, nebo bundle – série zpráv, které mají být provedeny současně. Bundle obsahuje 64bitovou časovou značku.

OSC je protokol nezávislý na druhu transportu. To znamená, že ve skutečnosti nezáleží na tom, jak se zpráva dostane z bodu A do bodu B. V dnešní době to znamená, že OSC zprávy se v naprosté většině případů posílají po internetové síti. Je tedy zapotřebí zprávy nasměrovat na správnou IP adresu a na správný port. Zařízení přijímající zprávy OSC buď poskytne výchozí číslo portu, nebo jej lze zvolit ručně. [8]

1.6 Digital Audio Workstation

Digital Audio Workstation (DAW), je elektronické zařízení nebo aplikační software používaný pro nahrávání, úpravu, produkci a postprodukci zvukových souborů. DAW přicházejí v široké škále konfigurací od jediného softwarového programu na notebooku přes integrovanou samostatnou jednotku až po vysoce komplexní konfiguraci mnoha komponent řízených centrálním počítačem. Bez ohledu na konfiguraci mají moderní DAW centrální rozhraní, které umožňuje uživateli upravovat a míchat více nahrávek a stop do finální stopy.

Pojem DAW v dnešní době nejčastěji odkazuje přímo na software, ale tradičně má počítačový DAW čtyři základní součásti: počítač, zvukovou kartu nebo jiné zvukové rozhraní, software pro úpravu zvuku a alespoň jedno uživatelské vstupní zařízení pro přidávání nebo úpravu dat, kterým může být například klávesnice nebo MIDI ovladač.

Počítač funguje jako hostitel zvukové karty, zatímco software poskytuje rozhraní a funkce pro úpravu zvuku. Zvuková karta obvykle převádí analogové zvukové signály do digitální podoby a digitální zpět na analogový zvuk při přehrávání. Může také pomoci při dalším zpracování zvuku. Software řídí všechny související hardwarové komponenty a poskytuje uživatelské rozhraní umožňující nahrávání, úpravy, přehrávání nebo práci s MIDI a OSC zprávami.

Počítačové DAW mají rozsáhlé možnosti nahrávání, úpravy, přehrávání a některé mají také funkce související s videem. Mohou například poskytnout prakticky neomezený počet stop pro nahrávání, polyfonii, rozsáhlé efektové jednotky, virtuální syntezátory nebo nástroje založené na nahrávkách, které lze použít pro tvorbu hudby. Pro integraci pluginů do DAW je aktuálně nejrozšířenější softwarové rozhraní VST. [9]

1.6.1 Ableton Live

Ableton Live je populární DAW, které nabízí řadu funkcí pro hudební produkci a živé vystupování. Ableton Live disponuje několika režimy. Režim *Arrangement View* je lineární rozhraní založené na časové ose, které se podobá tradičnímu rozvržení DAW. Umožňuje uspořádat a upravit hudební nápady do strukturované kompozice. *Session View* je rozhraní založené na mřížce, ve které lze organizovat a spouštět zvukové klipy, MIDI klipy a různá zařízení. Je to hlavní režim používaný pro živé vystupování a improvizaci, a zároveň byl tento režim použit při vizualizaci hudby v rámci této diplomové práce.

Klipy: V zobrazení *Session View* lze vytvářet zvukové nebo MIDI klipy, které nesou informaci o zvukovém obsahu v podobě jednorázového přehrání nebo smyčky. Tyto klipy lze umístit do různých buněk v mřížce. Každá buňka představuje určité místo a dobu trvání v rámci hudební časové osy.
Scény: Scéna je vodorovná řada složená z několika klipů. Klipy je možné uspořádat do různých scén a vytvořit tak různé hudební úseky nebo variace. Scény se obvykle používají ke spuštění více klipů současně, což umožňuje dynamické a spontánní uspořádání během živých vystoupení.
Spouštění: Klipy se spouští jednotlivě nebo ve skupinách (scénách) kliknutím na příslušná tlačítka spuštění. To umožňuje spouštět různé hudební prvky v určitém čase a vytvářet tak nelineární zážitek z vystoupení. Klipy lze přehrávat, zastavovat a nahrávat do smyčky v reálném čase.
Efekty a zařízení: Ableton Live poskytuje širokou škálu vestavěných zvukových efektů, nástrojů a MIDI efektů, které lze aplikovat na jednotlivé stopy nebo na hlavní stopu. Tyto efekty lze ovládat a modulovat v reálném čase, což dodává vystoupení kreativitu a variabilitu.

2. DIGITÁLNÍ VIZUALIZACE ZVUKOVÉHO SIGNÁLU

Vizualizace zvuku, respektive hudby, může nabývat různých podob. Zjednodušeně by se dala rozdělit na dvě základní formy – a to formu uměleckou a technickou.

Technická vizualizace zvukového signálu je vizuální reprezentace například časového průběhu signálu nebo obálky jeho časového průběhu. Nejčastěji se provádí s pomocí osciloskopu. Ve své nejstarší umělecké podobě se jedná o tanec. Dále jde o statická výtvarná díla inspirovaná hudbou, nebo grafické partitury. S první elektronickou dynamickou vizualizací hudby přišla v roce 1976 firma Atari. Atari Video Music bylo elektronické zařízení, které po připojení k HiFi systému promítalo na display vizuální animace, které reagovaly na výstup z HiFi systému. S rozvojem výpočetní techniky a digitálních systémů se začaly vizualizace hudby objevovat v běžně dostupných softwarových multimediálních přehrávačích. Jako vhodné příklady lze uvést například Winamp, Soundjam nebo Windows Media Player.

Vizualizace hudby má v dnešní době velký vliv na téměř všechny oblasti hudebního průmyslu. Lze ji zaznamenat na koncertech v podobě LED panelů, speciálně ovládaných světel, laserů nebo projekce video sekvencí. Významnou roli pro mnoho posluchačů hraje vizuální podoba hudby na internetu, a to nejen v oblasti videoklipů, ale i v oblasti streamovacích platforem. Videoklipy samotné jsou velmi drahé a časově náročné na výrobu. U poměrně velkého množství skladeb lze tedy pozorovat pouze jednoduchý audioreaktivní vizuál. Svůj vliv na vizualizaci hudby měla i pandemie viru SARS-CoV-2 která v podstatě znemožnila navštěvování živých koncertů a podnítila silnou vlnu nových způsobů vizualizace hudby a její přenos do virtuálního prostředí.

2.1 Světelná vizualizace zvukového signálu

Světelnou vizualizací zvukového signálu myslíme především osvětlování koncertů, popřípadě divadelních představení. Osvětlování začalo velmi jednoduše. Dříve se ovládalo pomocí velkých panelů, které vyžadovaly ruční manipulaci, přičemž každé světlo a parametr byly ovládány samostatně, což bylo velmi neefektivní. S vývojem elektroniky byly vytvořeny elektronické stmívače, které mohly zajistit počítačové stmívání mnoha kanálů světla současně. To však vedlo k dalšímu problému. Každý výrobce měl jiný protokol a systémy nebyly vzájemně kompatibilní.

2.1.1 DMX protokol

Jak průmysl rostl a osvětlení se stávaly stále složitějšími, byla potřeba křížově kompatibilního vybavení ještě důležitější. V roce 1986 USITT Engineering Commission sponzorovala zasedání na výroční konferenci v Oaklandu v Kalifornii. Od této relace začal projekt, jehož výsledkem byl USITT DMX512 – standard pro digitální přenos

dat pro stmívače a ovladače. Tento protokol prošel několika revizemi, byl široce přijat a je dnes nejrozšířenějším řídicím protokolem v oblasti osvětlování.

Protokol DMX je ve skutečnosti velmi jednoduchý. Je rozdělen na dva samostatné komponenty, a to adresu a hodnotu, která je s touto adresou spojena. Protokol DMX 512 má 512 adres, z nichž každá může nabývat hodnotu mezi nulou a 255, což odpovídá 8 bitům. Jednou z důležitých vlastností DMX je schopnost řetězení. To znamená, že lze propojit mnoho zařízení dohromady v řetězci, přičemž výstup z jednoho zařízení je propojen se vstupem jiného zařízení. Na konci řetězce poté musí být zakončovací odpor. Bez řádného ukončení se signál odráží zpět a může zmást zařízení k němu připojená. To může způsobit nestabilitu, blikání a další problémy.

Moderní světla mají mnoho parametrů, které můžeme ovládat, jako je například natočení ve dvou osách, barva a mnoho dalšího. Každý z těchto parametrů může být pomocí tohoto protokolu řízen osvětlovacím pultem. Protokol DMX posílá všechny informace všem zařízením, je na přijímajícím zařízení, aby si vzalo, co potřebuje. Osvětlovací pulty obsahují operační systém, který umožňuje naprogramovat, co je zapotřebí, aby světla dělala, a poté tyto příkazy převést na DMX signál. Počáteční DMX adresu pro každé světlo lze nastavit na osvětlovacím pultu. [10]

2.2 Softwarová vizualizace zvukového signálu

Kromě světelné vizualizace se v dnešní době hojně využívá vizualizace softwarové pomocí speciálních obrazových efektů (VFX). Díky komunikačním protokolům a široké kompatibilitě je možné vytvářet speciální obrazové audioreaktivní scény nejen v softwaru, který je k tomu přímo určen, ale téměř v libovolném prostředí, které umí zpracovávat audio data, nebo data odeslaná pomocí komunikačního protokolu přijímat a dále s nimi pracovat. Programy využívají samostatnou audio stopu a její spektrogram, nebo mohou pracovat například s MIDI a OSC zprávami, které mění parametry scény, nebo spouští události. Některý software dokonce umožňuje synchronizaci se světly pomocí DMX protokolu.

2.2.1 Resolume Arena

Resolume Arena je standardem mezi VJ softwarem. Nabízí intuitivní grafické rozhraní pro práci se statickými (tj. negenerativními) obrazy nebo videem a je vnímán jako silný nástroj při jejich prolínání a efektování. Program umí v živém čase analyzovat audio stopy a měnit parametry vizuální scény na základě jen určité části spektra signálu. Užitečnou funkcí může být například možnost promítání více projektory, které umožňuje projekci v úhlu až tři sta šedesát stupňů, nebo možnost projektory mapovat na libovolné objekty. Resolume Arena nabízí velké množství efektů, pluginů a podporuje pluginy vlastních i třetích stran. Lze také integrovat pluginy naprogramované v OpenGL. Software lze ovládat pomocí DMX přímo z osvětlovacího pultu, pomocí MIDI ovladače nebo přes telefon pomocí OSC zpráv. [11]

2.2.2 Max MSP

Software Max MSP je vizuální programovací prostředí pro Windows a MacOS, které je speciálně uzpůsobené pro práci s multimediálním obsahem, především hudbou a videem. Je hojně využíván skladateli, umělci a programátory k tvorbě pluginů, performancí a instalací. Max je modulární systém, který pracuje z velké části se sdílenými knihovnami. Umožňuje nejen tvorbu funkcí a programů, ale také tvorbu jejich grafických rozhraní. To umožňuje jejich praktické použití v kreativním procesu i uživateli, kteří nerozumí celému kódu. Za dobu svého fungování si vytvořil velkou uživatelskou základnu programátorů, která ho obohatila množstvím volně dostupných rozšíření, a navíc disponuje výbornou dokumentací.

Programy a funkce vytvořené v Maxu (patchery) lze vyexportovat do samostatných aplikací a distribuovat zdarma nebo i prodávat komerčně. Kromě toho lze Max použít k vytváření audio a MIDI pluginů pro DAW Ableton Live prostřednictvím jeho rozšíření Max for Live. [12]

2.2.3 Pure data

Pure Data (Pd) je open source vizuální programovací prostředí, které lze spustit na široké škále zařízení od osobních počítačů po vestavěná zařízení (např. Raspberry Pi) a chytré telefony. Je to hlavní větev rodiny *patcher* programovacích jazyků známé jako Max, původně vyvinuté Millerem Puckettem.

Pd je stejně jako Max vizuální programovací prostředí, které umožňuje hudebníkům, vizuálním umělcům a vývojářům vytvářet software graficky bez psaní řádků kódu. Pd lze použít ke zpracování a generování zvuku, videa, 2D/3D grafiky a lze bezproblémově využít vstupních zařízení a MIDI. Pd může snadno pracovat přes místní a vzdálené sítě a ovládat motorové systémy, osvětlovací zařízení a další vybavení. Umožňuje také vybrat si množství vyrovnávací paměti, které chcete věnovat jedné instanci. Je vhodný pro osvojení základních metod zpracování multimédií a vizuálního programování i pro realizaci složitých systémů pro rozsáhlé projekty. [13]

2.2.4 Unity

Unity je multiplatformní herní engine vyvinutý společností Unity Technologies, poprvé vydaný v červnu 2005 na Apple Worldwide Developers Conference jako herní engine pro Mac OS X. Od té doby byl engine rozšiřován tak, aby podporoval různé desktopové, mobilní, konzolové a virtuální platformy. Je obzvláště populární pro vývoj mobilních her pro iOS a Android. Díky snadnému používání je vhodný pro začínající vývojáře a oblíbili si ho i nezávislí tvůrci her.

Engine lze použít k vytváření trojrozměrných (3D) a dvourozměrných (2D) her, stejně jako interaktivních simulací. Byl použit ve spoustě dalších odvětvích, jako je film, automobilový průmysl, architektura, strojírenství, stavebnictví nebo ozbrojené síly Spojených států. [14]

2.2.5 Unreal Engine

Unreal Engine (UE) je 3D počítačový grafický herní engine vyvinutý společností Epic Games, poprvé představený v roce 1998 ve hře typu FPS. Původně byl vyvinut pro PC FPS hry. Od té doby se používá v různých žánrech her a dočkal se přijetí v jiných odvětvích, zejména ve filmovém a televizním průmyslu. Unreal Engine, napsaný v jazyce C++, se vyznačuje vysokou mírou přenositelnosti a podporuje širokou škálu desktopů, mobilních zařízení, konzolí a platforem virtuální reality. [15]

Nejnovější generace, Unreal Engine 5, byla uvedena na trh v dubnu 2022. Používá vizuální skriptovací systém *Blueprints*, který funguje na podobném principu jako skriptovací systémy programů jako Max MSP a Pure Data. Jeho zdrojový kód je dostupný na GitHubu po registraci účtu a komerční využití je zaručeno na základě licenčního modelu. Epic se vzdává své licenční marže za hry, dokud vývojáři nevydělají 1 milion USD v příjmech. [16]

Unreal Engine byl také používán obory mimo herní a umělecké zaměření díky jeho dostupnosti a sadám funkcí. Byl použit jako základ pro nástroj virtuální reality pro zkoumání molekul farmaceutických léčiv ve spolupráci s dalšími výzkumníky, virtuální prostředí pro zkoumání a navrhování nových budov a automobilů, anesteziologický školící software pro americké armádní lékaře nebo simulace místa činu pro více hráčů vyvinutá Akademií FBI. [17]

Tento software byl kvůli svým širokým možnostem využití zvolen pro tvorbu vizuálních efektů v rámci této diplomové práce.

3. NÁSTROJE PRO TVORBU VFX V PROSTŘEDÍ UNREAL ENGINE 5

V této kapitole jsou popsány jednotlivé nástroje, které se běžně využívají k tvorbě speciálních efektů v UE. Jednotlivé nástroje lze poté kombinovat různými způsoby.

3.1 Částicový systém Niagara

Částicový systém Niagara, který je přímou součástí UE, je jeden z nejpoužívanějších modulů UE pro tvorbu vizuálních efektů. Částicové systémy se používají ve fyzice počítačových her, počítačové grafice, animaci nebo jako speciální efekty pro multimediální tvorbu.

Částicové systémy jsou definovány jako skupina bodů v prostoru, které se řídí souborem pravidel definujících chování a vzhled. Částicové systémy modelují jevy jako mračno částic, používající stochastické procesy ke zjednodušení definice dynamického systému a mechaniky tekutin, které je obtížné reprezentovat afinními transformacemi. Typicky to bývají vysoce chaotické systémy, přírodní jevy nebo procesy způsobené chemickými reakcemi. [18]

Na obrázku 3.1 lze vidět modul pro tvorbu částicového systému. Tento modul obsahuje několik dílčích částí – *Properties, Emitter Spawn, Emitter Update, Particle Spawn, Particle Update* a *Render*. Každý částicový systém se skládá z jednoho nebo více dílčích emitorů. V části *Properties* se nastavují obecné vlastnosti daného emitoru. *Emitter spawn* určuje místo a podmínky, za jakých se emitor objeví. Moduly ve skupině *Emitter Update* moduly kontrolují a mění parametry emitoru s každým snímkem. *Particle Spawn* se využívá jen jednou pro každou vytvořenou částici. Moduly v této části nastavují počáteční hodnoty jednotlivých částic. *Particle update* moduly poté aktualizují hodnoty parametrů částic každý jednotlivý snímek. Poslední část *Render* umožňuje nastavit parametry pro vykreslování (rendering). Všechny moduly ve všech částech se spouštějí v pořadí od shora dolů. Je potřeba brát v potaz, že krom nativních UE modulů je možné vytvořit vlastní moduly pro jakoukoli část systému Niagara pomocí bloku *Scratch Pad*, který dovoluje vytvářet uživatelské funkce pomocí systému *Blueprints*.

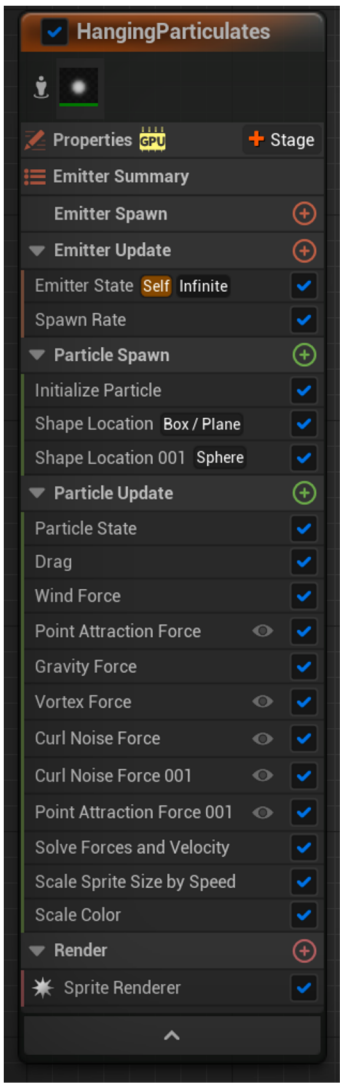
3.2 Dynamické materiálové instance

Materiál v UE je sada pokynů, které definují, jak má povrch vypadat při renderingu. Materiál může obsahovat textury, parametry a funkce, které řídí, jak povrch odráží světlo, jak vypadá za různých světelných podmínek a jak reaguje na interakce uživatele.

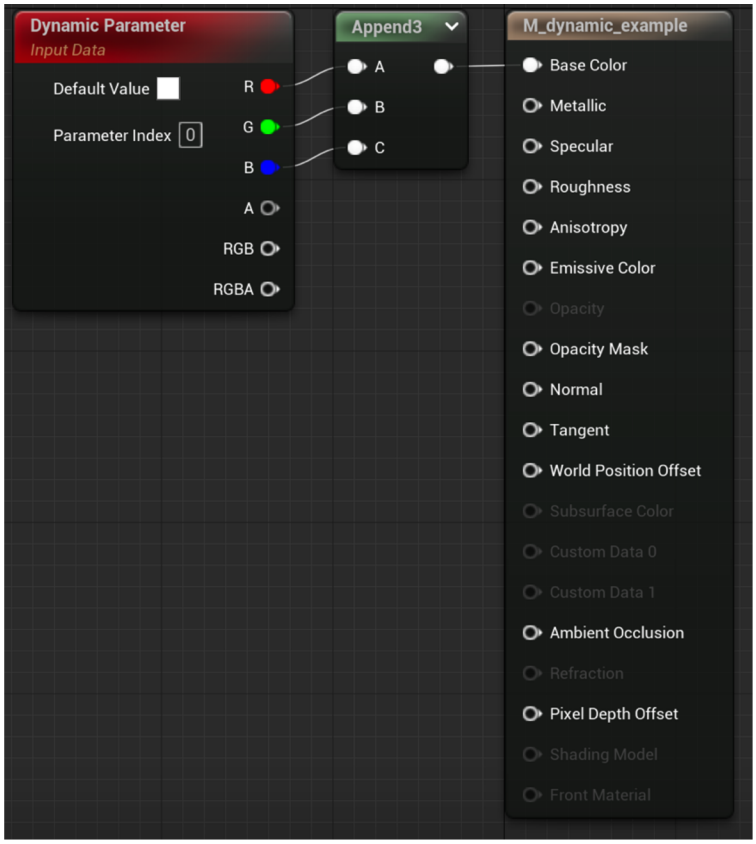
Instance materiálu je kopie materiálu, kterou lze za běhu upravovat. Tato instance dědí všechny vlastnosti rodičovského materiálu, ale může mít vlastní sadu parametrů, které lze upravovat nezávisle na rodičovském materiálu. To umožňuje vytvářet varianty

téhož materiálu, které mají různý vzhled na základě herních událostí nebo uživatelských vstupů.

Dynamickými materiálovými instancemi rozumíme materiálové instance, jejichž vlastnosti můžeme upravovat v reálném čase pomocí Blueprintu nebo C++. Od klasických materiálových instancí se liší tím, že obsahují dynamické parametry (Obr.3.2). Ty mohou být různých typů, například skalární parametry, vektorové parametry, parametry textur a mnoho dalších. Každý parametr může mít výchozí hodnotu a rozsah možných hodnot, které lze nastavit v kódu. Dynamické materiály lze použít k vytvoření různých efektů. Jako příklad lze uvést animované textury, které v průběhu času mění svoji barvu. Dále lze uvést morfující textury, které v průběhu času mění tvar nebo formu, popřípadě textury, které mění svoje atributy při poškození. Dynamické parametry nabízejí výkonný a flexibilní způsob ovládání vzhledu materiálů v UE a lze je použít k vytváření dynamických, realistických prostředí. [19]



Obr. 3.1 Modul pro tvorbu částicového systému



Obr. 3.2 materiál s dynamickým parametrem *Base Color*

3.3 Post process efekty

Post process efekty v UE představují soubor nástrojů a technik, které se používají ke zlepšení vizuální kvality hry nebo simulace. Tyto efekty se aplikují až po renderu scény a umožňují dodatečnou kontrolu nad výsledným obrazem. Efekty následného zpracování umožňují umělcům a návrhářům definovat celkový vzhled a dojem ze scény pomocí kombinovaného výběru vlastností a funkcí, které ovlivňují barvy, tónování, osvětlení nebo hloubku ostrosti. Pro přístup k těmto funkcím je potřeba do úrovně přidat *Post Process Volume*, což je objekt, který definuje oblast, kde jsou postprocesní efekty aktivní. Lze umístit více takových objektů a samostatně tak definovat vzhled určité oblasti, popřípadě je lze nastavit tak, aby ovlivňovaly celou scénu. Tvorba post process efektu probíhá v editoru materiálu, kde je materiálová doména změněna na *Post Process.* [20]

4. NAVRŽENÝ SYSTÉM VIZUALIZACE HUDBY

Hudba je jedinečná umělecká forma vyjádření. Má sílu ovlivňovat emoce, odvést člověka do imaginárních světů, zahalit publikum do časového zvukového zážitku. Na rozdíl od sochy nebo obrazu je živá hudba omezena na daný časový úsek. S postupem času se neustále mění, roste a ustupuje na intenzitě. Jako dočasné médium je omezena na postupné plynutí času. Vizualizace hudby poskytuje alternativní médium, které dokáže zachytit krátký snímek nebo historii hudebního vystoupení. Následující kapitola popisuje konkrétní řešení vizualizace hudby začleněním 3D počítačové animace jako vizuálního rozšíření živého hudebního vystoupení.

4.1 Výběr vhodného řešení a prostředí realizace

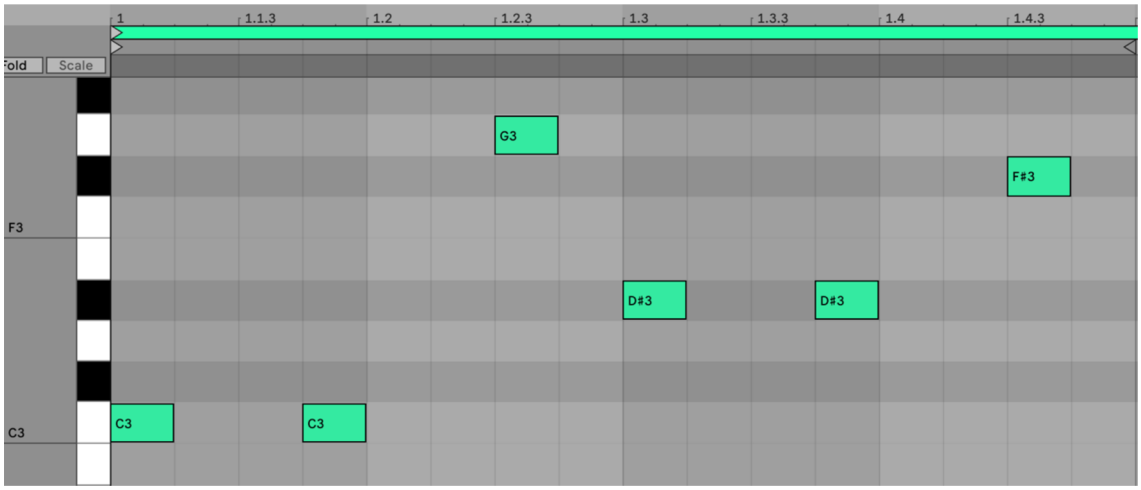
Před začátkem práce na konkrétním řešení vizualizace hudby je zapotřebí se zamyslet, k jakému účelu a za jakých podmínek bude sloužit. Je velmi obtížné realizovat jeden univerzální nástroj pro všechna použití, a proto se tato práce zaměřuje na jednu konkrétní oblast, a to na tvorbu nástroje pro generativní 3D vizualizaci audio dat pro DJe a hudební producenty pracující v DAW Ableton. Ten není pouze prostředí pro tvorbu, úpravu, mix nebo nahrávání hudby. Od ostatních DAW se liší díky sloupcovému uspořádání a skýtá zázemí nejen producentům a muzikantům, ale také DJům, kteří k hraní nevyužívají již vytvořené skladby, ale hudbu budují živě pomocí MIDI smyček a samplů. Jeho hlavní výhodou je vlastní programovací prostředí, ve kterém lze vytvářet uživatelské zásuvné moduly.

Nástroj vytvořený v rámci této práce bude možné použít pro komunikaci mezi DAW Ableton a prostředím Unreal Engine 5, ve kterém bude probíhat generativní 3D vizualizace hudebních dat. Možné využití se poté nabízí hned v několika oblastech, a to především při vizualizaci živého vystoupení, k tvorbě audioreaktivních vizuálů a videoklipů pro především elektronickou hudbu, nebo k tvorbě multimediálních instalací.

4.2 Získání hudebních dat

K vytvoření vizuální scény podle hudebních dat je nutné nejdříve data získat a vhodným způsobem je třídit. Obecně lze druhy hudebních dat v Abletonu rozdělit na dvě skupiny. Prvním druhem jsou vzorky (samply), které jsou v podstatě předem nahraný hudební signál reprezentován jeho časovým průběhem. Druhým druhem jsou hudební MIDI data. Ta v Abletonu reprezentují okamžik stisku noty, sílu stisku, jeho výšku a další data. Typicky bývají zaznamenána v tzv. *piano rollu* (Obr.4.1). Tyto signály potom mohou spouštět samplované zvuky nebo zvuky z virtuálních syntezátorů. Rozšíření Abletonu

Max for Live nabízí možnost vytvoření vlastních nástrojů, které lze v prostředí přímo implementovat. V tomto prostředí byly vytvořeny nástroje, které zaznamenávají oba druhy hudebních dat po lokální síti je odesílají k dalšímu zpracování.



Obr. 4.1 Piano roll v prostředí Ableton

4.2.1 OSC MIDI sender

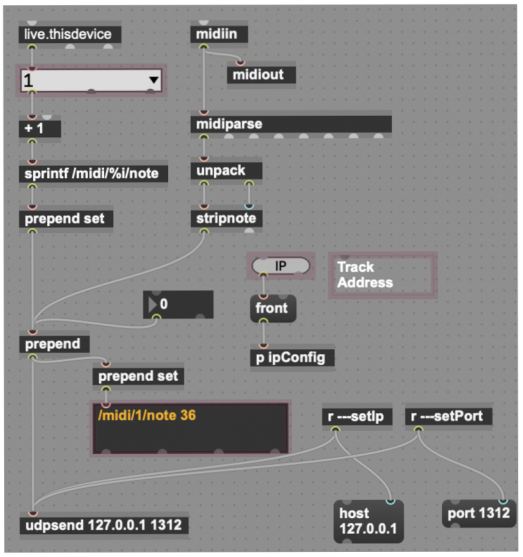
OSC MIDI sender je nástroj, který byl vytvořen k zaznamenání MIDI zpráv typu *Note ON/OFF* a data v nich obsažená. Dále umožňuje uživatelsky nastavit identifikátor *Track adress*, který v budoucnu usnadní přiřazování vizuálních efektů jednomu nebo více nástrojům najednou. Tato data se poté společně posílají po lokální síti jako OSC zprávy, které bude následně zpracovávat Unreal Engine.

4.2.2 OSC Envelope sender

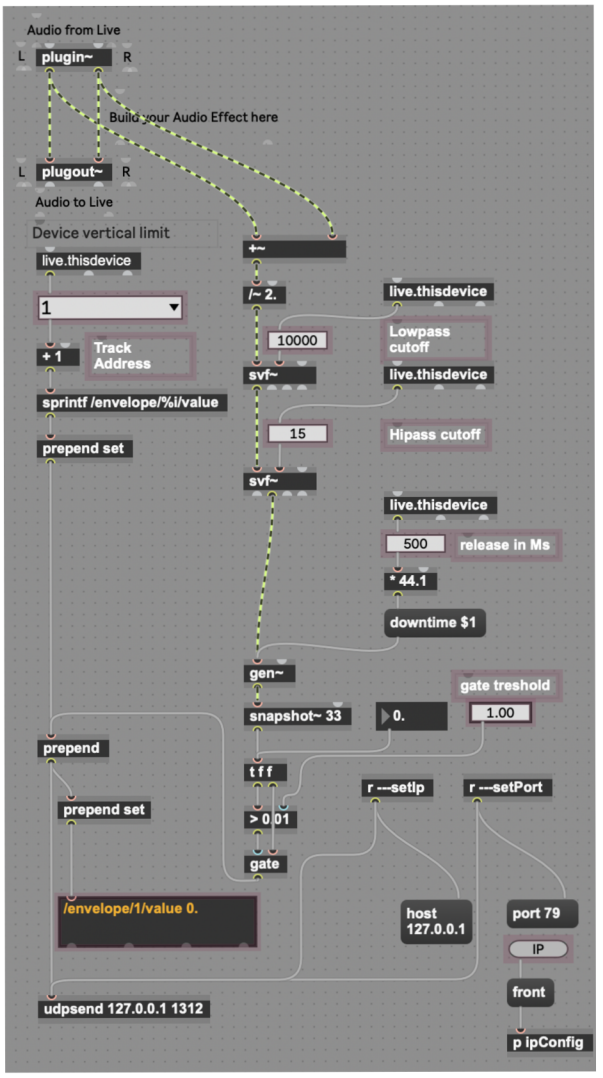
OSC Envelope sender je nástroj, který byl vytvořen ke zpracovávání hodnoty amplitudy signálu. Nejprve je stereo audio signál sečten do jedné stopy, kvůli zachování původní dynamiky signálu je následně dělen dvěma. Poté je pomocí bloku *snapshot* převeden na číselnou hodnotu. Aktuální hodnota obálky signálu je posílána po lokální síti stejným způsobem, jako u MIDI senderu (4.2.1).

Pro snazší práci při použití živých bicích nástrojů nebo předem nahraných hudebních smyček byl také vytvořen *Gated envelope sender*, který funguje obdobným způsobem, ale byl obohacen o kmitočtový filtr a šumovou bránu. Mezní frekvence filtru lze uživatelsky nastavit, stejně jako prahovou hodnotu šumové brány. Pomocí filtru lze velmi snadno zpracovávat pouze část frekvenčního spektra, která může odpovídat například jednomu určitému bicímu nástroji. Šumová brána poté zajišťuje, aby se do obálky nedostaly nechtěné přeslechy.

Na následujících obrázcích (Obr.4.2 a Obr.4.3) lze vidět Patchery vytvořených nástrojů pro odesílání hudebních dat z Abletonu.



Obr. 4.2 OSC MIDI Sender



Obr. 4.3 OSC Gated envelope sender

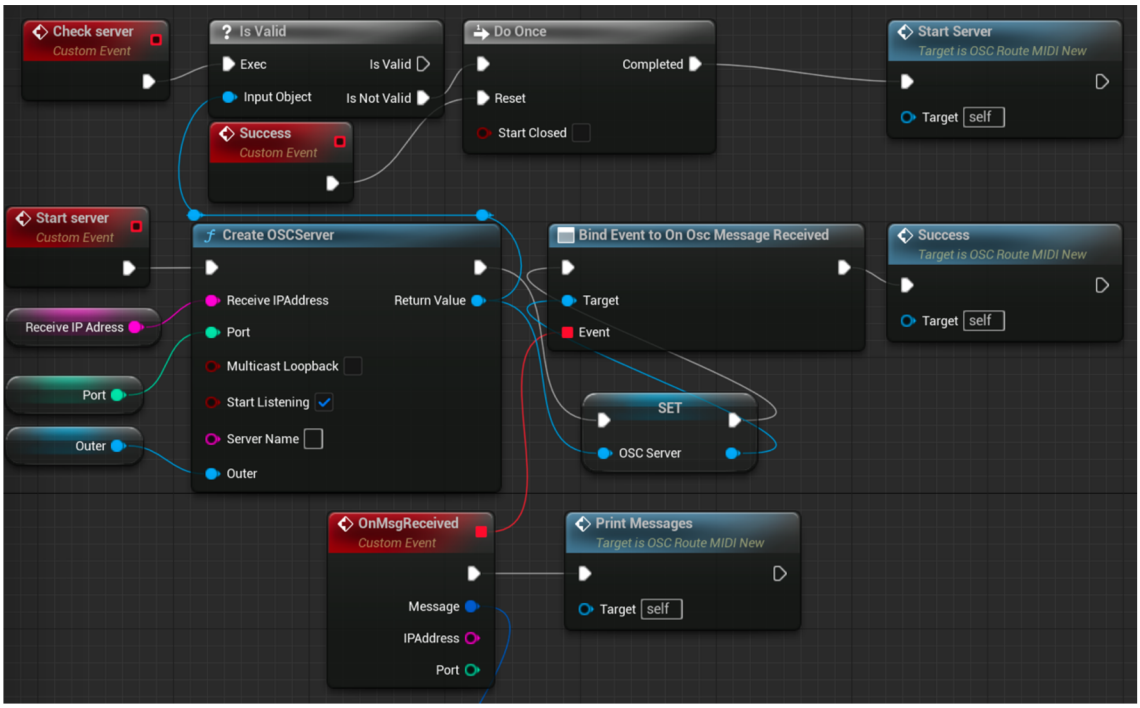
4.3 Zpracování hudebních dat

Ke zpracování OSC zpráv a jejich následnému použití při vizualizaci dochází v prostřední Unreal Engine, verze 5.1.1. Tento software využívá skriptovací systém *Blueprints*, v kterém lze vizuálně programovat za pomocí bloků a uzlů, podobně jako u softwaru Max.

4.3.1 OSC server

Blok *OSC server* v UE funguje jako koncový bod naslouchající zprávám odeslaným do místní instance UE. Podporuje multicastovou smyčku a analýzu OSC zpráv a svazků. Umožňuje také serveru konkrétně určit, které IP adresy bude naslouchat (whitelisting).

Při každém spuštění simulace je zavolána funkce *Start server* a při každém snímku obrazovky je zavolána funkce *Check server*. Server naslouchá OSC zprávám a na jejich základě poté spouští další funkce. V případě, že zpráva není validní, dochází k restartu serveru, což zajišťuje stabilní chod programu.



Obr. 4.4 OSC server

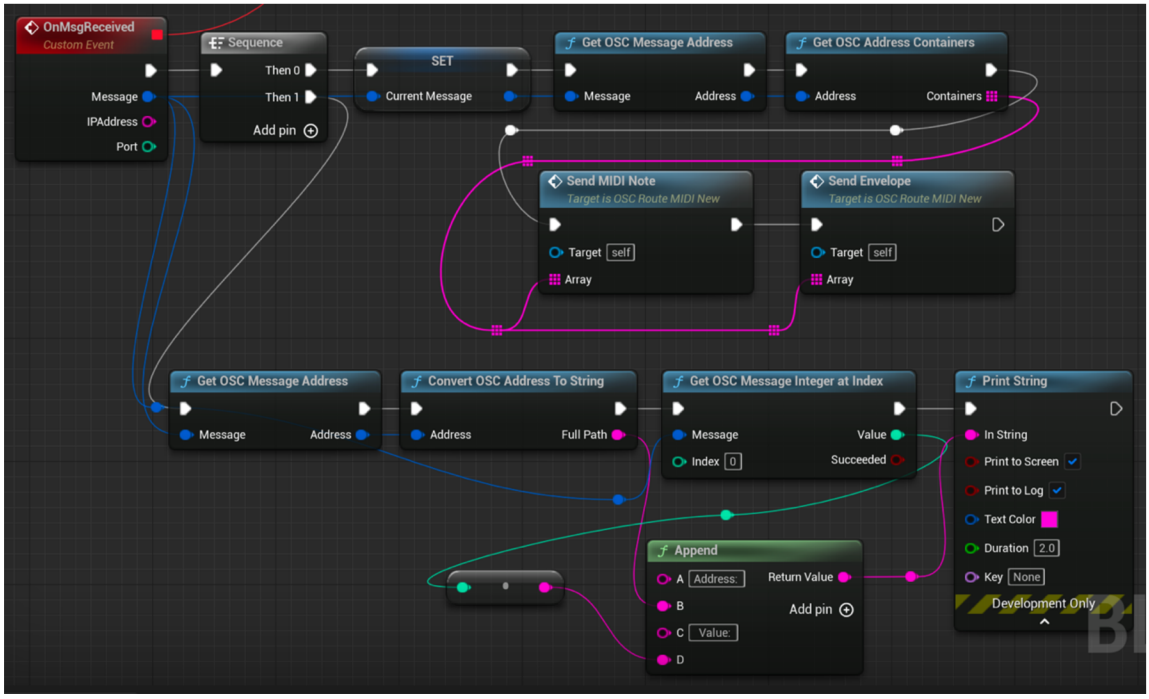
Funkce *Print messages* je uživatelsky vytvořená funkce. Tiskne všechny příchozí zprávy i pokud obsahují informace jiného druhu, než by měly být. Slouží především pro kontrolu příchozích dat, a byla využita hlavně během tvorby funkce programů pro odesílání i přijímání OSC zpráv.

4.3.2 Svázání událostí s OSC zprávami

Po příjmu OSC zprávy blokem *OSC Server* dochází ke spuštění dvou uživatelských událostí. Pro kontrolu správnosti dat jsou zprávy převedeny na textový řetězec a dále se tisknou na obrazovku.

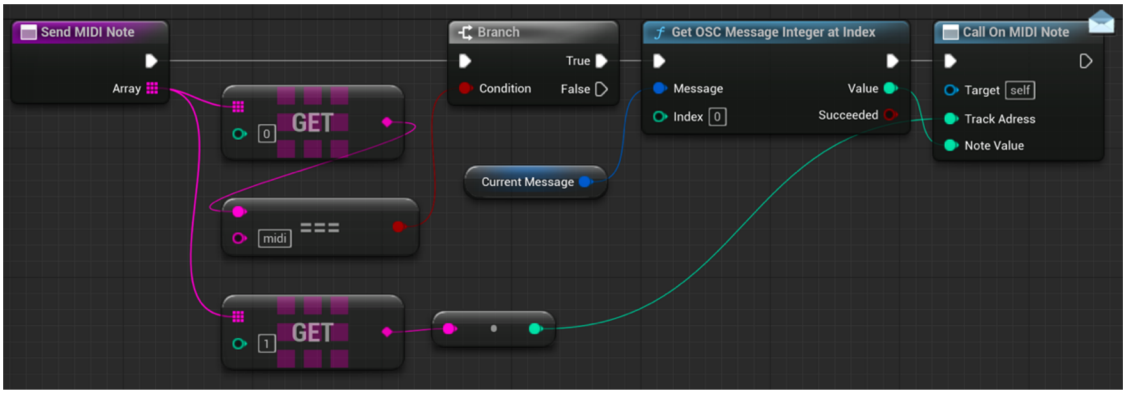
Fuknce *Get OSC Message adress* vrací adresy řetězců zabalených ve zprávě. Tyto adresy jsou potom uspořádány do pole v náležitém pořadí. V poli zprávy se uchovává typ hudebních dat (MIDI nebo obálka) a hodnota *Track adress*, která je uživatelsky zvolená v prostředí Ableton. Poslední hodnotou je číslo, které odpovídá buď aktuální hodnotě obálky, nebo parametru *Note Value*, která reprezentuje výšku tónu podle MIDI protokolu a nabývá hodnot 0–127.

Toto třídění umožňuje nejenom ovládat jeden vizuální efekt více nástroji, kterým je přiřazena stejná adresa, ale také ovládat vizuální efekty jednou konkrétní notou nebo jejich výčtem. To je výhodné především kvůli sadám bicích nástrojů a samplerům. V praxi bývá každému jednotlivému bubnu nebo samplu právě jeden tón z piano rollu.

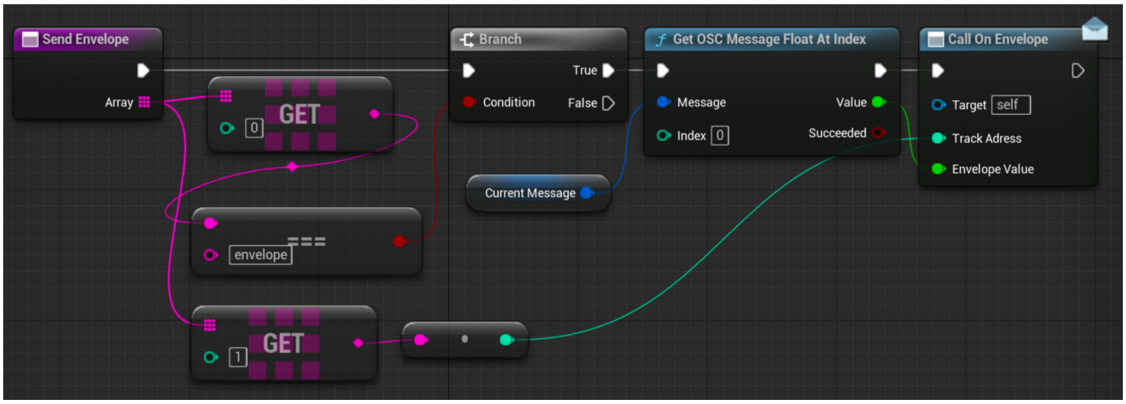


Obr. 4.5 Uspořádání OSC zpráv do pole a jejich tisk

Pokud hodnoty *Track Adr*ess a *Note Valu*e odpovídají daným kritériím, dochází k zavolání funkce *On MIDI Note*, která vyvolává konkrétní události na vizuální scéně. Funkce *On MIDI Note* je tzv. Event Dispatcher. Jedná se o způsob komunikace mezi Blueprinty v UE, kdy jeden odešle událost a ostatní, kteří naslouchají této události, jsou upozorněni. Tyto Event Dispatchery společně s upozorněním na zprávu také funkci, která naslouchá, předají hodnoty *Track Adress* a *Note Value*. Pokud tyto hodnoty splňují uživatelsky stanovená kritéria, funkce se spustí.



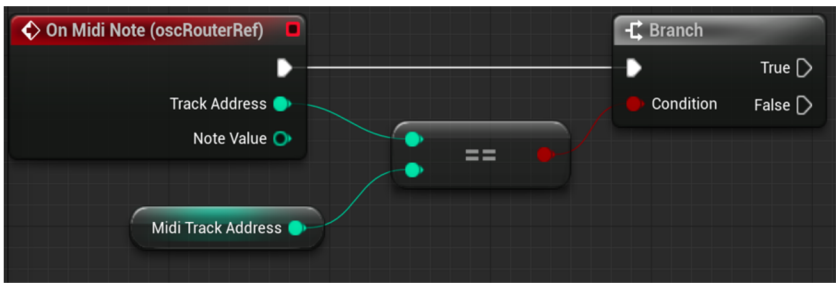
Obr. 4.6 Event dispatcher pro MIDI



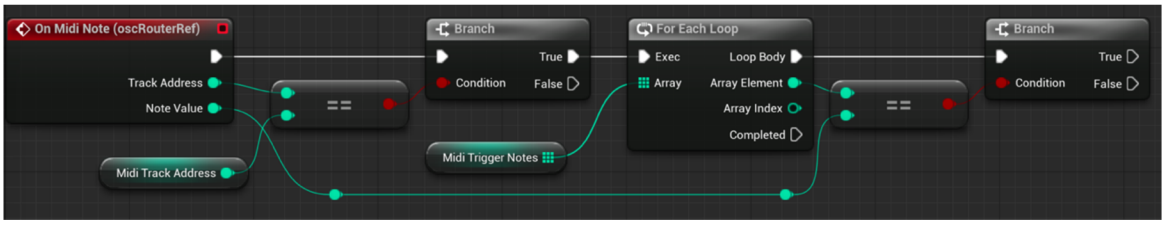
Obr. 4.7 Event dispatcher pro obálku

4.4 Spouštění událostí na vizuální scéně

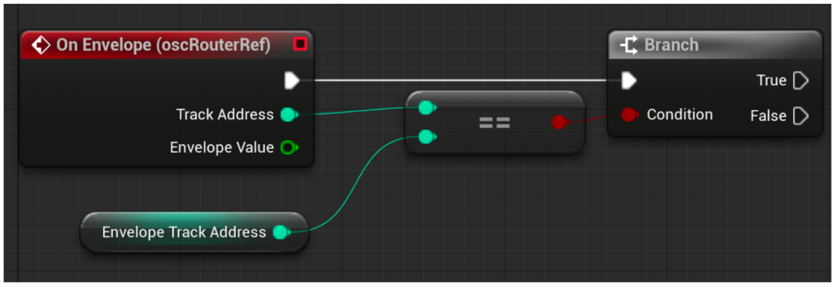
Ke spuštění události je zapotřebí stanovit kritéria, za kterých se spustí. Prvním kritériem je uživatelská hodnota *Track Adress*. Pokud není využit pin funkce *Note Value*, a hodnota *Track* Adress signálu je shodná s uživatelsky stanovenou adresou, tak se funkce spustí (Obr. 4.8.). Tato adresa může být identická pro více MIDI stop. Události lze také spouštět pouze některými MIDI notami, a to na základě jejich výčtu, který je stanoven uživatelsky (Obr. 4.9.). Při použití obálky signálu (Obr. 4.10) funkce aktivně posílá hodnoty obálky (*Envelope Value*) ze zvolené adresy.



Obr. 4.8 Spouštění události na základě hodnoty *Track Adress* a MIDI signálu



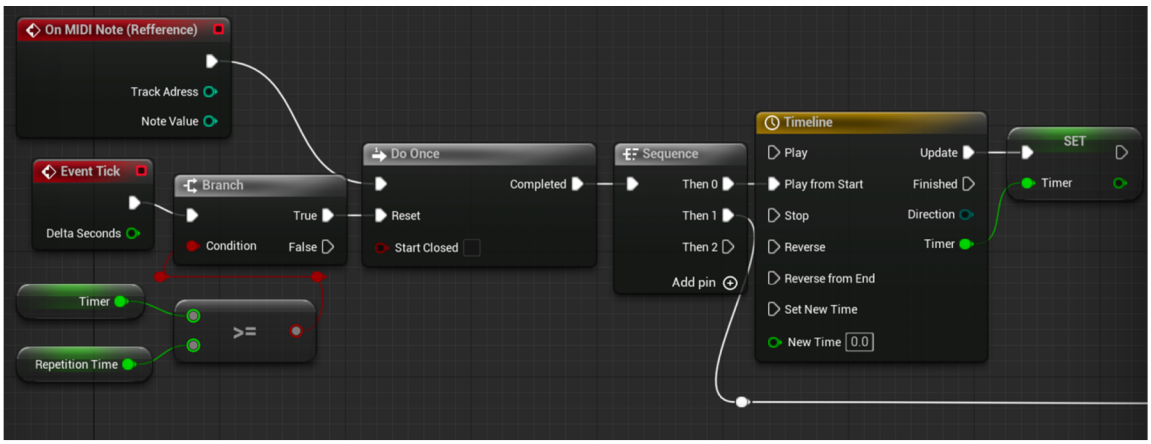
Obr. 4.9 Spouštění události na základě *Track Adress* a výčtu specifických not



Obr. 4.10 Spouštění události na základě hodnoty *Track Adress* a obálky signálu

4.4.1 Opakování událostí na základě uplynulého času

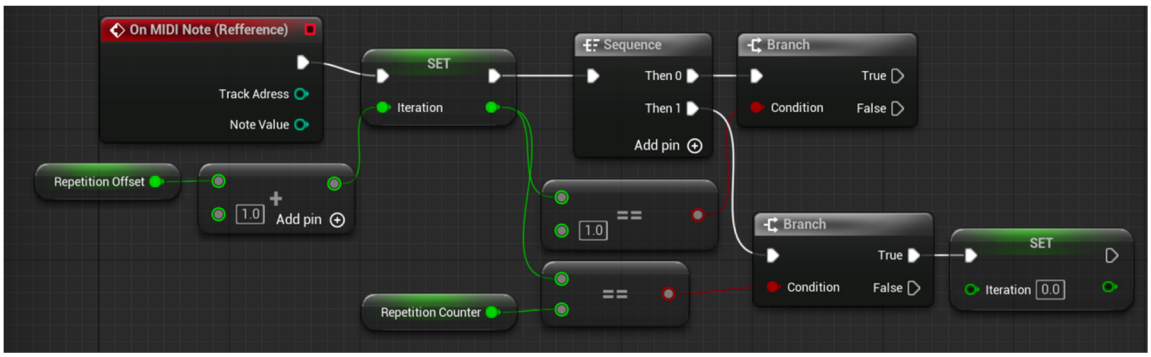
Vzhledem k charakteru hudebních dat byla implementována funkce pro spouštění události na základě uplynulého času (Obr. 4.11). Po přijmutí OSC zprávy *On MIDI Note* se spustí časový odpočet. Po příchodu každé zprávy program neustále kontroluje, jestli odpočet překročil uživatelsky stanovenou časovou hranici *Repetition Time*. Pokud ano, dojde k resetu funkce. Tato funkce tedy umožňuje události spouštět až po uplynutí stanoveného času od posledního spuštění události. To má za důsledek lepší kontrolu nad celým systémem.



Obr. 4.11 Spouštění následující události po uplynutí daného času

4.4.2 Opakování událostí na základě uplynulého počtu iterací

Tato funkce je obdobou funkce z kapitoly 4.4.1. Na rozdíl od předchozí funkce ale nekontroluje uplynulý čas od posledního spuštění události, ale počet pokusů o spuštění. Při každé příchozí zprávě z dané adresy nebo výčtu not z adresy nabude hodnota *Iteration* vyšší hodnoty. Pokud tato hodnota dosáhne uživatelsky stanovené hodnoty, dojde k restartu počítání a spuštění funkce. Události lze tím pádem spouštět při každém násobku uživatelsky stanovené hodnoty. Blueprint této funkce je vyobrazen na Obr. 4.12.



Obr. 4.12 Spouštění následující události uplynutí daného počtu iterací

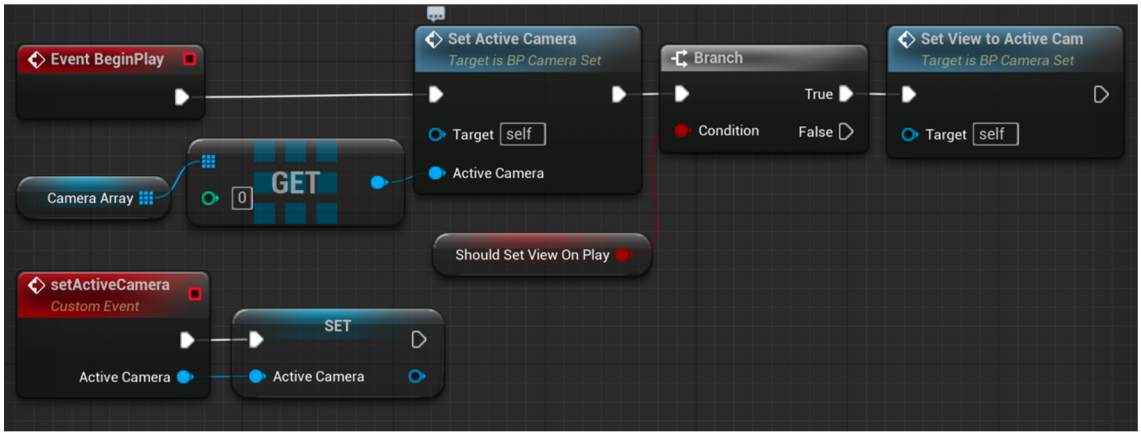
4.5 Systém pro ovládání kamery

Při vizualizaci hudby klasickým způsobem se v naprosté většině případů používají statické videosekvence. Hlavní výhodou vizualizace hudby v prostředí UE je, že probíhá v 3D prostředí. Abychom využili plného potenciálu vizualizace ve 3D prostředí byl navržen a implementován systém pro ovládání kamery.

Jedná se o sadu Blueprintů, která umožňují manuálně přepínat mezi libovolnými statickými i pohyblivými kamerami. Systém dále nabízí možnost zaměření kamery na uživatelsky stanovený cíl, měnění cílů v reálném čase a spouštění různých otřesů kamery pro docílení dramatičtější atmosféry. Všechny funkce kamerového systému se standardně ovládají pomocí MIDI, obdobným způsobem jako bylo popsáno v kapitole 4.4. Vzhledem k charakteru hudby a hudebních MIDI signálů by bylo velmi komplikované měnit kamery například na základě specifického tónu nebo nástroje, jako tomu je u vizuálních efektů. Vzhledem k repetetivnosti hudby by velmi snadno docházelo k chaotickému chování a přepínání kamery. Z tohoto důvodu je k ovládání kamerového systému použita MIDI stopa, která odesílá pouze MIDI signály, ale nerozeznívá žádnou nahrávku nebo nástroj. Tuto stopu lze poté předpřipravit pro jednotlivé hudební smyčky a scény, nebo ji ovládat živě za pomocí MIDI kontroleru.

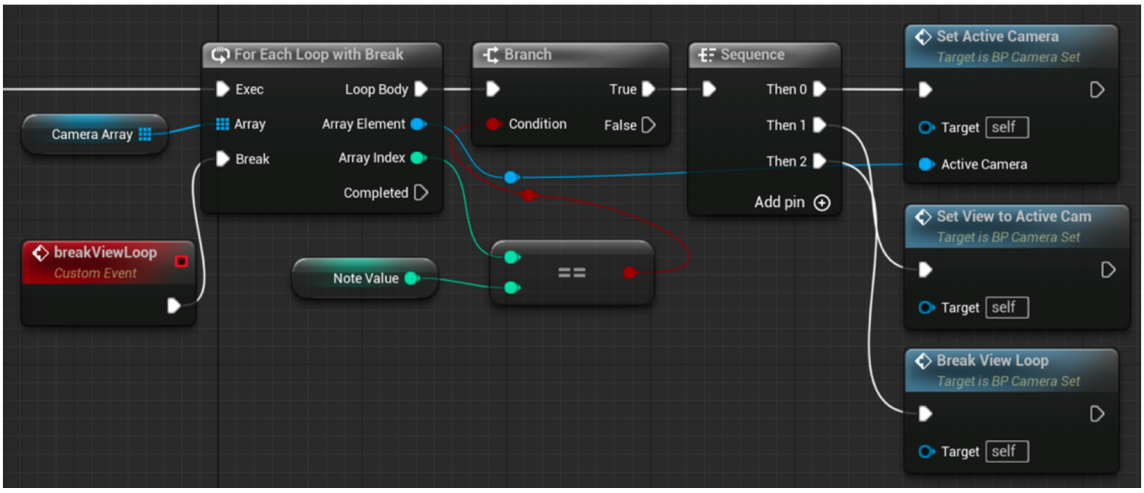
4.5.1 Přepínaní kamer

Ve chvíli, kdy se program spustí, se automaticky nastaví aktivní kamera na první kameru z pole kamer. Ta zůstává aktivní, dokud nepřijde signál, který spustí funkci pro její změnu.

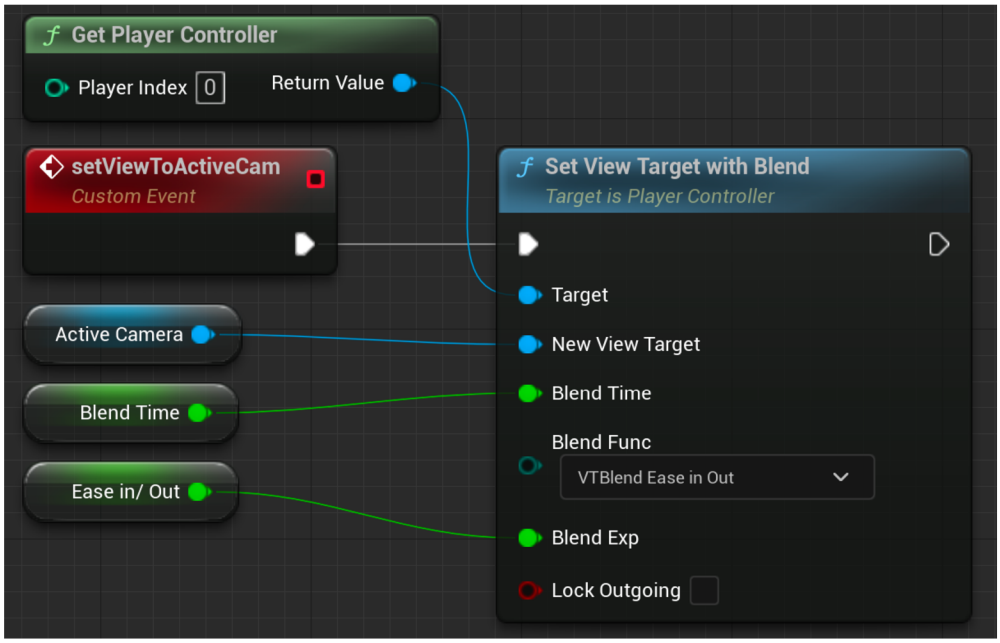


Obr. 4.13 Iniciace kamery po spuštění vizualizace

Funkce *On MIDI note* při spuštěném programu kontroluje každou příchozí OSC zprávu. Pokud tato zpráva splní kritéria pro změnu kamery, tak se funkce spustí. Pro přepínaní v kamerovém systému je vyhrazena první oktáva specifické MIDI stopy určené k ovládání tohoto systému, kde každá nota odpovídá jednomu elementu z výčtu kamer.



Obr. 4.14 Kontrola podmínek, volání funkcí pro zvolení aktivní kamery a přepnutí pohledu na aktivní kameru

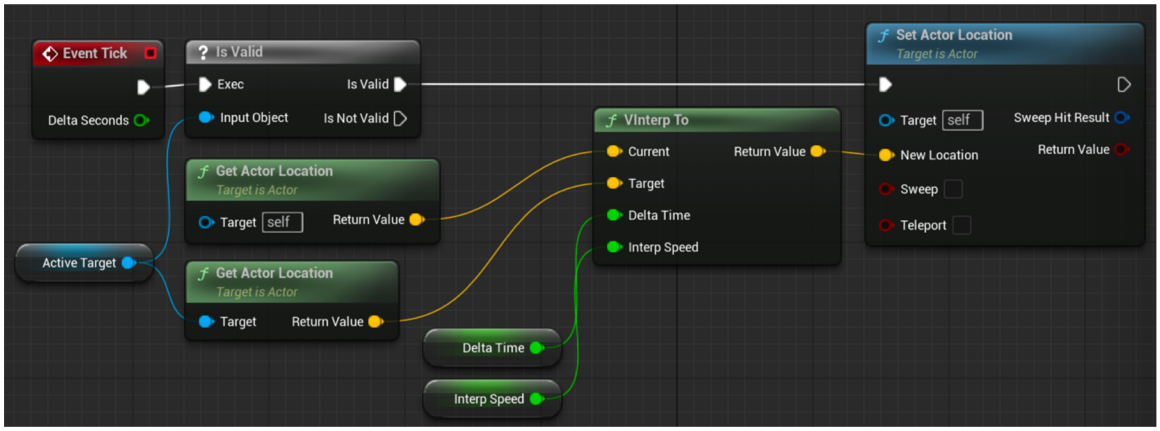


Obr. 4.15 Funkce pro přepnutí pohledu na aktivní kameru

4.5.2 Měnění cílů kamery

Součástí kamerového systému je i modul pro měnění cílů kamery. Vytvořený Blueprint sleduje pozici určitého komponentu scény a na tuto pozici také zaměřuje kameru. Komponenty mohou být jak statické, tak pohyblivé. Jednotlivé komponenty, které může kamera sledovat, jsou uspořádané do pole, přičemž každá hodnota pole komponent reprezentuje jednu konkrétní MIDI notu z druhé oktávy specifické MIDI stopy pro ovládání kamerového systému, obdobně jako u přepínaní kamer na Obr. 4.14. Libovolná kamera tedy může sledovat libovolný objekt a může svůj cíl měnit na základě předpřipravené MIDI stopy, nebo živě za pomocí MIDI ovladače.

Změna cíle neprobíhá náhle, ale je využito interpolace kamery (Obr. 4.16). To znamená, že kamera plynule přechází mezi dvěma polohami v určitém časovém úseku, a to se provádí výpočtem mezihodnot mezi počátečním a koncovým bodem pohybu kamery. Následuje plynulý přechod kamery mezi těmito hodnotami.



Obr. 4.16 Interpolace směru kamery na aktivní cíl

4.5.3 Spouštění otřesů kamery

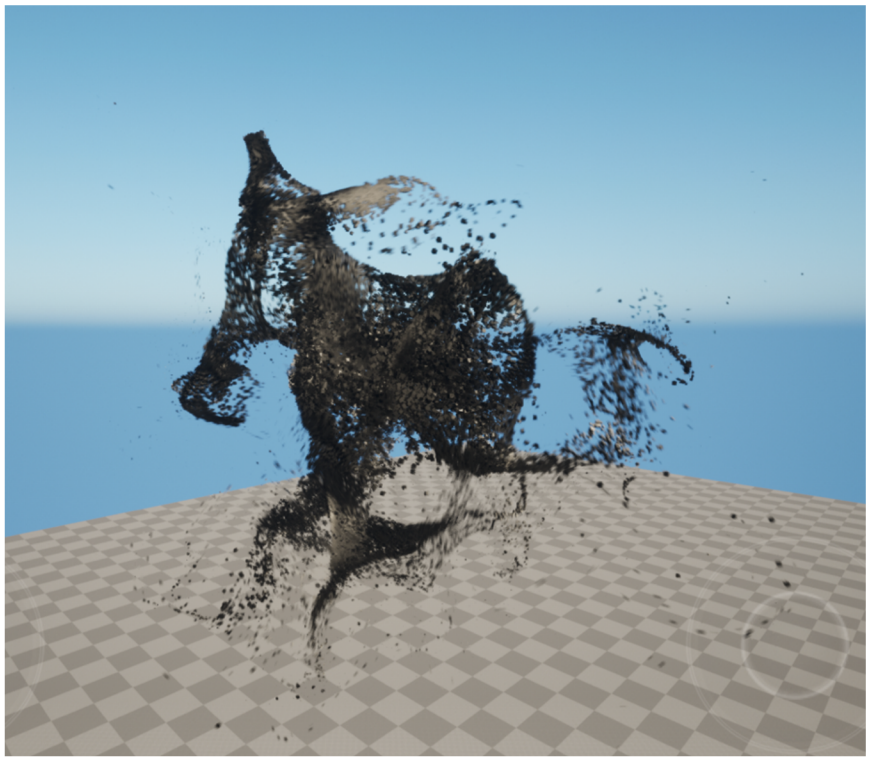
Další součástí kamerového systému je spouštění otřesů kamery. Ty jsou užitečné k navození realističtější atmosféry, používají se k přidání pocitu nárazu, pohybu nebo vibrací do kamery. Často se používají ve videohrách k simulaci účinků výbuchů, zemětřesení nebo jiných silných událostí, které by způsobily otřesy nebo vibrace kamery. V případě hudby to může být například velmi silný úder bicího nástroje. Je vytvořen soubor různých otřesů kamery, které jsou následně spouštěny opět ze specifické MIDI stopy pro ovládání kamerového systému.

5. TVORBA AUDIOREAKTIVNÍCH VIZUÁLNÍCH EFEKTŮ

Tato kapitola demonstruje možnosti využití nástroje vytvořeného pro vzájemnou komunikaci DAW Ableton a Unreal Engine 5. Je vysvětlen postup tvorby různých druhů vizuálních efektů v rámci vizuálního prostředí a jejich implementace v rámci generativní vizualizace hudby v reálném čase. Ke každému vizuálnímu efektu byl vytvořen i Blueprint pro spouštění daného efektu při vizualizaci hudby.

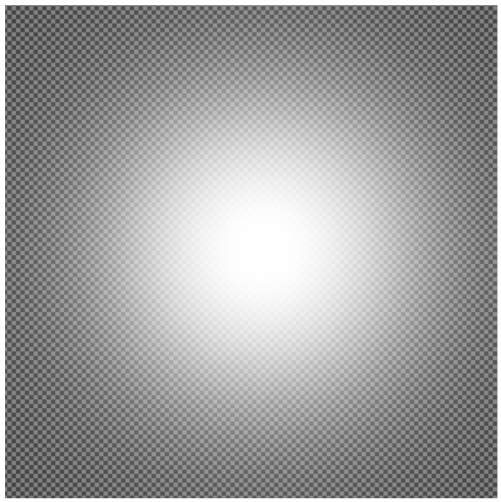
5.1 Chaotický částicový systém

Efekt vytvořený k základní demonstraci je částicový systém. Obsahuje deset tisíc částic a jeho simulace je prováděna pomocí GPU. Částice vznikají ve stanovené oblasti a jejich životnost je náhodná v intervalu od 5 do 8 sekund. Systém je ovlivňován několika vírovými silami, gravitační silou a silou, která přitahuje částice do jednoho bodu. To zajišťuje vizuálně příjemný chaotický systém, který se v čase dynamicky mění, ale nemění se středobod jeho polohy.



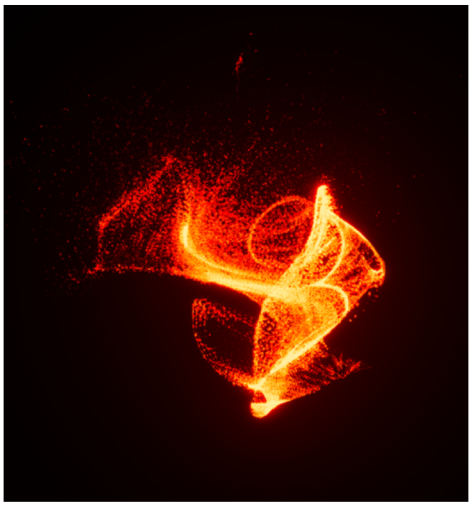
Obr. 5.1 Chaotický částicový systém

Částicovému systému je ve výchozím stavu přidělen základní 3D model částic. V projektu byl změněn způsob vykreslování z *Mesh Renderer* na *Sprite renderer*, což umožňuje využití pouhé textury pro model částic a změnit vzhled z kostek na jiskry. Pro efekt jisker byla vytvořena speciální textura (Obr. 4.4), a následně z ní byl vytvořen materiál, jehož hlavním atributem je *Blend Mode Additive*. Aditivní mód překrývání zajišťuje to, že pokud se částice překrývají, jejich barva se sčítá. Ve vizuální podobě to znamená, že místa s větším množstvím částic jsou světlejší než místa s malým množstvím částic.

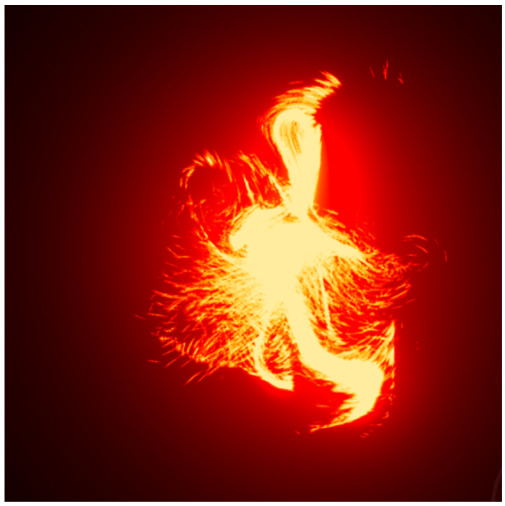


Obr. 5.2 Textura částic

Velikost částic je zpočátku uniformní, ale s měnící se rychlostí částic dochází k modulaci jejich délky pomocí funkce *Scale sprite size by speed*, což dodává lepší vzhled systému. Orientace částic je daná vektorem jejich rychlosti.



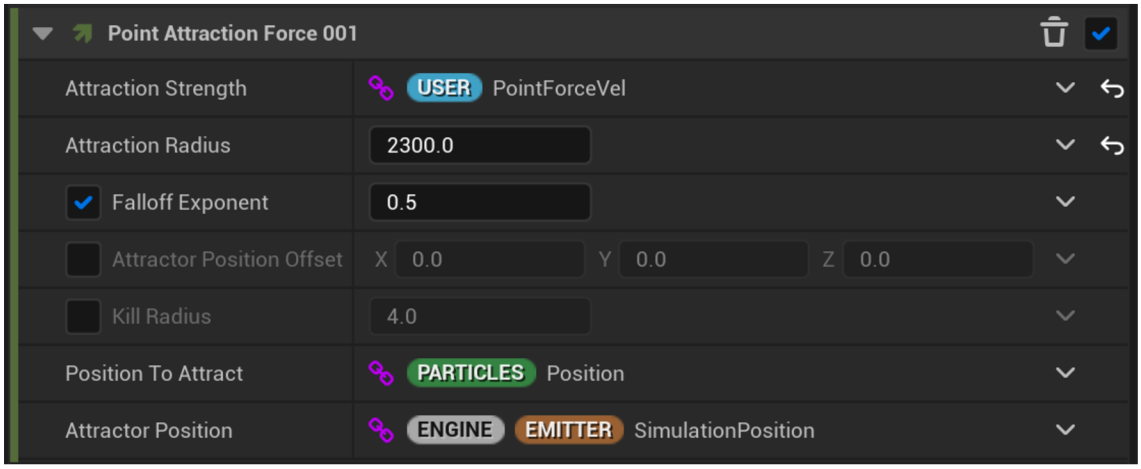
Obr. 5.3 Částicový systém po aplikaci textury a barvy



Obr. 5.4 Částicový systém po modulaci délky částic

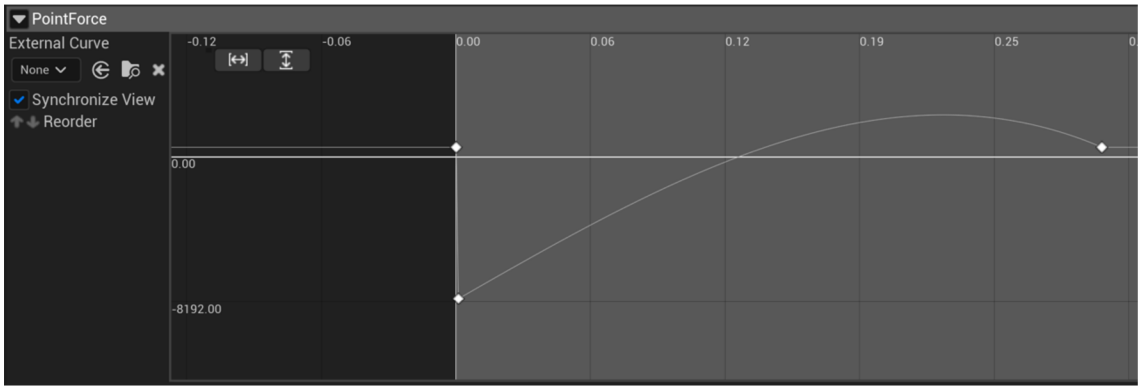
5.1.1 Využití částicového systému při vizualizaci hudby

Vytvořený částicový systém je následně upraven tak, aby vhodně reagoval na hudební signál. V první řadě je potřeba určit, které parametry budeme na základě hudebních dat měnit. Pro demonstraci reakce na MIDI signál byly zvoleny dva různé parametry, a to velikost síly, která přitahuje částice do středu systému *Attraction Strenght* a parametr, který určuje jejich zpomalení *Drag*. Tyto dva parametry je zapotřebí definovat jako uživatelské, jinak by nebylo možné měnit jejich hodnoty uživatelem na základě obálky nebo MIDI signálu v reálném čase.



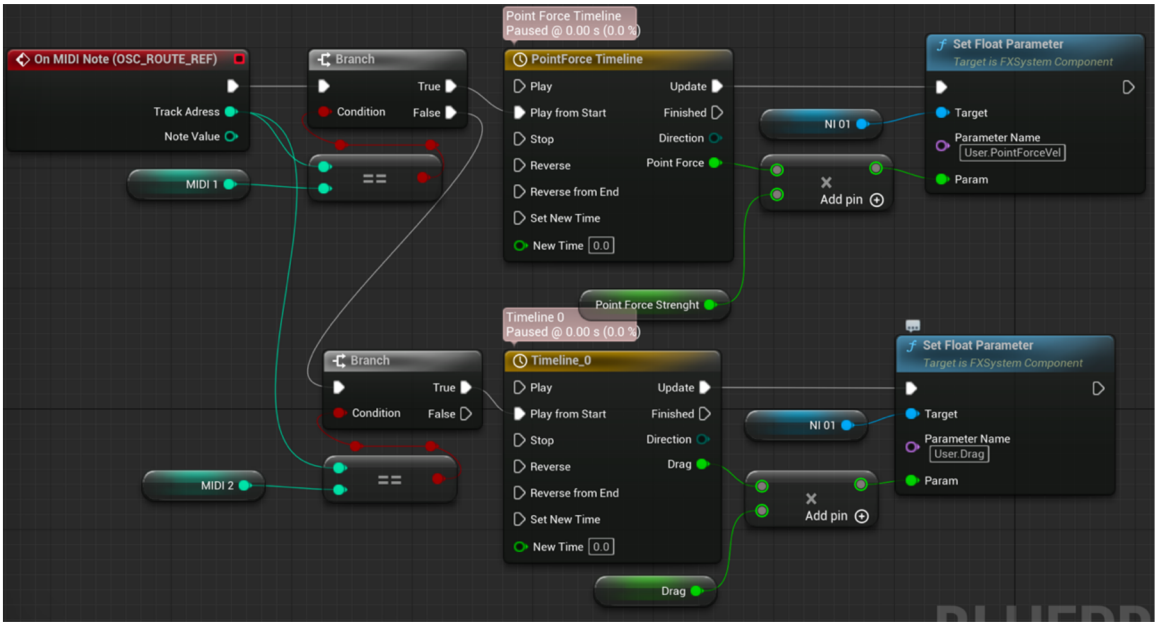
Obr. 5.5 Uživatelsky ovládaný parametr *Attraction Strength*

Dalším krokem je vytvoření Blueprintu, který zajistí, že při úderu MIDI signálu se parametr změní. K tomuto účelu je využit Event dispatcher vytvořený v kapitole 4.3.2. Událost *On MIDI Note* nejprve ověří podmínky pro její spuštění (správná adresa, popřípadě konkrétní noty) a poté na základě uživatelem stanovené časové osy dynamicky změní daný parametr částicového systému.



Obr. 5.6 Časová osa pro parametr *Attraction force* spuštěná pomocí MIDI signálu

Na následujícím obrázku lze vidět kompletní Blueprint pro spuštění události pomocí MIDI signálu. Signály z uživatelsky stanovené adresy *MIDI 1* spouští modulaci parametru *Attraction Force*, signály z adresy *MIDI 2* poté modulaci parametru *Drag*, opět pomocí časové osy.



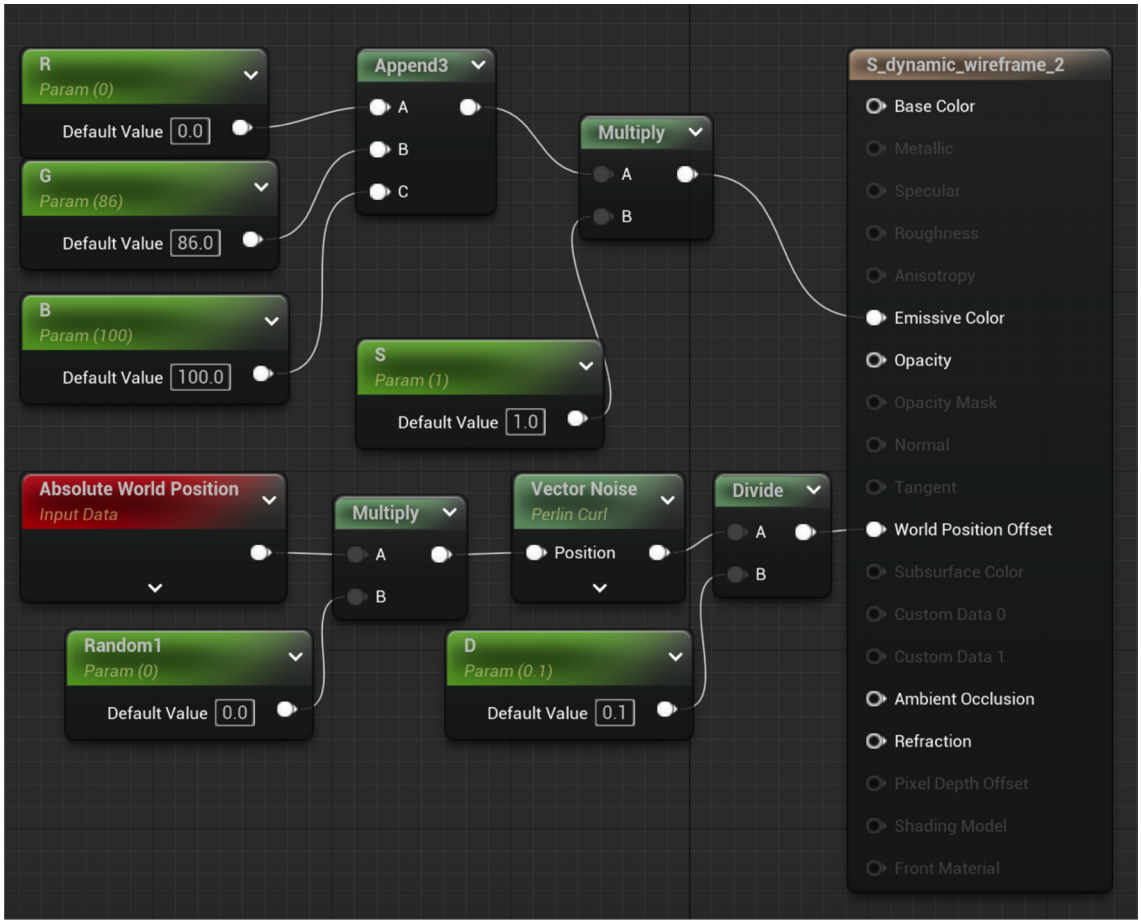
Obr. 5.7 Modulace parametrů částicového systému za pomocí MIDI signálu

Obr. 5.8 Částicový systém v klidu

Obr. 5.9 Částicový systém při vybuzením MIDI signálem z adresy MIDI 1

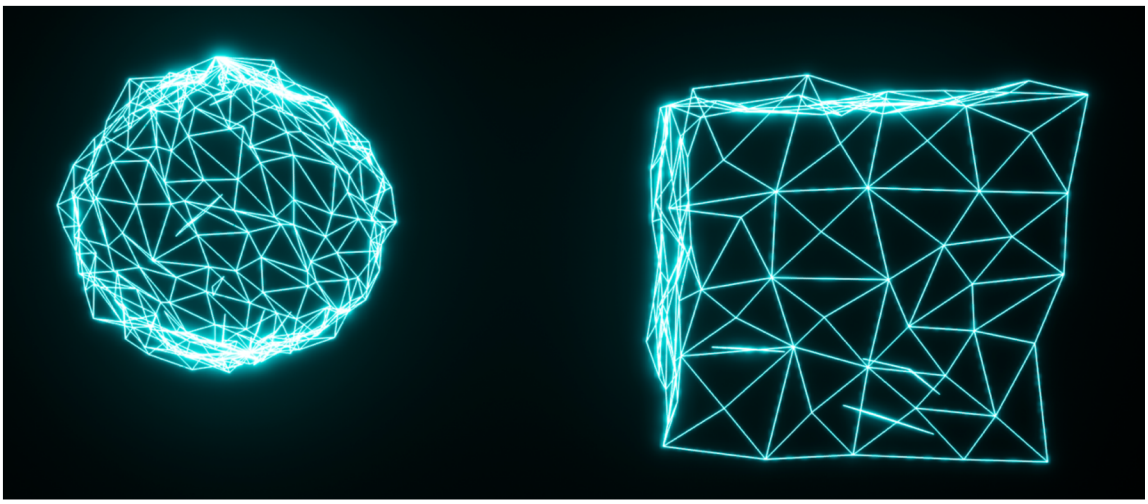
5.2 Dynamický wireframe materiál

Dynamický materiál vytvořený v rámci této kapitoly reprezentuje základní způsob tvorby dynamických materiálů a lze ho samostatně použít k vizualizaci hudby. Způsob fungování tohoto materiálu je vcelku jednoduchý. Zvolený 3D model, na který bude materiál aplikován, nebude mít viditelný svůj celý povrch. V atributech materiálu je zvolena možnost *wireframe* zobrazení. Wireframe je vizuální reprezentace geometrie polygonové sítě nebo objektu ve 3D prostoru. Používá se k vizualizaci a manipulaci tvaru objektu nebo k identifikaci potenciálních problémů s geometrií, jako jsou chybějící nebo překrývající se polygony. Objekt zobrazen v režimu wireframe je zobrazen jako síť čar, které obkreslují vrcholy a hrany těchto polygonů. V tomto případě je wireframe zobrazení použito čistě k docílení zajímavé vizuální podoby.

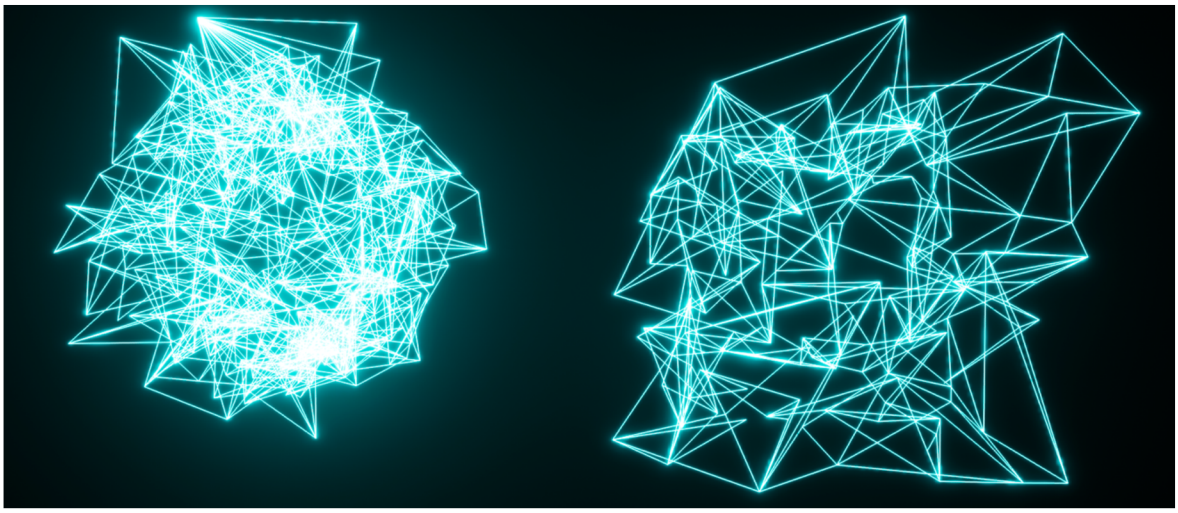


Obr. 5.10 Jednoduchý dynamický wireframe materiál

Vyzařovací barva, kterou je síť zbarvena, je zvolená uživatelsky a následně je násobena hodnotou, která se mění podle časové osy v závislosti na MIDI události, obdobně jako na Obr. 5.6, alternativně může hodnota tohoto parametru sledovat hodnotu zvukové obálky. Na pin *world position offset*, který určuje vychýlení jednotlivých vrcholů polygonové sítě, je připojen blok *Vector noise*. Vrcholy sítě jsou vychylovány náhodným šumem typu *Perlin*. Sílu šumu určuje časová osa, která se spouští úderem MIDI noty, případně hodnotami obálky zvuku. Na následujících obrázcích 5.11 a 5.12 lze vidět aplikace dynamického materiálu na dva různé objekty.



Obr. 5.11 Jednoduchý dynamický wireframe materiál při vybuzení slabým signálem



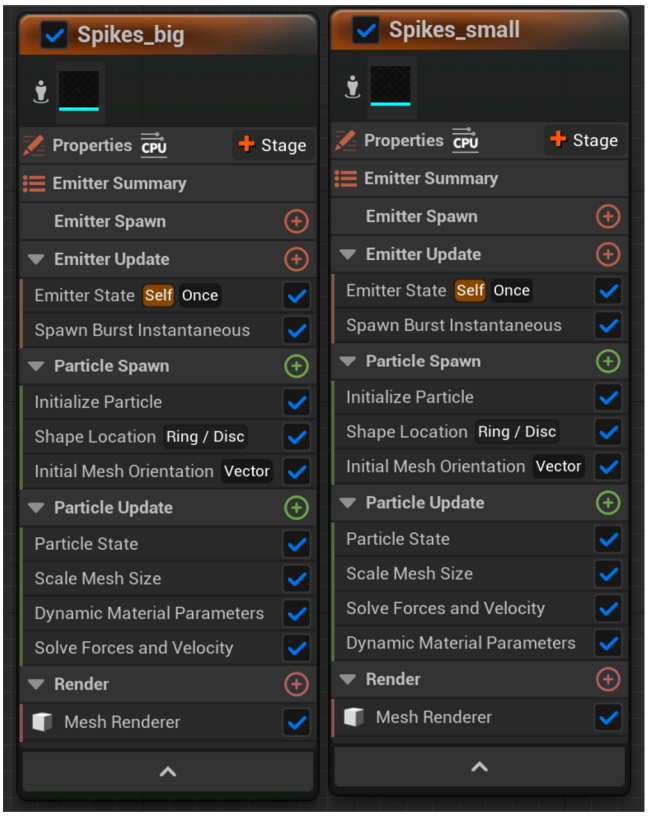
Obr. 5.12 Jednoduchý dynamický wireframe materiál při vybuzení silným signálem

5.3 Částicový systém s využitím dynamické materiálové instance

Některé nástroje pro tvorbu speciálních efektů lze různě kombinovat. Následně popsaný efekt kombinuje částicový systém, který při vykreslování částic využívá dynamický materiál vytvořený v kapitole 5.2.

Cílem tohoto efektu je vytvořit pomyslné bodáky, které vyrůstají ze země a následně svoji velikost zase zmenšují, dokud se neztratí. Toho je docíleno dvěma různými emitory, které fungují na stejné bázi.

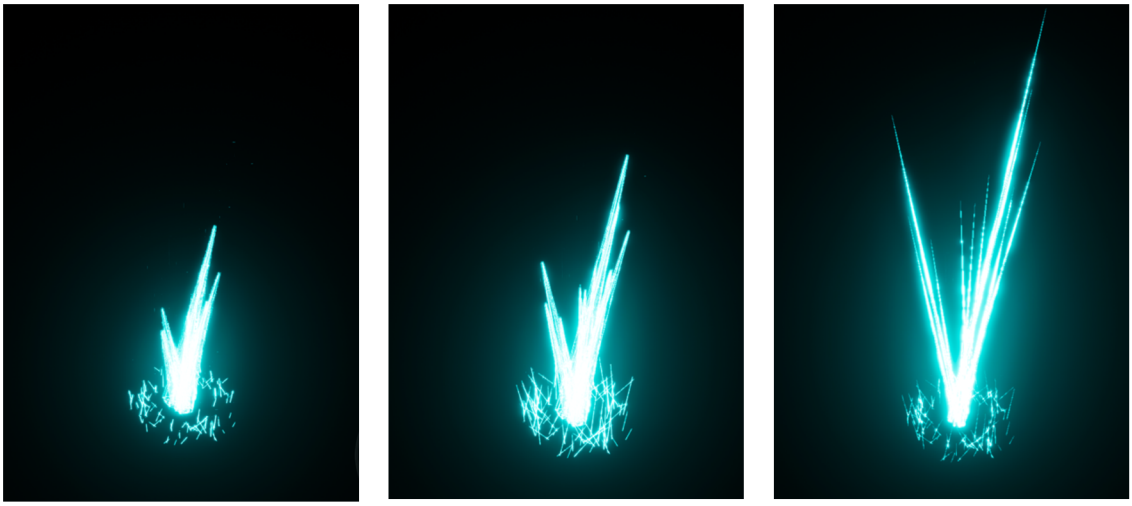
V sekci *Emitter Update* je zvolen modul *Spawn Burst instantaneous*. To má za důsledek, že všechny částice objeví v jeden přesný okamžik. V následujícím bloku *Particle Spawn* je zvolen tvar, ve kterém se mají objevit. K tomu byl zvolen kruh s malým poloměrem. Modul *Initial Mesh Orientation* dále nastavuje vstupní orientaci modelu, který bude použit při vykreslování částic. V tomto případě jsou modely zarovnány podle osy Z souřadnicového systému. Tento blok dále nabízí i možnost rotace modelu. Pomocí vektoru s náhodnou hodnotou v daném rozmezí je docíleno toho, že jsou jednotlivé modely na počátku jejich života nezávisle vychýleny od osy. Pokud by toto vychýlení nebylo nastaveno, všechny modely by byly orientovány zcela stejně a nebylo by dosáhnuto kýženého vizuálního efektu.



Obr. 5.13 Emitory částicového systému při použití dynamické materiálové instance

V sekci *Particle Update* poté blok *Scale Mesh Size* aktivně mění velikost vykreslovaných modelů v čase. Velikost modelu není měněna uniformě, ale v závislosti na souřadnicovém systému, což má za důsledek růst modelů pouze v požadovaném směru. Pomyslné bodáky tedy zůstávají přibližně stejně tlusté, mění se především jejich délka. Modul *Dynamic Material Parameters* následně uchovává uživatelské parametry materiálu, které bude možné měnit dynamicky v dynamické materiálové instanci. Pro tento efekt byl zvolen materiál vytvořený v kapitole 5.2 a dynamickým parametrem jsou zde hodnoty *R*, *G* a *B* vektoru barvy.

Druhý emitor funguje v podstatě stejně. Hlavním rozdílem je, že na jeho výstupu jsou modely menší velikosti, které se objevují v širším okolí prvního emitoru.



Obr. 5.14 Částicový systém s využitím dynamického materiálu

5.4 Dynamický materiál s živým video vstupem

Vizualizace hudby pomocí živého přenosu videa je vhodným způsobem vizualizace, který může zvýšit celkový zážitek publika. Zachycením pohybů a výrazů interpreta a jejich zobrazením na obrazovkách mohou diváci vidět energii a emoce při vystoupení způsobem, který přesahuje pouhý zvuk, nebo syntetická vizualizace.

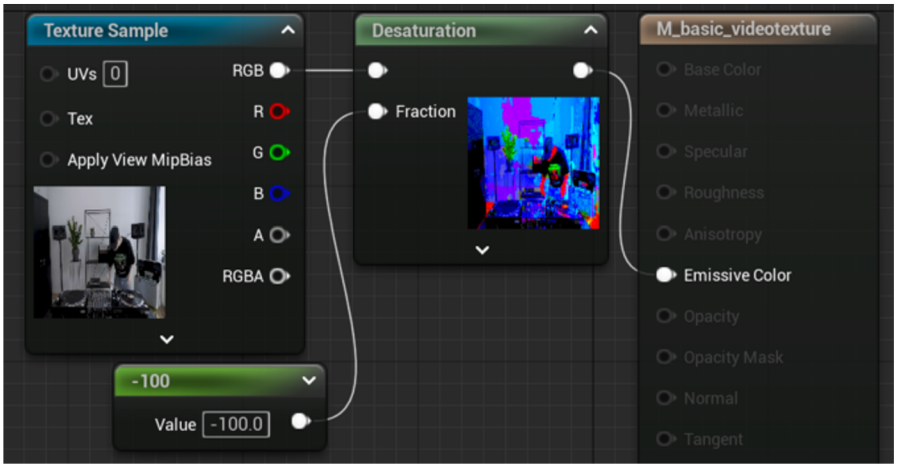
K tvorbě materiálu s texturou živého videa bylo potřeba do scény přidat přehrávač *Media Player*. Do přehrávače je poté možné nahrát statickou videosekvenci, nebo zvolit živý vstup z webkamery, popřípadě externí kamery připojené k počítači. Při přidání přehrávače lze v možnostech zaškrtnou volbu *create Video output MedieTexture asset*, což má za výsledek, že se společně s přehrávačem rovnou vytvoří i video textura, kterou potom můžeme použít v rámci libovolného materiálu.

Pro potřeby vizualizace hudby byla vytvořena sada vizuálních efektů speciálně pro video textury.

5.4.1 Saturace obrazu

Efekt saturace neboli sytosti obrazu je výborným příkladem toho, že ne vždy je zapotřebí složitých, rafinovaných komplexních efektů k dosažení zajímavého výsledku. Saturace je řízena informacemi, které jsou uloženy v chrominančních (barevných) kanálech videosignálu.

V digitálním videu se sytost obvykle vyjadřuje v procentech, přičemž sytost 0 % znamená zcela šedý obraz a sytost 100 % představuje nejintenzivnější a nejživější barvy. Pokud jsou barvy ve videu posunuty mimo svůj normální rozsah, může dojít ke zkreslení a ztrátě detailů. Tímto způsobem vznikají artefakty, které mohou mít různou podobu. Nejběžnějším artefaktem, který vzniká při přesycení videa, je ořezávání barev. K tomu dochází, když jsou barvy ve videu tak intenzivní, že ztrácejí detaily a jsou zkreslené. To může způsobit, že barvy "prosakují" do jiných oblastí obrazu, což má za následek ztrátu rozlišení a jasnosti. Dalším artefaktem, který může vzniknout při přesycení videa, je *color banding*. Ke color bandingu dochází, když je v obraze nedostatek barevných informací, což má za následek viditelné barevné pruhy. To může být obzvláště patrné v přechodech nebo v oblastech obrazu s jemnými barevnými odchylkami a může způsobit, že obraz bude vypadat méně hladce a realisticky. Přesycení může také způsobit posterizaci, ke které dochází, když se barevná informace v obraze zredukuje na omezený počet barev. To může vést ke stylizovanějšímu, grafičtějšímu vzhledu, ale může také způsobit ztrátu detailů a čistoty obrazu. V běžné video produkci tyto artefakty představují problém, v případě vizualizace hudby je lze využít jako umělecké prvky.



Obr. 5.15 Blueprint pro saturaci video textury

V Unreal Engine se konkrétní metoda, použitá v bloku desaturace textury, může lišit v závislosti na verzi enginu a konkrétní implementaci. Běžným přístupem k desaturaci je však převod barvy z barevného prostoru RGB do barevného prostoru HSL (Hue, Saturation, Lightness).

V barevném prostoru HSL představuje složka sytosti *S* intenzitu nebo čistotu barvy. Při desaturaci se hodnota sytosti obvykle sníží nebo nastaví na nulu, což vede k odstínům na škále šedi, nebo tlumenému vzhledu. Saturace textury funguje opačným způsobem.

Převod barvy z barevného modelu RGB na model HSL lze provést podle následujících kroků a vorců 5.1 – 5.10.

1. Normalizace RGB hodnot na hodnoty 0–1:

$$R_n = \frac{R}{255}, G_n = \frac{G}{255}, B_n = \frac{B}{255} \tag{5.1}$$

2. Výpočet minimálních a maximálních hodnot normalizovaných složek RGB:

$$Max = \max(R_n, G_n, B_n) \tag{5.2}$$

$$Min = \min(R_n, G_n, B_n) \tag{5.3}$$

3. Výpočet odstínu *H*:

Maximum se nachází v *R*:
$$H = \frac{(G_n - B_n)}{Max - Min} \tag{5.4}$$

Maximum se nachází v *G*:
$$H = 2 + \frac{(B_n - R_n)}{Max - Min} \tag{5.5}$$

Maximum se nachází v *B*:
$$H = 4 + \frac{(R_n - G_n)}{Max - Min} \tag{5.6}$$

Pokud *Max* = *Min*, je barva odstínem šedé a hodnota *H* je obvykle nastavena na 0.

4. Převod do stupnice 0–360 stupňů:

$$H_n = H \cdot 60 \tag{5.7}$$

5. Výpočet *L*:

$$L = \frac{Max + Min}{2} \tag{5.8}$$

6. Výpočet *S*:
   a. Pokud L ≤ 0,5

$$S = \frac{Max - Min}{Max + Min} \tag{5.9}$$

   b. Pokud *L* > 0,5

$$S = Max - \frac{Min}{2 - Max - Min} \tag{5.10}$$

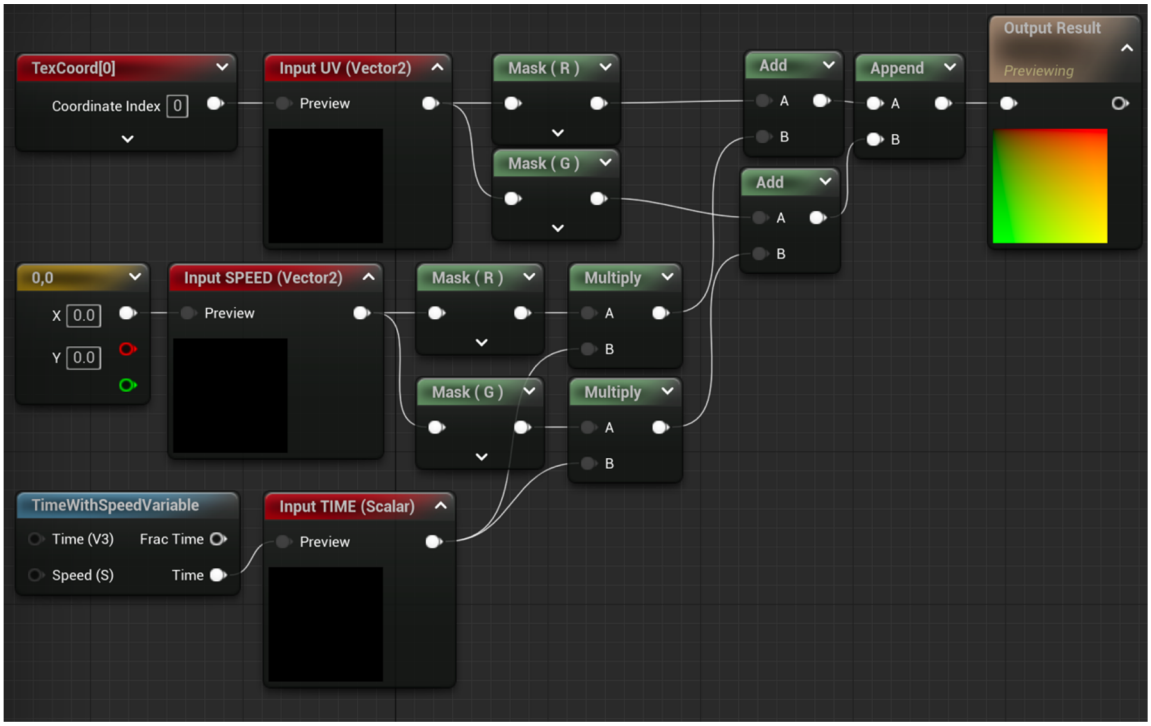
Pokud *Max* = *Min*, je barva odstínem šedé a hodnota *S* je obvykle nastavena na 0.

Výsledkem těchto výpočtů pro každý pixel textury jsou hodnoty HSL pro danou barvu RGB. Tyto hodnoty představují složky odstínu (H), sytosti (S) a jasu (L). [21]

5.4.2 Glitch efekt

Efekt popsaný v této kapitole opět reprezentuje vizuální artefakty, které při běžné video produkci bývají problémem, ale v rámci vizualizace hudby jsou použity jako umělecký efekt. Digitální video data jsou manuálně poškozována a manipulována k dosažení glitch efektů, které v analogové doméně vznikají fyzickým poškozením nosičů, selháním nebo vadami hardwaru, popřípadě vnějším rušením signálu.

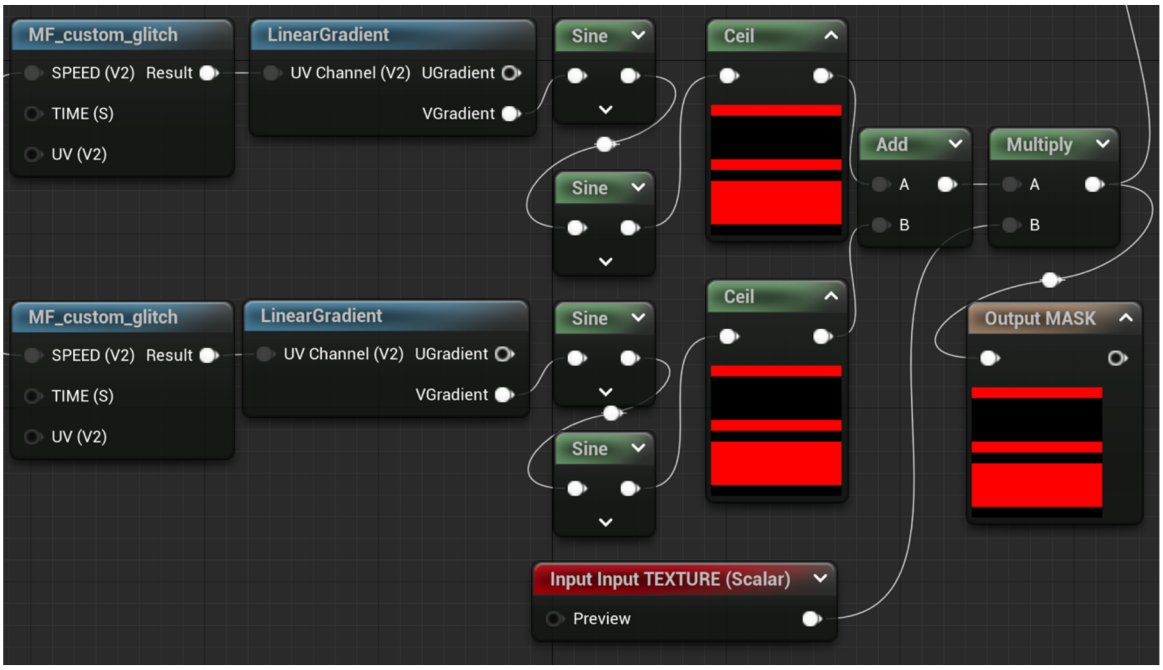
První částí efektu je vytvoření materiálové funkce, jejíž výstup je maska v podobě různě širokých černých a bílých pásů, které aktivně probíhají ve vertikálním směru. Jednoduchá uživatelsky vytvořená funkce *MF_custom_glitch* generuje vlastní UV mapu. UV mapou se označuje dvourozměrný souřadnicový systém, který se používá k mapování textur na povrch 3D modelů. UV souřadnice se používají k určení toho, jak mají být pixely textury aplikovány na vrcholy sítě. K hodnotám mapy vytvořené touto funkcí jsou přičítány hodnoty rychlosti posunu v čase. Výsledná UV mapa má dvě vrstvy, každou s různou rychlostí posunu.



Obr. 5.16 Jednoduchý dynamický wireframe materiál

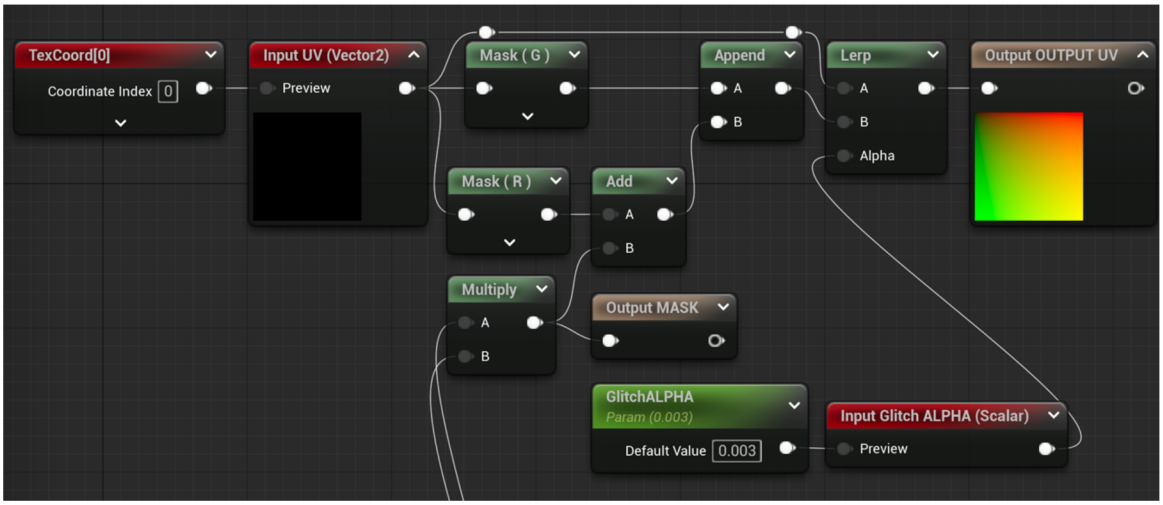
Tato funkce je implementována dovnitř materiálové funkce *MP_custom_glitch2*. Blok *LinearGradient* zde generuje lineární gradient ve směru U a V na základě daných UV souřadnic. Jako vstup tohoto bloku je výstup funkce *MF_custom_glitch*. Články *Sine* poté zajišťuje oscilaci gradientů plynule tam a zpátky. Tyto signály jsou následně zaokrouhleny na celou hodnotu pomocí funkce *ceil*, čímž zmizí přechody ve škále šedi a zůstanou pouze černé a bílé pruhy. Ty jsou posléze násobeny uživatelsky zvolenou

glitch texturou, pomocí které dosáhneme požadovaného efektu. Výsledkem tohoto násobení je maska, která je uložena jako první výstup funkce *MP_custom_glitch2* a kterou budeme dále používat v rámci dynamického glitch materiálu.



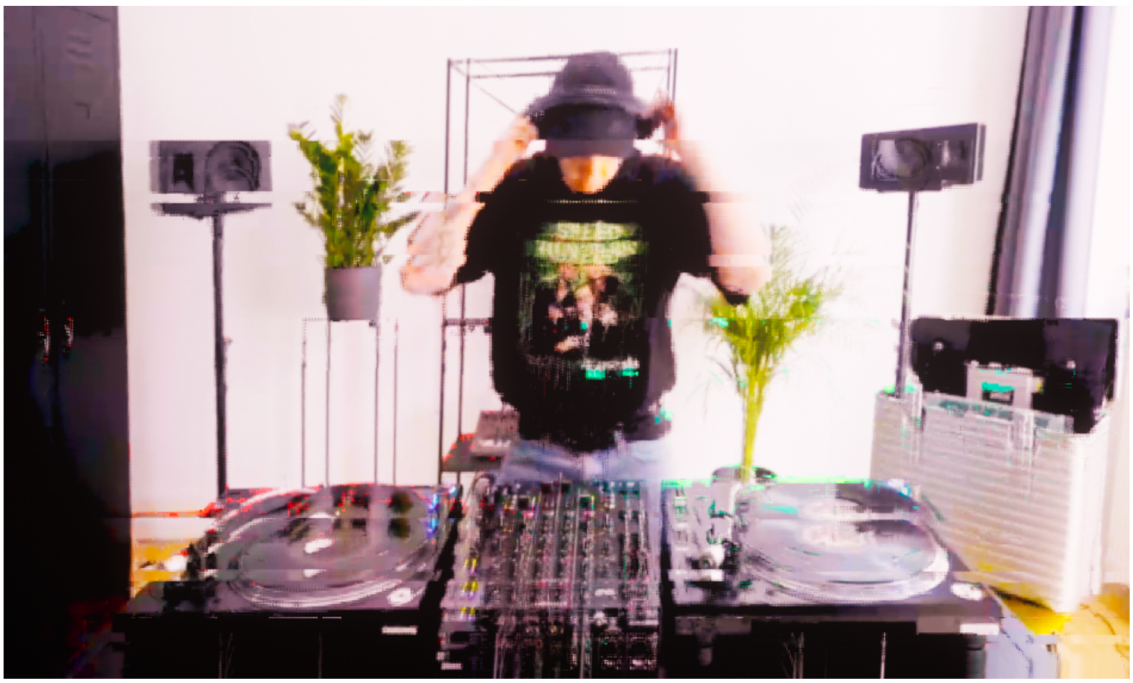
Obr. 5.17 Výsledná pohyblivá maska

Tato maska se dále prolíná se standardní UV maskou. Prolínání je provedeno pomocí funkce lineární interpolace. Ta přijímá dva vstupy a hodnotu *Alpha*, která určuje poměr prolnutí mezi oběma vstupy. Výsledná upravená UV mapa je uložena jako další výstup funkce.



Obr. 5.18 Výsledná pohyblivá UV maska

Pomocí funkce *MP_custom_glitch2* jsou vytvořeny dvě různé UV masky. Jedna prolnutá pohyblivou glitch texturou a druhá bez ní. Ty jsou aplikovány na video texturu. Tyto dvě textury jsou od sebe následně odečteny, čímž vzniká pohyblivá textura, která má proměnlivá světlá místa v bodech, kde se tyto dvě textury v závislosti na vychýlení UV maskou různí. Výsledek tohoto procesu je následně použit pro tvorbu *Opacity Mask*, která dělá určitá místa textury průsvitná. Tento výsledek je také použit v kombinaci s vytvořenou pohyblivou maskou (Obr. 5.15) a barevným filtrem k manipulaci zdrojové video textury a následným ovládáním vyzařovací barvy.



Obr. 5.19 Výsledný glitch effect

5.5 Post process efekt

Pro demonstraci post process efektu byl vytvořen efekt, který uplatňuje Fresnelův odraz na povrch všech modelů nezávisle na jejich materiálu. Tento efekt je jeden ze způsobů, jak simulovat chování světla odrážejícího se od povrchu pod různými úhly. Funguje tak, že moduluje množství odrazu na povrchu v závislosti na úhlu pohledu. Když světlo dopadá na povrch, část se odráží zpět pod stejným úhlem jako dopadající světlo, zatímco část se láme (ohýbá) a prochází povrchem. Velikost odrazu a lomu závisí na úhlu dopadu a indexu lomu povrchu.

Koeficienty odrazu R_s a R_p podle rovnic 5.11 a 5.12 udávají poměr intenzity odraženého a dopadajícího svazku. Jejich hodnoty záleží na polarizaci dopadajícího světla. Rozlišujeme polarizaci *s* a *p*. Při *s* polarizaci je vektor intenzity dopadajícího světla kolmý na rovinu dopadu, v případě *p* polarizace je naopak součástí této roviny.

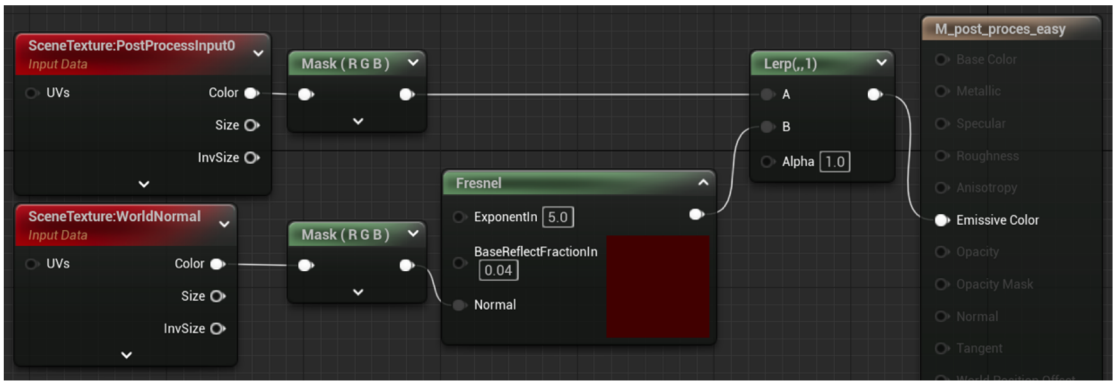
$$R_s = \left[\frac{n_1 \cdot \cos(\theta_i) - n_2 \cdot \cos(\theta_t)}{n_1 \cdot \cos(\theta_i) - n_2 \cdot \cos(\theta_t)}\right]^2 \tag{5.11}$$

$$R_p = \left[\frac{n_1 \cdot \cos(\theta_t) - n_2 \cdot \cos(\theta_i)}{n_1 \cdot \cos(\theta_t) - n_2 \cdot \cos(\theta_i)}\right]^2 \tag{5.12}$$

Kde n_1 a n_2 jsou indexy lomu jednotlivých prostředí, úhly θ_i, θ_r, θ_t jako úhel dopadu, odrazu, lomu. Mezi nimi platí Snellův zákon $n_1 \cdot \sin(\theta_i) = n_2 \cdot \sin(\theta_t)$

Rovnice 5.13 pro nepolarizovanou Fresnelovu odrazivost *R*, která kombinuje koeficienty odrazu pro rovnoběžnou i kolmou polarizaci a kterou využívá blok *Fressnel* v UE, je dána následujícím vzorcem.

$$R = 0{,}5 \cdot (R_s + R_p) \tag{5.13}$$



Obr. 5.20 Postprocess efekt využívající Fresnelův odraz

Fresnelův člen v modelech pro stínování, jako jsou ty používané v UE, funguje tak, že se vypočítá úhel mezi dopadajícím světlem a normálou povrchu v každém bodě povrchu. Počítá se také úhel mezi směrem pohledu a normálou povrchu. Na základě těchto úhlů se pak Fresnelův člen používá k řízení množství odrazu na povrchu.

Když je úhel pohledu rovnoběžný normále povrchu, dochází k nejmenšímu odrazu. Jak se úhel pohledu stává kolmějším, množství odrazu se zvyšuje. Pokud normála povrchu směřuje přímo na kameru, je na výstupu funkce hodnota 0, naopak pokud normála povrchu směřuje kolmo ke kameře, je na výstupu funkce hodnota 1. To vytváří přirozeně vypadající efekt, který simuluje chování světla na skutečných površích. Tento efekt je možné použít pro tvorbu řady vizuálních efektů, jako jsou vinětace, detekce hran nebo změny hloubky ostrosti. Následující obrázky (Obr. 5.19 a Obr.5.20) vyobrazují prostředí, na které je použit. [22]

Obr. 5.21 Původní render prostředí

Obr. 5.22 Prostředí po aplikaci efektu využívajícího Fresnelův odraz

6. TVORBA PILOTNÍHO PROJEKTU

Pro kompletní a plnohodnotnou demonstraci bylo vytvořeno speciální interaktivní 3D prostředí. V rámci tohoto prostředí je využito nástroje pro komunikaci mezi oběma používanými softwary (kapitoly 4.2 a 4.3) a jsou aplikovány různé druhy efektů (kapitola 5) v kombinaci se systémem pro ovládání kamery popsaným v kapitole 4.5.

Byl vytvořen plán a celková koncepce prostředí. Z důvodu celkové estetičnosti a ušetření části výkonu byla zvolena podoba pomyslné měsíční krajiny. Tato krajina totiž vypadá zajímavě a vizuálně příjemně i bez použití velkého množství 3D modelů, jako jsou rozlišné objekty nebo vegetace. Rozvržení prostředí bylo naplánováno radiálně jako krajina s kráterem uprostřed, která je dokola obklopena vysokými skalami, a to především kvůli plánovanému dominantnímu částicovému systému přímo uprostřed scény a celkovému vyzření.

Následující kapitoly stručně popisují tvorbu prostředí, implementaci osvětlovacích prvků, texturování materiálů a následnou implementaci audiovizuálních efektů.

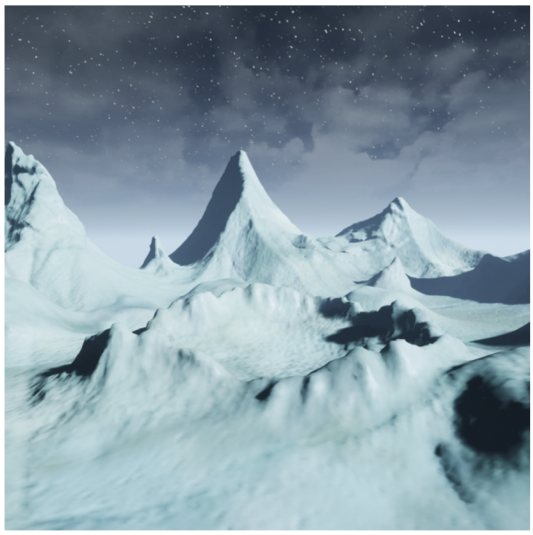
6.1 Implementace osvětlovacích komponentů

Nejprve byla vytvořena prázdná úroveň. Prvním krokem ve tvorbě prostředí je nastavení osvětlení. Ke snadnému nastavení osvětlení slouží nástroj *Environmental Light Mixer*. Je to okno editoru, ve kterém lze vytvářet a upravovat komponenty, které je jinak nutné implementovat manuálně. Jedná se o jednotlivé světelné komponenty prostředí, jako jsou obloha, mraky, atmosferické světlo a sluneční svit. Umožňuje rychle upravovat tyto komponenty a zvolit si množství detailů vlastností, ke kterým chceme následně přístup. Pomocí tohoto nástroje byla do prostředí přidána exponenciální výšková mlha, která vytváří větší hustotu v nízkých místech mapy a menší hustotu ve vysokých místech. Následně byl přidán Blueprint *BP_Sky_Sphere*, který se používá k vytvoření pozadí herního světa a k vytvoření pohlcujícího prostředí pro hráče. *BP_Sky_Sphere* obsahuje řadu funkcí, které lze přizpůsobit tak, aby vytvářely různé typy oblohy, například jasná obloha, zatažená obloha nebo noční obloha. Tyto funkce zahrnují model kupole oblohy, systém slunce a měsíce, vrstvu mraků, hvězdnou krajinu a barevný gradient horizontu.

6.2 Tvarování povrchu krajiny

Dalším krokem ve tvorbě prostředí je tvarování základního povrchu krajiny. Nejprve bylo zapotřebí změnit režim enginu ze základního módu na mód *Landscape*. V tomto režimu úprav krajiny byl terén tvarován pomocí různých nástrojů, jako je například zploštění, eroze a zvednutí/snížení. Pro vytvoření detailnější struktury a reálnějších tvarů byly použity alfa štětce, což jsou obrázky ve stupních šedi, které definují tvar a intenzitu tahu štětce. Tyto štětce byly následně aplikovány na nástroje pro tvarování terénu. Na terén byl dále aplikován vhodný materiál, který zahrnuje jak klasickou barevnou texturu, tak

i normálovou texturu. Normálová textura funguje tak, že kóduje informaci o orientaci normál povrchu v každém bodě na povrchu objektu. Tyto informace pak grafický engine používá ke změně způsobu interakce světla s povrchem a vytváří tak vzhled detailů povrchu. Dále je využita ORD textura, která ve svých kanálech nese informace o pohlcování světla, hrubosti a vychýlení geometrie v každém bodě textury. [23]



Obr. 6.1 Prostředí se světelnými komponenty a vytvarovaným povrchem

Obr. 6.2 Prostředí se světelnými komponenty, vytvarovaným povrchem a aplikovaným materiálem

6.3 Implementace vizuálních efektů

Pro toto 3D prostředí byla vytvořena sada speciálních efektů. Tyto efekty byly umístěny na předem zvolené, popřípadě náhodně generované pozice. Všechny níže popsané efekty jsou audioreaktivní a společně s prostředím utváří jeden celek. V rámci prostředí se jednotlivé efekty spouští na základě hudebních dat, případně mohou při změně hudební scény vznikat nebo zanikat.

6.3.1 Hanging particles

Jednoduchý částicový systém, který v závislosti na hudebním signálu umístí svítící částici na náhodné místo v předem stanoveném okruhu. Je vhodný pro melodie a jako výplň, spíše než dominantní vizuální prvek.

6.3.2 Sparks

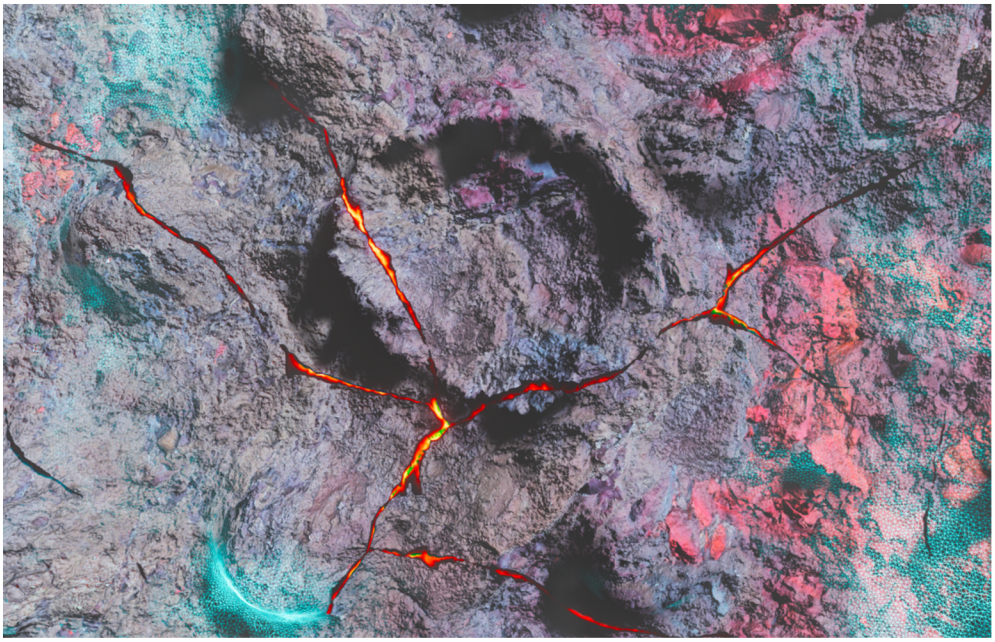
Tento částicový systém je dominantním prvkem celé vizuální scény. Je umístěn do kráteru, který se nachází uprostřed scény a je využíván jako cíl, na který je zaměřená kamera, nebo jako cíl, okolo které kamera obíhá. Vlastnosti systému jsou z velké části převzaty ze systému vytvořeném v kapitole 5.1. Audioreaktivním prvkem je zde velikost proměnné *Curl Noise Force*, která vychyluje částice pomocí vířivého šumu.

6.3.3 Dynamic wireframe overlay materiál

Dynamický materiál vytvořený v kapitole 5.2 byl modifikován a použit k upravení celkové atmosféry vizuální scény. Díky nové funkci Unreal Enginu 5.1. *Overlay Material* bylo možné a snadné aplikovat tento materiál jako další vrstvu, která překrývá materiál krajiny. Aby efekt nepůsobil rušivě, byl použit pouze na okrajové části scény, nikoli na středovou část, kde je dominantním prvkem velký částicový systém. Toho bylo docíleno násobením textury s radiálním exponenciálním gradientem s pevně daným poloměrem. K docílení zajímavějšího vizuálního efektu je také původní wireframe textura násobena šumovou texturou.

6.3.4 Lava Overlay

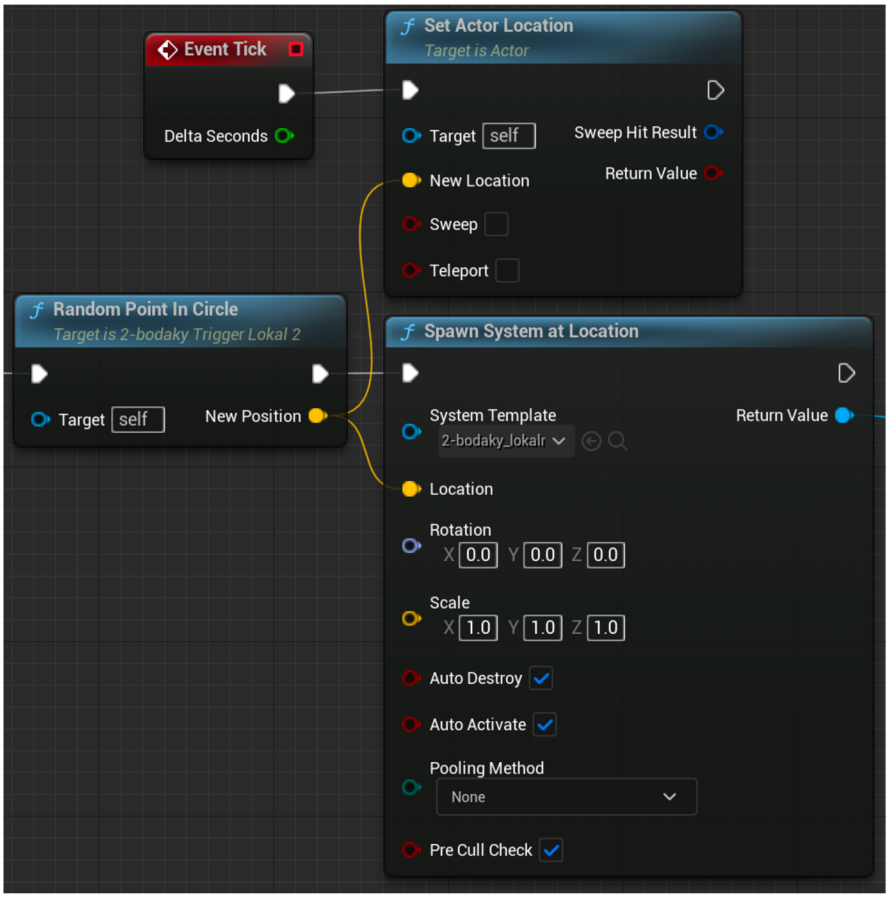
Pro dotvoření celkového dojmu a atmosféry byl následně přidán další efekt, tentokrát v podobě lávy, která byla opět přidána jako materiál překrývající původní materiál krajiny. Provedení materiálu je formou jednoduché textury, krycí maskou a vyzařovací barvou, jež je aplikována pouze na vnitřní část praskliny.



Obr. 6.3 Pohled na krajinu z vrchu po aplikaci překryvných materiálu

6.3.5 Spikes

Dalším prvkem tohoto prostředí jsou *Spikes*. Jedná se o využití systému z kapitoly 5.3. Hlavní podstatou tohoto prvku je, že se na základě hudebních signálů objevuje pokaždé na náhodném místě ve stanoveném rozsahu. Při použití funkcí *Set Actor Position* a *Spawn System at Location* (Obr. 6.3) lze docílit toho, že pokud bude zvolen cíl kamery právě na tento prvek, bude se její natočení měnit v závislosti na tom, na jaké pozici se efekt objeví.



Obr. 6.4 Implementace funkcí *Set Actor Location* a *Spawn System at Location* v Blueprintu pro spuštění efektu

6.3.6 Post Process materiál

Poslední z vytvořených vizuálních efektů pro pilotní projekt je postprocesní materiál, který je kombinací efektů z kapitol 5.4.1 a 5.4.2. Místo původní video textury byl zvolen blok *SceneTexture:PostProcessInput0*, který aplikuje materiál na celou obrazovku až na závěr, po vykreslení všeho ostatního. Míru saturace obrazu i množství prolnutí a manipulace glitch texturou jsou řízeny pomocí samostatného Blueprintu, který ovládá parametry z materiálové kolekce parametrů, jež je použita pro řízení post process materiálu samotného.

6.4 Funkční zapojení systému pro komunikaci programů, ovládání nástrojů a speciálních efektů

Pro funkčnost všech nástrojů a efektů je nutné přichystat jejich funkční zapojení. To je demonstrováno na pilotním projektu, který obsahuje předpřipravený hudební projekt v Abletonu a plně funkční vizuální scénu v UE.

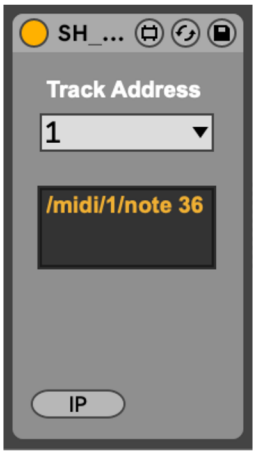
6.4.1 Ableton

V režimu *Live Session* programu Ableton byla předem vytvořena hudební skladba, která obsahuje několik různých nástrojů a několik scén.

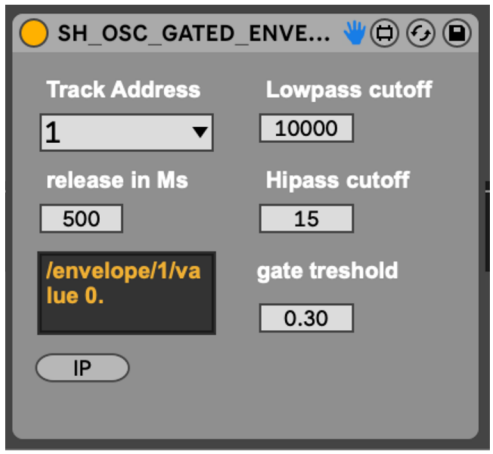
Do každého kanálu, který je použit k vizualizaci, byl přidán nástroj pro odesílání dat – *OSC MIDI Sender* (kapitola 4.2.1) nebo *OSC Gated Envelope Sender* (kapitola *4.2.2*). V těchto nástrojích bylo také zapotřebí zvolit hodnotu *Track Adress*, *IP Adress* a *port*, v případě posílání obálky signálu také mezní kmitočty filtrů a mezní hodnotu šumové brány.

6.4.2 Unreal Engine

Do vytvořeného prostředí v UE byl nejprve přidán Blueprint *OSC MIDI Router* (kapitola 4.3.2). Ten zajišťuje přijímaní OSC zpráv, jejich třídění a komunikaci s Blueprinty, které spouštějí události na vizuální scéně. V detailech tohoto Blueprintu byla zvolena stejná adresa a stejný port jako v Abletonu. Při zaškrtnutí pole *Print Messages* se všechny zprávy přijímané OSC serverem tisknou na plochu, lze tudíž snáze specifikovat možné problémy při párování obou programů.



Obr. 6.5 Grafické rozhraní OSC MIDI Sender

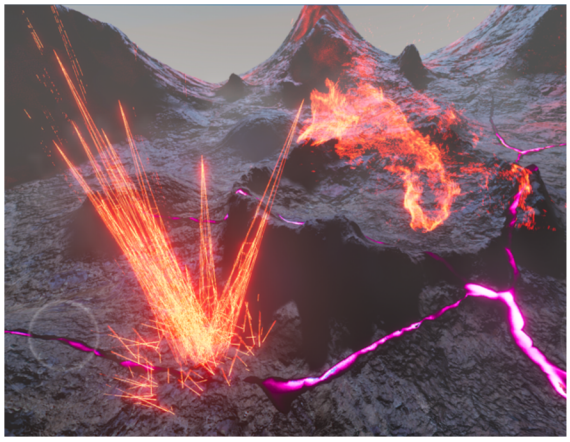


Obr. 6.6 Grafické rozhraní OSC Gated Envelope Sender

Dále byly do prostředí přidány Blueprinty pro spouštění vizuálních efektů. S Blueprintem se na scénu automaticky přidá i daný efekt. U efektů s předem stanovenou lokací tedy bylo zapotřebí umístit Blueprint na správné místo. Ve všech bylo nutné zvolit referenci na *OSC_MIDI_Router*, jinak by nedošlo k vzájemné komunikaci. Efektové Blueprity naslouchají každé příchozí zprávě a pokud dojde ke splnění podmínek pro spuštění, proběhne naprogramovaná efektová akce. Podmínky pro spuštění se nastavují manuálně v detailech Blueprintu. Tam je nutné zvolit hodnotu *Track Adress*, jejíž hodnota musí být stejná jako hodnota *Track Adress* stopy v Abletonu, pomocí které chceme efekty spouštět. Pro specifikaci konkrétních not, které by měly akce spouštět je nutné přidat hodnoty not do pole *Trigger Notes*. Pokud toto pole zůstane prázdné, efekt se spustí při libovolné notě z dané adresy. Při použití obálky lze frekvenční oblast spuštění efektu kontrolovat filtry a šumovou bránou, což se provádí přímo v Abletonu.

Podobným procesem bylo nutné postupovat při implementaci kamerového systému. Nejprve bylo zapotřebí manuálně umístit na scénu kamery, které chceme používat. Pro volbu kamer byl přidán Blueprint *BP_Camera_Set* a pro změnu cílů, na které bude kamera zaměřena, byl přidán *BP_look_at_targets*. V jeho detailech *BP_Camera_Set* bylo potřeba opět zvolit referenci na router, ale také na Blueprint pro měnění cílů. Následně bylo nutné do pole kamer přidat kamery. V detailech *BP_look_at_targets* bylo potřeba zvolit referenci a následně přidat cíle, na které se budou kamery zaměřovat, do pole cílů.

Po správném zapojení všech komponent, které bylo popsáno v předešlých odstavcích, zbývá při tvorbě této vizualizace poslední část, a tou je závěrečná optimalizace, aby vizuál nebyl příliš chaotický. Některé efekty bylo třeba obohatit o blok z kapitol 4.4.1 a 4.4.2, která stabilizují systém proti chaotickému chování, popřípadě upravit časové linky, podle kterých se modulují parametry efektů.



Obr. 6.7 Pilotní projekt v provozu

Obr. 6.8 Pilotní projekt v provozu (post process efekt)

ZÁVĚR

V rámci teoretické části diplomové práce byla zpracována rešerše na téma zvukového signálu, jeho charakteristik a reprezentace zvukového signálu v časové i frekvenční oblasti. Byl popsán způsob digitálního zpracování zvukových signálů a způsoby digitální komunikace mezi softwary a zařízeními, které pracují se zvukem a obrazem. Byl popsán software Ableton, ve kterém je část práce realizována. Dále bylo zpracováno téma digitální vizualizace zvukového signálu včetně softwarů, které se k tomuto účelu v dnešní době používají. Software Unreal Engine 5.1, ve kterém je provedena druhá část práce, byl popsán, stejně jako byly popsány principy tvorby speciálních efektů v tomto programu.

Praktická část diplomové práce se poté zabývá konkrétním návrhem vizualizace hudby dle předem stanovených cílů.

V DAW Ableton byl vytvořen program, který zaznamenává MIDI data, a jejich parametry následně v reálném čase zpracovává. Tato data jsou přiřazena konkrétní adrese a pomocí OSC zpráv jsou odeslána po lokální síti k dalšímu zpracování. Obdobným způsobem funguje druhý vytvořený program, který však zaznamenává tvar obálky hudebního signálu. Tento sledovač obálky byl následně modifikován takovým způsobem, aby bylo možné signál ve frekvenční oblasti filtrovat a také stanovit mezní hranici hlasitosti, od které se obálka zaznamenává pomocí šumové brány.

Vizualizace pomocí dat z hudebního softwaru probíhá v prostředí Unreal Engine 5.1. Byl naprogramován nástroj, který data z OSC zpráv na lokální síti přijímá. Dále provádí kontrolu validity dat a validní data třídí podle jejich typu a podle uživatelsky stanovených identifikátorů, které následně usnadňují přiřazování efektů na vizuální scéně jednotlivým nástrojům, zvukům, nebo jejich skupinám.

Druhý nástroj vytvořený v UE poté slouží ke spouštění událostí na vizuální scéně. Prvním krokem je kontrola kritérií, přičemž pokud OSC zpráva uživatelsky stanovená kritéria splňuje, dochází ke spuštění události. Ty mohou následně modulovat libovolný parametr vizuální scény nebo například spouštět procesy pomocí časových os. Byl navržen také způsob spouštění událostí na základě časového odpočtu nebo repetice OSC zpráv pro zamezení chaotického chování vizualizace.

Byl vytvořen systém pro ovládání kamer, který umožňuje měnit pohled mezi různými kamerami na základě hudebních dat. Tento systém také skýtá možnost sledování daného objektu kamerou a spouštění otřesů kamery.

K demonstraci funkčnosti navrženého řešení vizualizace hudby byla vytvořena sada efektů na bázi částicových systémů, dynamických materiálových instancí, post process efektů a jejich kombinací. Tyto efekty byly následně použity při tvorbě 3D interaktivní scény v rámci pilotního projektu, která prezentuje kombinaci všech výše zmíněných nástrojů a efektů.

Navržené řešení vizualizace zvuku umožňuje tvorbu vizuální scény na základě MIDI dat z daného kanálu, na základě konkrétní MIDI noty (nebo jejich výčtu) nebo na základě

tvaru obálky zvukového signálu. Běžně dostupné systémy tvoří vizualizaci hudby pouze na základě obálky zvuku a statických 2D videosekvencí, obrazů nebo textur. Jedinečností tohoto systému je, že vizualizace probíhá na základě MIDI dat nebo obálky zvuku generativně v 3D prostředí a vizuál je pokaždé zcela unikátní. Velkou výhodou tohoto systému je široké spektrum jeho použití.

Nástroj byl vyvinut především pro vizualizaci smyčkových DJ setů, nicméně jeho použití je v konečném důsledku velmi široké. Lze ho použít prakticky při jakékoli práci s hudbou v programu Ableton, ať už je to živý DJ set, samostatná skladba nebo improvizace. Další možností využití tohoto nástroje může být přenesení hudebního zážitku do čistě virtuálního prostředí, tvorba videoklipů nebo audioreaktivních vizuálů pro streamovací platformy.

LITERATURA

[1] SYROVÝ, Václav. Hudební akustika. 3., dopl. vyd. V Praze: Akademie múzických umění, 2013. Akustická knihovna Zvukového studia Hudební fakulty AMU. ISBN 978-80-7331-297-8.

[2] Audio signal. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-12-01]. Dostupné z: https://en.wikipedia.org/wiki/Audio_signal

[3] DE LEON, Phillip L. Computer Music in Undergraduate Digital Signal Processing [online]. [cit. 2022-12-01].

[4] MIŠUREC, Jiří a Zdeněk SMÉKAL. Číslicové zpracování signálů. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. ISBN 978-80-214-4448-5.

[5] Schimmel, J. 20. 12. 2002. Komunikační rozhraní MIDI. Elektrorevue [online], no. 69. Available from: http://www.elektrorevue.cz/clanky/02069/index.html#tab3 [cit. 2013-28-12]

[6] Oficiální dokumentace protokolu MIDI [online]. MIDI Association, 2021 [cit. 2021-12-11]. Dostupné z: https://www.midi.org/.

[7] BREVE, Bernardo, Stefano CIRILLO, Mariano CUOFANO a Domenico DESI-ATO. Perceiving space through sound: mapping human movements into MIDI [online], University of Salerno, Fisciano, 2020-07-07, s. 49-56 [cit. 2022-05-26]. Dostupné z: doi:10.18293/DMSVIVA20-011.

[8] Open Sound Control [online]. 2021 [cit. 2022-12-01]. Dostupné z: https://ccrma.stanford.edu/groups/osc/index.html

[9] Kefauver, Alan P.; Patschke, David (2007-01-01). Fundamentals of Digital Audio, New Edition. A-R Editions, Inc. p. 133. ISBN 9780895796110.

[10] DMX 512: Information and a Real World Guide to DMX-512 Protocol [online]. [cit. 2022-12-03]. Dostupné z: http://www.dmx-512.com

[11] Resolume [online]. [cit. 2022-12-04]. Dostupné z: https://resolume.com

[12] Max: Max - Product Site [online]. [cit. 2022-12-04]. Dostupné z: https://cycling74.com/products/max

[13] Pure Data [online]. [cit. 2022-12-04]. Dostupné z: https://puredata.info

[14] Unity [online]. Unity Technologies, 2022 [cit. 2022-12-04]. Dostupné z: https://unity.com

[15] Unreal Engine [online]. Epic Games, 2022 [cit. 2022-12-04]. Dostupné z: https://www.unrealengine.com/en-US¨

[16] YEE, Erica. The real reason Epic landed a $15 billion valuation is not Fortnite's viral video game success [online]. 21.12.2018 [cit. 2022-12-04]. Dostupné z: https://www.cnbc.com/2018/12/14/the-reason-epic-landed-a-15-billion-valuation-is-not-fortnite-success.html

[17] BRIGHTMAN, James. Epic Games launches Unreal Government Network for serious games applications [online]. [cit. 2022-12-04]. Dostupné z: https://www.gamesindustry.biz/epic-games-launches-unreal-government-network-for-serious-games-applications
[18] HASTINGS a Ratan K GUHA. Interactive Evolution of Particle Systems for Computer Graphics and Animation [online]. 05.2009 [cit. 2022-12-06].
[19] Instanced Materials [online]. [cit. 2023-05-11]. Dostupné z: https://docs.unrealengine.com/5.0/en-US/instanced-materials-in-unreal-engine/
[20] Post process materials [online]. [cit. 2023-05-11]. Dostupné z: https://docs.unrealengine.com/5.0/en-US/post-process-materials-in-unreal-engine/
[21] Math behind colorspace conversions, RGB-HSL [online]. [cit. 2023-05-18]. Dostupné z: https://www.niwa.nu/2013/05/math-behind-colorspace-conversions-rgb-hsl/
[22] Using Fresnel in your Materials: Guide for using the Fresnel Material node. [online]. [cit. 2023-05-17]. Dostupné z: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/HowTo/Fresnel/
[23] Fresnel Equations [online]. [cit. 2023-05-17]. Dostupné z: https://en.wikipedia.org/wiki/Fresnel_equations
[24] Texturování [online]. [cit. 2023-05-11]. Dostupné z: https://cs.wikipedia.org/wiki/Texturování

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

FEKT	Fakulta elektrotechniky a komunikačních technologií
VUT	Vysoké učení technické v Brně
AD	Analog to Digital
DA	Digital to Analog
ADSR	Atack-decay-sustain-release
DFT	Diskrétní Fourierova transformace
FFT	Rychlá Fourierova transformace
MIDI	Musical Instrument Digital Interface
OSC	Open Sound Control
CNMAT	Center for New Music and Audio Technologies
DAW	Digital audio workstation
HiFi	High Fidelity
DMX	Digital Multiplex
LED	Light Emitting Diode
VFX	Visual Effects
FPS	First Person Shooter
VJ	Video Jockey
DJ	Disc Jockey
Pd	Pure Data
UE	Unreal Engine
GPU	Graphics Processing Unit

SEZNAM PŘÍLOH

Příloha A – Zdrojové soubory pro Max for Live a Unreal Engine 5.1

Příloha B – Návod pro uživatele

Všechny přílohy jsou formou elektronické přílohy, která je nahrána v IS VUT Brno.