Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Vatovani

Igra generirana na osnovi podane glasbe

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Peter Peer

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Igra generirana na osnovi podane glasbe

Tematika naloge:

Implementirajte igro, ki na podlagi poljubno izbrane pesmi generira vsebino igre ter poskrbi za dobro uporabniško izkušnjo s pravilno sinhronizacijo poteka pesmi in generirane vsebine na osnovi oblikovane igralnosti. Predstavite potrebna razvoja orodja, uporabljene algoritme za analizo glasbe, potek implementacije, na koncu pa naredite evalvacijo igre z uporabo ustrezne ankete med igralci ter na osnovi samodejno zbranih podatkov odigranih iger.

Zahvaljujem se svojemu mentorju za pomoč in nasvete pri izdelavi diplomskega dela. Prav tako bi se rad zahvalil družini in prijateljem za spodbudo in motivacijo med delom. Še posebej bi se rad zahvalil svoji sestri Lauri za pomoč pri pravopisu.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
FFT	Fast Fourier Transform	Hitra Fourierjeva transformacija
IDE	Integrated Development Environment	Integrirano razvojno okolje
SF	Spectral Flux	Spektralni pretok
MPEG	Moving Picture Experts Group	Ekspertna skupina premikajoče slike
MP3	MPEG Audio Layer III	MPEG zvočni sloj III
REST	Representational State Transfer	Prenos predstavitvenega stanja
HTTP	Hypertext Transfer Protocol	Hipertekstovni prenosni protkol
BPM	Beats Per Minute	Udari na minuto

Povzetek

Naslov: Igra generirana na osnovi podane glasbe

Avtor: Tadej Vatovani

V diplomski nalogi se ukvarjamo s problemom, kako iz pesmi izluščiti informacije, ki bi jih lahko nato uporabili v računalniški igri. Pri tem gremo čez osnove procesiranja signalov, kako transformirati podatke v frekvenčno domeno preko hitre Fourierjeve transformacije in kako le-te uporabljamo in razumemo. Nato opišemo nekatere algoritme, ki se jih uporablja pri zaznavanju nastopov tonov in našo implementacijo ter izboljšavo enega od teh algoritmov. Po pregledu algoritmov, ki smo jih uporabili za interpretacijo pesmi, nato predstavimo, kako se jih uporablja v igri. Ko končamo z opisom igre in njeno implementacijo, še ocenimo, kako uspešni smo bili v našem zadanem cilju.

Ključne besede: glasba, Unity, C#, igra, 3D, vizualizacija.

Abstract

Title: Song based generated game

Author: Tadej Vatovani

In the thesis we aim to analyze a song and gain information with which we can build a video game. We first go through the basics of signal processing, transforming data into the frequency domain through the fast Fourier transform and how we can use and interpret the data. After that we describe the main algorithms that are currently used for onset detection and how we implemented and improved one of them. Then we show how we used our results in a video game. When we are finished describing our implementation and the game, we go through the feedback we gained from our players and evaluate how successful we were.

Keywords: music, Unity, C#, game, 3D, visualization.

Poglavje 1

Uvod

Glasba nas spremlja vse življenje. Nanjo vežemo naše najbolj pomembne spomine in trenutke. V tej diplomski nalogi razvijamo še dodaten način, kako doživljati našo glasbo in sicer preko računalniške igre.

Pri diplomski nalogi smo imeli dva glavna cilja:

- Ustvariti razširitev v Unity, s katero se lahko ustvarja igre bazirane na glasbi podane z igralčeve strani.
- Uporabiti to razširitev in ustvariti igro.

V drugem poglavju prikažemo orodja, ki smo jih uporabili pri diplomski nalogi. Pri temu opišemo zakaj in kako smo izbrana orodja uporabili.

V tretjem poglavju predstavimo osnove digitalnega procesiranja signala, transformacije zvočnega signala v frekvenčno domeno in pomen frekvenc. Nato predstavimo glavne informacije, ki jih hočemo pridobiti iz pesmi. Pri tem je glavni algoritem, na katerega se osredotočamo spektralni pretok, s čimer hočemo zaznati, kdaj v pesmi nastopijo toni. Potem prikažemo energijo frekvenc in strukturo pesmi.

Ko končamo s pregledom in implementacijo algoritmov, ki jih uporabljamo pri analizi glasbe, se premaknemo na implementacijo naše razširitve v računalniški igri. Ko predstavimo v četrtem poglavju nekaj osnov, ki jih rabimo pri generaciji 3D objektov, opišemo, kako se naša igra poveže z našo

razširitvijo.

V petem poglavju predstavimo, kako izgleda in deluje naša igra, kaj v njej igralci delajo in kakšen odziv smo od njih dobili. Glede na odziv igralcev in statistiko, ki smo jo zbrali medtem, ko so igrali, ovrednotimo, kako uspešni smo bili pri doseganju našega zadanega cilja.

Poglavje 2

Uporabljena orodja

2.1 Pogon Unity

Unity je igralni pogon za razvoj iger za različne platforme. Igralni pogon je razvijalno okolje za ustvarjanje iger, ki olajša delo, saj poskrbi za velik del razvoja igre, ki bi bil brez njega za posameznika nedosegljiv.

Glavne naloge za katere je zadolžen Unity:

- 3D in 2D izris.
- Uporabniški vmesnik pri menijih.
- Uporabnikove vhodne naprave.
- Detekcija trkov in fizika.
- Predvajanje zvoka.
- Kreiranje scene.

Za Unity smo se odločili zaradi hitrega razvoja, saj je lahek za uporabo in zelo razširjen, zaradi česar ima veliko gradiva, s katerim si lahko pomagamo [8].

2.2 C#

Za nadzor nad objekti v Unity se uporablja skripte. Skripte nam omogočajo dostop do večine objektov in njihovih lastnosti, s čimer lahko v igri definiramo obnašanje njenih sestavnih delov. Skripte se piše v C# programskem jeziku. C# je objektno orientiran programski jezik, ki sledi večini modernih paradigem programskih jezikov. C# uporablja samodejno zbiranje smeti (angl. garbage collector), zaradi česar programerju ni treba skrbeti za dodelitev in sprostitev spomina, kar zelo olajša delo pri pisanju iger, saj delamo z veliko dinamičnimi objekti. C# sloni na Microsoftovem .NET ogrodju, kar razširi njegov dostop do različnih funkcij.

2.3 Visual Studio IDE

Visual Studio je integrirano razvojno okolje, ki se ga uporablja pri pisanju programskih jezikov, ki slonijo na .NET ogrodju. Podpira razhroščevanje kode, pregled sintakse, formatiranje in samodejno dokončanje kode.

2.4 3ds max

3ds max je program namenjen 3D modeliranju in animaciji. Glavna funkcija je ustvarjanje in urejanje poligonske mreže, ki predstavlja naše 3D modele. Ko smo ustvarili poligonsko mrežo, smo preko 3ds maxa nastavili material, ki ga uporablja posamezni del 3D modela. Končni 3D model smo nato izvozili iz 3ds maxa v Unity [1].

2.5 NodeJS

Med igranjem igre zbiramo statistiko. Ker uporabnike nismo hoteli obremeniti s pošiljanjem statistike, smo se odločili, da bomo postopek avtomatizirali s pomočjo strežnika. Za razvoj strežnika smo uporabili NodeJS. Za NodeJS

smo se odločili, saj je enostaven za uporabo in primeren za REST arhitekturo, na kateri sloni strežnik [9].

Poglavje 3

Analiza glasbe

3.1 Branje in uvoz datotek

Ko uporabnik zažene igro, s svojega lokalnega diska izbere datoteko, ki vsebuje glasbo. Pri tem smo se osredotočili na en tip zapisa zvoka in sicer MP3. Za MP3 zapis smo se odločili predvsem zaradi njegove razširjenosti, kar omogoča dostopnost večjemu številu igralcev.

3.1.1 MP3 tip datoteke

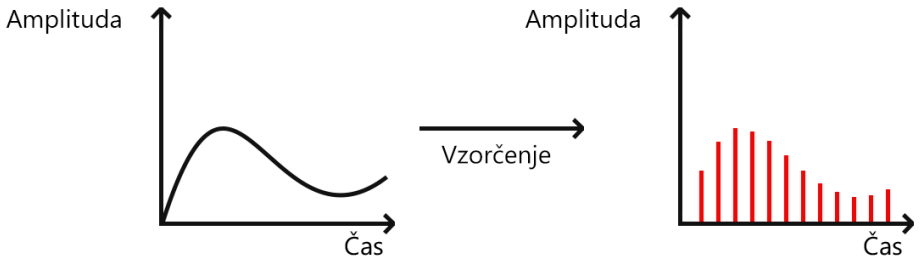
MP3 je stisnjen zapis zvoka z izgubo, razvit s strani MPEG. Glavni cilj MP3 zapisa je učinkovit zapis zvoka, kateri minimalno vpliva na kvaliteto zvoka. To doseže s tem, da iz zapisa odstrani nepotrebne frekvence.

Ker fokus te diplomske naloge ni v različnih zapisih zvoka, ampak analiza glasbe, smo za dekodiranje zapisa uporabili odprto kodno knjižnico mpg123 [3].

3.1.2 Vzorčenje

Pri digitalnem procesiranju signalov je vzorčenje postopek, ki spremeni nek zvezni signal v diskretni signal, sestavljen iz zaporedja vzorcev, prikazan na sliki 3.1. Vzorec je vrednost signala v neki točki v času. Pri izbiri števila vzorcev se odločamo med natančnostjo in količino podatkov, ki jih moramo

obdelati, kar vpliva na hitrost algoritma.



Slika 3.1: Primer vzorčenja neprekinjenega signala.

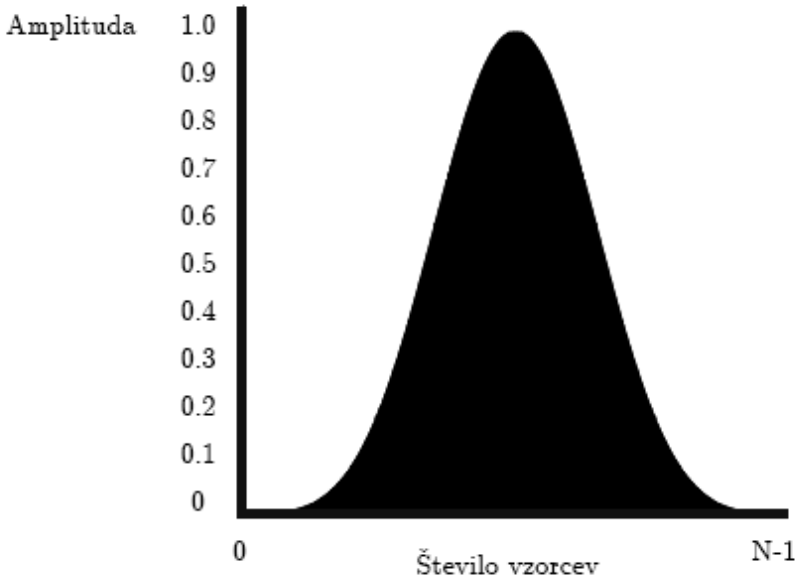
3.1.3 Hitra Fourierjeva transformacija

Zvočni signal, ki ga pridobimo iz MP3 datoteke, moramo nato preoblikovati.

Preden podatke transformiramo preko FFT, jih preoblikujemo s tem, da njihove vrednosti pomnožimo z Blackman-Harris okensko funkcijo(angl. window function), prikazano na sliki 3.2 . Okenske funkcije so funkcije, ki imajo izven nekega intervala nično vrednost. S tem se znebimo podatkov, ki so izven intervalov, ki nas zanimajo. Okenske funkcije so zglajene krivulje v obliki zvonca (podobne normalni porazdelitvi), saj s tem podatke prilagodijo FFT, ki domneva, da bo funkcija, ki jo transformira, neskončna in periodična. Prav tako se znebimo skrajnih vrednosti, ki bi lahko negativno vplivale na naše rezultate.

Fourierjeva hitra transformacija je funkcija, ki nam naše podatke premakne iz časovnega v frekvenčni prostor. Do frekvenčnih podatkov pridemo s tem, da vsak vzorec pesmi razdelimo na skupek več sinusnih funkcij. S tem pridobimo informacijo, katera frekvenca je nastopila in s kakšno amplitudo [4].

Rezultat FFT je 38 vzorcev na sekundo, s katerimi nato delamo nadaljnjo analizo.



Slika 3.2: Porazdelitev funkcije Blackman-Harris.

3.2 Pomen frekvenc

Vsaka frekvenca je odgovorna za svoj ton, zato lahko razdelimo frekvence po različnih tonih, ki jih hočemo najti.

Frekvence smo razdelili v tri skupine (opisane z intervali vrednosti frekvenc):

- Nizke: Pri tem so mišljeni predvsem basi, bobni in nižji vokali. Nahajajo se med 0 Hz in 110 Hz.
- Srednje: Večinoma glavna melodija pesmi. Nahajajo se med 110 Hz in 8190 Hz.
- Visoke: Višji toni, na primer činele bobnov, razni žvenketi ali višji toni petja. Nahajajo se med 8190 Hz in 13150 Hz.

Za kvalifikacijo v skupine smo najprej začeli z določitvijo intervalov glede na frekvence glavnih inštrumentov in vokalov, nato pa z eksperimentiranjem

v igri in opazovanjem rezultatov, ki smo jih dobili z različnimi intervali, še natančneje določili.

S tem ko frekvence omejimo na določene skupine, lažje izoliramo določene dogodke in tone, ki nastopijo v glasbi.

3.3 Glavni algoritmi

Za vizualizacijo in generacijo igre nas zanimajo tri glavne informacije:

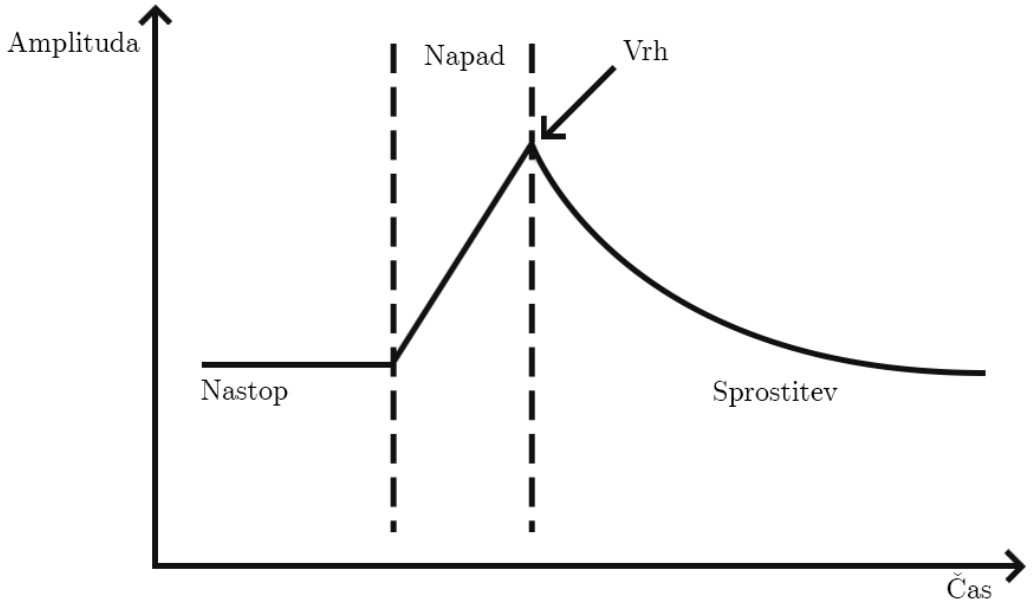
- Nastop glasbenih tonov.
- Energija frekvenčnih skupin.
- Struktura pesmi.

3.3.1 Nastop glasbenih tonov

Na sliki 3.3 vidimo sestavo tona. Pri tonu nas zanima čas nastopa ton. Ko zaznamo nastope tonov, pridobimo informacijo, kdaj v pesmi se nekaj dogaja. Če je nastopil nek ton, označimo kje v času je vrh tega tona, s čimer lahko določene sestavne dele igre sinhroniziramo z glasbo [5].

Za zaznavanje nastopov tonov poznamo tri glavne algoritme:

- Spektralni pretok: Ta algoritem dela na principu pretoka, kar pomeni, da za vsak vzorec pogleda spremembo v amplitudi frekvenc iz prejšnjega vzorca [7].
- Fazno odstopanje: Namesto da se osredotočamo na amplitude frekvenc, raje gledamo fazno razliko v sinusni funkciji zvoka. Naprej se izračuna frekvenca iz razlike faz med trenutnim in prejšnjim vzorcem, nato pa se iz razlike med frekvencami izračuna fazno odstopanje [6].
- Kompleksno odstopanje: Pri kompleksnem odstopanju kombiniramo amplitudo frekvenc in spremembo faz. Pri tem glede na faze in amplitudo algoritem napove, kakšno bo stanje v naslednjem oknu glede na



Slika 3.3: Sestava glasbenega tona.

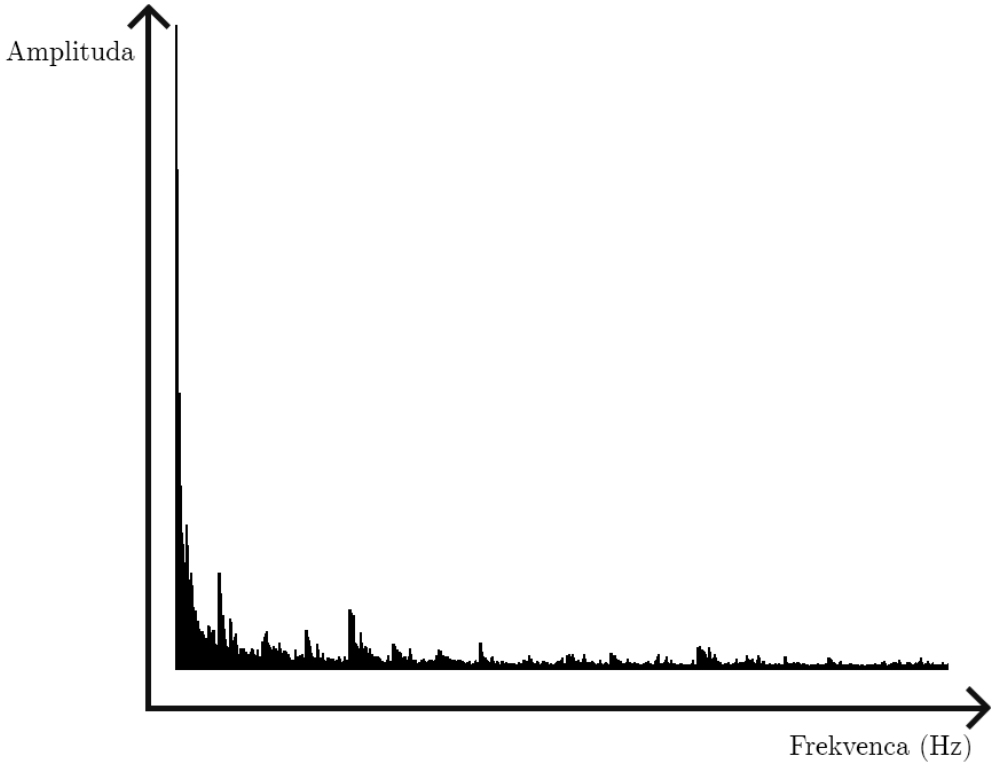
stanje prejšnjih dveh oken. Nato z razliko med napovedjo in resničnim stanjem odkrijemo odstopanja [7].

Algoritem, ki smo ga izbrali v svoji implementaciji, je spektralni pretok, saj ima glede na raziskavo [7] najmanjšo napako in največjo natančnost ter je tudi enostavnejši za implementacijo.

3.3.2 Spektralni pretok

Spektralni pretok je funkcija, ki nam vrne vrednost spektra skozi čas. Spekter je sestavljen iz energije pesmi, tj. amplituda vseh frekvenc v nekem času. Spekter smo nato razdelili na več vzorcev, ki predstavljajo spekter v nekem trenutku v času. Primer vrednosti enega izmed vzorcev lahko vidimo v sliki 3.4.

Za izračun spektralnega pretoka algoritem izračuna razliko med trenutnim in prejšnjim vzorcem. Vrednost vzorca je povprečje amplitud frekvenc. Pri enačbi 3.1 s predstavlja amplitudo frekvence, i frekvenco in k trenutni vzorec.



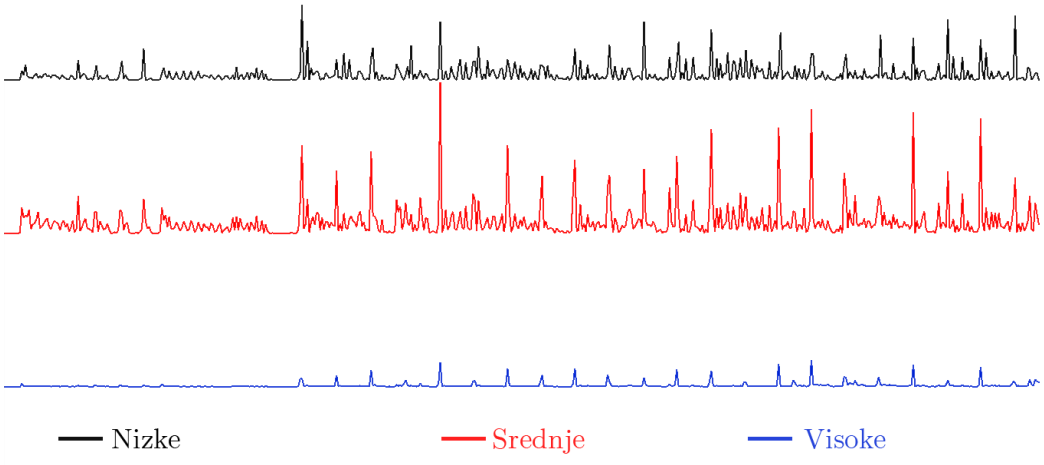
Slika 3.4: Primer vzorca spektra.

$$SF(k) = \sum_{i=0}^{n-1} s(k,i)–s(k-1,i) \tag{3.1}$$

Za povečanje natančnosti smo vzorce razdelili po skupinah frekvenc, ki smo jih določili v poglavju 3.2. S tem smo lahko pridobili nastop tonov v različnih skupinah, s čimer se lahko osredotočamo na določene dele pesmi.

Kot vidimo v sliki 3.5 so rezultat spektralnega pretoka 3 funkcije. Vsaka funkcija predstavlja eno od frekvenčnih skupin in njeno vrednost skozi čas.

Iz rezultata, ki nam ga vrne spektralni pretok, nato izračunamo prage, pri katerih pričakujemo nastop tonov. Prage pridobimo tako, da za vsak vzorec vzamemo vrednosti petih sosednjih vzorcev in izračunamo njihovo povprečno vrednost. To vrednost nato pomnožimo z neko konstanto. Konstanta določi,



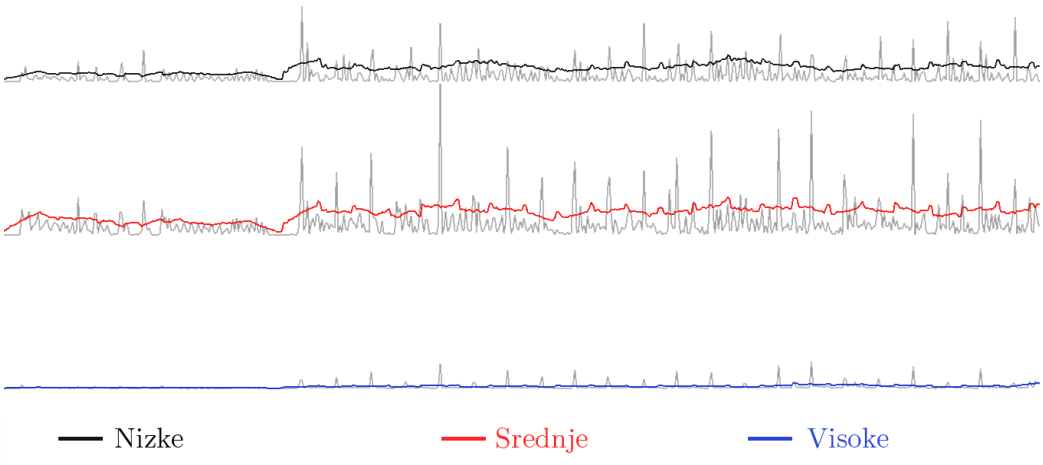
Slika 3.5: Rezultat spektralnega pretoka.

koliko večji mora biti pretok od praga, da je tam nastopil nek ton. Ker smo pretok razdelili na več skupin, lahko to konstanto določimo za vsako skupino posebej:

- Nizke frekvence: 2,15.
- Srednje frekvence: 1,75.
- Visoke frekvence: 1,35.

Nižje frekvence imajo višjo amplitudo, zato imajo večje konstante. Podobno kot pri frekvenčnih skupinah smo z eksperimentiranjem v igri in opazovanjem rezultatov nato natančneje določili konstante. Večja konstanta pomeni, da upoštevamo le večja odstopanja, zato imamo manj lažno pozitivnih rezultatov, vendar v primeru, da je povprečje zelo visoko, izgubimo nekaj nastopov, ki bi jih drugače lahko zaznali.

Ko pridobimo prage za vsako frekvenčno skupino, poiščemo nastope tonov. Primer pridobljenih pragov lahko vidimo na sliki 3.6. Od spektralnega pretoka odštejemo prage, nato iščemo vrhove. Vrh zaznamo, če je zaporedna vrednost manjša od prejšnje. Ker nastopi običajno niso zaporedni, ignoriramo zaporedne nastope.



Slika 3.6: Pragi, ki smo jih pridobili iz spektralnega pretoka.

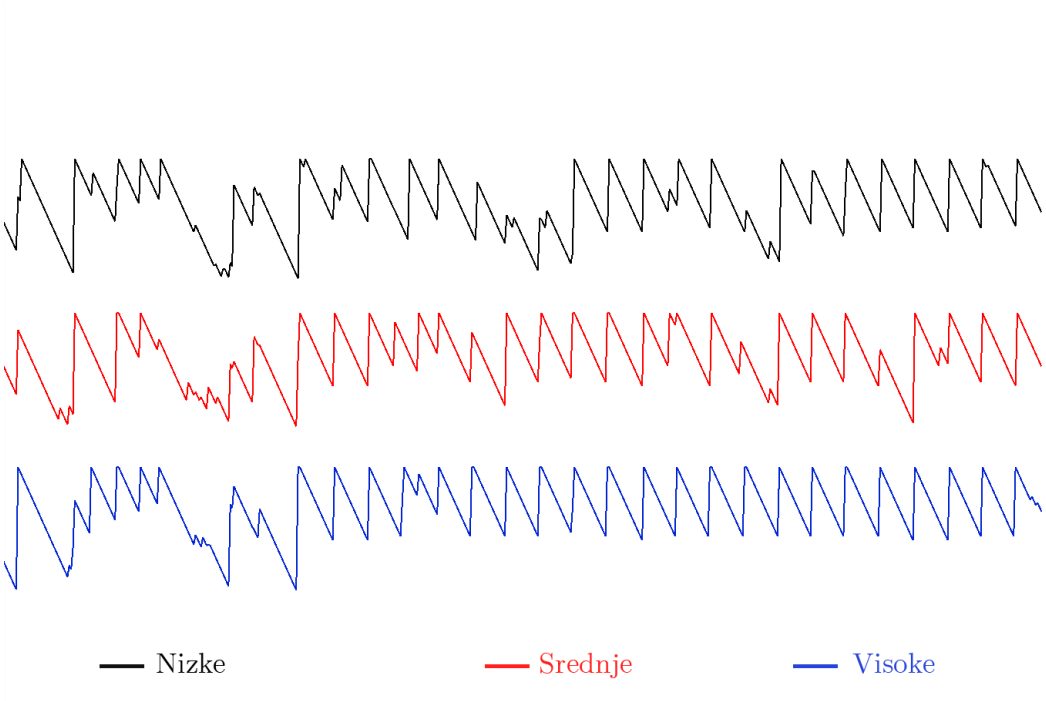
Ko smo zaključili s tem postopkom, imamo zabeležene vse nastope tonov v pesmi.

3.3.3 Energija frekvenčnih skupin

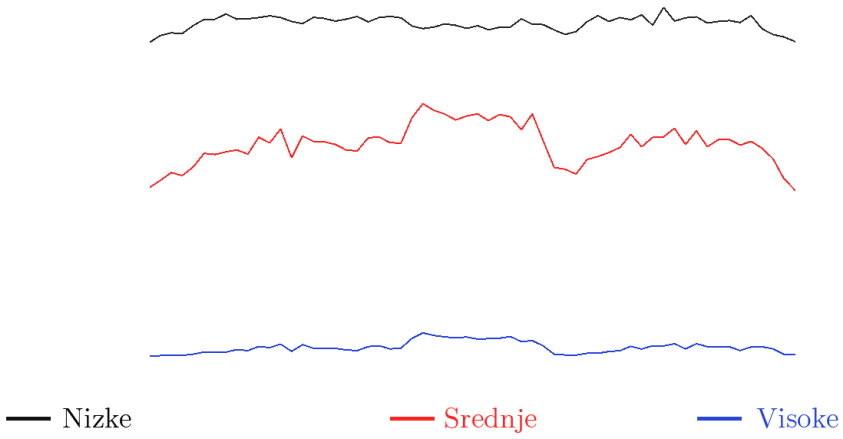
Za vsako frekvenčno skupino pridobimo energijo. Do energije pridemo tako, da spektralni pretok delimo s pragom in rezultat omejimo med 0 in 2. Vrednosti, kot se jih vidi v sliki 3.7, nato povežemo s sestavnimi deli igre.

3.3.4 Struktura pesmi

Vse amplitude razdelimo na dele s 76 vzorci. Za 76 vzorcev smo se odločili, ker s tem dobimo globalne maksimume, vendar se vrednosti še vedno zmerno hitro spreminjajo. Za vsak posamezni del izračunamo povprečje. Iz teh povprečij nato dobimo krivuljo poteka pesmi. S pomočjo strukture pesmi vidimo, kako niha energija pesmi, iz česar lahko razberemo različne dele pesmi. V sliki 3.8 se v srednjih frekvencah vidi začetek pesmi, ko krivulja narašča, nato so najvišje amplitude, kjer v pesmi nastopi inštrumentalni solo.



Slika 3.7: Energija frekvenčnih skupin.



Slika 3.8: Struktura pesmi.

Poglavje 4

Implementacija igre

4.1 Generiranje sveta igre

Glavna skripta, ki generira svet igre je zadolžena za:

- Generacijo poligonske mreže.
- Postavitev sestavnih delov igre.

Pri igri je mapa razdeljena na več delov. S tem optimiziramo igro, da ne izrisuje nepotrebnih delov mape, ki jih uporabnik ne vidi. V igri je aktivnih 5 delov naenkrat. Tipično en del predstavlja 5 sekund pesmi.

4.1.1 Poligonska mreža

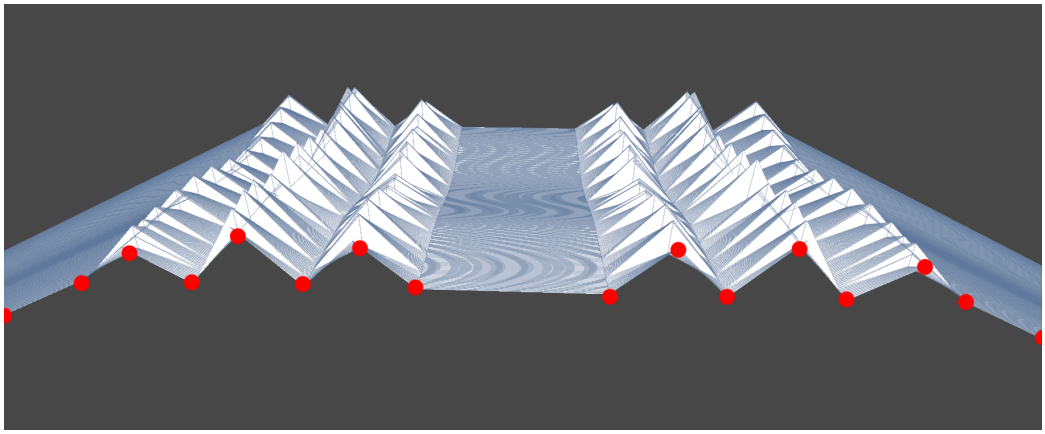
Poligonska mreža je način, kako predstaviti 3D objekte v računalniški grafiki. Sestavljena je iz oglišč in trikotnikov. Vsaka poligonska mreža ima definiran seznam oglišč in seznam, ki poveže oglišča v trikotnike. V igri se iz glasbe generirata dve glavni poligonski mreži:

- Ozadje.
- Cesta po kateri se premika igralec.

Preden se pri generaciji uporablja struktura pesmi, jo zgladimo in normaliziramo, s čimer se znebimo artefaktov, ki bi nastali pri prevelikih razlikah v vrednostih strukture. Zgladimo jo tako, da med dvema sosednjima vrednostima dodamo vrednosti, ki jih določimo s kosinus funkcijo, medtem ko jo normaliziramo z deljenjem s povprečno vrednostjo strukture.

4.1.2 Ozadje

Ozadje je skupek gričev, ki se premikajo glede na glasbo. Vsak del ozadja vzame 200 vzorcev iz glasbe in nato zgradi griče. En grič predstavlja eno izmed frekvenčnih skupin. Za vsak vzorec v mrežo dodamo 16 oglišč, ki so na sliki 4.1 označena z rdečimi točkami. Osnovno y koordinato oglišč določimo s strukturo pesmi, nato pa za vsako frekvenčno skupino povečamo y za njihovo energijo. Ko so griči zgrajeni, se med igranjem njihova y os prilagaja glede na trenutno energijo v pesmi.

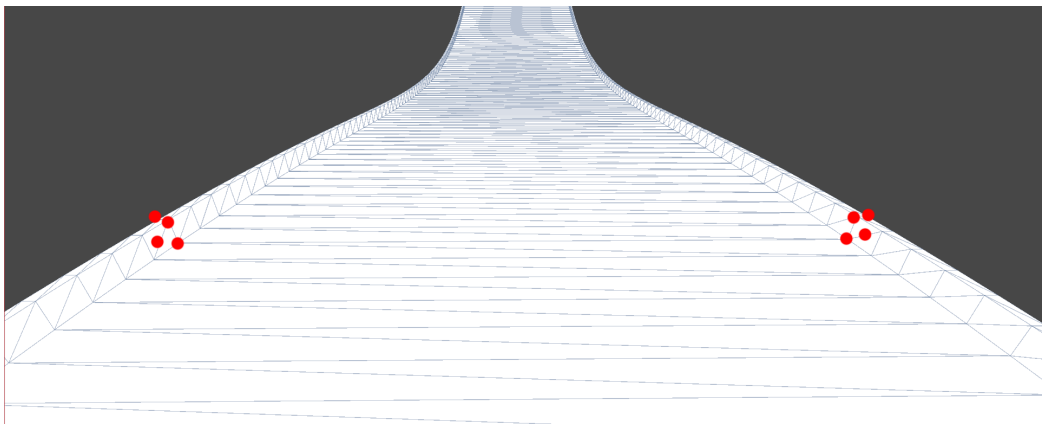


Slika 4.1: Generirana poligonska mreža ozadja.

4.1.3 Cesta po kateri se premika igralec

Cesta je zgrajena podobno kot ozadje, le da upošteva samo strukturo pesmi. Za vsak vzorec v mrežo dodamo 8 oglišč, ki so na sliki 4.2 označena z rdečimi

točkami. Za y oglišč vzamemo strukturo pesmi. Ker je cesta samo ena, se upošteva samo srednjo frekvenčno skupino.



Slika 4.2: Generirana poligonska mreža ceste.

4.1.4 Postavitev sestavnih delov igre

Ko skripta generira poligonske mreže, v igro postavi različne sestavne dele, s katerimi se igralec srečuje. Sestavni deli so diamanti, bombe, ovire in okrasne luči. Te dele skripta postavi glede na čas, v katerem smo zaznali nastope tonov.

4.1.5 Materiali in senčilniki

Materiali in senčilniki določajo, kako se izrišejo 3D objekti oziroma poligonske mreže v igri. Materiali določijo lastnosti objektov, medtem ko senčilniki te lastnosti uporabijo pri izrisu objektov.

Pri igri se uporablja štiri glavne senčilnike:

- Privzeti senčilnik v Unity.
- Senčilnik neba, ki na nebu (angl. skybox) izriše več krogov, katerim lahko preko materiala določaš velikost, barvo in svetlost.
- Senčilnik bomb in ovir, ki ustvarijo učinek stekla oziroma energije.

- Senčilnik luči, ki ustvari podoben učinek kot snop luči (angl. spot light), saj bi bil izračun večih snopov luči preveč računsko zahteven.

4.2 Sprožilci in modifikatorji

Glavni skripti, ki sta zadolženi za upravljanje s sestavnimi deli igre, v ritmu glasbe, sta sprožilec in modifikator. Oba kot parameter prejmeta katero frekvenčno skupino naj nadzirata.

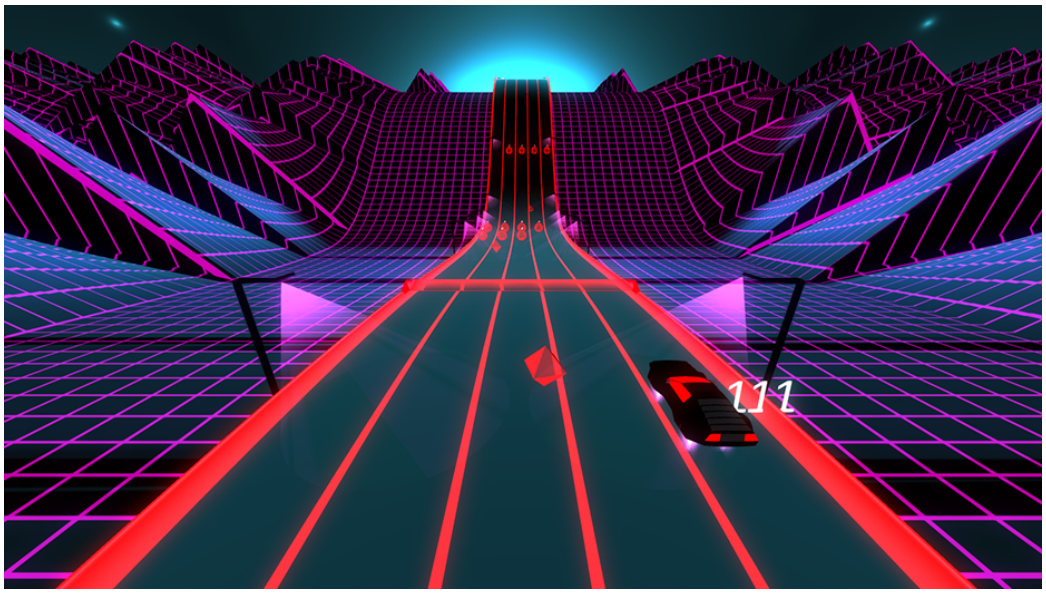
Sprožilci so zadolženi za dogodke v igri, ki se občasno sprožijo, zato so povezani z nastopi tonov. Igra uporablja tri sprožilce:

- MaterialEmissionColorTrigger: zamenja emisijsko barvo materiala.
- MaterialTintColorTrigger: zamenja odtenek barve, ki ga uporabljajo prosojni materiali.
- SkyboxColorTrigger: zamenja barvo, ki jo vidimo v nebu igre.

Modifikatorji stalno spreminjajo sestavne dele igre, da se ujemajo z glasbo, zato so povezani na energijo frekvenčnih skupin. Igra uporablja štiri modifikatorje:

- LightColorModifier: spremeni barvo svetlobnega vira.
- LightIntensityModifier: spremeni svetlost svetlobnega vira.
- SizeModifier: spremeni velikost objekta.
- SkyboxLightIntensityModifier: spremeni svetlost elementov na nebu igre.

4.3 Sestavni deli igre

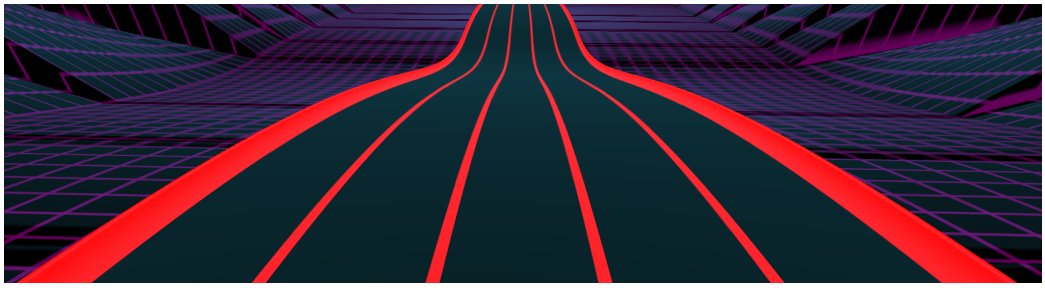


Slika 4.3: Zaslonska slika igre, glavni del igre je cesta, po katerih se premika igralec.

4.3.1 Kontrole

Igralec ima nadzor nad avtomobilom, ki ga lahko premika levo ali desno po glavni cesti, sestavljeni iz pasov, ki so razvidni s slike 4.4. Na voljo ima tudi skok, s čimer se lahko izogiba oviram. Igralec ima na izbiro igranje z miško ali tipkovnico. Če uporablja tipkovnico, se premika levo in desno s pritiskom tipk A in D, če uporablja miško, pa se levo in desno premika s premikom miške.

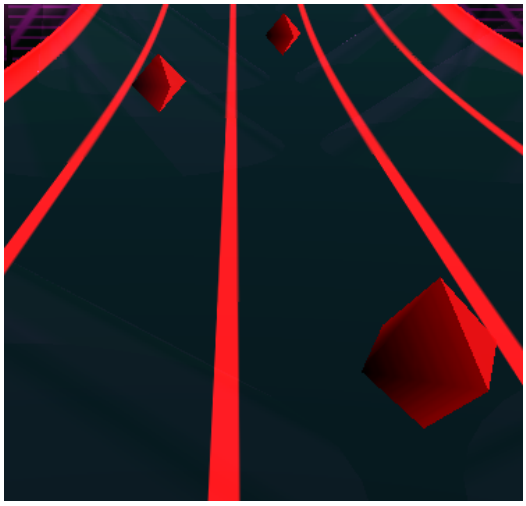
Igralec lahko skoči s pritiskom levega miškinega gumba ali s pritiskom tipke presledek. Ko igralec skoči, se avtomobil preko animacije premakne v zrak in pridobi tudi nekaj časa neranljivosti, s čimer preprečimo izgubo točk, v primeru, ko avtomobil ni dovolj visoko v animaciji, a je igralec pravočasno pritisnil tipko.



Slika 4.4: Pasovi, po katerih se premika igralec.

4.3.2 Zbiranje točk

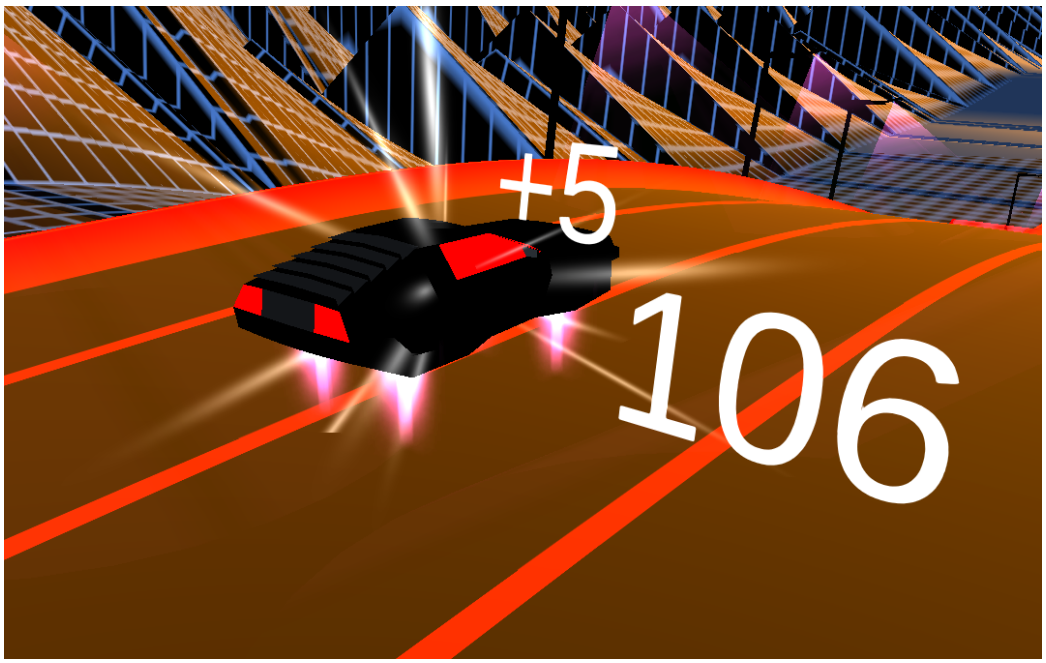
Cilj igre je pridobiti čim več točk. Igralec pridobiva točke s pobiranjem diamantov, ki so prikazani na sliki 4.5. Ko igralec pobere več diamantov zaporedoma, si poveča svoj večkratnik, kar pomeni, da za naslednji pobran diamant pridobi več točk. Ko igralec konča igro, se njegovo število točk primerja z njegovimi prejšnjimi rezultati. Če je število točk večje od prejšnjega, se to število shrani, s čimer lahko igralci tudi tekmujejo med seboj.



Slika 4.5: Diamanti, ki jih pobira igralec.

Koliko točk ima igralec se vidi s številom, ki se mu prikaže poleg njegovega avtomobila, kot je prikazano na sliki 4.6. Za povratno informacijo, da je igralec pobral diamant, zaigra zvočni učinek (angl. sound effect) in ustvari

učinek delcev (angl. particle effect).



Slika 4.6: Igralec je pobral diamant, s čimer se mu poveča število točk in sproži učinek delcev.

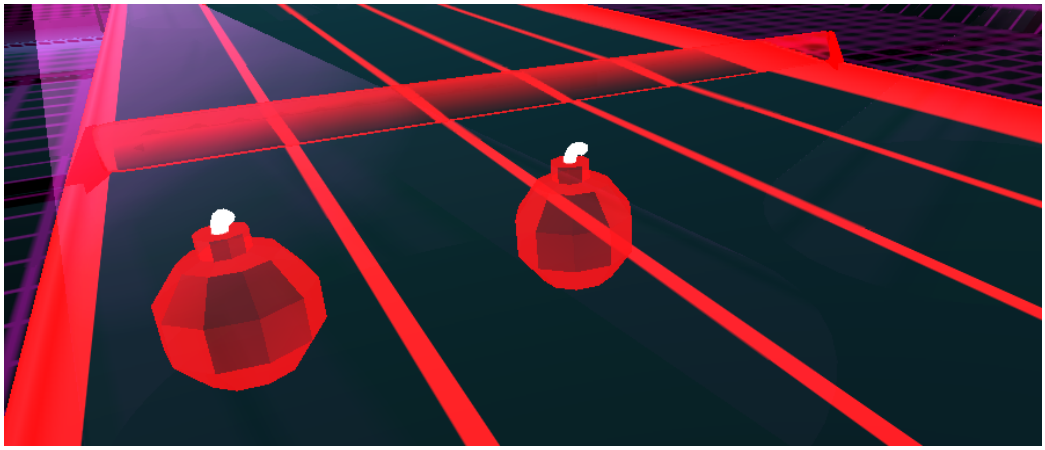
4.3.3 Izogibanje bombam in oviram

Med zbiranjem točk se mora igralec izogibati bombam in oviram, katere lahko vidite na sliki 4.7. Če igralec zadene bombo, izgubi nekaj točk, če pa zadene oviro izgubi svoj večkratnik. Bombam se lahko igralec izogne s premikom levo, desno ali s skokom, oviram pa se lahko izogne le s skokom. Podobno kot diamanti imata bomba in ovira svojo povratno informacijo s svojim zvočnim učinkom in učinkom delcev.

4.3.4 Težavnosti

Igralec lahko izbira med štirimi težavnostmi:

- Lahka: najlažja težavnost, mišljena za igralce, ki ne igrajo veliko iger.



Slika 4.7: Bombe in ovire, katerim se izogiba igralec.

- Normalna: težavnost mišljena za povprečnega igralca, ki prvič igra to igro.
- Težka: težavnost mišljena za igralca, ki že nekaj časa igra igro.
- Zelo težka: najtežja težavnost mišljena za izkušene igralce.

Težavnost vpliva na:

- Število bomb in kako pogosto se generirajo.
- Kolikšen odstotek sestavnih delov igre je diamant, bomba ali ovira.
- Koliko točk izgubijo igralci, ko zadenejo bombo.

Poleg omenjenih vplivov je še nekaj posebnih razlik, ki se pojavijo samo na določenih težavnostih. Pri zelo težki težavnosti se točke začnejo premikati, medtem ko so v drugih težavnostih vedno v enakem pasu. Prav tako pri težki in zelo težki težavnosti, ko igralec zadene bombo, izgubi svoj večkratnik.

4.3.5 Rang

Na koncu igre se igralcu, glede na to, kako se je izkazal, določi rang. Tako mu v primeru, da ni dosegel najvišjega ranga, igra ponudi dodaten cilj. Rangi

imajo navdih iz japonskih iger [2], kjer gredo ocene igralcev do S ranga, veliko iger ima nato tudi SSS rang, za katerega smo se odločili tudi mi:

- F ali 0: Igralec ga pridobi, če je pobral manj kot 60 % točk ali zadel več kot 10 bomb ali 10 ovir.
- D ali 1: Igralec ga pridobi, če je pobral več kot 60 % točk in zadel manj kot 8 bomb ali 8 ovir.
- C ali 2: Igralec ga pridobi, če je pobral več kot 65 % točk in zadel manj kot 6 bomb ali 6 ovir.
- B ali 3: Igralec ga pridobi, če je pobral več kot 75 % točk in zadel manj kot 3 bombe ali 3 ovire.
- A ali 4: Igralec ga pridobi, če je pobral več kot 80 % točk in zadel manj kot 2 bombi ali 2 oviri.
- S ali 5: Igralec ga pridobi, če je pobral več kot 90 % točk in zadel manj kot 1 bombo ali 1 oviro.
- SSS ali 6: Igralec ga pridobi, če je pobral več kot 95 % točk in zadel 0 bomb ali 0 ovir.

Vse ocene so odvisne tudi od dolžine pesmi, saj s tem igralci niso kaznovani, če igrajo daljše pesmi. To upoštevamo tako, da zgornje absolutne meje povečamo za vsake 3 minute pesmi.

4.3.6 Ostali učinki

Ostali implementirani učinki so:

- Z energijo utripajo svetloba in luči, ki so v igri.
- Z večjo energijo barve bolj zažarijo (angl. bloom).
- Ko igralec pobere diamant, se mu glede na energijo zatrese kamera.
- Glede na vrednost strukture pesmi se igralec premika hitreje ali počasneje.

Poglavje 5

Evalvacija

Končni rezultat diplome je igra, ki se prilagaja pesmi, ki jo podajo igralci. Pri evaluaciji smo se odločili za uporabo ankete, pri čemer nekatere podatke, ki jih lahko pridobimo brez ankete, zbiramo samodejno.

5.1 Anketa

Z igro smo podali 4 pesmi različnih zvrsti (elektronska, rock, klasična glasba in jazz), s čimer smo zajeli velik delež različnih scenarijev, nad katerimi bo moral delati algoritem.

Z anketo na sliki 5.1 smo želeli izvedeti:

- Kako dobro se je igra ujemala s podanimi pesmimi?
- Kako dobro se je igra ujemala z ostalimi pesmimi, ki so jih morda igrali anketiranci?
- Ali je igra ustrezno težavna?
- Kako bi ocenili igro?

Prav tako smo anketirancem podali možnost, da napišejo še svoje komentarje o igri.

Prejeli smo 12 odgovorov, pri katerih je bil razpon ocen med 1 (slabo) in 7 (dobro), in jih zbrali v tabelo 5.1:

Vprašanje	Odgovori												Povprečje
Pesem 1	7,	6,	5,	5,	6,	6,	6,	6,	6,	6,	4,	7	5,83
Pesem 2	7,	7,	5,	7,	5,	6,	5,	5,	6,	6,	6,	6	5,92
Pesem 3	7,	4,	5,	6,	5,	3,	5,	7,	5,	6,	5,	5,	5,25
Pesem 4	7,	5,	5,	4,	5,	1,	5,	7,	2,	6,	5,	4,	4,67
Ostale pesmi	7,	/,	5,	/,	/,	4,	4,	5,	7,	3,	4,	6,	5
Težavnost	4,	4,	4,	5,	5,	4,	4,	5,	4,	5,	4,	3,	4,25
Ocena igre	7,	6,	5,	5,	5,	5,	5,	7,	7,	5,	5,	6,	5,67

Tabela 5.1: Odgovori na anketo.

Pesem 1 in 2 predstavljata elektronsko in rock zvrst glasbe. Pri obeh zvrsteh se algoritem dobro obnese, saj so prisotni bobni in basi, pri katerih najlažje zaznamo nastope tonov. Obe pesmi sta imeli hitrejši tempo, ki se dobro ujema s stilom igre, zato so bile povprečne ocene dobre.

Pesem 3 je spadala v zvrst klasične glasbe, s počasnejšim tempom. Tukaj je povprečna ocena malo padla, vendar so imeli igralci še vedno občutek, da se igra večinoma ujema z glasbo.

Najslabše se je odrezala pesem 4. Pesem 4 je bila iz jazz glasbene zvrsti, kjer so igralci imeli občutek, da se vizualni stil igre ne ujema s pesmijo.

Pri ostalih pesmih, ki so jih igralci igrali je povprečna ocena 5, s tem vidimo, da so igralci imeli občutek, da se igra pretežno ujema s pesmijo, vendar predvidevamo, da je nastal podoben problem, kot so ga imeli s pesmijo 4.

Pri težavnosti se je večina strinjala, da je igra ustrezno težavna, kar ni presenetljivo, saj ima igra na voljo več težavnosti, kar jim ponuja možnost, da se igra prilagaja njihovim sposobnostim.

Igra je bila igralcem všeč, saj so ji dali visoko oceno.

Pri anketi smo zbirali tudi komentarje, pri čemer smo prišli do treh slabosti, ki bi jih lahko izboljšali:

- Igralci so slabo razlikovali med bombami in diamanti, saj so si bili vizualno preveč podobni. To se lahko reši s spremembo diamantov in bomb v bolj razločljive materiale.
- Eden od igralcev je pripomnil, da je bilo preveč vizualnih učinkov. Kar bi se rešilo z dodatnimi opcijami v nastavitvah igre.
- Četudi je algoritem za analizo pravilen, se igra tematsko ne ujema z glasbo, ki ima počasnejši tempo. Razširitev, ki bi rešila ta problem je opisana v poglavju 6.

5.2 Zbiranje statistike

Podatke, na katere lahko gledamo objektivno, smo zbirali samodejno. Ti so:

- Koliko odstotkov diamantov je igralec povprečno zbral v igri?
- Koliko bomb in ovir je igralec povprečno zadel v igri?
- Povprečen rang, ki ga je igralec prejel.
- Koliko iger je igralec končal?
- Koliko iger je igralec odigral?

Podatke smo zbirali preko REST strežnika, ki je preko HTTP zahtev sprejemal podatke in jih nato shranil v PostgreSQL bazi. Ko igra pošlje statistiko, prejme identifikacijo, katero si lokalno shrani, da naslednjič, ko igralec pošlje statistiko, strežnik ve, da je to isti igralec. Igro je igralo več igralcev, kot smo dobili odgovorov na anketo, zato smo zbrali statistiko na 43. različnih računalnikih, kjer je bilo vsaj 43 različnih igralcev. Ko smo podatke pridobili iz strežnika, smo jih zbrali v tabeli 5.2.

Težavnost	Lahka	Normalna	Težka	Zelo težka
Diamanti	0,93%	0,87%	0,92%	0,88%
Bombe	0,95	1,65	0,77	0,74
Ovire	0,84	1,00	0,42	0,47
Rang	2,58	2,19	2,83	3,20
Odigrane igre	97 (19)	96 (41)	34 (8)	51 (11)

Vrednosti so povprečja N igralcev, ki se nahajajo v oklepaju ob vrednosti odigranih in končanih iger. Če igralec ni odigral igre pri neki težavnosti, se ga pri tisti težavnosti ne upošteva.

Tabela 5.2: Statistika igralcev.

V povprečju so igralci zbrali večino diamantov, kar pomeni, da je igra težka zaradi ovir in bomb.

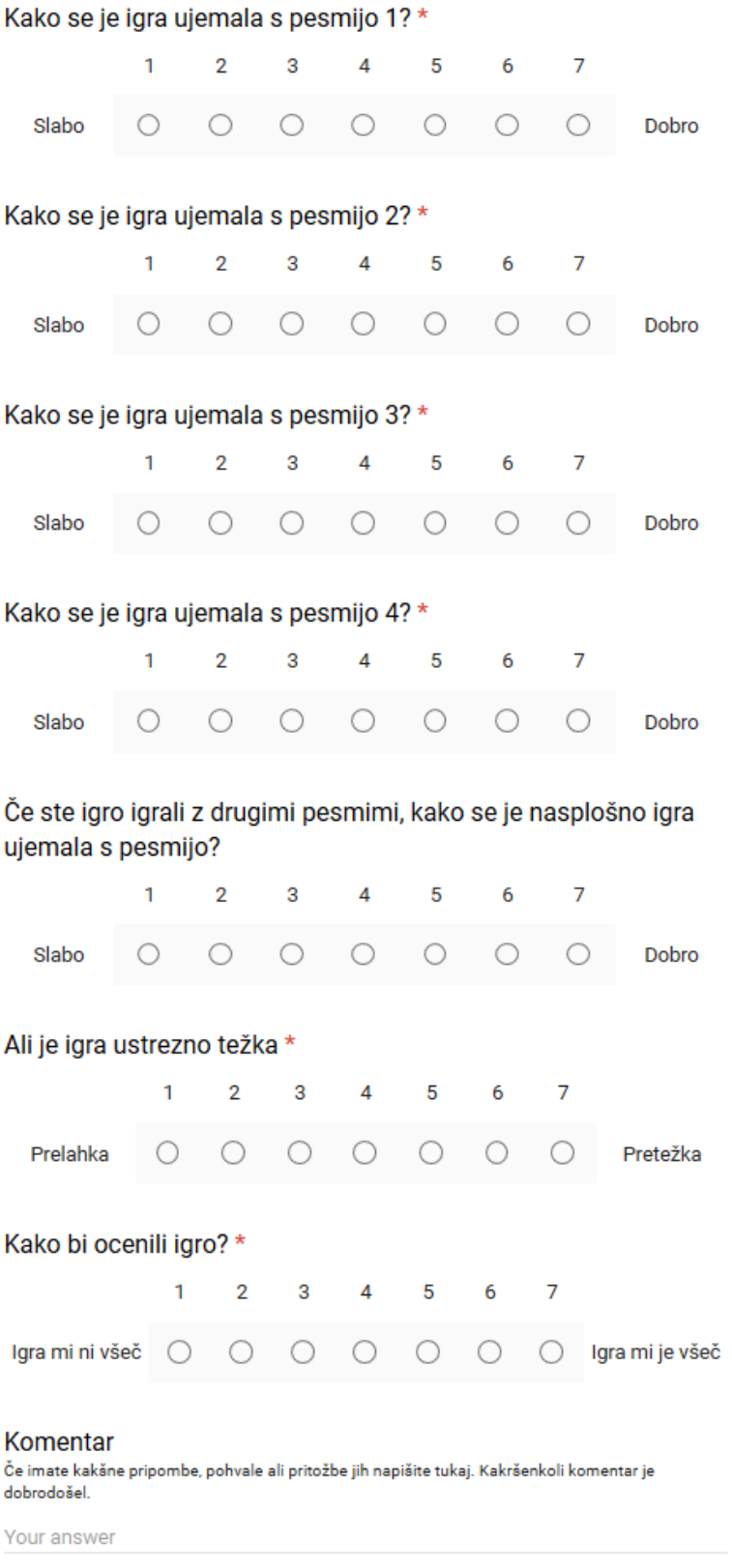
Število zadetih bomb in ovir se računa na vsakih 30 sekund, kar pomeni, da so v povprečju igralci zadeli 1,03 bomb in 0,68 ovir vsakih 30 sekund.

Povprečno največ bomb in ovir je zadelo uporabnikov pri normalni težavnosti, ki pa je tudi bila najbolj popularna težavnost. Pričakujemo, da je to težavnost, s katero je začela večina igralcev. Posledično so bili rezultati slabši, saj je bila to njihova prva igra. Enako se opazi tudi pri rangu, ki je pri normalni težavnosti najnižji.

Kljub temu da imata težka in zelo težka težavnost največ ovir in bomb, je bilo tam najmanj zadetih bomb in ovir ter najboljši povprečni rang, iz česar sklepamo, da so se zanje odločili le boljši igralci.

Povprečni rang, ki so ga igralci dosegli je 2,7. To je okoli C ranga, kar je zadovoljivo povprečje.

Povprečni igralec je odigral 8 pesmi. Od teh osmih pesmi jih je do konca odigralo 80 %, s čimer smo kar zadovoljni.



Slika 5.1: Anketa.

Poglavje 6

Zaključek

V diplomski nalogi smo si kot cilj zadali razviti razširitev za Unity, ki bi nam omogočila razvoj iger, ki slonijo na pesmih. Ta cilj smo dosegli, saj smo z našo rešitvijo uspešno razvili igro, pri kateri so igralci mnenja, da se igra dobro ujema z njihovimi pesmimi. Pri diplomi smo razvili precej enostavno igro, ki se je osredotočila predvsem na vizualizacijo, vendar smo dokazali, da je koncept uresničljiv.

S pomočjo povratne informacije uporabnikov smo prišli do nekaj nadaljnjih izboljšav:

- Pri spektralnem pretoku konstante narediti dinamične. Možna rešitev je prilagoditev konstant glede na strukturo pesmi. S tem bi odstranili trenutno slabost algoritma, ki se zatakne pri veliko zaporednih visokih amplitudah, saj so konstante prevelike, zaradi česar izgubimo nekatere nastope tonov.

- Za ločevanje glasbene zvrsti pesmi, bi pri pesmih ocenili njihov tempo. Do tega pridemo enostavno, saj lahko preko zaznavanja nastopov tonov pridobimo informacijo o BPM. Nato bi za igro ustvarili več tem, ki bi se izbrale glede na zaznan tempo.

Za konec bi še radi omenili, da bo ta razširitev uporabljena za razvoj nove, kompleksnejše igre, ki bo imela več časa za razvoj. Igra bo najbrž objavljena

na STEAM (platforma za prodajo računalniških iger), tako da če v 2019 opazite igro bazirano na glasbi, je možno, da je prišla ven kot rezultat te diplome.

Literatura

[1] 3ds max features overview. Dosegljivo: `https://www.autodesk.eu/products/3ds-max/features`. [Dostopano: 21. 8. 2018].

[2] Japonski rangi. Dosegljivo: `https://www.giantbomb.com/s-rank/3015-2962/`. [Dostopano: 21. 8. 2018].

[3] Fast console MPEG audio player and decoder library. Dosegljivo: `https://www.mpg123.de/`. [Dostopano: 29. 8. 2018].

[4] Understanding FFT and windowing. Dosegljivo: `http://www.ni.com/white-paper/4844/en/`, 2016. [Dostopano: 21. 8. 2018].

[5] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[6] J. P. Bello and M. Sandler. Phase-based note onset detection for music signals. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, volume 5, pages V–441, 2003.

[7] S. Dixon. Onset detection revisited. In *2006 International Conference on Digital Audio Effects (ICDAFx '06)*, pages 133–137, 2006.

[8] M. Labschütz, K. Krösl, M. Aquino, F. Grashäftl, and S. Kohl. Content Creation for a 3D Game with Maya and Unity3D. In *2011 Central European Seminar on Computer Graphics (CESCG '11)*, pages 75–82, 2011.

[9] S. Tilkov and S. Vinoski. Node.js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.