Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gašper Žgajnar

Uporaba senzorja Kinect za učenje konceptov glasbe

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00126/2013
Datum: 11.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **GAŠPER ŽGAJNAR**

Naslov: **UPORABA SENZORJA KINECT ZA UČENJE KONCEPTOV GLASBE**
USING KINECT FOR LEARNING MUSIC CONCEPTS

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

V okviru diplomskega dela razvijte igro, ki omogoča učenje glasbenih konceptov s tem, da igralec s pomočjo gibov rok in nog lovi dogodke v skladbi. Položaj igralca detektirajte s pomočjo senzorja Kinect. Za realizacijo težavnostnih stopenj v igri razvijte postopek, ki smiselno zmanjšuje število not v skladbi, parametre igre pa ovrednotite na skupini uporabnikov.

Mentor:

doc. dr. Matija Marolt

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Gašper Žgajnar, z vpisno številko **63090178**, sem avtor diplomskega dela z naslovom:

Uporaba senzorja Kinect za učenje konceptov glasbe

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matija Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 24. septembra 2013 Podpis avtorja:

Kazalo

Kratice

MIDI Musical Instrument Digital Interface

RGB Red Green Blue

USB Universal Serial Bus

SDK Software Development Kit

IDE Integrated Development Environment

MIT Massachusetts Institute of Technology

MMA MIDI Manufacturers Association

MPQN Microseconds Per Quarter Note

BPM Beats Per Minute

WAVE Waveform Audio File Format

Povzetek

V diplomskem delu je predstavljena igra razvita v okolju XNA Game Studio, ki omogoča igranje MIDI skladbe s pomočjo senzorja Microsoft Kinect. Gre za igro v kateri igralec s pomočjo gibov rok in nog, ki jih zaznava Kinect, poskuša uloviti note, ki se premikajo proti enemu izmed ciljev na zaslonu. Note, pridobljene iz datoteke MIDI, so predstavljene kot 3D objekti v obliki kocke. Cilj proti kateremu leti nota, je določen na podlagi števila ciljnih škatel in višine note.

Pesem je analizirana in obdelana s pomočjo zunanje knjižnice za obdelavo MIDI datotek. Število not, ki se pojavijo v igri, je odvisno od težavnosti, ki jo izbere uporabnik. Za zmanjševanje števila not je bil razvit algoritem, ki deluje na principu uteži. Te se izračunajo iz lastnosti posameznih not.

Narejeno je bilo tudi testiranje igre. Manjši vzorec uporabnikov je ocenil kolikšno število ciljnih škatel je primernejše za igro.

Ključne besede: Kinect, datoteka MIDI, igra, XNA, nota, C#.

Abstract

This thesis presents game developed using XNA Game Studio. The game enables you to play MIDI songs using Microsoft Kinect. The goal of the game is to successfully catch as many notes as possible using hands and feet detected by Kinect. The game gets notes from MIDI file and presents them as 3D objects in shape of cube. Notes are moving toward one of the destinations on the screen. Which destination box is selected for the specific note, is based on the number of destination boxes and the height of a note.

The song is analyzed and parsed using third parry open source library for processing MIDI files. The number of notes, which appear in a game, depends on a chosen difficulty. To reduce that number the algorithm was developed. It works using weights, which are calculated on a basis of each note's properties.

There was also made a simple game testing. A smaller sample of users evaluated, which number of destination boxes is more suitable for the game.

Key words: Kinect, MIDI file, game, XNA, note, C#.

Poglavje 1

Uvod

Računalniške igre sodijo med popularnejše načine krajšanja časa. Poleg svoje zabavne vrednosti pa lahko delujejo tudi kot didaktičen pripomoček. Z razvojem novih tehnologij nastajajo novi načini uporabe in interakcije z računalniki. S tem pa se odpirajo tudi nove možnosti za razvijalce, da uporabnikom ponudijo bogatejše uporabniške izkušnje pri rabi različnih aplikacij. Med te tehnologije sodi tudi senzor Kinect. Od običajnih kamer se loči po zmožnosti zajema globinske slike, ki z ustrezno programsko opremo, omogoča zaznavo in prepoznavno okolice v treh dimenzijah.

V diplomskem delu bom opisal razvoj računalniške igre v okolju XNA, ki za interakcijo uporablja senzor Kinect. Predstavil bom njeno delovanje, funkcionalnosti in uporabljene tehnologije. Na začetku sta predstavljena po en komercialni in en odprtokodni primer igre, ki sta osnovani na podobnem konceptu. Izmed uporabljenih tehnologij sta najpodrobneje predstavljena senzor Kinect, ki je potrebna strojna oprema za igranje igre, in lastnosti MIDI datotek s pomočjo katerih je realiziran glasbeni del aplikacije. Poleg opisa in delovanja igre, sta podrobneje predstavljena algoritem za filtriranje števila not in algoritem za določanje cilja posamezni noti. Izdelano je bilo tudi preprosto testiranje igre. Nekaj potencialnih uporabnikov je podalo svoje mnenje katero število ciljnih škatel je najustreznejše za igranje.

Osnovni koncept igre je igranje glasbenih not iz izbrane MIDI skladbe

s pomočjo senzorja Kinect. Podanih je nekaj pesmi in iz izbrane se izdela zaporedje not, ki jih je potrebno zaigrati. Posamezna nota je predstavljena v obliki preproste 3D škatle, ki se premika proti uporabniku. Iz centra zaslona leti proti enemu izmed ciljev, ki se nahajajo na levi in desni strani zaslona. Tudi posamezni cilji so predstavljeni kot 3D objekt. Na katerega izmed možnih ciljev bo letela posamezna nota, predvsem zavisi od njenih lastnosti (višina in vrsta note). Točno destinacijo določa namenski algoritem. Uporabnik uspešno zaigra (ujame) noto, če z roko ali nogo prekrije ciljno škatlo v trenutku, ko se v njej nahaja nota.

Analiza in predvajanje podane skladbe je narejeno s pomočjo odprtokodne knjižnice za obdelavo MIDI datotek – C# MIDI Toolkit. Iz izbrane MIDI skladbe se pridobijo vsi parametri pesmi (tempo, dolžina takta...) in posamezne note, ki nastopajo v njej. Število not, ki se nato pojavijo v igri je odvisno od izbrane težavnosti. Za filtriranje, zmanjševanje števila not je bil razvit algoritem, ki na podlagi njihovih lastnosti določi ustrezne uteži.

Nove tehnologije ponujajo bogatejšo, zabavnejšo uporabniško izkušnjo. Zato tudi motivacija za izdelavo igre, opisane v tem diplomskem delu, izhaja iz želje po uporabi senzorja Kinect. Ta omogoča drugačno interakcijo uporabnika z igro kot smo jo navajeni pri klasični strojni opremi za upravljanje z računalniki. Tako se lahko z uporabo senzorja Kinect izdela preprosta a kljub temu zabavna igra. Želja pa je bila tudi izdelati igro, ki bi poleg zabavne vrednosti vsebovala nek izobraževalni element. Ker je koncept igre glasbeno usmerjen, so zato na škatlah napisane note, ki jo posamezen 3D objekt predstavlja. S tem je tudi igralcu omogočen nekoliko drugačen vpogled v note, ki nastopajo v skladbi, ki jo igra.

Poglavje 2

Pregled dveh, konceptualno podobnih iger

Trg z računalniškimi igrami je zelo obsežen, zato tudi že obstaja kar nekaj komercialnih kot tudi odprtokodnih primerov iger, ki so po konceptu in načinu igranja podobni igri opisani v tej diplomi. Vsem tem igram je značilno, da ponujajo igranje glasbenih posnetkov na različne načine. V nadaljevanju sta predstavljena dva popularnejša primera; en komercialni in en odprtokodni.

2.1 Guitar Hero

Serija Guitar Hero [1] sodi med eno izmed najdonosnejših v igralni industriji. Prva izvedenka je bila objavljena leta 2005 nato pa je sledilo še pet glavnih izdaj pod istim naslovom. Ciljne naprave so na začetku bile igralne konzole, kasneje pa so dodali tudi osebne računalnike in mobilne naprave.

Glavna značilnost igre je, da je za igranje potreben poseben krmilnik, v obliki kitare. Med igranjem je na zaslonu prikazan podaljšek vratu kitare po katerem se proti igralcu premikajo note. Te so predstavljene kot barvni okrogli markerji, ki potujejo proti spodnjemu delu zaslona sinhronizirano z glasbo v ozadju. Barve not na zaslonu se ujemajo s tipkami na kitari. Ko nota doseže dno zaslona, mora igralec pritisniti pravilno tipko ali kombinacijo tipk

Slika 2.1: Igralni meni pri Guitar Hero [1].

na kitari, ki jih nato zaigra s pritiskom tipke, ki predstavlja strune kitare. Glede na uspešno ali neuspešno zaigrane note se spreminja priljubljenost igralca pred navideznimi poslušalci, ki lahko igranje tudi prekinejo, če z njim niso zadovoljni.

Težavnost igre je določena s številom tipk na katere se igra pesem in številom ter hitrostjo not, ki jih je potrebno zaigrati.

Z novejšo izdajo igre so bili dodani novi instrumenti. Poleg običajne kitare se lahko igra tudi bobne ali bas kitaro, z mikrofonom pa se lahko vokalno spremlja pesem.

Pesmi, ki jih uporabnik lahko zaigra, izhajajo predvsem iz glasbene zvrsti rock.

2.2 Frets on fire

Frets on fire [2] je za razliko od Guitar Hero zastonjska odprtokodna igra za igranje pesmi na računalniku. Igralec za igranje ne potrebuje posebnega krmilnika (kitare) ampak lahko igra s pomočjo računalniške tipkovnice. Igra

Slika 2.2: Igralni meni pri Frets on Fire [2].

podpira vse tri osrednje računalniške platforme – Windows, Linux in Mac OS X.

Koncept igre je podoben kot pri Guitar Hero. Ravno tako se proti dnu zaslona premikajo note, ki jih je potrebno zaigrati s pomočjo tipkovnice.

Glavna prednost pred Guitar Hero je možnost dodajanja lastnih pesmi v igro. Igra vsebuje tudi vgrajen urejevalnik pesmi, kjer lahko uporabnik prilagaja in ustvarja nove skladbe.

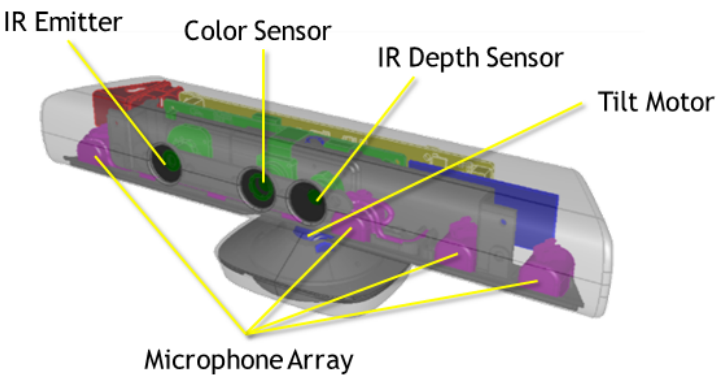
Poglavje 3

Uporabljene tehnologije

3.1 Microsoft Kinect

Premikanje po menujih in upravljanje z igro je narejeno z uporabo senzorja Kinect [4]. Kinect je senzor za zaznavo gibanja s pomočjo globinske slike. Podjetje Microsoft ga je razvilo leta 2010 kot dodatek za igralno konzolo Xbox 360. Dve leti pozneje so na trg poslali prilagojeno inačico senzorja za osebne računalnike. Poglavitna razlika med njima je v velikosti zaznanega polja. Kinect za Windows podpira t.i. bližinski način (ang. near mode), kar omogoča zaznavo objektov z globinsko kamero na razdalji od 40 centimetrov naprej. Kinect za Xbox 360 pa omogoča zaznavanje na razdaljah od 80cm do 4m. Vidno polje, ki ga pokrivata kameri znaša 57.5 stopinj po horizontalni in 43.5 stopinj po vertikalni osi. Po tej se Kinect lahko dodatno premakne še za 27 stopinj navzgor ali navzdol preko vgrajenih električnih motorjev.

Glavna prednost senzorja Kinect pred navadnimi kamerami je zmožnost zaznave globinske slike. Namesto matrike slikovnih točk v različnih barvnih spektrih Kinect vrne množico točk, ki imajo poleg x in y tudi z koordinato – koordinato globine. Z informacijo o oddaljenosti posameznih objektov v okolici je možno načrtovati aplikacije, ki za svoje delovanje potrebujejo 3D sliko okolice ali objektov v njej. Tako je z ustreznimi algoritmi za obdelavo podatkov pridobljenih iz senzorja možno zaznati in slediti objektom in



Slika 3.1: Zgradba senzorja Kinect [3].

njihovemu gibanju, ki ga izvajajo pred Kinectom.

Delovanje senzorja

Senzor Kinect je sestavljen iz dveh kamer – infrardeča in navadna RGB kamere, vira (projektorja) infrardeče svetlobe in štirih mikrofonov. RGB kamera podpira ločljivost do 1280x960 slikovnih pik, infrardeča pa 640x480 pikslov. Poleg senzorjev pa je pomemben gradnik tudi vezje za procesiranje vseh pridobljenih signalov.

Kinect omogoča zaznavanje 3D slike. Celotna globinska slika se izdela znotraj senzorja in nato pošlje preko USB kabla na računalnik; podobno kot pri prenosu slike iz navadne kamere, le da tu namesto 2D slikovnih pik po vodniku potujejo vrednosti s podatki o globini.

Obstaja več različnih tehnik za pridobivanje globinske slike. Običajno se 3D informacijo sestavi iz dveh slik posnetih iz različnih kotov in nato s pomočjo triangulacije izračuna oddaljenost posameznih objektov. Kinect pa deluje nekoliko drugače. Namesto dveh, medsebojno malo zamaknjenih kamer, Kinect pridobi globinsko sliko s pomočjo infrardeče kamere in projektorja infrardeče svetlobe. Ta projicira infrardeče žarke na objekte v okolici v naključnem, strojno generiranem zaporedju. Žarki se na okolici pojavijo kot rdeče, očem nevidne pike, zazna pa jih infrardeča kamera. Na podlagi

analize odboja žarka se določi oddaljenost posamezne točke.

Kinect kot naprava zgolj zaznava okolico in oddaljenost objektov, vse nadaljnje obdelave podatkov pa se vršijo na računalniku z ustrezno programsko opremo.

Na Kinectu so nameščeni tudi štirje mikrofoni. Ti poleg osnovne funkcije snemanja zvoka omogočajo tudi določanje oddaljenosti in smeri zvočnega vira. To je možno ravno zaradi štirih, različno razporejenih mikrofonov do katerih, zaradi medsebojne oddaljenosti, zvok potuje različno dolgo.

3.2 XNA Game Studio 4.0

XNA Game Studio [5] je programsko okolje za izdelavo iger v razvojnem orodju Visual Studio. XNA Game Studio vključuje ogrodje XNA, ki je zbirka orodji in knjižnic za razvoj iger, ki bazirajo na .NET ogrodju.

Razvito je bilo s strani Microsofta in omogoča izdelavo preprostih iger za platforme Windows, Windows Phone in Xbox 360. Programska jezika, ki ju XNA uradno podpira, sta C# in Visual Basic .NET.

Ogrodje ponuja preprost igralni pogon in s tem omogoča hiter razvoj enostavnejših iger. Osnovni razred projekta narejenega z XNA Game Studiem deduje od razreda Microsoft.Xna.Framework.Game (primer 3.1). Ta vsebuje osnovne komponente potrebne za izdelavo igre. Poleg metode za inicializacijo in nalaganje vsebine je tudi že implementirana osnovna zanka značilna za vsako igro, ki se izvaja ciklično v povprečju 60-krat na sekundo. V njej se osvežuje in izrisuje vsebina na zaslon. Poleg vseh metod pa razred zagotavlja tudi okno, ki prikazuje igro z vsemi njenimi elementi.

Ena izmed prednosti ogrodja je cevovod za nalaganje vsebine (ang. content pipeline). Gre za nabor procesov, ki omogočajo enostavno nalaganje različnih gradnikov igre (kot so 3D objekti, teksture, grafični slogi...), iz datoteke v obliko primerno za uporabo v igri.

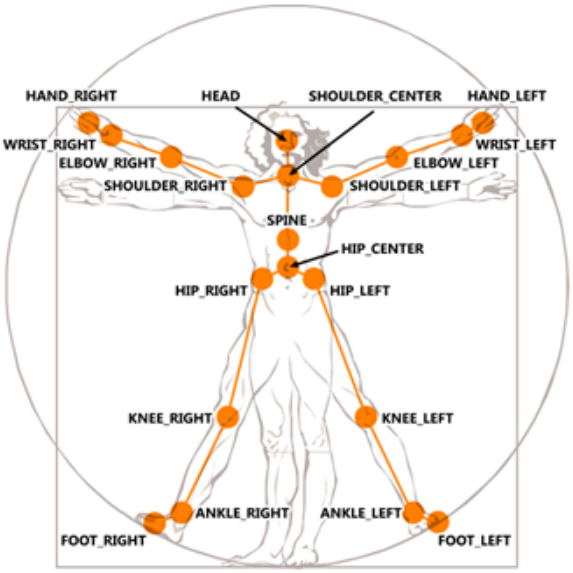
Trenutna in ob enem zadnja verzija XNA Game Studia je 4.0. Od letošnjega leta, 2013, ni več v razvoju.

Listing 3.1: Primer osnovnega razreda igre narejene v okolju XNA

```
public class Game1 : Microsoft.Xna.Framework.Game{
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        public Game1(){
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }
        protected override void Initialize(){
            base.Initialize();
        }
        protected override void LoadContent(){
            spriteBatch = new SpriteBatch(GraphicsDevice);
        }
        protected override void Update(GameTime gameTime){
            base.Update(gameTime);
        }
        protected override void Draw(GameTime gameTime){
            GraphicsDevice.Clear(Color.CornflowerBlue);
            base.Draw(gameTime);
        }
    }
```

3.3 Kinect SDK 1.7

Microsoft zagotavlja tudi programsko opremo za razvoj aplikacij, ki uporabljajo senzor Kinect. Skupek programske kode je zapakiran v programsko razvojno opremo Kinect SDK [6] z najnovejšo verzijo 1.7. Gre za zbirko gonilnikov, orodji, programskih vmesnikov in različnih primerov kode, ki omogočajo lažji in kvalitetnejši razvoj aplikacij. Paket omogoča povezavo senzorja z računalnikom in pridobivanje vseh signalov (globinska slika, RGB slika, zvok...) v ustrezni obliki. Kinect SDK vsebuje tudi algoritme za



Slika 3.2: Skelet z okončinami, ki jih zazna Kinect [4].

obdelavo podatkov. Tako so, namesto množice 3D točk, razvijalcu ponujeni že obdelani podatki, ki vsebujejo le Implementirano je prepoznavanje in sledenje skeletu uporabnika. Kinect lahko istočasno sledi šestim osebam in prepozna dva različna skeleta s po dvajsetimi, v stoječem, ali s po desetimi sklepi v sedečem položaju. Kinect SDK vsebuje tudi algoritme za zaznavo obraza, glasovne ukaze (ne za vse jezike) in različne gibe rok. Ti omogočajo interakcijo uporabnika z aplikacijo. Mednje sodita predvsem detekcija stiska roke v pest, ki lahko nadomešča daljši klik miške (ang. hold down) in izteg roke, ki lahko predstavlja klik miške.

Glavna informacija, potrebna za delovanje igre, je položaj igralca in njegovih rok ter nog. S pomočjo razvojne knjižnice in algoritma za detekcijo skeleta je znan podatek o položaju posameznih okončin; prikazane so na sliki 3.2. Poleg prepoznave delov telesa, pa se iz knjižnice uporablja tudi algoritem za razpoznavo stiska roke v pest.

3.4 Microsoft Visual Studio 2010

Za programiranje in uporabo XNA Game Studia 4.0 in Kinect SDK 1.7 je potrebno integrirano razvojno okolje (IDE) Microsoft Visual Studio. Deluje na računalnikih z operacijskim sistemom Windows in se uporablja za razvoj različnih vrst aplikacij za različne platforme razvite pri Microsoftu.

Visual Studio vsebuje urejevalnik programske kode, vgrajen razhroščevalnik, samo dokončanje ukazov, preverjanje sintaktične pravilnosti kode, ponuja pa tudi različna orodja za gradnjo grafičnih vmesnikov. Okolje podpira več programskih jezikov (C/C++, Visual Basic .NET, C#, Python...).

3.5 C#

C# je en izmed podprtih programskih jezikov za razvoj aplikacij v Visual Studiu in uporabo XNA orodji ter Kinect SDK-ja. Je objektno orientiran programski jezik, ki je bil razvit pri Microsoftu v okviru razvoja .NET ogrodja.

3.6 C# MIDI Toolkit

C# MIDI Toolkit je odprtokodni projekt dostopen pod licenco MIT [7] avtorja Leslieja Sanforda. Knjižnica je ena redkih, ki omogoča delo z MIDI datotekami v programskem jeziku C#.

Knjižnica omogoča obdelavo MIDI datotek. Podano skladbo pretvori iz formata MIDI v ustrezne programske objekte, ki so primernejši za programiranje. Tako je lažje manipuliranje s skladbo, posameznimi sledmi v njej in z vsemi MIDI dogodki, ki jo sestavljajo.

Poleg obdelave MIDI datotek pa knjižnica tudi omogoča sprejemanje (snemanje) in pošiljanje (predvajanje) MIDI sporočil. Knjižnica za predvajanje ponuja tri različne načine. Na izhodno napravo se lahko pošilja posamezne MIDI dogodke. Ta način je uporaben, če aplikacija predvaja posamezne tone, ki časovno niso povezani med seboj. Drugi način omogoča sestavo toka MIDI

dogodkov, ki se nato pošlje na izhodno napravo in predvaja zaporedno. Tretji način pa ponuja predvajanje celotne MIDI skladbe podane v sekvenci, ki jo sestavljajo sledi in druge lastnosti pesmi.

3.7 MIDI

MIDI [8] (Musical Instrument Digital Interface) je standard, ki opisuje protokol, digitalni vmesnik in priključke za povezavo in komunikacijo glasbenih instrumentov in računalnikov. MIDI tehnologija je bila standardizirana leta 1983. Z njo upravlja MMA – MIDI Manufactures Association. Cilj standarda je bil razvoj vmesnika za povezavo glasbenih instrumentov različnih proizvajalcev, ki bi omogočal standardne funkcije za medsebojno komunikacijo in delovanje naprav povezanih z MIDI kablom. Tako lahko en instrument nadzira delovanje drugega in komunicira z računalniki, na katerih je naložena različna programska oprema.

Podatki, ki se prenašajo med napravami so definirani v sklopu formata MIDI datotek.

3.7.1 MIDI datoteke

Namen MIDI datotek [9] je omogočiti izmenjavo časovno opredeljenih (ang. time-stamped) MIDI dogodkov med različnimi programi na istem ali različnih računalnikih.

Standardna MIDI datoteka je datotečna oblika za shranjevanje pesmi in pripadajočih podatkov v obliki primerni za uporabo v glasbeni programski in strojni opremi. Vsebuje informacije o sledeh, ki sestavljajo skladbo. V sledeh so shranjeni podatki o uporabljenih instrumentih, sekvenci glasbenih not in njihovih lastnostih potrebnih za predvajanje skladbe. Standard omogoča izdelavo glasbenega posnetka in uporabo tega na drugih računalnik z različno programsko opremo.

MIDI datoteka ni glasbeni posnetek temveč, podobno kot notno črtovje, vsebuje navodila in pravila kako se določena pesem zaigra. Tako MIDI dato-

teka zavzame precej manj prostora na disku kot bi dejanski posnetek skladbe.

Specifikacija MIDI datoteke

Vsi podatki v MIDI datoteki so shranjeni po pravilu debelega konca (ang. big endian, najpomembnejši bit na prvem mestu). Pred večino sklopov podatkov je navedena njihova dolžina, kar omogoča večjo prilagodljivost.

MIDI datoteke so organizirane v podatkovne kose (ang. chunks). Vsak kos sestoji iz 4-znakovnega imena (tipa) in štiri bajtnega parametra, ki predstavlja dolžino kosa. Taka oblika omogoča dodajanje novih (vrst) kosov. MIDI datoteka se vedno prične z glavo kosa (ang. header chunk), ki ji sledi ena ali več sledi kosa (track chunk).

Glava MIDI datoteke

V glavi kosa (ang. header chunk) so navedene osnovne informacije o MIDI datoteki. Sestavlja jo pet elementov predstavljenih v tabeli 3.1.

Vrednosti za ID kosa (chunk ID) in za velikost glave sta vedno enaki. Tretji parameter lahko zavzame vrednosti 0, 1 ali 2, saj obstajajo tri različne vrste MIDI datotek. Razlikujejo se v številu in lastnostih sledi, ki jih datoteka vsebuje. MIDI datoteka tipa 0 ima le eno sled, tip 1 in 2 pa lahko vsebujeta do 65536 sledi, točno število pa je določeno s četrtim parametrom glave. Zadnji parameter, časovni razdelek (ang. time division), določa kako se pretvori delta časovne enote (ang. delta times) v realne časovne enote. Vrednost je predstavljena s številom tikov na bit ali s številom sličic na sekund. Kateri način je uporabljen, določa zgornji bit besede.

Sled MIDI datoteke (ang. track chunk)

Sledi MIDI datoteke vsebujejo podatke o skladbi. Vsaka sled je sestavljena iz ID kosa (chunk ID), velikosti kosa in MIDI dogodkov (3.2). ID kosa je vedno 'MTrk', velikost pa je odvisna od števila MIDI dogodkov v sledi.

dolžina (v bajtih)	tip	opis	vrednost
4	char[4]	ID kosa (ang. chunk ID)	MThd
4	dword	velikost	6
2	word	vrsta MIDI datoteke	0, 1 ali 2
2	word	število sledi	1-65636
2	word	časovni razdelek (ang. time division)	

Tabela 3.1: Struktura glave MIDI datoteke [9].

dolžina (v bajtih)	tip	opis	vrednost
4	char[4]	ID kosa (ang. chunk ID)	MTrk
4	dword	velikost	
n	dogodki sledi (ang. track event data)		

Tabela 3.2: Struktura sledi MIDI datoteke [9].

MIDI dogodki (ang. MIDI events)

MIDI dogodki opisujejo glasbeno vsebino MIDI datoteke. Definirani so trije različni dogodki – kanalni dogodek (ang. channel event), meta dogodek (ang. meta event) in sistemsko ekskluzivni dogodek (system exclusive event). Vsak dogodek, poleg podatkov, vsebuje tudi delta čas (ang. delta time). Delta čas je število, ki predstavlja relativni časovni interval med dvema sosednima dogodkoma. Če je delta čas 0, potem se dogodka izvedeta (zaigrata) istočasno. Za izračun absolutnega časa je potrebno upoštevati časovni razdelek (ang. time division), tempo in časovni podpis (ang. time signature) posamezne sledi.

MIDI kanalni dogodki (ang. channel events)

MIDI skladba je sestavljena iz različnih not z različnimi glasbenimi lastnostmi, ki so lahko zaigrane na različne instrumente. Vse te lastnosti so definirane

delta time	tip dogodka	`MIDI kanal`	parameter 1	parameter 2
`n`	4 biti	4 biti	1 bajt	1 bajt

Tabela 3.3: Struktura kanalnega dogodka [9].

vrsta dogodka	MIDI kanal	`številka note`	hitrost
`Note On/ Note Off`	0-15	0-127	0-127

Tabela 3.4: Struktura dogodkov note on/note off [9].

z MIDI kanalnimi dogodki (ang. channel events). Struktura le-teh je predstavljena v tabeli 3.3. MIDI kanalni dogodki so najpogostejša vrsta v MIDI datotekah.

Definiranih je sedem različnih MIDI kanalnih dogodkov:

- Note On
- Note Off
- Note aftertouch
- Upravljalnik, ki javi spremembo v stanju MIDI kanala (ang. controller)
- Program change
- Channel Aftertouch
- Pitch bend

Dogodka najpomembnejša za diplomsko nalogo, Note on in Note off, sta predstavljena v tabeli 3.4.

Oba dogodka sestavlja številka MIDI kanala po katerem sta poslana in dva parametra. Številka note predstavlja vrsto note in oktavo v kateri se

ton številka oktave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

Tabela 3.5: Oktave in toni, ki jih podpira MIDI datoteka.

zaigra. MIDI datoteka podpira enajst oktav 3.5. Parameter hitrost pa določa s kolikšno intenziteto (glasnostjo) je nota zaigrana.

Note on in Note off običajno nastopata v paru. Prvi določa začetek igranja note, drugi pa kdaj ta nota preneha z igranjem (kdaj je MIDI tipka sproščena (ang. released)).

Meta dogodki (ang. meta events)

Meta dogodki predstavljajo vse preostale informacije o skladbi (npr. besedilo pesmi), ki se ne pošiljajo ali sprejemajo preko MIDI vhoda.
Definiranih je 15 različnih meta dogodkov:

- številka zaporedja (ang. sequence number)
- besedilni dogodek (ang. text event)

- opozorilo o avtorskih pravicah (ang. copyright notice)
- ime zaporedja/sledi (ang. sequence/track name)
- ime glasbila (ang. instrument name)
- besedilo pesmi (ang. lyrics)
- označevalec pomembnih točk v zaporedju (ang. marker)
- iztočnica novega zvoka ali posebnega dejanja (ang. cue point)
- MIDI Channel Prefix
- konec sledi (ang. end of track)
- določanje tempa (ang. set tempo)
- SMPTE odmik (ang. SMPTE offset)
- Key signature
- Sequencer specific

Vsaka sled se konča z dogodkom konec sledi (ang. end of track).

Od meta dogodkov sta za diplomsko delo najpomembnejša določanje tempa (ang. set tempo) in časovni podpis (ang. time signature).
Tempo sekvence je podan v mikrosekundah na četrtinko (ang. microsecond per quarter note - MPQN). Privzeta vrednost je 500.000 MPQN oz. 120 bitov na minuto (ang. bits per minute - BPM).

Formuli za pretvarjanje med MPQN in BPM:

$t = 60.000.000$ (število mikrosekund v minuti)

$$BPM = \frac{t}{MPQN}$$

$$MPQN = \frac{t}{BPM}$$

Časovni podpis (ang. time signature) sestavljajo štiri vrednosti – števec (ang. numerator), imenovalca (ang. denominator), pulz metronoma (ang. metronome pulse) in število $\frac{1}{32}$ not na MIDI četrtinko (ang. number of 32nd notes per MIDI quarter-note).

Števec in imenovalec skupaj tvorita vrsto takta po katerem se igra skladba.

Denumerator je predstavljen z negativno potenco števila 2. Metronom pulse specificira kako pogost je tik metronoma (24 pomen en kik na četrtinko, 48 en klik na polovinko).

Privzete vrednosti so $\frac{4}{4}$, 24 (en pulz metronoma na četrtinko) in 8 (8 $\frac{1}{32}$ not na četrtinko).

Sistemsko ekskluzivni dogodki (ang. system exclusive events)

Sistemsko ekskluzivni dogodki se uporabljajo za kontrolo MIDI strojne in programske opreme, ki potrebuje posebne podatke za delovanje. Standardizirani so trije sistemsko ekskluzivni dogodki, proizvajalci pa lahko dodajo svoje.

Poglavje 4

Implementacija

V okviru diplomskega dela je bila razvita računalniška igra za igranje MIDI pesmi s pomočjo senzorja Kinect. MIDI skladbe so zaradi svoje strukture primerne za enostavno razdelitev na manjše komponente – note, ki se jih nato lahko vizualno predstavi s 3D objekti. Osrednjo vlogo interakcije uporabnika z igro opravlja senzor Kinect. Omogoča navigacijo po menuju in deluje kot navidezni instrument preko katerega igralec lahko zaigra note pesmi.

4.1 Koncept igre

Pred začetkom igre je potrebno izbrati pesem, težavnost in število ciljnih škatel. Ker so MIDI skladbe lahko sestavljene iz več sledi, je posebej potrebno izbrati še sled. Težavnost je razdeljena na tri stopnje in določa število (gostoto) not, ki se pojavijo v igri. Algoritem za filtriranje števila not je opisan pod točko 4.5.

Po izbranih nastavitvah se igra lahko prične. Igralec mora čim uspešneje zaigrati note, ki se premikajo proti njemu. Posamezna nota je v igri predstavljena kot 3D objekt v obliki preproste škatle. Cilj kamor letijo je lahko na levi ali desni strani zaslona. Na vsaki strani je izbrano število ciljnih škatel. Note se premikajo (letijo) iz centra zaslona proti eni izmed ciljnih škatel. Nota je uspešno zaigrana, če igralec z roko ali nogo prekrije ciljno škatlo v

trenutku, ko se ta nahaja v njej. Poleg iztegnjene dlani pa igralec lahko tudi ujame (zaigra) noto z roko stisnjeno v pest in si s tem prisluži bonus točke. V primeru neuspešno zaigrane note, se predvaja zvok, ki predstavlja napako. Cilj igre je, da igralec pravilno zaigra čim več not in s tem osvoji veliko točk.

4.2 Delovanje

Izbrana MIDI pesem se pretvori v ustrezne objekte igre. Obdelava skladbe je narejena s pomočjo odprtokodne knjižnice C# MIDI Toolkit. Knjižnica podano MIDI datoteko pretvori v programske objekte, preko katerih je enostaven dostop do glave datoteke, posameznih sledi in njihovih lastnosti.

Za izbrano sled se kreirajo note. Nota je v MIDI datoteki predstavljena kot kombinacija kanalnih dogodkov (ang. channel events) Note on in Note off, ki skupaj tvorita podatek o začetku in koncu igranja note. Dogodek Note on predstavlja začetek nove note. Določa višino, vrsto in jakost tona ter pozicijo note v pesmi (kdaj se prične izvajati). Ta je določena z delta časom, ki predstavlja relativno število časovnih enot, ki potečejo od predhodnega MIDI dogodka. Note off dogodek se pojavi za Note on. Ker se med njima lahko izvede še veliko drugih dogodkov (npr. nov Note on dogodek v primeru, da sta zaigrani dve noti istočasno), je potrebno poiskati ustrezen Note off dogodek. Par se določi na podlagi številke note, ki jo ima tudi Note off.

Knjižnica na podlagi delta časa, ki je definiran za vsak MIDI dogodek, izračuna tudi absolutni čas pojavitve dogodka v pesmi. Tako je lažje računanje dolžine note, ki je preprosto razlika med absolutnim časom Note off in Note on dogodka v pesmi. Dolžina note se uporablja v algoritmu za filtriranje not opisanim pod točko 4.5.

Potrebno je tudi izračunati čas, ki poteče med sosednima notama. Ta je enak razliki med absolutnima časoma zaporednih Note on dogodkov. Podatek je pomemben zaradi sinhronizacije predvajanja skladbe s pošiljanjem 3D objektov, ki predstavljajo note. Sinhronizacija je potrebna, ker se predvajanje skladbe in pošiljanje not izvaja v ločenih nitih. S tem se reši problem

zamika, ki nastane, če vizualne note prožijo predvajanje.

Nota je v igri predstavljena kot 3D objekt v obliki kocke. Barva škatle je odvisna od oktave v kateri je zaigrana nota. MIDI datoteke podpirajo enajst oktav 3.5. Temnejša kot je barva nižjo oktavo predstavlja. Barvni spekter se giblje od vijolične preko različnih odtenkov modre, zelen in rumene do rdeče barve. Na vsaki škatli je tudi oznaka vrste note.

Z začetkom igre, se v ločeni niti prične pošiljanje zaporedja škatel. Zamiki med posameznimi kockami ustrezajo času, ki poteče me sosednima notama. Škatle letijo proti levi ali desni strani zaslona. Na vsaki strani je določeno število ciljnih škatel. Cilj proti kateremu leti posamezna vizualna nota določa algoritem opisan pod točko 4.6. Ko pride prva kocka v stik s ciljno škatlo, se prične predvajanje MIDI datoteke. Predvajanje skladbe je narejeno s pomočjo iste knjižnice kot obdelava.

Čas, ki poteče med sosednima notama, je izračunan iz MIDI datoteke in je predstavljen v relativnih časovnih enota. Pretvorba v milisekunde je odvisna od časovnega razdelka (ang. time division) MIDI datoteke in tempa trenutne sledi.

Formula za pretvorbo iz relativnih časovnih enot v milisekunde:

$tempo$ – tempo v mikrosekundah na četrtinko (MPQN)

$division$ – časovni razdelek v enoti število tikov na bit na četrtinko
(ang. ticks per beat on every quarter note)

t – dolžina enega tika v milisekundah

$$t = \frac{tempo}{division * 1000}$$

Ko vizualna nota prileti do ciljne škatle, jo je potrebno zaigrati. Stik med

ciljno škatlo in kocko, ki simbolizira noto, se določi s preverjanjem dotika posameznih sfer, ki navidezno obdajajo vsako škatlo (ang. bounding sphere).

Vlogo instrumenta nadomešča senzor Kinect. V časovnih intervalih pošilja na računalnik globinsko sliko okolice vključno z uporabnikom. Podatki, ki jih računalnik prejme, se obdelajo z algoritmi iz razvojne knjižnice Kinect SDK opisane v poglavju 3.3. Iz globinske slike se tako zazna položaj uporabnikovega okostja. Skelet je sestavljen iz dvajsetih okončin. Za igro so najpomembnejši položaji dlani in stopal. Pozicija teh okončin se primerja s položajem posameznih ciljnih škatel. Če dlan ali stopalo prekriva enega izmed ciljev v trenutku, ko se v njem nahaja škatla note, se ta uspešno zaigra in igralec prejme točko. Implementirano je tudi preverjanje stanja dlani. Ta je lahko v iztegnjenem položaju ali stisnjena v pest. Algoritem za prepoznavo stanja je tudi del razvojne knjižnice za Kinect. Za noto zaigrano s stisnjeno pestjo, igralec prejme deset točk.

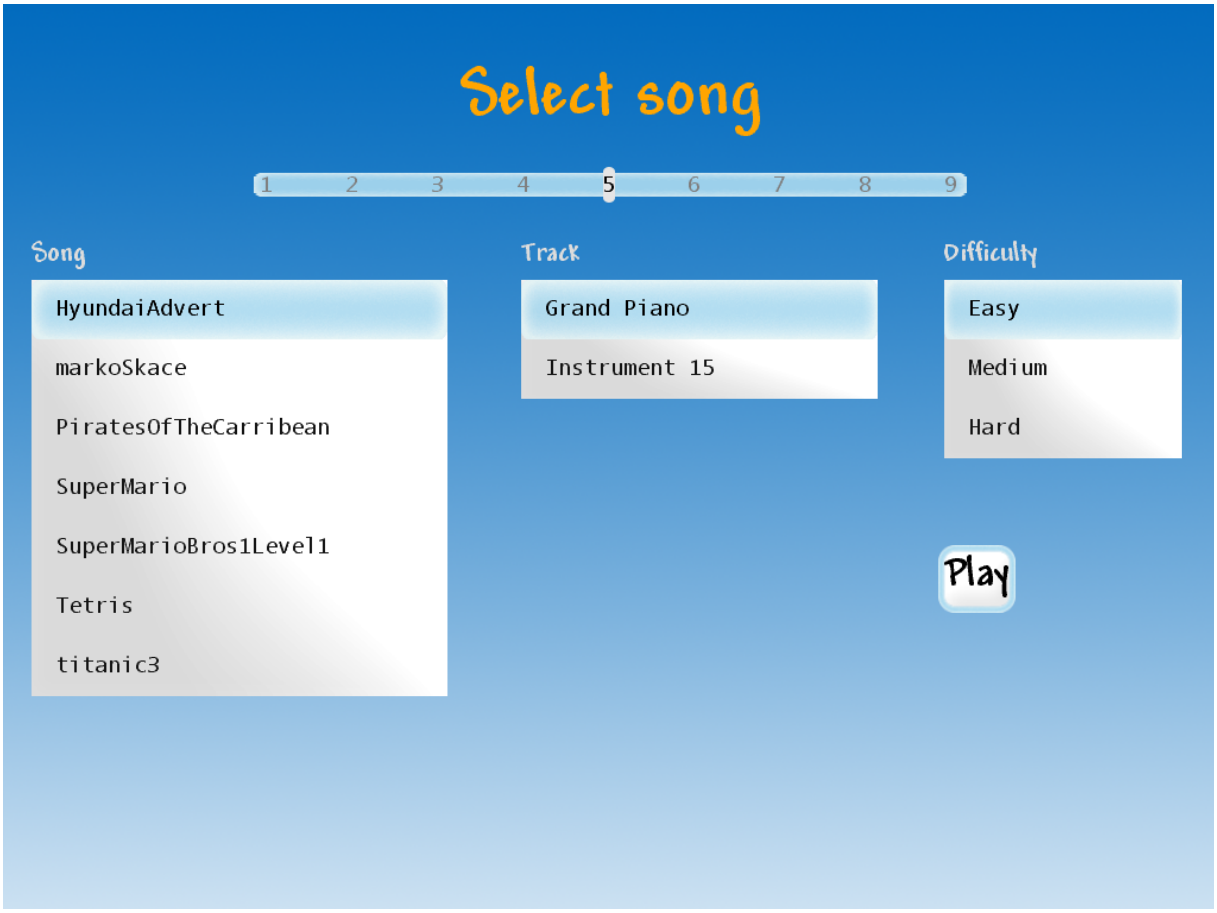
Če igralec zgreši ali zamudi noto, se zaigra zvok, ki ponazarja napako. Zvočni posnetek je formata WAVE (Waveform Audio File Format) in predvaja se s pomočjo knjižnice, ki je del ogrodja XNA Game Studio.

4.3 Meni nastavitev

Pred začetkom igre je potrebno izbrati pesem, sled, težavnost in število ciljnih škatel. Ker se za interakcijo uporabnika z aplikacijo uporablja senzor Kinect, se vse nastavitve določajo z gesto rok. S premikom roke se premika tudi kazalec na zaslonu, s stiskom roke v pest pa se označen element izbere.

MIDI skladbo sestavlja ena ali več sledi. Sled običajno vsebuje glasbene podatke za različne instrumente, ki igrajo različne dele pesmi (glavna melodija, spremljava, ritem...). Zato se po izbrani pesmi, prikaže seznam sledi, ki jih MIDI datoteka vsebuje.

Tretji seznam je namenjen izbiri težavnosti. Težavnost igre je definirana s številom not, ki jih potrebno zaigrati. Na izbiro so tri stopnje. Če je izbrana prva, najlažja, se v igri pojavijo le tiste note, ki so na prvem mestu v taktu.



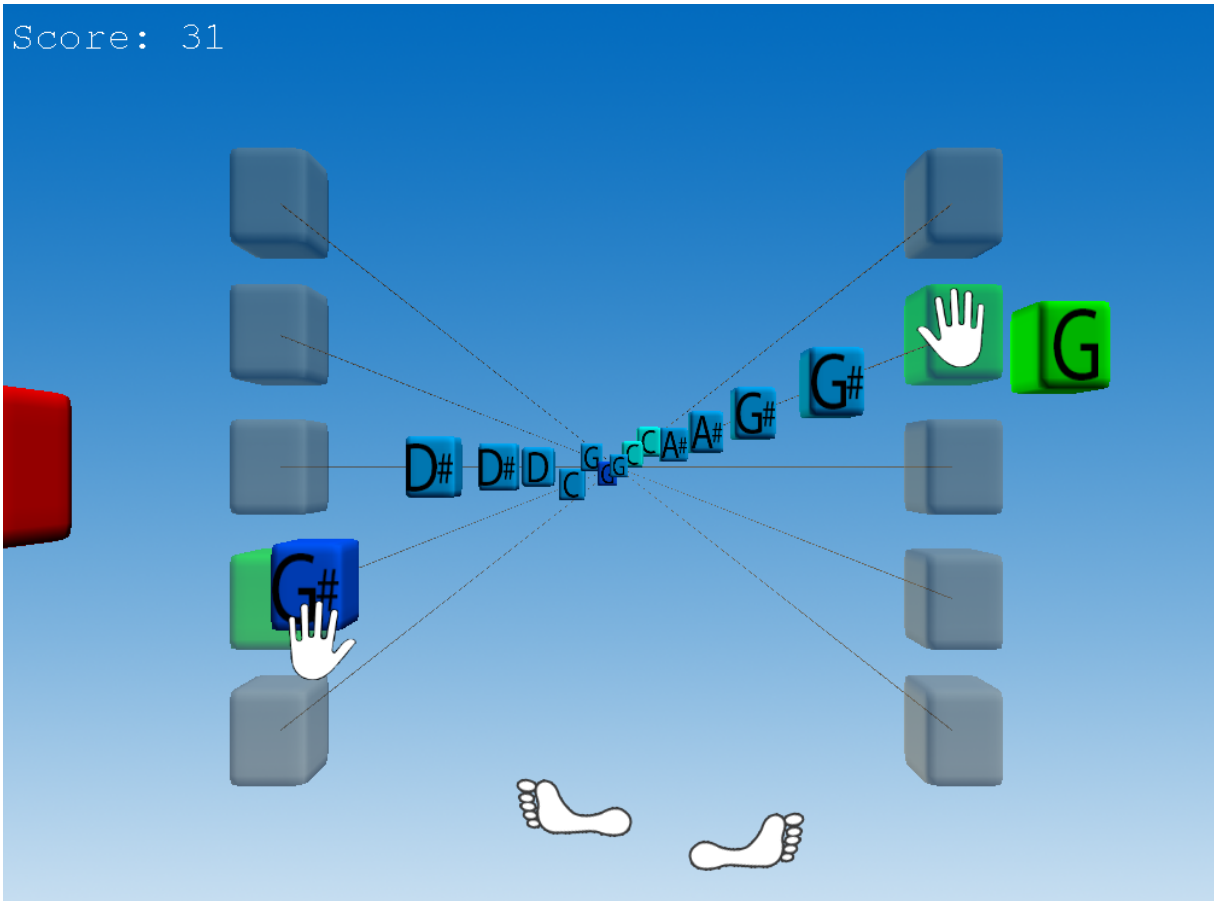
Slika 4.1: Meni za izbiro pesmi, sledi, težavnosti in števila ciljnih škatel.

Pri srednji težavnosti se iz vsakega takta odstrani polovico not. Pri najtežji pa je potrebno zaigrati vse note izbrane sledi.

Igralcu je na voljo tudi nastavitev števila ciljnih škatel. Vizualne note letijo proti levemu in desnemu delu zaslona in na vsaki strani je enako število ciljnih škatel, ki se giblje med 1 in 9.

4.4 Igralni meni

Igralni prostor je razdeljen na levi in desni del. Na vsaki strani je enako število ciljnih škatel. Te označujejo prostor, kjer je potrebno z ustreznim



Slika 4.2: Igralni meni med igro.

gibom zaigrati noto. Posamezni cilj je predstavljen kot polprosojen 3D objekt v obliki kocke. Pot po kateri potujejo vizualne note je izrisana v obliki ravne črte, ki vodi iz centra zaslona proti pripadajoči ciljni škatli.

Vizualne note letijo proti cilju iz sredine zaslona, iz navidezne neskončnosti. Tako kot ciljne škatle so tudi note predstavljene v obliki 3D kock. Barva škatle zavisi od oktave v kateri se nota zaigra. Na vsaki kocki je tudi tekstura z napisom tona, ki ga predstavlja.

4.5 Algoritem za filtriranje števila not

Stopnja težavnost je v igri definirana s številom not, ki jih je potrebno, iz izbrane pesmi (in ustrezne sledi), zaigrati. Za filtriranje števila not je bil razvit algoritem, ki deluje na principu uteži.

Algoritem podano pesem (sled) razdeli na posamezne takte. Dolžina takta se izračuna po formuli:

$division$ – časovni razdelek v enoti število tikov na bit na četrtinko
q – število četrtink v taktu
t – število tikov na vsak takt

$$t = division * q * 4$$

Oba podatka sta del MIDI datoteke.

Med dodajanjem not v takte, se določajo tudi lastnosti takta. Shrani se najkrajša in najdaljša nota, prva nota v taktu in najkrajši ter najdaljši čas, ki poteče med sosednima notama. Te lastnosti so potrebne za računanje uteži not. Po končanem razvrščanju tako vsak takt sestavljajo note katerih vsota dolžin ustreza dolžini takta.

Zmanjševanje števila not se izvaja z izločanjem not iz posameznih taktov na podlagi velikosti uteži. Te se izračunajo iz lastnosti pripadajoče note. Velikost uteži je tako odvisna od položaja note v taktu, dolžine note in dolžine pretečenega časa med sosednjima notama. Prvi noti v taktu se dodeli vrednost 2, vse ostale pa prejmejo 0. Vsak takt ima najkrajšo in najdaljšo noto in dolžine vseh ostalih se gibljejo med tema dvema vrednostma. Da so dolžine lahko primerljive, je potrebno to število pretvoriti na interval med 0 in 1. Tako se najprej od vsake dolžine odšteje najmanjša dolžina v taktu, nato pa se rezultat deli še z razliko največje in najmanjše dolžine. Daljša kot je nota višja utež dobi.

Kot tretja lastnost na velikost uteži vpliva dolžina pretečenega časa od predhodne note. Ravno tako je potrebno vrednost pretvoriti na interval med

0 in 1. Preslikava poteka na podoben način kot pri dolžini note. Najprej se od vrednosti odšteje najkrajši pretečeni čas, rezultat pa se nato deli z razliko najdaljšega in najkrajšega časa med sosednima notama. Več časa kot preteče, višja utež je dodeljena drugi noti.

Vse tri uteži se nato seštejejo in vsota določa vrednost note.

Ko so izračunane vse uteži, algoritem prične z izločanjem. Število odstranjenih not je enako vhodnemu parametru algoritma, ki v odstotkih predstavlja število not, ki jih je potrebno odstraniti. Glede na njegovo vrednost algoritem iz vsakega takta odstrani ustrezno število not z najmanjšimi utežmi.

Listing 4.1: Izsek kode iz algoritma za filtriranje števila not.

```
foreach (Measure measure in track)
{
    measure.measurePropertiesComp();
    measure.computeWeights();

    int measureLength = measure.midiNotes.Count;

    int numberOfNotesToRemove = (int)(measureLength * removeFaktor);
    if (numberOfNotesToRemove == measureLength)
    {
        numberOfNotesToRemove = measureLength - 1;
    }
    measure.removeLowWeightNotes(numberOfNotesToRemove);
}
```

4.6 Algoritem za razporeditev not na ciljne škatle

3D objekti - škatle, ki predstavljajo posamezne note pesmi se premikajo proti igralcu iz centra proti levemu ali desnemu predelu zaslona, kjer se nahaja enako število ciljnih škatel. Izbiro cilja posameznim notam določa algoritem za razporeditev not na ciljne škatle.

Najprej se določi stran zaslona – levo ali desno. Note, ki se v zaporedju nahajajo na lihih mestih potujejo na levo, note na sodih mestih pa na desno stran zaslona. Z izmeničnim dodeljevanjem se doseže najenakomernejša razporeditev not.

Ko je določena stran zaslona, pride na vrsto izbira ene izmed ciljnih škatel, ki zavisi od višine note. Višji noti je dodeljena višja škatla. Algoritem iz pesmi pridobi najvišjo in najnižjo noto. Na podlagi tega podatka se izračuna dolžina intervala not, ki nastopajo v pesmi. Ta interval se razdeli na toliko pod intervalov kot je ciljnih škatel. Vsaki noti se nato dodeli tisti pod interval, ki vključuje vrednost, ki jo zavzema višina note. S tem višje note letijo proti višjim in nižje proti nižjim škatlam. Z upoštevanjem dolžine intervala, ki ga zavzamejo note v skladbi, pa ciljne škatle nastopajo približno enakomerno.

Poglavje 5

Testiranje in evalvacija

Pri izdelavi računalniških iger je za kvalitetno igralno izkušnjo pomemben del razvoja tudi testiranje s potencialnimi uporabniki. Igralci lahko ponudijo kvalitetne predloge za izboljšave, saj razvijalci velikokrat težko razmišljajo o problemu izven svojih začrtanih idej. Zato je bilo v okviru diplomskega dela izvedeno tudi preprosto testiranje igre.

Predvsem nas je zanimalo katero število ciljnih škatel je primernejše za igro. Tako smo na manjšem vzorcu ljudi (< 10) testirali različne nastavitve in vprašali za njihovo mnenje.

Ciljne škatle se nahajajo na levi in desni strani igralnega zaslona. Število je na obeh straneh enako in se giblje med 1 in 9. Število ciljnih škatel posredno vpliva na potrebo po aktivnosti uporabnika med igro, ki je večja pri večjem številu škatel.

Potek testiranja

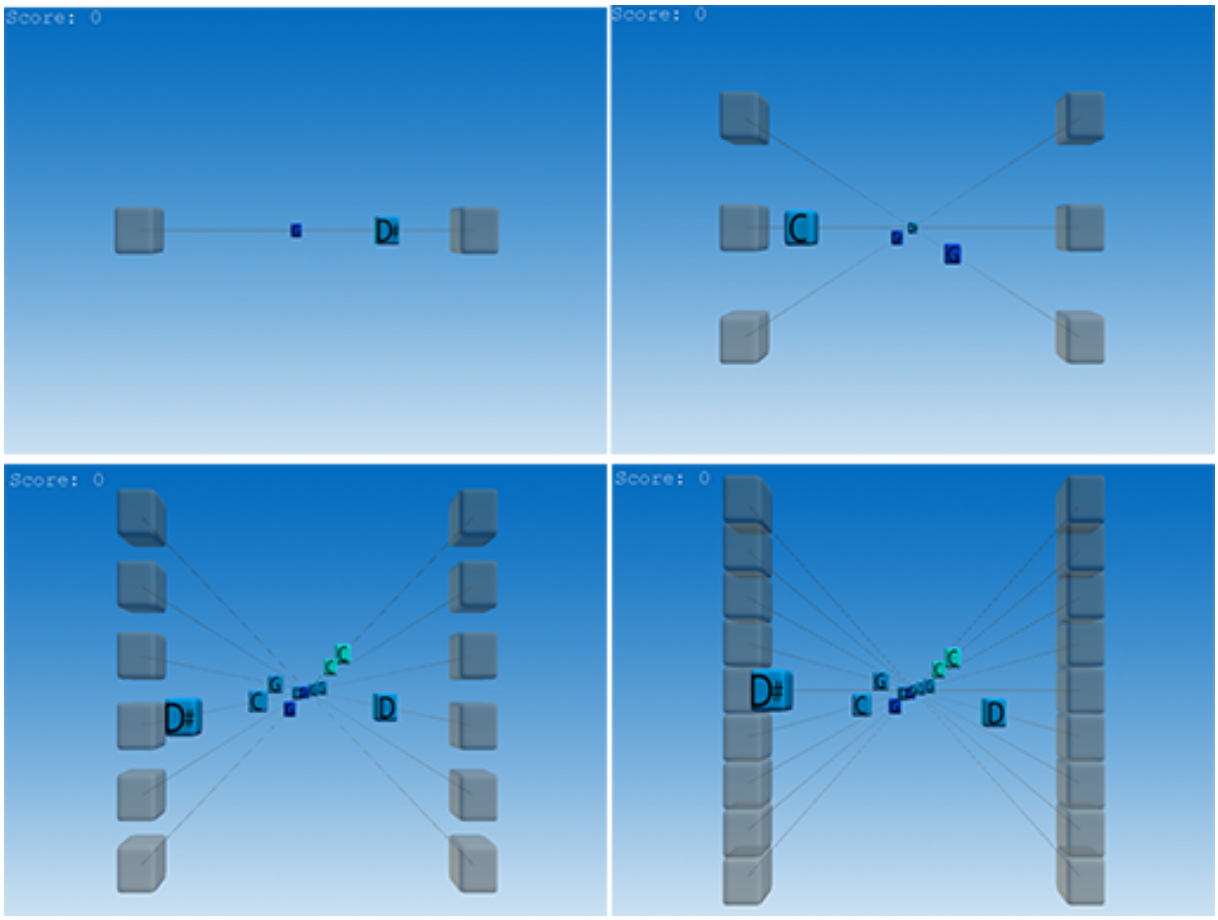
Vsak uporabnik je na srednji težavnosti preizkusil vsa možna števila (od 1 do 9) ciljnih škatel. Izmed vseh je izbral tri tista, ki so se mu zdela najprimernejša za igro. Nato je te tri možnosti preizkusil še na ostalih dveh težavnostih. Po tem testu je izbral končno število ciljnih škatel, ki je po njegovem mnenju najustreznejše za igro.

Rezultati

Rezultati so bili pri večini testiranih uporabnikov podobni. V ožji izbor najprimernejšega števila ciljnih škatel so vsi uporabniki izbrali števila 4, 5 in 6. Glavni argument zakaj manjše število (1, 2 ali 3) ni primerno, je bil, da zaradi manjšega števila ciljnih škatel igra postane preveč enostavna in s tem dolgočasna.

Števila večja od 6 (7, 8 in 9) pa povzročijo druge težave. Zaradi prevelike gostote ciljnih škatel, igra postane težko pregledna, saj so uporabniki težje pravočasno prepoznali proti kateri ciljni škatli potuje posamezna nota. Kot drugi problem pa so tudi navedli, da je pri večjem številu ciljnih škatel, težje dovolj hitro premakniti roko na pravilno škatlo.

Kot najprimernejše število ciljnih škatel je večina testnih uporabnikov določila število 5. Za to število so bili mnenja, da je ravno pravšnje razmerje med zahtevnostjo in zanimivostjo igre. Med škatlami je dovolj prostora, da lahko roko enostavno postaviš na željeno škatlo in tudi razpon med najnižjo in najvišjo ni prevelik. Podano je bilo tudi mnenje, da je število 5 ustrezno tudi iz simbolnega pomena, saj je notno črtovje tudi sestavljeno iz petih črt.



Slika 5.1: Primeri igre z eno, tremi, šestimi in devetimi ciljnimi škatlami.

Poglavje 6

Zaključek

V diplomskem delu je predstavljena računalniška igra za igranje MIDI skladb in vse tehnologije uporabljene za njen razvoj.

Opisano je delovanje senzorja Kinect in njegove lastnosti. Ker se uporablja za interakcijo uporabnika z igro, so opisani potrebni podatki, ki jih Kinect z ustrezno programsko opremo ponuja, in kako se do njih dostopa.

Podrobneje je predstavljena struktura standarda datotek MIDI. Te imajo ključno vlogo za pravilno uporabo pesmi v igri.

Glavni del diplomske naloge je bil razvoj računalniške igre. Tako je v nadaljevanju opisan njen koncept in delovanje. Predstavljeni so posamezni deli igre in dva razvita algoritma.

Izvedeno je bilo tudi preprosto testiranje igre na manjšem vzorcu potencialnih uporabnikov. Opisan je potek testiranja in odzivi uporabnikov.

Razviti koncept igre ima še veliko prostora za nadgradnje in dodajanje novih funkcionalnosti. Ker Kinect podpira sledenje skeleta dveh uporabnikov hkrati, bi bilo možno realizirati večigralnost (ang. multiplayer). Uporabnika bi lahko med seboj tekmovala ali pa igrala isto skladbo, vendar vsak drugo sled.

Konceptualno bi se igri lahko dodalo več izobraževalnega dela. Ker je igra glasbeno usmerjena, bi lahko uporabnik bolj nazorno spremljal lastnosti skladbe, ki jo igra. Tako bi se lahko skozi računalniško igro tudi glasbeno

izobraževal.

Literatura

[1] Wikipedia, Guitar Hero, September 2013. [Online]. Available: http://en.wikipedia.org/wiki/Guitar_Hero

[2] Wikipedia, Frets on Fire, September 2013. [Online]. Available: http://en.wikipedia.org/wiki/Frets_on_Fire

[3] Developer Network, Kinect for Windows Sensor Components and Specifications, September 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/jj131033.aspx

[4] Developer Network, Kinect Sensor, September 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/hh438998.aspx

[5] Developer Network, XNA Game Studio 4.0, September 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb200104(v=xnagamestudio.40).aspx

[6] Developer Network, Kinect for Windows SDK, September 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/hh855347.aspx

[7] Open Source Initiative, The MIT License (MIT), September 2013. [Online]. Available: http://opensource.org/licenses/mit-license.php

[8] Wikipedia, MIDI, September 2013. [Online]. Available: http://en.wikipedia.org/wiki/MIDI

[9] The Sonic Spot, MIDI File Format, September 2013. [Online]. Available: http://www.sonicspot.com/guide/midifiles.html