UNIVERZA V LJUBLANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matevž Baloh

Razvoj aplikacije za učenje klavirja

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Matija Marolt

Ljubljana 2012

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: 00232/2012
Datum: 04.04.2012

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

Kandidat: **MATEVŽ BALOH**

Naslov: **RAZVOJ APLIKACIJE ZA UČENJE KLAVIRJA**
DEVELOPMENT OF A PIANO LEARNING TOOL

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V delu preučite kako bi uspešno formulo, ki jo pri igranju inštrumentov uporablja igra "Guitar Hero", lahko prenesli na aplikacijo za učenje igranja klavirja. Razvijte konkretno aplikacijo in ocenite njeno primernost za učenje.

Mentor:

doc. dr. Matija Marolt

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Matevž Baloh, z vpisno številko 63090220, sem avtor diplomskega dela z naslovom

Razvoj aplikacije za učenje klavirja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije 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 21. septembra 2012. Podpis avtorja:

KAZALO

Uvod....1
Zasnova....2
2.1 Metodologije razvoja....4
2.2 Razvojni načrt....4
Razvoj igre....9
3.1 Povezava med sintetizatorjem in igro....11
3.2 Grafični vmesnik....16
Razvoj aplikacije za učenje klavirja....26
Razvoj aplikacije za pomoč učitelju klavirja....39
5.1 Funkcije za pomoč učitelju....41
Kritika končne aplikacije....46
Sklepne ugotovitve....49

KAZALO SLIK

Slika 2.1: Vmesnik za igro 'Guitar Hero' 2
Slika 2.2: Posnetek zaslona v igri 'Guitar Hero III: Legends of Rock' 3
Slika 3.1: Primerjava posnetega in predvajanega notnega zapisa..... 12
Slika 3.2: Prikazana okna ob prejemu sporočila iz sintetizatorja 13
Slika 3.3: Obrazec za določanje zakasnitve sporočila med sintetizatorjem in računalnikom .. 15
Slika 3.4: Grafični vmesnik aplikacije ob začetku razvoja..... 16
Slika 3.5: Grafični vmesnik aplikacije z izrisom pravokotnikov, ki predstavljajo sled igranega. 17
Slika 3.6: Grafični vmesnik z označenim območjem, kjer se pojavijo pravokotniki 18
Slika 3.7: Izsek grafičnega vmesnika ob dveh zaporednih iteracijah programa. Zgornji pravokotnik predstavlja ton, ki se že ni zaključil, spodnji pa ton, ki se je že zaključil. Pravokotniki se gibajo navzdol. 20
Slika 4.1: Grafični vmesnik z razdeljenim poljem, po katerim se gibajo pravokotniki. Delitev v skupine tipk. 31
Slika 4.2: Izsek grafičnega vmesnika z razdeljenim poljem, po katerem se gibajo pravokotniki Delitev v časovna obdobja. 34
Slika 4.3: Izsek grafičnega vmesnika. Prikaz tonov Fn, Gn, An in Hn. 35
Slika 4.4: Izsek grafičnega vmesnika. Prikaz pravokotnikov v barvi tipke, kateri ustrezajo. .. 37
Slika 5.1: Izsek grafičnega vmesnika, prikaz sledi igranega. Smer gibanja pravokotnikov je navzgor. Zgornja tona nista bila pritisnjena istočasno, spodnja tona pa ja. 42
Slika 5.2: Izsek grafičnega vmesnika. Barva tona prikazuje glasnost igranega od zelene (tiho) do rumene in nato rdeče (glasno). 44
Slika 6.1: Grafični vmesnik zaključene aplikacije. 46

KAZALO FORMUL

Formula 3.1: Formula za izračun zakasnitve med predvajanim in posnetim signalom. f_m označuje število udarcev na minuto (120). 12

POVZETEK

Diplomsko delo opredeljuje primernost uporabe formule, ki jo definira igra 'Guitar Hero' v aplikaciji, katere primarni namen je pomoč uporabniku pri učenju igranja na klavir. Primernost je opredeljena skozi postopek razvoja konkretne aplikacije.

Delo opisuje poizkus uporabe formule za izdelavo igre, katere primarni namen je zabava, sekundarni namen pa učenje klavirja. Poleg tega opisuje poizkus razvoja aplikacije, katere primarni namen je učenje klavirja in ne zabava. Namen take aplikacije je nuditi uporabniku vse prednosti, ki jih običajno nudi učitelj igranja na klavir. Tretji poizkus, ki ga delo opisuje, je razvoj aplikacije, katere namen je pomoč učitelju igranja na klavir pri pedagoškem delu.

Ugotovitve, ki določajo razpon primernosti formule za aplikacijo, katere namen je en od naštetih, izvirajo iz predvidenih in nepredvidenih težav, ki so se pojavile med razvojem. Delo pa se osredotoči predvsem na težave med razvojem z namenom, da morebitnim razvijalcem, kateri želijo razviti aplikacijo s podobnimi cilji kot jih zastavlja ena od opisanih aplikacij, nudi vpogled v razvoj in omogoča predvidevanje težav in kompromisov, katere bodo morali sprejeti.

ABSTRACT

This thesis analyzes the appropriateness of the formula defined by the game 'Guitar Hero' in an application, which aims to help it's users learn how to play the piano. The appropriateness is determined through the development of an application.

The thesis describes an attempt at the development of a game, the primary intention of which is to be fun, with the secondary purpose of teaching how to play the piano. After this, it describes an attempt at the development of an application, the primary intention of which is to teach how to play the piano and no longer to be fun. The purpose of such an application is to offer its users all the benefits, usually offered by a piano teacher. The third attempt, which the thesis describes, is the development of an application, the purpose of which is to aid a piano teacher in their teaching.

The conclusions, which determine the appropriateness of the formula for an application, the purpose of which is similar to one which is described, all come from the foreseen and unforeseen problems, which were encountered during development. The thesis focuses on the problems during development with the goal of offering potential developers an insight into which problems they can expect during development, allowing them to better foresee the problems they will encounter and determine which compromises they will have to take.

Poglavje 1

Uvod

Guitar Hero, prevedeno Heroj Kitare, je lastno ime serije glasbenih, računalniških iger. Prva igra v seriji je izšla leta 2005 pod imenom Guitar Hero. Sledile so ji Guitar Hero II (2006), Guitar Hero III Legends of Rock (2007), Guitar Hero World Tour (2008), Guitar Hero 5 (2009) in Guitar Hero: Warriors of Rock (2010).

Serija iger je znana po načinu igranja, ki predvideva uporabo vmesnika, katerega oblika spominja na kitaro. Ideja implementacije metod, s katerimi je serija postala uspešna, tudi na druge inštrumente, je plod naravnega razvoja raziskovanja primernosti uporabe uspešnih projektov.

Drugi pomemben vzrok za razvoj prihaja iz pomanjkanja aplikacij, ki uspešno izkoriščajo prednosti računalnikov v pedagoškem postopku. V tem primeru gre za postopek učenja igranja na klavir.

Primarni načrt razvoja je predvideval razvoj igre, katero uporabnik igra z namenom zabave, sproti pa se nauči igranja na klavir. Zaradi konceptualnih težav smo načrt spremenili in razvili aplikacijo, katera se ne predstavlja kot igra, marveč kot aplikacija, katere namen je pomoč učitelju pri pedagoškem delu. Tekom razvoja pa smo prišli do ugotovitev o primernosti formule 'Guitar Hero' za klavir.

Poglavje 2

Zasnova

Igre v seriji Guitar Hero se igra s posebnim vmesnikom, ki ga prikazuje slika 2.1. Primarni vmesnik je oblikovan kot manjša kitara, s petimi gumbi na vratu kitare in podolgovatim stikalom na trupu, s katerim uporabnik simulira brenkanje.

Slika 2.1: Vmesnik za igro 'Guitar Hero'

http://en.wikipedia.org/wiki/File:Guitar_Hero_World_Tour_Guitar_Controller_PS3.png

Med igranjem skladbe se na zaslonu pojavi pet pasov, po katerih se od vrha proti dnu enakomerno gibajo ikone, ki določajo trenutke, ko naj igralec zabrenka, kot prikazuje slika 2.2. Vsak od petih pasov pa predstavlja gumb na vratu vmesnika, katerega mora igralec držati pritisnjenega, ob pritisku stikala za brenkanje.

Slika 2.2: Posnetek zaslona v igri 'Guitar Hero III: Legends of Rock'

http://images2.wikia.nocookie.net/__cb20110521201451/guitarhero/images/b/b5/Guitar-GH3-hammeron.jpg

Igra je razdeljena na več težavnostnih stopenj. Najlažja stopnja vključuje le uporabo treh gumbov, katere držimo s kazalcem, sredincem in prstancem. Naslednji nivo vključi uporabo četrtega gumba, katerega držimo s šibkejšim mezincem. Zadnja dva nivoja zahtevnosti pa vključujeta vseh pet gumbov. Posledično mora igralec pri teh dveh nivojih roko premikati po vratu vmesnika. Četudi ima roka pet prstov, kar je enako število kot gumbov na vratu vmesnika, uporabljamo palec pri podporo vrata vmesnika, tako da ga ne moramo uporabiti za pritisk gumbov.

Kljub temu, da je gibanja roke malo, pa veljajo igre v seriji 'Guitar Hero' za težke za obvladati. Nekateri kritiki [5] celo trdijo, da je v določenih situacijah igranje igre težje kot igranje iste pesmi na dejanski električni kitari.

Igra je dostopna novincem. Navodila za igranje lahko povemo v eni povedi in vsakemu igralcu bo takoj jasno, kaj mora napraviti in kaj je cilj. Poleg tega so začetni nivoji zahtevnosti dovolj enostavni, da jih lahko večina zaigra spodobno že prvič. Prve pesmi, ki jih igralec poizkusi igrati so privlačne, popularne in enostavne.

Igralec, ki igro že dalj časa igra, s časom pridobi boljšo sposobnost igranja. Tako igro počasi začne obvladati, a ključna lastnost igre je, da jo je težko v celoti obvladati. Težavnostna stopnja igre sega od zelo enostavne do take, kateri velika večina igralcev nikoli ne bo kos.

Kombinacija dostopnosti novincem in težavnosti obvladanja je zelo pomemben aspekt vsake igre in posledično pomemben cilj, ki ga moramo upoštevati že med načrtovanjem igre [2].

2.1 Metodologije razvoja

Metodologija razvoja je teoretično določilo korakov in postopkov, ki so potrebni za razvoj neke aplikacije. Obstaja več metodologij, a za potrebo razumevanja korakov je potrebno razumeti le slapovni model, poznan tudi kot zaporedni model (ang. waterfall model) [7].

Slapovna ali zaporedna metodologija razvoja je stara metodologija razvoja programske opreme. V praksi dela po točno definiranih korakih z jasno določenimi začetki in zaključki korakov.

Metodologija je definirana s petimi koraki: analiza zahtev, načrtovanje, izvedba, vpeljava in vzdrževanje. Koraki si sledijo zaporedno, iz česar izvira ime metodologije [8].

Zaporedje korakov omogoča enostavno razumevanje potrebnih postopkov, a hkrati povečuje potrebo po kakovostni izvedbi vsakega koraka. Vračanje nazaj v metodologiji namreč ni mogoče. Edina razumna pot vrnitve je vrnitev na korak analize, kar v praksi pomeni ponovni začetek projekta.

2.2 Razvojni načrt

Primarni načrt je načrt, ki smo ga poslali mentorju, z namenom, da preverimo ali je načrt smiseln in v skladu z razpisano temo.

Načrt je kot končni izdelek predvideval končano aplikacijo, ki naj bi služila namenu učenja klavirja, a se uporabniku predstavljala kot igra. Predvidevali smo, da bi služila vsem težavnostnim stopnjam od začetnikom do naprednih igralcev klavirja.

Kot programski jezik smo si zastavili jezik C# in razvojno okolje Microsoft Visual Studio 2010. Razlog za to so naše predhodne izkušnje s povezovanjem eksternih naprav in računalniških aplikacij v jeziku. Predvidevali smo, da bo vzpostavitev zanesljive povezave med aplikacijo in eksternim sintetizatorjem največja ovira pri razvoju. Razvojno okolje Microsoft Visual Studio 2010 smo dobili skozi projekt MSDNAA.

Načrt je predvideval, da bi sintetizator, na katerega bi igrali, z računalnikom povezali s kablom. Naš sintetizator je kot izhod podpiral MIDI vtič. Računalnik takega vtiča ni imel, zato smo kot predvideno metodo povezave izbrali MIDI-to-USB kabel, to je kabel, ki ima na enem koncu vtič USB in na drugem koncu vtič tipa MIDI. Signale, ki prihajajo iz enega ali

drugega konca pa pretvori v signale, katerih pomen je enak, a so kompatibilni z vtičem in standardom na drugi strani.

Za uporabniški vmesnik je primarni načrt predvideval igralno okno, na dnu katerega je izrisana cela ali delna klaviatura, po ideji podobna kot oznake na dnu pasov v igri 'Guitar Hero'. Proti tipkam naj bi se od vrha proti klaviaturi, navpično navzdol gibali simboli, predvidoma pravokotniki, ki predstavljajo čas, ko mora tipka biti pritisnjena. Ti pravokotniki naj bi začeli na vrhu okna aplikacije, direktno nad tipko, katero predstavljajo. Tudi to je povzeto po modelu, ki ga uporablja Guitar Hero.

Igre iz serije 'Guitar Hero' podpirajo le omejeno število pesmi. Le-te so v originalni izvedbi, kakovost izvajanja pa določa naša natančnost igranja. S tem so omejeni le na predvidene, pripravljene pesmi. Naša aplikacija predvideva, da bomo navodila za izvajanje pesmi prebrali iz MIDI datoteke, to je glasbena datoteka, ki vsebuje navodila za sintetizacijo igranja neke skladbe.

MIDI protokol nam ni bil poznan, posledično smo ključne dele protokola morali preučiti. Predvidevali smo razvoj ogrodja, ki bo skrbelo za pravilno in zanesljivo povezavo med klaviaturo in aplikacijo. MIDI protokol je sicer dobro dokumentiran, a kompleksen za samostojno implementacijo v popolnosti. V tem koraku smo predvidevali, da se bodo lahko pojavile nepredvidene težave pri implementaciji komunikacije. Kot možno alternativo smo predlagali izdelavo in implementacijo protokola, katerega bi uporabljal hipotetični sintetizator, kompatibilen s programom. Izdelava in implementacija tega protokola bi se izvedla le v primeru, da bi implementacija komunikacije preko MIDI protokola spodletela. K sreči se je kmalu izkazalo, da obstaja odprtokodna knjižnica C# MIDI Toolkit [6], ki podpira branje in predvajanje MIDI datotek in komunikacijo z MIDI napravami, kot so sintetizatorji.

Po prvem posvetu z mentorjem, smo načrtu dodali uporabo knjižnice C# MIDI Toolkit. Kot cilj je še vedno bila zastavljena zaključena aplikacija, ki naj bi služila učenju klavirja, a naj bi se uporabniku predstavila kot igra.

Ob zaključku koraka analize zahtev smo imeli kreiran konceptualni načrt pričakovanih dosežkov. Ob zaključku projekta smo želeli imeti aplikacijo, ki se čimbolj drži formule, ki jo definira 'Guitar Hero'. Aplikacija naj bi bila predvsem uporabna, namenjena učenju igranja na klavir in naj bi bila zaključena.

Sledil je korak načrtovanja aplikacije, v katerem smo kreirali bolj podroben načrt z mejniki v razvoju ter časovnega obdobja, v katerem naj bi posamezni mejnik dosegli.

Prvi zastavljen mejnik je bila vzpostavitev zanesljive povezave med aplikacijo in sintetizatorjem, saj smo ta korak smatral za najbolj tveganega. Dobra praksa razvoja aplikacij določa, da se najprej opravijo najbolj tvegani deli razvoja.

Drugi zastavljen mejnik je bil dodatek funkcionalnosti tabeliranja in predvajanja MIDI dogodkov iz MIDI datotek.

Tretji mejnik je vseboval aplikacijo z dokončanim uporabniškim vmesnikom ter dodano funkcionalnostjo izpisa rezultatov. Med tem ko je večina grobe funkcionalnosti dokončana že pri drugem mejniku, tretji vključuje še analizo igranja in podatke, ki so uporabniku zanimivi. Doseg tega mejnika naj bi naznanil zaključek faze izdelave.

Za tem je bilo predvideno še obdobje testiranja in morebitnih nadgradenj. Ta čas pa naj bi služil tudi kot rezerven čas za primer, da bi med razvojem naleteli na večje težave.

Kot odskočno desko za razvoj smo si izbrali aplikacijo, ki pride s knjižnico C# MIDI Toolkit. Znotraj knjižnice, izbrana aplikacija služi kot demonstracijsko orodje za delo s knjižnico, vsebuje pa že nekaj kosov kode, ki jih razvoj aplikacije potrebuje. Posledično smo se odločili, da bo kot temelj naši aplikaciji služil ta demonstracijski program, imenovan SequencerDemo. Tekom razvoja aplikacije tega imena nismo spreminjali, tako da služi tudi kot razvojno ime naši aplikaciji.

SequencerDemo je aplikacija, ki demonstrira nekaj orodij, ki jih nudi knjižnica C# MIDI Toolkit. Na dnu aplikacije se izriše klaviatura. Ko z miško stisnemo gumb na tej klaviaturi, se predvaja ton, ki ustreza gumbu.

Demonstracijska aplikacija nudi tudi predvajanja MIDI datotek. Datoteko naložimo z uporabo menija na vrhu. Najprej izberemo 'File', nato 'Open…'. Nato izberemo MIDI datoteko, ki jo želimo predvajati in jo odpremo. Predvajanje izbrane datoteke upravljamo z gumbi 'Stop', 'Start' in 'Continue' na samem oknu. Premikanje po MIDI datoteki nam omogoča drsnik nad gumbi za upravljanje s predvajanjem.

Poleg tega nam demonstracijski program omogoča še igranje klavirja preko tipkovnice. Tipke na srednji vrstici s črkami ustrezajo belim tipkam in ustrezne tipke zgornje vrstice s črkami ustrezajo črnim tipkam. Igranje črnih tipk na slovenski tipkovnici ne deluje dobro, saj je predvidena uporaba tipkovnice, kjer sta tipki 'y' in 'z' zamenjani, tako da je 'y' v zgornji vrstici s črkami in 'z' v spodnji vrstici s črkami.

Odprtokodna knjižnica C# MIDI Toolkit je zasnovana na delu z MIDI sporočili. Različni razredi, ki jih knjižnica podpira, med seboj komunicirajo s temi MIDI sporočili, ki so definirana v svojem razredu. Parametri MIDI sporočil so isti, ki definirajo sporočilo, ki se

prenese med sintetizatorjem in računalnikom. Najpomembnejši parametri sporočila so poimenovani 'command', 'data1', 'data2' in 'MidiChannel'.

Command, Slovensko 'ukaz', je parameter, ki hrani podatek o ukazu, ki ga sporočilo nosi. Nam sta najpomembnejša tipa ukaza poimenovana 'NoteOn' in 'NoteOff', ki predstavljata dogodek pritiska tipke na sintetizatorju in dogodek spusta tipke. Vključno z 'NoteOn' in 'NoteOff' obstaja skupaj 8 unikatnih ukazov, katere definira 4 bite dolg zapis, pri kateremu je bit z najvišjo vrednostjo vedno enak 1.

Data1, Slovensko 'Podatek 1', v primerih ukaza 'NoteOn' in 'NoteOff' nosi zaporedno vrednost tipke, na katero se dogodek nanaša. Vrednost parametra data1 je lahko med 0 in 127. Vrednost 0 predstavlja pritisk ali spust tipke, ki ustreza tonu C0, vrednost 1 predstavlja pritisk ali spust tipke, ki ustreza tonu C#0 in tako se nadaljuje do vrednosti 127, ki predstavlja tipko, ki ustreza tonu G10 [9].

Data2, Slovensko 'Podatek 2', je parameter, ki v primeru ukaza 'NoteOn' nosi hitrost pritiska tipke, ki ustreza glasnosti tona, ki ga pritisk tipke sproži. Vrednost tega parametra je lahko od 0 do 127, kjer 127 pomeni maksimalno hitrost pritiska in posledično maksimalno glasnost tona, 1 predstavlja minimalno hitrost pritiska in posledično minimalno glasnost tona, 0 pa je ekvivalent ukazu tipa 'NoteOff'. Nekateri sintetizatorju in MIDI datoteke namreč namesto ukaza 'NoteOff', ob dogodku spusta tipke, pošljejo ukaz 'NoteOn' z glasnostjo 0.

MidiChannel, Slovensko 'MIDI kanal', je zadnji parameter, ki je potreben za uspešno procesiranje sporočila. Parameter definira MIDI kanal, kateremu je sporočilo namenjeno. Teh kanalov je lahko maksimalno 16 in jih definira 4 bitno število. Komunikacija med sintetizatorjem in računalnikom poteka na prvem kanalu, ki nosi indeks 0.

MIDI datoteke so zapisane v obliki zaporedja MIDI sporočil. Branje in predvajanje datotek poteka tako, da se sporočila iz datoteke preberejo in uvrstijo na pravo mesto v časovnik. Le-ta se sproži vsako milisekundo in izvede vse MIDI dogodke, ki so predvideni za tisto milisekundo.

Knjižnica podpira komunikacijo z zunanjimi napravami. To se izvede z uporabo razredov 'InputDevice' za vhodne naprave in 'OutputDevice' za izhodne naprave. Med razvojem smo uporabljali razred 'InputDevice', saj smo sporočila pošiljali iz sintetizatorja proti računalniku.

Delo z vhodnimi napravami poteka tako, da iz seznama primernih naprav izberemo tisto, s katero se želimo povezati. Knjižnica podpira metodo, ki nam ta seznam tvori. Povezavo pa vzpostavimo tako, da ustvarimo objekt tipa 'InputDevice', kateremu podamo kot parameter indeks naprave, s katero se želimo povezati. Indeks dobimo iz seznama primernih

naprav. Nato ustvarjenemu objektu določimo metodo, ki se izvede ob prejetem MIDI sporočilu. Nazadnje pa kličemo še metodo 'StartRecording()', ki vzpostavi povezavo in začne s proženjem metode, katero smo določili, da se izvede ob prejemu MIDI sporočila.

Poglavje 3

Razvoj igre

Razvojni načrt je predvideval razvoj aplikacije, katere namen je učenje klavirja, uporabniku pa je predstavljena kot igra. To poglavje opisuje razvoj aplikacije kot igre in opredelitev primernosti formule 'Guitar Hero' za igro s klavirjem.

Izdelava računalniških iger je disciplina, ki zahteva kolaboracijo ekip iz mnogih disciplin. Industrija računalniških iger se je razvila do take mere, da je ustvarila nove, samostojne discipline, kot si oblikovalci nivojev (Angleško level designer).

Tako kot oblikovanje in izdelava namiznih iger, ima razvoj računalniških iger določene specifike, katerih se mora zavedati vsak član ekipe, ne glede na vlogo, ki jo izpolnjuje [1]. Omejitve računalniških iger pretežno izvirajo iz sposobnosti računalnika in sposobnosti igralca. Kot primer: igralec si ni sposoben zapomniti ogromnega števila podatkov, računalnik pa ni sposoben igralcu prikazati okolja preko celotnega nabora človekovih čutil.

Poznavanje disciplin računalništva, kot so programiranje, oblikovanje uporabniških vmesnikov in uporabe omrežnih tehnologij, sicer ni edini predpogoj za izdelavo računalniških iger, je pa nujno potrebno.

Samostojni razvoj računalniških iger zahteva poznavanje veliko disciplin. Posameznik, ki se sam loti razvoja računalniških iger mora imeti osnovno razumevanje programiranja, pisanja zgodbe in dialoga, grafične umetnosti, modeliranja, oblikovanja nivojev, uporabe glasbe, testiranja in drugih. Poznavanje vseh teh disciplin zahteva veliko izkušenj, a ne zadostuje za izdelavo kakovostnega izdelka. Kot primer: ni ključno, da je kakovost zvoka popolna, marveč je ključna uporaba zvoka v povezavi z vsemi ostalimi elementi. Pri zvoku, katerega namen je, da igralca opozori na nek dogodek, je mnogo manj pomembna kakovost izbranega zvoka, kot uspešnost cilja, da zvok igralca dejansko opozori. Izjemno prijeten in

kakovosten zvok, kateri igralca ne uspe opozoriti na dogodek, ni uporaben. Ta primer lahko služi le kot primer povezanosti med elementi in disciplinami, ki so povezane v razvoj računalniških iger.

Samostojni razvijalec mora torej poznati vse discipline, katere uporablja v svoji igri in povezave med njimi. Posledično je razvoj počasen in prepleten z nepredvidenimi težavami. Poleg tega počasnost razvoja omejuje maksimalno velikost projekta.

Osnovna ideja igre, ki jo razvijamo, je zasnovana na formuli, ki jo definira 'Guitar Hero'. Smernice razvoja predvidevajo, da se bo igra formule čimbolj držala.

Poleg osnovne ideje je smernica razvoja ta, da bo igra uspešna metoda za učenje igranja na klavir. Osnoven cilj igre je namreč ta, da bo igralca pomagala naučiti igrati na klavir.

Tretja smernica je privlačnost same igre. Vsaka igra mora biti globoka in igralec mora ostati zainteresiran za nadaljevanje z igranjem. Različne igre dosegajo to na različne načine, med drugim s poudarkom na zabavnost, dobro zgodbo, uporabo tehnike Skinnerjeve kletke ali preprosto edinstvenost izkušnje. Za našo igro smo si izbrali edinstvenost izkušnje s potencialno razširitvijo, ki bi uporabljale metode, povezane s tehniko Skinnerjeve kletke.

Namen igre je, da igralca pomaga naučiti igrati klavir, a uporabnik naj nebi vstopil v vlogo igralca z namenom, da se nauči igranja na klavir. Uporabnik naj bi igro igral zato, ker je zabavna, sproti pa naj bi se naučil igranja na klavir. Ker je postopek učenja igranja na klavir dolg, bo sicer bolj verjetno, da bo igra služila zgolj toliko, da bo igralec prebil prvo oviro igranja, katere vzrok je popolno pomanjkanje znanja na neko temo. Igra naj bi služila toliko, da bi se igralec naučil nekaj enostavnih melodij, s čimer bi pridobil dovolj znanja, da mu klavir nebi bil popolnoma tuj. S tem bi nato lahko nadaljeval na poti učenja igranja na klavir, bodisi s pomočjo igre, ali sam.

Uporaba edinstvenosti izkušnje, kot metode za vzdrževanje zanimanja igralca ni popolno. Edinstvenost izkušnje namreč relativno hitro mine. Posledično kmalu pride v poštev tehnika Skinnerjeve kletke, katere namen je ravno obraten.

Skinnerjeva kletka je ime tehnike pogojevanja dejanj, ki sodelujočega nagrajuje vsakič ko stori dejanje, katerega želimo da stori. Originalni eksperiment je vseboval škatlo, v kateri sta bila golob in gumb. Ko je golob pritisnil gumb je dobil hrano. Kmalu se je naučil, da to dejanje nosi nagrado. Do tedaj so psihologi vedeli, da lahko pogojujejo reakcijo posameznikov, s tem eksperimentom pa so ugotovili, da lahko pogojujejo tudi akcijo – dejanje. Kasneje so ugotovili, da to deluje tudi na ljudeh, a da so fiziološke potrebe, kot je hrana, slab način nagrajevanja, saj sodelujoči v nekem koraku doseže nasičenost. V primeru

hrane to pomeni, da je sodelujoči sit. Če je nekdo sit ne bo več delal akcije, katero nagrajujemo s hrano. Ugotovili pa so, da potrebe, katere niso fiziološke po značaju, v splošnem ne dosežejo nasičenosti. Primeri takih potreb sta avtoriteta in denar [4].

Sodobne igre pogosto uporabljajo tehnike, ki so zakoreninjene v Skinnerjevi kletki. Vsaka igra, ki vsebuje nivoje napredovanja, točke ali podobno obliko numeričnega napredovanja, zelo verjetno uporablja tehniko Skinnerjeve kletke.

Naša igra naj bi implementirala Skinnerjevo kletko v obliki ocenjevanja in ovrednotenja igranja z uporabo točkovne ocene. Igra naj bi vedla v katerem trenutku mora uporabnik pritisniti katero tipko. Ta podatek bi dobila iz MIDI datoteke, katero program predvaja. Vedela bi tudi, v katerem trenutku je uporabnik dejansko pritisnil tipko. Iz teh podatkov naj bi izračunala neko številčno oceno.

Igralec bi v zaporednih igranjih neke pesmi napredoval. Posledično bi se tudi ta številka višala, kar bi odražalo igralčevo napredovanje. Pričakovano je, da naj bi ta številka, ki predstavlja merljiv napredek, igralca držala zainteresiranega v igranje igre, vsaj dokler se njegov napredek ne ustavi. To-je stanje, ko napredek ni več tako hiter, kot na začetku. V tem stanju je verjetno, da bo igralec bodisi končal z igranjem, bodisi začel z drugo pesmijo.

3.1 Povezava med sintetizatorjem in igro

Prvi predvideni mejnik razvojnega načrta je bil vzpostavitev zanesljive povezave med klaviaturo in igro. S strani fakultete smo prejeli kabel MIDI-to-USB, ki na eni strani vsebuje dva moška MIDI vtiča, na drugi pa en moški USB vtič. MIDI vtiča sta označena z oznako 'In', kar pomeni 'Vhod', in 'Out', kar pomeni 'Izhod'. Oznaki označujeta smer pretoka podatkov, gledano iz strani računalnika. MIDI vtič, ki nosi oznako 'In' torej predstavlja vhod v računalnik. Tega bomo priključili v sintetizator, ki bo v računalnik pošiljala podatke o pritisnjenih tipkah. Vtiča, ki nosi oznako 'Out' ne bomo uporabili, saj računalnik sintetizatorju ne bo pošiljal ničesar.

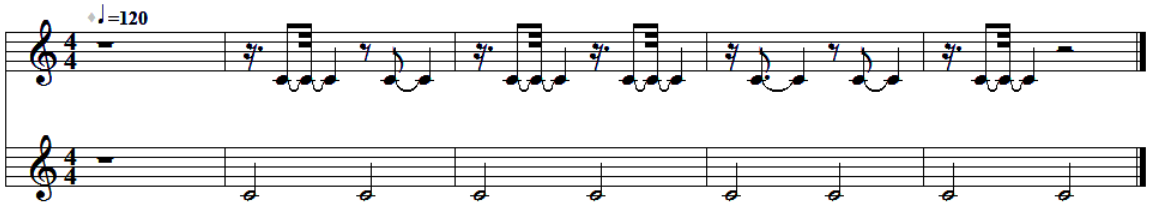
Sintetizator, katerega bomo uporabljali, ima dva ženska MIDI vtiča. Oznaki nad vtičema sta enaki kot na moških vtičih MIDI-to-USB kabla. En ima oznako 'In', drugi 'Out'. Pomen obeh je enak, kot pri kablu, le da je tokrat gledano iz strani sintetizatorja. Vtič z oznako 'Out' je tisti, katerega sintetizator uporablja, ko pošilja podatke. Za delovanje igre je potrebno, da sintetizator ob pritisku tipke, računalniku pošlje podatek, zato bomo na sintetizatorju uporabili vtič, z oznako 'Out'.

Morda zveni proti-intuitivno, a za pravilno delovanje moramo MIDI vtič z oznako 'In' na MIDI-to-USB kablu, vstaviti v vtič z oznako 'Out' na sintetizatorju.

Na operacijskem sistemu Okna 7 je primerni gonilnik poiskal sam operacijski sistem. Naprava in gonilnik je v upravljalniku naprav uvrščena v razdelek krmilnikov zvoka, grafike in iger pod imenom 'USB2.0-MIDI'

Vzpostavitev povezave med sintetizatorjem in računalnikom je bila s tem zaključena. Povezavo smo preizkusili z brezplačno različico programa za komponiranje glasbe imenovanim Noteworthy Cmposer [10].

Povezava je delovala, a prva potencialna težava se je pojavila v tem koraku. Težavo prikazuje slika 3.1. Spodnja vrstica slike prikazuje predvajano zaporedje, zgornja pa posneto zaporedje. Tipko na klavirju smo stisnil istočasno kot smo slišal predvajano noto. Ob prejemu ukaza za začetek snemanja, program hkrati začne snemanje in predvajanje vnesenega zaporedja, v tem primeru spodnje vrstice.



Slika 3.1: Primerjava posnetega in predvajanega notnega zapisa.

Ta preizkus smo ponovili nekajkrat in vsakič ugotovili, da zakasnitev med predvajanjem note in beleženjem istočasno pritisnjene note na sintetizatorju, najpogosteje traja šestnajstinko in pol pri hitrosti 120 udarcev na minuto.

Ena šestnajstinka traja eno šestnajstino celinke. Celinka traja štiri dobe. Ena doba traja časovno obdobje, katero določi število udarcev na minuto f_m, v tem primeru 120.

$$t = (1{,}5) \cdot (\frac{\frac{60}{f_m} \cdot 4}{16}) = 0{,}1875\text{s}$$

Formula 3.1: Formula za izračun zakasnitve med predvajanim in posnetim signalom. f_m označuje število udarcev na minuto (120).

Zakasnitev, izračunana v formuli 3.1, se je v tem koraku izkazala kot 0,1875 sekunde, kar je nesprejemljivo veliko. Vzrok za tako visok čas zakasnitve je lahko sintetizator, ki morda prepočasi pošilja podatke. Lahko je kabel, ki morda prepočasi pretvarja MIDI podatke v signal, katerega lahko prenese preko USB kabla. Lahko je gonilnik, ali pa sam program, s

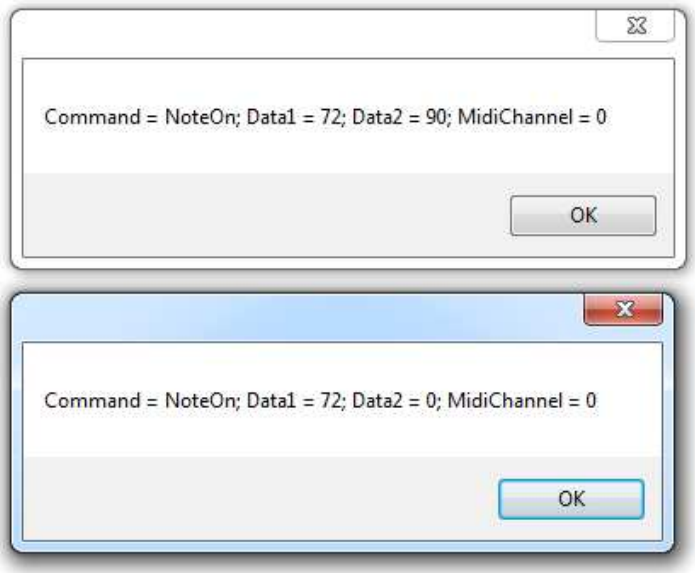
katerim smo to izmerili. V vsakem primeru v tem koraku nismo imeli veliko možnosti razhroščevanja, saj menjava nobenega od kosov ni prišla v poštev.

Izdelavo same igre smo zasnovali na že opisanem programu SequencerDemo. SequencerDemo služi kot temelj, na katerem naj bi razvili zastavljeno igro. Prva verzija SequencerDemo ni vsebovala nobene funkcionalnosti povezave z eksternimi napravami, četudi je knjižnica, katero naj bi SequencerDemo demonstriral, to funkcionalnost podpirala.

Prvi korak razvoja je torej bil ta, da programu dodamo sposobnost povezave z vhodno napravo. Naprava mora biti vhodna, ker bomo iz nje prejemali podatke o pritisnjenih tipkah.

Za vzpostavitev povezave z vhodno napravo, knjižnica predvideva kreacijo objekta tipa 'InputDevice', določitve metode, ki se izvede ob dogodku prejema novega sporočila in nazadnje klica metode StartRecording(). Klic te metode vzpostavi povezavo z napravo, ki je sedaj dodeljena izključno naši aplikaciji. To pomeni, da druge aplikacije ne morejo dostopati do te naprave. Povezava bo ostala vzpostavljena, dokler se ne izvede klic metode StopRecording(). Glede stanja aplikacije pa velja, da se aplikacija, katera ima vzpostavljeno povezavo z neko napravo, ne bo zaključila, dokler se povezava ne prekine. Posledično je pri programiranju zelo pomembno, da se dosledno zapira povezavo ob zaprtju obrazca, kateri vsebuje aktivno povezavo.

Uspešnost vzpostavitve povezave smo sprva preizkusili tako, da smo v funkcijo, ki se izvede ob prejemu MIDI sporočila iz vhodne naprave, dodal ukaz izpisa podatkov o sporočilu v obliki okna. Primer izpisa prikazuje slika 3.2.



Slika 3.2: Prikazana okna ob prejemu sporočila iz sintetizatorja

Šele v tem trenutku smo se zavedali, da pritisk na gumb sproži dva dogodka. Najprej se je pojavilo zgornje opozorilo. MIDI ukaz na njem nosi ukaz 'NoteOn', ki je v uporabljeni dokumentaciji MIDI [9] definiran kot dogodek, ki ustreza pritisku tipke. Parameter 'Data1' ustreza zaporedni številki pritisnjene tipke, v tem primeru je njena vrednost 72, kar ustreza tonu C6. Ob nekaj poizkusih smo ugotovil, da parameter 'Data2' nosi vrednost med 0 in 127, ki ustreza hitrosti pritiska, posledično glasnosti. 'MidiChannel' nosi vrednost 0, kar je bilo predvideno, saj povezane naprave komunicirajo preko prvega razpoložljivega MIDI kanala, ki je najpogosteje kanal z indeksom 0.

Drugo prejeto sporočilo tudi predstavlja ukaz 'NoteOn', četudi se pošlje v trenutku, ko je tipka spuščena. MIDI dokumentacija za ta dogodek predvideva sporočilo z ukazom 'NoteOff'. Sprva smo pregledali, da ne gre za napako v logiki knjižnice in ugotovil, da je na več mestih v kodi ukaz 'NoteOff' po pomenu enačen z ukazom 'NoeOn', ki ima parameter 'Data2' nastavljen na 0. Data2 predstavlja glasnost, torej sporočilo z ukazom NoteOn in glasnostjo 0 predstavlja konec predvajanja tiste note. Kljub zavajajočemu imenu drugega prejetega sporočila, je le-ta skladen z uporabljeno MIDI dokumentacijo in predstavlja dogodek spusta tipke. Parameter data1 nosi indeks note in njej ustrezne tipke, katera je bila spuščena.

V tem koraku smo razumeli pomen glavnih parametrov MIDI sporočila. V preteklih preizkusih smo ugotovili, da je program NoteWorthy Composer, med simultanim predvajanjem in snemanjem, trpel zakasnitev pri snemanju, ki znaša približno 0,1875 sekunde. Preizkušanje zakasnitve je postala prioriteta.

Izvedba merjenja zakasnitve je enostavna. Po ideji shranimo čas, ko naj bi sporočilo prišlo in čas, ko sporočilo dejansko prispe. Razlika med časoma ustreza zakasnitvi. Težava pa je, da moramo dejansko stisniti tipko na sintetizatorju, da se pošlje signal. Človeški refleksi pa niso dovolj hitri, da bi lahko ob točno določenem času stisnili tipko. Posledično smo merjenje zasnovali na principu ritma. Na aplikaciji naj bi se vsake pol sekunde, za trenutek pojavil tekst. Pojav teksta naznanja trenutek, ko naj bi sporočilo prišlo. Izpis uspešnosti pa se bo pojavil na dnu okna aplikacije.

Slika 3.3: Obrazec za določanje zakasnitve sporočila med sintetizatorjem in računalnikom

Rdeči »XX« je tekst, ki vsake pol sekunde utripne. Na levi in desni od rdečega teksta je sekvenca dvajsetih odebeljenih navpičnih črt (znak |). Ob začetku sinhronizacije sta ti dve sekvenci znakov prazni, rdeči »XX« pa je skrit. Tekom delovanja se sekvenca znakov dopolnjuje od zunaj proti sredini. Namen tega je bolje določanje trenutka, ko naj bi uporabnik pritisnil tipko na klavirju. Nova črta se izriše vsakih 25 milisekund, skupaj jih je 20, kar ustreza izrisu celotne sekvence vsakih 500 milisekund, torej 0,5 sekunde. Pričakovan čas prejema sporočila določimo ob trenutku, ko se pojavi rdeči »XX«, dejanski čas prejema pa se določi v metodi, ki jo sproži dogodek prejema sporočila.

S to programsko kodo smo določili realno zakasnitev med pritiskom gumba na sintetizatorju ter prejemu in obdelavi sporočila v aplikaciji. Realna zakasnitev pa žal ni zelo enakomerna. V večini primerov namreč znaša med 10 in 100 milisekund. Zakasnitev 10 milisekund je popolnoma sprejemljiva, a zakasnitve nad, ocenjeno, 20 milisekund, pa niso, saj je zakasnitev tako velika, da jo človeško uho zazna.

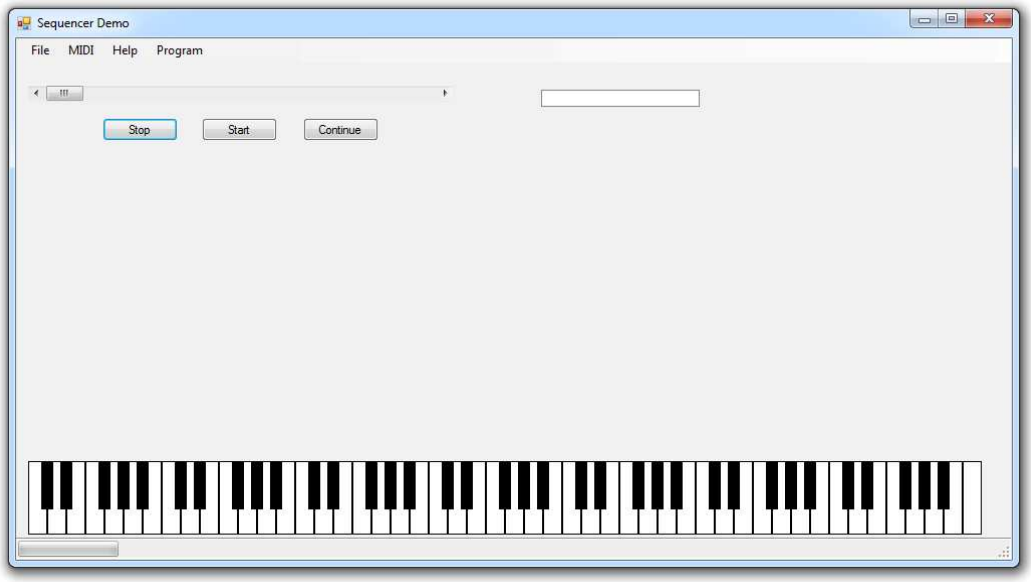
S pomočjo napisanega merilnika zakasnitve smo tako definirali zakasnitev, ki znaša do 100 milisekund. Če to primerjamo s prejšnjo izmerjeno vrednostjo iz programa NoteWorthy Composer, ki je znašala 185 milisekund, lahko ugotovimo, da je primerljiva. Program NoteWrothy Composer je nastavljen tako, da posnete signale zaokroži na določeno natančnost, zato je pričakovano, da bo efektivna zakasnitev še dodatna zaradi tega zaokroženja.

Po posvetu z mentorjem predvidevamo, da je za tako visoko zakasnitev kriv MIDI-to-USB kabel, ki ga uporabljamo. Kabel profesionalne kakovosti bi verjetno znižal zakasnitev. Za dodatni preizkus te trditve smo naložili program Synthesia [11]. To je program s podobno zasnovo kot igra, ki jo razvijamo, le da je komercialne kakovosti. Razvil ga je Synthesia LLC. Po naložitvi programa smo vanj vnesli MIDI datoteko, zagnali predvajanje in pritisnili tipko

na sintetizatorju. Program Synthesia nam ob prejemu signala iz MIDI sintetizatorja predvaja odmev tona, kateremu ustreza pritisnjena tipka. Zakasnitev med pritiskom in odmevom na računalniku je opazen. Kombinacija izmerjene zakasnitve v programu NoteWorthy Composer in zakasnitve, zaznane med uporabo programa Synthesia, ki sta oba komercialne kakovosti, nam je služila kot dokaz, da zakasnitev, izmerjena v naši igri, ni presenetljiva ali kot posledica slabe programske kode.

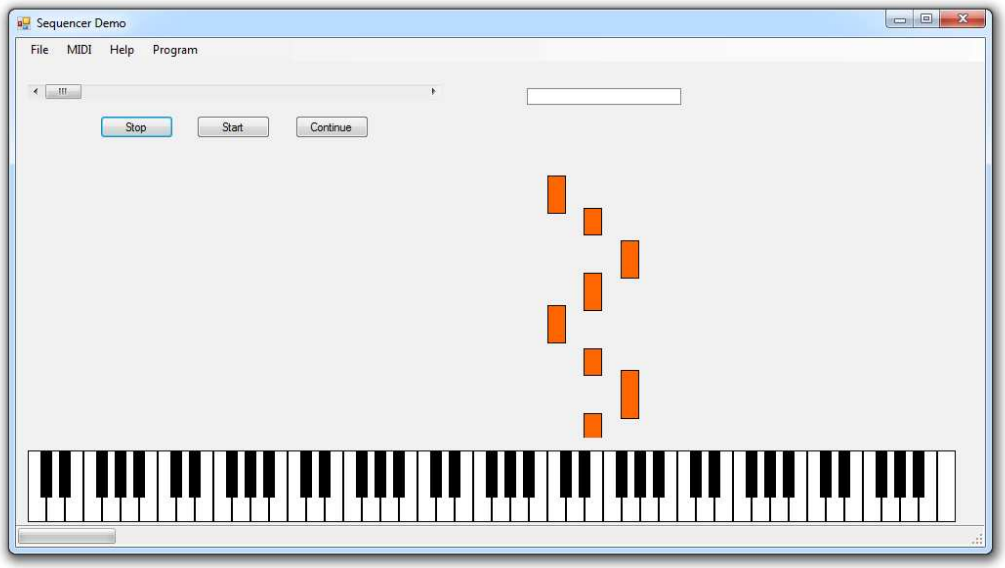
3.2 Grafični vmesnik

Po zaključku izdelave merilnika zakasnitve smo začeli z izdelave ogrodja dejanskega igralnega polja igre. Kot zasnovo smo uporabili okno, katerega nudi SequencerDemo, saj je bilo primerno za naše potrebe. Okno smo povečali, da je omogočalo rezervacijo prostora za potrebe pasov, po katerih se bodo gibali pravokotniki, ki bodo določali predvidene čase, ko naj bi uporabnik pritisnil tipke na sintetizatorju.



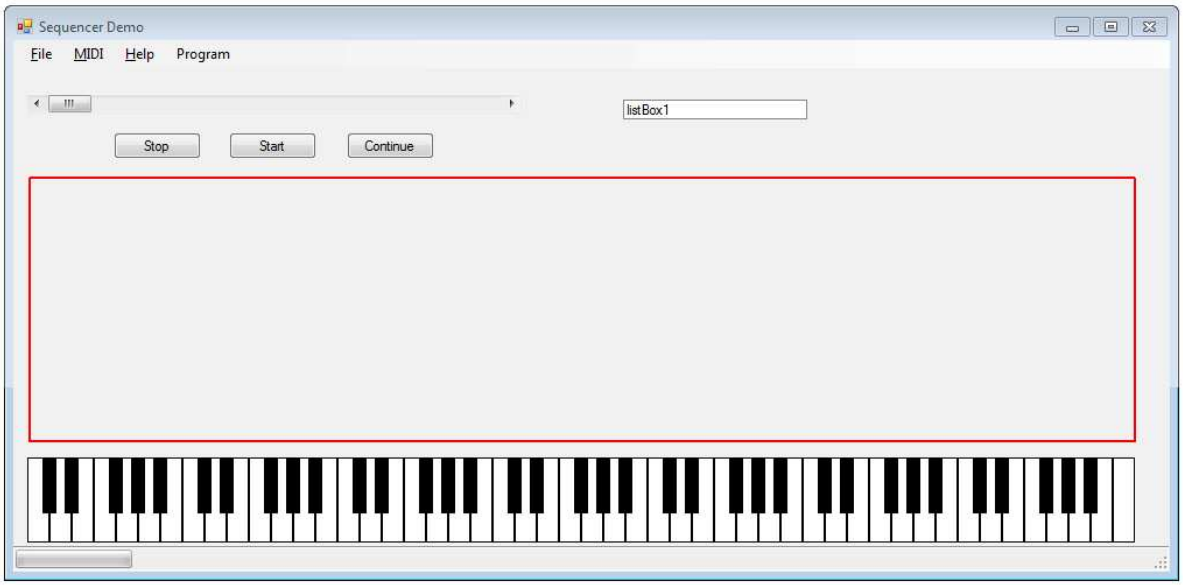
Slika 3.4: Grafični vmesnik aplikacije ob začetku razvoja

Prvo naslednje dejanje je bila izdelava funkcij, ki omogočajo gibanje pravokotnikov. V tem trenutku še nismo imeli funkcije, ki izvede zakasnitev med predvajanjem, zato so se pravokotniki pojavili istočasno kot se je MIDI zapis predvajal. Posledično pravokotniki niso predstavljali navodil za igranje, temveč sled že zaigranega. A ker je šlo za vmesni korak pri izdelavi funkcionalnosti, je to bilo sprejemljivo.



Slika 3.5: Grafični vmesnik aplikacije z izrisom pravokotnikov, ki predstavljajo sled igranega.

Tehnološka izvedba zastavljenega dela vključuje seznam tipa List z imenom 'active_rectangles', ki hrani vse aktivne pravokotnike, ki so tipa RectangleShape. Izris krmili časovnik, poimenovan 'shape_mover', ki se sproži vsakih 50 milisekund. Prva izvedba programa je ob utripu časovnika izvedla programsko kodo, ki je pogledala lokacijo vsakega od aktivnih pravokotnikov, to je pravokotnikov v seznamu 'active_rectangles'. Lokacija je sestavljena iz koodinat X in Y. Točka, kjer imata obe koordinate vrednost 0 je v zgornjem levem kotu tistega objekt, ki vsebuje objekt, kateremu preverjamo koordinate. V našem primeru gre za objekt tipa 'ShapeContainer', katerega velikost in pozicijo prikazuje slika 3.6.



Slika 3.6: Grafični vmesnik z označenim območjem, kjer se pojavijo pravokotniki

Točka s koordinatami X = 0 in Y = 0 se torej nahaja v zgornjem, levem kotu rdečega pravokotnika. Koordinate točke, ki se nahaja v spodnjem, desnem kotu ustrezajo parametrom objekta tipa ShapeContainer, katerega pravokotnik ponazarja. V tem kotu je koordinata X enaka širini objekta, koordinata Y pa je enaka višini objekta.

Iz tega lahko ugotovimo, da vrednost koordinate Y rase s premikanjem navzdol. Ker je predvideno, da se bodo pravokotniki premikali navzdol, bomo to dosegli s postopnim večanjem koordinate Y, ki definira lokacijo vsakega od aktivnih pravokotnikov.

Ker obstaja nevarnost, da se bo med predvajanjem pesmi nabrala ogromna količina takih pravokotnikov, jih moramo sproti brisati. Med procesiranjem spreminjanja lokacije, sproti gradimo tudi seznam pravokotnikov, ki so že prepotovali višino zaslona in se nahajajo pod zaslonom. Ob zaključku spreminjanja lokacije vseh aktivnih pravokotnikov, vse pravokotnike, ki se nahajajo v seznamu pravokotnikov, ki niso več vidni, odstranimo iz seznama aktivnih pravokotnikov in jih zbrišemo.

S tem je prva verzija procesiranja pravokotnikov zaključena, a za izvajanje procesiranja, moramo pravokotnike ustvariti in jih dodati na seznam aktivnih pravokotnikov. Prva izvedba je to storila z beleženjem časa, ko je prispelo sporočilo tipa 'NoteOn' do trenutka, ko prispe sporočilo tipa 'NoteOff' ali ekvivalenten 'NoteOn' z glasnostjo 0. V tem trenutku se iz razlike časov preračuna primerna višina pravokotnika, ki ustreza času pritiska

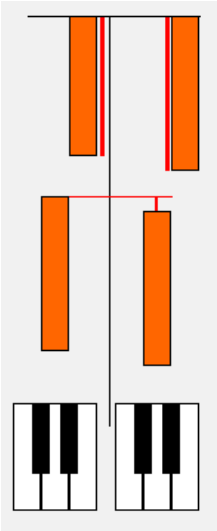
tipke. S pomočjo višine in indeksa tona se preračuna primerna začetna pozicija pravokotnika, ki se ga vstavi v seznam aktivnih pravokotnikov za procesiranje.

Ta verzija kode je delovala, a je še vedno ustvarjala pravokotnike ob prejetemu signala, bodisi iz sintetizatorja ali MIDI datoteke. Posledično pravokotniki predstavljajo sled igranega in ne navodila za igranje. Poleg tega so se pravokotniki ustvarjali ob zaključku tona, torej so se v trenutku spusta tipke nenadoma ustvarili v polni velikosti. Niso se ustvarili ob pritisku note in rasli do spusta, nakar bi se začeli premikati. Implementacija slednjega je nato postala prioriteta.

V tem trenutku smo imeli seznam aktivnih pravokotnikov ter objekt tipa ShapeContainer na grafičnem vmesniku. Programska koda je bila napisana tako, da smo v seznam aktivnih elementov dodali pravokotnik istočasno kot smo ga dodali v objekt tipa ShapeContainer, kar je potrebno za prikaz na grafičnem vmesniku. Prav tako smo ga iz seznama aktivnih elementov odstranili istočasno kot smo ga odstranili iz seznama objekta tipa ShapeContainer. Iz tega smo ugotovili, da je seznam aktivnih elementov nepotreben, saj je vedno enak seznamu pravokotnikov, ki jih objekt tipa ShapeContainer vsebuje.

Pomen, katerega smo do sedaj pripisali seznamu aktivnih elementov smo v tem koraku zamenjali. Pomen pravokotnikov v tem seznamu je od tedaj predstavljal vse pravokotnike, ki šele rasejo. Torej pravokotnike, ki predstavljajo tone, kateri so se že začeli, a se še niso zaključili. Procesiranje teh tonov je namreč drugačno od ostalih. Medtem ko moramo že končanim tonom spreminjati koordinato Y, moramo pravokotnikom, katerih ton še ni zaključen, spreminjati višino, saj šele rasejo.

Kodo dodajanja pravokotnikov v seznam smo spremenili, da ni več dodajala pravokotnika ob prejemu sporočila o zaključenem tonu, marveč ob prejemu sporočila o pritisku tipke. V tem trenutku smo pravokotnik dodali na seznam aktivnih pravokotnikov z višino enako 5 pik. Ob vsaki iteraciji programske kode, katera skrbi za premike pravokotnikov, smo spremenili to, da je sedaj vsem pravokotnikom, ki se nahajajo v seznamu aktivnih pravokotnikov, spremenila višino, medtem ko vsem, ki se v seznamu ne nahajajo, spreminjamo pozicijo.



Slika 3.7: Izsek grafičnega vmesnika ob dveh zaporednih iteracijah programa. Zgornji pravokotnik predstavlja ton, ki se že ni zaključil, spodnji pa ton, ki se je že zaključil. Pravokotniki se gibajo navzdol.

Slika 3.7 prikazuje izsek okna ob nekem času, ter ob naslednji iteraciji. Smer gibanja pravokotnikov je navzdol. Ton D, na izrisani klaviaturi srednji, je že zaključen. Zato se v drugi sliki nahaja za neko število pik nižje, a ohranja svojo višino. V programu se ta ton ne nahaja v seznamu aktivnih tonov, saj je že zaključen. Ton E, na izrisanem delu klaviature tretji, pa se še vedno izvaja, torej še ni zaključen. Ker še ne vemo točne dolžine tona, v naslednji iteraciji samo povečamo njegovo višino, ki predstavlja trajanje tona. V programu se pravokotnik nahaja v seznamu aktivnih pravokotnikov.

S funkcionalnostjo izrisa sledi predvajanih tonov smo lahko v programu demonstrirali potek igranja, tako da smo bodisi predvajali neko MIDI datoteko ali priključili sintetizator in nanj zaigrali neko melodijo.

Med preizkušanjem programa smo tako opazili eno ključno težavo, ki je že dalj časa med razvojem postajala očitna. Igranje take igre je težko. Že enostavne melodije, kot na primer melodija pesmi 'Abraham ima sedem sinov', izriše veliko število pravokotnikov, katerim je sprva vsakemu težko slediti, začetnikom pa ne postane enostavno niti po nekaj poizkusih. Pri nekoliko bolj napredni pesmi, ki uporablja istočasni pritisk dveh tipk, je ravno tako težko slediti zapisu tudi igralcu, ki ni začetnik na klavirju. Kompleksne pesmi, ki zahtevajo obširno uporabo obeh rok pa sploh ni smiselno igrati, saj človek preprosto ne more slediti dogajanju na zaslonu in pravočasno prevajati navodila iz zaslona na ločeno napravo - sintetizator.

Težava prevajanja navodil, ki jih igralec vidi na zaslonu na sintetizator, je problem, ki bi odgnal ogromno igralcev zelo kmalu. Veliko igralcev, ki začnejo igrati igro s predstavo, da je že začetniško igranje na klavir težko, bi od izkušnje prišli le s potrjenim mnenjem, četudi za kompleksnost igranja nebi bil kriv klavir, marveč igra.

Če vzamemo pod drobnogled mehaniko igranja pri igri 'Guitar Hero' in jo primerjamo z mehaniko igranja pri naši igri, lahko opazimo nekaj očitnih podobnosti. Pri obeh obstajajo neki elementi, ki se premikajo po časovnem traku in ko prispejo na neko točko, moramo storiti neko dejanje, da inštrument zaigra. A to dejanje je na sintetizatorju izrazito različno kot pri 'Guitar Hero' vmesniku.

V primeru igre 'Guitar Hero' mora igralec ob trenutku, ki ga določajo oznake na časovnem pasu, imeti pritisnjene tipke, ki jih oznake določajo in udariti po stikalu, ki predstavlja brenkalo. Desničar bo torej z levo roko krmilil pritiske na gumbe, ki so na vratu vmesnika, z desno roko pa stikalo, ki predstavlja brenkanje.

Ta ločitev dejanj pa je globlja, namreč ločuje izvajanje melodije na eno roko in izvajanje ritma pa na drugo. Leva roka krmili tipke na vratu vmesnika. Ta roka torej krmili melodijo, saj ni pomembno kdaj tipke na vratu spustimo in kdaj jih pritisnemo, pomembno je le, da so stisnjene ob trenutku, ki ga določajo oznake na časovnem pasu. Z desno roko pa krmilimo samo ritem, saj z njo zgolj udarjamo po stikalu, ki predstavlja brenkalo ob tistih trenutkih, ki jih določajo oznake na časovnem pasu.

Minimalna zahteva za igranje klavirja ali sintetizatorja pa je igranje enoglasne melodije, brez kakršnekoli spremljave. Za igranje take melodije moramo uporabljati eno roko, s katero istočasno izvajamo ritem in melodijo. To dejanje igranja mnogo bolj oteži začetniku.

Naslednji problem, ki je vsekakor pomemben, se nanaša na naravo igranja klavirja v primerjavi z igranem na kitaro, posebej pesmi in zaporedja v igri 'Guitar Hero'. Večino pesmi v igri 'Guitar Hero' sestavljajo deli, ki se ponavljajo. V večini pesmi, ki jih Guitar Hero vključuje, je kitara inštrument, s katerim spremljamo druge. Pesmi, za katere to ne drži, pa se pojavijo šele proti koncu igre, ko je pričakovano, da bo igralec že vešč igranja. Tudi v teh pesmih so deli, kjer kitara ne spremlja le ločeni odseki, ko igralec kitare prevzame vodilno vlogo igralca melodije pesmi.

Značaj igranja na klavir pa je drugačen. Klavir je zelo vsestranski inštrument, ki ga uporabljajo v veliko žanrih glasbe. Vloga klavirja ali sintetizatorja je najbolj podobna spremljevalni vlogi kitare v nekaterih žanrih elektronske glasbe. Le-ti uporabljajo sintetizator kot spremljevalni inštrument, kateri ima pogosto zelo ponavljajočo melodijo.

Drugi primeri uporabe klavirja so za enostavno spremljavo, najpogosteje zborovske zasedbe. Pesmarice pogosto, poleg besedila, vsebujejo nad besedilom zapisane oznake. Te oznake so namenjene predvsem igralcem kitare, klavirja ali drugih inštrumentov, da jim povejo tonaliteto, to je tonsko višino, na kateri morejo zbor spremljati. Z vajo pianist ali kitarist pridobi sposobnost, da mu ta podatek zadostuje za uspešno spremljanje zbora. Melodijo, s katero bo zbor spremljal - torej konkretne tipke, ki jih bo med igranjem pritiskal - pa si mora pianist spolnit sproti.

Igra bi lahko prišla prav za učenje zgoraj omenjenih primerov, a pianisti, ki so na nivoju, da se želijo učiti avtomatičnega spremljanja, take igre preprosto ne potrebujejo več, saj so se sposobni te sposobnosti učiti sami. Funkcionalnost, ki bi jim lahko prišla prav je funkcionalnost branja oznak v neki pesmi. Žal pa MIDI ne vsebuje že pripravljenega podatka o primerni tonaliteti spremljave, določanje le-te programsko pa je zelo nezanesljivo. Poleg tega pa je osnova igre uporaba formule iz igre 'Guitar Hero', ki pa ne podpira ničesar podobnega.

Od naštetih primerov bi bila igra bolj primerna za učenje spremljanja elektronski glasbi in drugi glasbi, kjer klavir pogosto ponavlja neko zaporedja tonov dalj časa. Žal pa je take glasbe malo, poleg tega pa s tem strežemo zelo majhnemu segmentu ljudi, ki bi se radi naučili igranja na klavir.

Tretji pogost način igranja klavirja je spremljanje po notnem zapisu. V tem primeru je avtor notnega zapisa poskrbel, da ima klavir točno določeno spremljavo, ki jo definira notni zapis. V takem primeru naletimo na isto težavo kompleksnosti prevajanja padajočih pravokotnikov v note, kot v četrtem pogostem načinu igranja klavirja, ki je igranje melodije ali vodilne vloge.

Iz pogovorov s posamezniki, ki so začeli igrati klavir, lahko sklepamo, da ima velik vpliv na zanimanje za igranje inštrumentov popularna glasba. Večina ljudi ima ob začetku učenja v mislih neko skladbo ali pesem, katero bi se radi naučili. Ta skladba, vsaj v začetnih korakih, predstavlja nek cilj. Uspešno igranje te skladbe pa mejnik in uspeh. Med učenjem in vajo za doseg tega cilja, pa se igralec nauči veliko o igranju klavirja in si s tem pridobi prakso in vajo. Če igralcu uspe doseči cilj uspešnega igranja pesmi, je bolj verjetno, da bo kasneje ponovno poskusil še z drugo. Skupna lastnost teh pesmi pa je, da bi igralec rad, da bi zvenela čimbolj podobna originalni izvedbi. Igranje izključno spremljave pa tega ne bo doseglo. Od učenja želi uporabnik odnesti sposobnost, da lahko naučeno pesem spodobno zaigra tudi nekje, kjer ta igra ni dostopna. Drugače povedano, uporabniku pogosto ne bo zadostovalo, da

zna zaigrati spremljavo neki pesmi, ki jo igra računalnik. Rad bi, da brez pomoči računalnika zaigra nekaj, kar bodo poslušalci prepoznali.

Na področju učenja igranja na klavir lahko ta igra izpolni le zelo majhno vlogo. Ker je želja večine igralcev igranje melodije, morda tudi spremljave na melodijo, postane igranje igre zelo zahtevno. Že enostavne melodije bodo hitro preplavile igralčev zaslon s pravokotniki, katerim igralec ne bo mogel slediti. Vlogo, ki jo pa lahko zapolni, je prikaz zaporedja igranja posameznikom, ki ne znajo brati notnega zapisa.

Žal je v praksi učenje prevajanja pravokotnikov iz zaslona v pritiske na klaviaturi sintetizatorja dolgoročno težje in bolj naporno, kot učenje notnega zapisa. Četudi bo nekdo, ki notnega zapisa ne pozna hitreje razumel pravokotnike. Njihova krivulja učenja je namreč bolj položna na začetku, zato je ta metoda bolj privlačna novincem, a kakor je bilo že obdelano, kompleksnost branja zapisa s pravokotniki narašča eksponentno s številom pravokotnikov na zaslonu. Že pri enostavnih pesmih, kot je 'Abraham ima sedem sinov', je enostavnejše branje iz notnega zapisa, kot sledenje pravokotnikom. V primeru, da se nekdo nauči hitrega prevajanja med pravokotniki na zaslonu in klaviaturo na sintetizatorju, kar je po daljšem obdobju vaje gotovo možno, pa je omembe vredna še nesmiselnost te sposobnosti. Spomnimo se, da je primarni namen igre učenje igranja na klavir. Ta sposobnost pa nas, kot igralca, ne privede bližje temu cilju.

Med opisovanjem igranja avtomatične spremljave zboru, je bila omenjena tonaliteta. To je izraz, ki definira kateri ton predstavlja osnovo igranja, v glasbeni terminologiji toniko. Zaporedni toni na klaviaturi so med seboj odmaknjeni po pol tona. Le-ti so našteti v tabeli tonov, ki jih podpira format MIDI [9].

Neko melodijo lahko zaigramo tako, da si za osnovni ton ali tonaliteto, izberemo katerikoli ton na klaviaturi. Če nato za pol tona zvišamo vsak ton v zaporedju, ki tvori melodijo, bo predvajana melodija zvenela popolnoma enako, le nekoliko višje. V glasbeni terminologiji se temu reče modulacija. To početje je popolnoma normalen postopek, ki ga definira jasen algoritem. Krivulja kompleksnosti igranja pa ni enakomerna na vseh izbranih tonalitetah.

Najpogosteje se kot najosnovnejša tonaliteta pojavi C dur. Le-ta ima to lastnost, da za njegovo igranje ne potrebujemo nobene črne tipke na klavirju. V splošnem je torej najenostavnejše igrati melodije v C duru, saj ni potrebno skrbeti za pravilno uporabo črnih tipk. To velja posebej takrat, ko se igralec uči avtomatskega spremljanja, saj takrat note niso zapisane, ampak si jih igralec sproti izmišljuje.

Za razliko od tonalitete C dur, za Cis dur ali C# dur velja popolno obratno. Cis dur ima vseh sedem obstoječih višajev, in zaradi tega vključuje vseh pet črnih tipk, ki se pojavijo v vsaki oktavi. Posledično je igranje iste melodije v Cis duru mnogo težje od igranja v C duru, četudi je razlika med njima le izpeljan matematični algoritem.

Četudi lahko algoritem izvajamo tudi v obratno smer, s čimer bi lahko spreminjali tonaliteto vsake vnesene skladbe na primerno, enostavnejšo lestvico, to pogosto ni mogoče. Razlog pa je v človeških sposobnostih, specifično glasilkah. Pogosto zvišanje neke pesmi za pol tona pomeni, da pevci, ki so razvrščeni v glasove z definiranimi razponi, ne bodo mogli zapeti najvišjih tonov v pesmi. Enako velja za nižanje tonalitete. Posledično je pogosto potrebno, da spremljevalec, zaradi fizioloških lastnosti človeka, obdrži podano tonaliteto, četudi ni enostavna za igranje.

Pomen tonalitete je v igri 'Guitar Hero' trivialen. Tipk na vratu vmesnika je le pet, kar zelo omeji zmogljivosti prikaza igranja. V več pesmih so zaporedni trozvoki, ki so različnih tonalitet, predstavljeni z minimalno spremenjenimi, ali celo identičnimi prijemi tipk na vratu vmesnika. V glasbi je trozvok kombinacija treh tonov, ki so med seboj oddaljeni ton in pol ali dva tona.

Taka trivializacija spremembe tonalitete in vpliva na igranje je morda sprejemljiva za kitaro, ki ob vsakem udarcu po strunah ne proizvede le enega tona, marveč več tonov, v odvisnosti od tega, koliko strun zabrenkamo.

Pri igranju na klavir pa obstaja le malo načinov za poenostavljanje prijemov ali lažje razumevanje prijemov. Skratka taka trivializacija ne pride v poštev. Prijemi morajo biti točni, kakor jih definira notni zapis ali zapis s pravokotniki. Potreba po doslednem upoštevanju tonalitete in pogosta prepoved spreminjanja tonalitete je med razlogi, ki dodatno otežujejo igranje na klavir in posledično igranje naše igre za zabavo.

Ob navedenih ugotovitvah smo sprejeli, da formula 'Guitar Hero' ni primerna za uporabo v igri, katere namen je učenje klavirja, a se poskuša predstaviti kot igra, ki je zanimiva sama po sebi. V konceptu uporabe formule za učenje klavirja smo še videli potencial, a poskus prikaza razvite aplikacije kot igre, se nam je zdel nesmiseln. Koncept igre smo v tem koraku opustili in začeli na novo z razvojem aplikacije, katere jasen namen je učenje klavirja. Ciljna publika od tega trenutka niso več posamezniki, ki bi radi igrali igro, marveč posamezniki, kateri aplikacijo uporabljajo za izraziti namen učenja klavirja.

Ob zaključku tega koraka smo imeli aplikacijo, ki je bila sposobna meriti in prikazati zakasnitev v komunikaciji med sintetizatorjem in aplikacijo. Aplikacija je bila sposobna prikazati na klaviaturi pritisnjene tipke v obliki padajočih pravokotnikov, katerih namen je bil

demonstracija delovanja končne aplikacije. Poleg tega je bila sposobna predvajati MIDI datoteke in enako prikazovati padajoče pravokotnike.

Poglavje 4

Razvoj aplikacije za učenje klavirja

Razvoj igre smo opustili, a v formuli še vedno videli potencial, le namen aplikacije smo spremenili. Namesto igre, smo od tedaj razvijali aplikacijo, katere primarni namen je učenje igranja na klavir. Uporabnik bo k aplikaciji pristopil s primarnim namenom učenja in ne več igranja ali zabave.

Področje računalništva se pogosto sreča s področjem glasbe. Računalniki in aplikacije so zelo poenostavili izdelavo glasbe. Velik prispevek k temu je sposobnost pisanja glasbe s predogledom končnega izdelka. To sposobnost najpogosteje nudi MIDI.

MIDI ali 'Musical Instrument Digital Interface', kar prevedeno pomeni Digitalni vmesnik glasbenih inštrumentov, je razširjena specifikacija, ki omogoča sintetiziranje glasbenih inštrumentov s pomočjo primerne strojne ali programske opreme. Sprva so računalniki, ki so imeli vstavljeno MIDI kartico, podpirali istočasno predvajanje le nekaj tonov, a danes lahko preko MIDI naprav predvajamo skoraj poljubno mnogo tonov hkrati.

Uporaba MIDI standarda je danes razširjena na mnogo naprav, ki vključujejo naprave od glasbenih voščilnic do profesionalnih sintetizatorjev. Aplikacija, katero razvijamo, za vmesnik predvideva MIDI sintetizator, ki je najpogostejša oblika sintetizatorja na trgu.

Uporaba računalništva za poenostavljanje postopka razvoja glasbe pa vključuje tudi razvoj na področju obdelave glasbenih posnetkov. Medtem ko MIDI proizvaja sintetizirane zvoke, katere opisujejo matematične formule in algoritmi, klasični inštrumenti te zvoke proizvajajo s pomočjo fizikalnih pojavov, kot so vibriranje strun in membran.

Sintetizirane zvoke ustvarijo matematične formule, posledično so vedno popolnoma enaki. Pri uporabi klasičnih inštrumentov pa lahko s pomočjo tehnike igranja ter manjših ali večjih sprememb v tej tehniki, dobimo mnogo več nadzora nad ustvarjenimi toni. S tem lahko bolje nadzorujemo barvo tona in druge, pogosto subjektivne in nemerljive lastnosti tona. Ker

nam igranje klasičnih inštrumentov omogoča toliko več nadzora, kot sintetiziranje teh istih inštrumentov, je razvoj na področju obdelave posnetih, torej ne-sintetiziranih, zvokov ravno tako pomemben element na interdisciplinarnem področju računalništva in glasbe.

Na trgu aplikacij, katerih cilj je omogočanje izboljšanja posnetkov, obstaja veliko aplikacij, ki služijo vsem segmentov trga, od začetnikov do vodilnih v industriji.

Ker je igranje klasičnega inštrumenta, v večini primerov, bolj zaželeno, kot uporabo sintetizatorja, je na mestu vprašanje zakaj se nismo odločili za izdelavo aplikacije, ki kot vir podatkov za analizo kakovosti igranja, uporablja klavir in ne sintetizator. Večkrat smo namreč omenili, da je namen aplikacije ta, da uporabnika nauči igrati klavir, in ne le sintetizator.

Najočitnejši problem pri uporabi klasičnega klavirja z računalniško aplikacijo je ta, da klavir ne uporablja elektronskega ustvarjanja in pošiljanja signalov ali sporočil, katere bi lahko brali in analizirali. Klavir ustvarja zvok tako, da ob pritisku tipke, s kladivcem udari po struni, ki se nahaja znotraj ohišja. Delovanje je torej popolnoma mehanično.

Možno pa je signal posneti in analizirati posnetek. Iz posnetka se da, preko določenih algoritmov, kot so Fouriereva transformacija, določiti kateri ton je bil pritisnjen. S pomočjo analize amplitude, ki ustreza glasnosti, se da določiti tudi ob katerem trenutku je bila tipka pritisnjena.

Težava nastane, ko želimo analizirati hkratno igranje več tonov. Aplikacije, ki so danes na trgu, so večinoma sposobne analiziranja tonov in prevajanja le-teh v zapis podoben MIDI, katerega lahko analiziramo, a le če se igra po en ton hkrati.

Omejitev igranja le enega tona hkrati bi pomenila, da bi lahko aplikacija služila zelo malo časa v procesu učenja igranja na klavir. Iz tega razloga je bolj smiselna uporaba sintetizatorja kot vmesnika za igranje. Četudi taka implementacija ni popolna, je boljši kompromis, kot analiza zvočnega posnetka.

Poizkusi implementacije računalniške pomoči za učenje inštrumentov niso nič novega. Žal pa večina implementacij spodleti, saj lahko služijo v zelo omejenem obsegu. Pogosto je uspešnost aplikacije, katere namen je pomoč pri učenju igranja na inštrument, bolj pripisati oglaševanju in trženju kot uporabnosti aplikacije. Ključni cilj izdelave te aplikacije je, da bo uporabna v postopku učenja.

Sodobni sintetizatorji vstopne kakovosti pogosto nudijo funkcije, katerih namen je pomoč pri učenju igranja. Sintetizatorji pogosto vključujejo zbirko pesmi, katere se lahko igralec nauči igrati.

Pogost način učenja na samem sintetizatorju je s prikazom tona, ki naj bi ga uporabnik pritisnil, na zaslonu. Sintetizator počaka, dokler igralec ne pritisne te tipke in nato nadaljuje z melodijo, nakar počaka pri naslednji noti, ki naj bi jo igralec zaigral.

Na omejenem zaslonu, ki ga ima sintetizator na voljo je taka implementacija pomoči pri učenju verjetno najboljša možna, a naleti na nekaj ključnih problemov. Prvi problem je zahteva, da se uporabnik nauči branje iz notnega črtovja. Kot smo že opisali ima branje strmejšo krivuljo učenja kot prikaz pravokotnikov, ki jih uporablja naša aplikacija. Je pa res, da je učenje notnega črtovja dolgoročno koristno, celo potrebno. Ni pa nujno potreben prvi korak pri učenju. Še vedno namreč predvidevamo, da bo uporabnik najraje igral skladbe, ki so mu všeč in da ga bodo le-te motivirale za nadaljevanje in vajo. Primerno je, da uporabnika najprej naučimo igrati in šele nato uvedemo v učenje branja notnega zapisa in teorijo glasbe.

Poleg težave učenja notnega zapisa, katero lahko uporabnik preseže s tehniko poizkusov in napak, pa je na omejenem zaslonu, katerega sintetizatorji nudijo, težava to, da igralec ne more videti not, ki prihajajo. Igralec lahko torej vidi le noto, katero mora trenutno igrati. Ker se nota prikaže šele takrat, ko sintetizator pričakuje, da bo prikazana nota zaigrana, bo ob času, ko igralec noto iz zaslona prebere, že pozen z igranjem prebrane note. Ob prvih branjih not se bo igralec predvidoma ustavil pri vsaki noti in si jo poskušal zapomniti. To je pričakovano, a uči slabo navado - branje le trenutne note.

Glasba, ki je zapisana na papirju, v notnem zapisu, je vidna v celoti. Igralcu torej ni potrebno gledati trenutno igrane note dlje, kot potrebuje, da jo prebere. Posledično je pričakovano, da bo bral note, ki sledijo, da jih bo imel prebrane še preden jih bo moral zaigrati. Poleg tega mu tak način branja omogoča, da bo imel čas prebrati kompleksnejšo zaporedje, preden ga bo moral zaigrati. V računalništvu implementacijo takega načina branja pogosto imenujemo medpomnjenje. Ker pa način učenja prikazuje le noto, katero naj bi igralec trenutno igral, med pavzo pa ne prikazuje ničesar, se takšnega načina branja ne moramo naučiti, namesto tega se navadimo na branje le trenutno igrane note. Te navade pa se je pogosto težko odvaditi.

Opisano predstavlja težavo uporabniku, ki je že nekajkrat melodijo prebral in si jo je delno že zapomnil, ni je pa še sposoben zaigrati v celoti, brez pomoči notnega zapisa. Ko se pa uporabnik bliža zaključku učenja skladbe, ko je sposoben pesem zaigrati na pamet, a se občasno še moti v pritisku tipk, pa postane tak način učenja moteč. Sintetizator namreč ne bo nadaljeval z melodijo v primeru izpuščenega tona, dokler se igralec ne ustavi in nadaljuje iz točke, v kateri je ton izpustil. Zadeva, ki dodatno omejuje uporabnost takega sistema v vseh korakih učenja pesmi pa je ta, da bo sintetizator ton registriral samo od točke, ko naj bi se ton

pojavil dalje. Ton, katerega igralec pritisne prezgodaj se torej ne bo registriral in igranje bo ostalo na tisti točki.

Opisana metoda je poskus implementacije sistema, katerega namen je pomoč pri učenju igranja na sintetizator, a slabosti tega sistema vsekakor dopuščajo za izboljšave, katere lahko implementiramo z uporabo računalnika.

Osebni računalnik je naprava, ki je hitrejša in bolj vsestranska kot integrirano vezje, ki se nahaja v sintetizatorju. Poleg tega vključuje večji zaslon z večjim razpoložljivim številom barv, kot sintetizatorji vstopne kakovosti. Te sposobnosti moramo izkoristiti, saj mora aplikacija biti bolj uporabna kot opisan sistem za učenje, katerega že vsebuje sintetizator. Šele takrat jo bomo lahko označili kot uspešno. Hkrati pa smernica razvoja še vedno predvideva tesno vključitev formule 'Guitar Hero'.

Konceptualni cilj razvoja je v tem koraku razvoj aplikacije, katere namen je pomoč učenja klavirja. Uporabnik k aplikaciji pristopi z namenom učenja, aplikacija pa mu nudi pomoč na tej poti. Za upravljanje, aplikacija ne potrebuje napredne uporabe in po ideji nadomesti učitelja igranja na klavir.

Aplikacija, katero smo reciklirali iz igre, katero smo izdelovali v prejšnjem koraku, vključuje rešitve za določene probleme, kateri so lastni metodi učenja, ki jo nudi sintetizator.

Težavo prikaza le trenutne note reši trak, po katerem se gibajo pravokotniki. Metoda prikaza zaporedja igranja s pravokotniki pravzaprav ne dovoljuje branja le pravokotnikov na dnu, saj se pas nenehno giblje, posledično igralec mora brati pravokotnike še preden pridejo do dna, da jih lahko pravočasno prebere in zaigra. To sicer igralca ne navadi na pravilni način branja not, saj se bo branja le-teh moral naučiti kasneje, a mu za razliko od metode, ki jo uporablja sintetizator, ne ustvari slabe navade.

Težave prevajanja prebranega na zaslonu s tipkami na sintetizatorju, s katero smo se srečali že v koraku razvoja igre, sprememba konceptualnega načrta za aplikacijo ne odstrani, jo pa omili. En od konceptualnih ciljev je med razvojem igre bil, da bo uporabnik k igri prihajal z namenom zabavanja. To je pa težko doseči, če je uvodna krivulja učenja strma. Uporabnik se bo v tem primeru igre kmalu naveličal, saj ga ne bo privlekla z začetnimi uspehi. Edinstvenost aplikacije pa bo kmalu pojenjala, pogosto preden bo uporabnik pridobil sposobnost prevajanja pravokotnikov v pritiske tipk na sintetizatorju.

K aplikaciji, kakršno razvijamo v tem koraku, pa uporabnik ne pristopi z namenom zabave, marveč z namenom, da od izkušnje odnese sposobnost igranja klavirja. Od uporabnika se torej predvideva, da bo motiviran za učenje, kar pomeni, da se bo k aplikaciji vračal dalj časa, kot uporabnik, ki od aplikacije primarno želi zabavo. Ta začetna motivacija

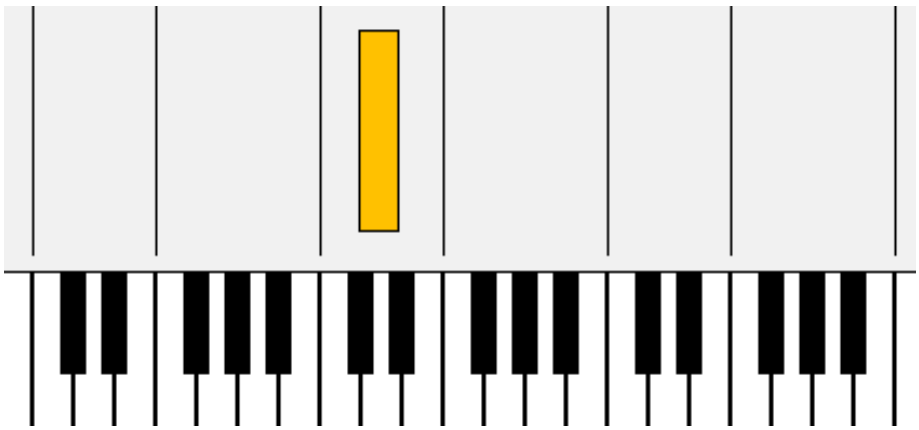
omogoča, da bo večje število uporabnikov pridobilo sposobnost prevajanja pravokotnikov v pritiske tipk, v času, dokler motivacija ne pojenja.

Kljub daljšem uvodnem času, pa moramo poskrbeti, da bo prevajanje pravokotnikov v pritiske tipk, čimbolj enostavno. Kot zgled za eno od izboljšav smo uporabili igro Synthesia. To je igra, komercialne kakovosti, ki je po zasnovi podobna igri, ki smo jo razvijali in analizirali v prvem koraku. Med lastnostmi igre pa so tudi vse slabosti, s katerimi smo se srečali med našim razvojem. Od omenjene igre smo vzeli idejo za razdelitev pasu, po katerem se gibljejo pravokotniki na manjše pasove, da lahko že na vrhu pasu identificiramo, kateremu tonu določen pravokotnik pripada. Ti manjši pasovi izkoriščajo lastnost klaviature, ki sama izkorišča lastnost oktave.

Oktava je razmak med toni, ki ustreza dvakratni povečavi frekvence tona. V splošni glasbeni notaciji, oktavo predstavlja številka, zapisana ob imenu tona. Ton A4 je ton A v četrti oktavi, ton A5 je ton A v peti oktavi. Frekvenca tona A5 je dvakrat višja od frekvence tona A4. Ena oktava vsebuje 12 tonov, če ne vključujemo ponovitve prvega tona. Če številko oktave zamenjamo s spremenljivko n, splošni zapis zaporedje določene oktave vsebuje tone Cn, C#n, Dn, D#n, En, Fn, F#n, Gn, G#n, An, A#n in Hn. Tonu Hn nato sledi ton Cn+1. Znak # predstavlja višaj, ki določeno noto zviša za pol tona. Na klaviaturi so to črne tipke. Ton C#n se torej nahaja med tonoma Cn in Dn. Lastnost, katero je potrebno izpostaviti določuje pomanjkanje imen dveh tonov. Med tonoma En in Fn ni tona E#n, poleg tega med tonoma Hn in Cn+1 ni tona H#n. Na teh dveh mestih, sta na klaviaturi dve beli tipki skupaj, brez da bi med njima bila črna tipka. Dodaten pomen pa določa neenako število črnih tipk. Med tonoma Cn in En sta natančno dve črni tipki, med tonoma En in Cn pa natančno 3 črne tipke.

Klaviatura izkorišča to lastnost z namenom, da lahko prepoznamo katerokoli ton na njej, ne da štejemo tone od začetka ali da potrebujemo dodatne oznake.

S človeško sposobnosti abstrakcije lahko tako določimo kateremu tonu pripada določen pravokotnik, če ga izrišemo na pas, kateri je razdeljen na neenakomerne pasove.



Slika 4.1: Grafični vmesnik z razdeljenim poljem, po katerim se gibajo pravokotniki. Delitev v skupine tipk.

Ob razumevanju logike razporeditve pasov lahko identificiramo kateremu tonu pripadajo tudi brez izrisane tipkovnice na dnu. To nam omogoča, da lahko ton identificiramo že na vrhu okna, preden se približa dnu. S tem nekoliko olajšamo postopek učenja prevajanja pravokotnikov na pritiske tipk, a postopek ostaja zahteven.

Drugi način, kako lahko poenostavimo prevajanje pravokotnikov v pritiske tipk, je ta, da poleg razdelitve po tipkah, razdelimo na pasove tudi čas. Ritem je namreč pomemben del glasbe in s pomočjo ritma lahko bolje definiramo trenutek, ko naj bi pritisnili tipko in koliko časa jo moramo držati.

V veliki večini skladb je ritem enakomeren. Ena doba je obdobje, ki jo definira oznaka na začetku notnega zapisa. Do sedaj smo s pomočjo te oznake računali zakasnitev prenosa sporočila med sintetizatorjem in računalnikom. Eno dobo definira številka, ki predstavlja število dob v eni minuti. Pomembna lastnost dobe je enakomernost. Toni, ki sestavljajo skladbo, pa so definirani glede na obdobje štirih dob. Celinka, najdaljša pogosta nota, predstavlja noto, ki traja štiri dobe; polovinka traja pol manj, torej dve dobi; četrtinka traja četrtino celinke, torej eno dobo. Obstajajo tudi osminke, šestnajstinke in krajše note, ki sledijo enakim pravilom, ki jim določajo trajanje. Note lahko poljubno združujemo, s čimer ustvarimo tone poljubnih dolžin. Če združimo eno četrtinko, ki traja eno dobo z eno osminko, ki traja pol dobe, dobimo neprekinjeno noto, ki traja eno dobo in pol.

Zaradi te logične razdelitve trajanja, ki so tako tesno povezane z definirano dolžino ene dobe, je smiselno razdeliti pas s pravokotniki po navpični osi, na obdobja, ki ustrezajo eni dobi. Najpogostejše note, ki se pojavijo v skladbi so med celinko in šestnajstinko. Četudi ime

celinka namiguje, da je le-ta najosnovnejša enota dobe, se v praksi izkaže, da je v veliki večini primerov bolj smiselno za osnovno enoto dobe izbrati četrtinko, kakor to počnemo mi.

V glasbeni terminologiji se pogosto pojavijo izrazi, ki opisujejo predvideno hitrost izvajanja. Najbolj sprejeti so izrazi v italijanščini, med katerimi je 'Allegro', kar pomeni veselo ali hitro. Četudi ni vsesplošno sprejetih definicij hitrosti izvajanja, je razumno v definicijo hitrosti 'Allegro' vključiti hitrost, ki definira dobo ene četrtinke kot 120 dob na minuto. S tem lahko določimo, da 120 dob na minuto v splošnem predstavlja hitro pesem. 120 dob na minuto pomeni, da ena doba traja pol sekunde. Dolžino dobe med igranjem ali petjem vzdržujemo s tem, da z nogo tolčemo ob tla v enakomernem ritmu. Tolčemo lahko seveda zgolj z omejeno hitrostjo, zato ne moremo tolči osmink, ki bi zahtevale, da ob tla udarimo štirikrat v sekundi. Podobno pa se ob udarjanju ob tla s hitrostjo celinke zgodi to, da je obdobje med dvema udarcema dolgo dve sekundi. S tako dolgim obdobjem pa ne moremo vzdrževati enakomerne hitrosti skozi celotno pesem. V hitri pesmi lahko torej tolčemo le vsake pol sekunde za četrinko ali vsako sekundo za polovinko. Katero izberemo je odvisno od pesmi ter od naše sposobnosti igranja. Za izdelavo aplikacije pa je precej pomembna konsistentnost. Koristno je, da je obnašanje aplikacije v podobnih situacijah čimbolj podobno. Posledično je koristno, da izberemo eno hitrost delitve na časovne pasove.

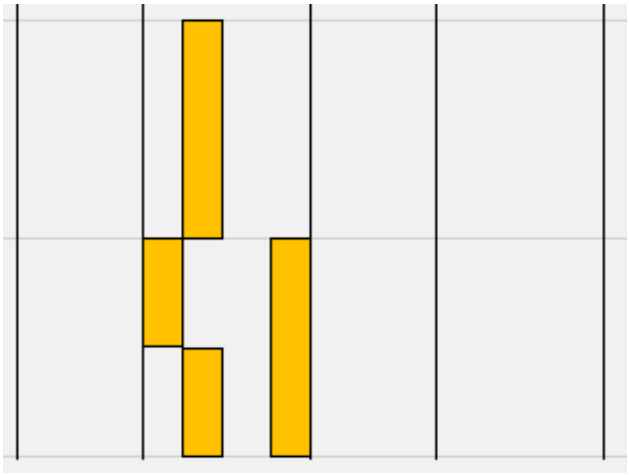
Iz pregleda hitrih pesmi smo določili, da sta najprimernejše note za določitev ene dobe četrtinka in polovinka, kjer polovinka traja dvakrat dlje kot četrinka. To smo preizkusili pri hitrosti pesmi, kjer je doba ene četrtinke definirana kot 120 dob na minuto. To ustreza hitrim pesmim, a če to hitrost razpolovimo na 60 dob na minuto, pridemo v območje hitrosti, ki so poimenovane 'lento' in 'largo'. 'Lento' pomeni počasi, 'largo' pa široko. Skratka hitrost 60 dob na minuto ustreza počasnim pesmim. Če opravimo podobno primerjavo, kot smo jo opravili pri hitrih pesmih, ugotovimo, da ena osminka traja pol sekunde, ena četrtinka traja eno sekundo, ena polovinka pa dve sekundi. Podobno kot pri celinki pri hitrih pesmih, je tolčenje z nogo vsaki dve sekundi preredko, da bi lahko vzdrževali isto hitrost dalj časa. Četrtinka tokrat traja eno sekundo, kar je primerna perioda, osminka traja pol sekunde, kar je tudi primerna perioda, šestnajstinka pa samo eno četrtino sekunde, kar je pa prehitro za udarjanje. Posledično lahko zaključimo, da je za počasne pesmi najprimerneje določiti četritnko ali osminko kot eno dobo. Katero si izberemo je tudi tokrat odvisno od pesmi in naše pripravljenosti.

Pri hitrih pesmih sta primerne note za definicijo ene dobe polovinka in četrtinka, pri počasnih pa četrtinka in osminka. Skupni element, ki ustreza hitrim in počasnim pesmim, je

torej četrinka in zato jo bomo uporabili kot dobo, ki določa časovno obdobje med vodoravnimi črtami, ki razdelijo pas s pravokotniki po navpični osi.

Obstaja sicer alternativa, da bi določili drugačno dobo za počasne in hitre pesmi, a implikacija te metode je, da bi na neki arbitrarni točki morali postaviti mejnik. Pesmi nimajo te lastnosti, da spadajo striktno med hitre in počasne, obstajajo pesmi vseh hitrosti, posledično obstajajo tudi pesmi, ki so blizu meje, katero si izberemo, ne glede na to kam mejo postavimo. Pesmi pa imajo sicer lastnost, da so zelo konstantne v hitrosti, a jih lahko občasno spreminjajo. Pogosto je, da imajo pesmi hitre in počasne dele. Le-to bi zahtevalo spremembo obdobja, ki ga časovni razdelki predstavljajo. Kot primer, če si kot mejnik izberemo hitrost 100 dob na minuto, kjer pesmi, ki so počasnejše od 100 dob na minuto razdelimo na obdobja, ki ustrezajo dobi ene četrtinke, pesmi, ki so hitrejše od 100 dob na minuto pa razdelimo na obdobja, ki ustrezajo dobi ene polovinke, lahko naletimo na pesem, ki v nekem trenutku preide iz hitrosti 90 na hitrost 120 dob na minuto. V prvem delu bo uporabljala razdelke, ki bodo trajali enako dolgo kot ena četrtinka, torej 0,75 sekunde, v drugem delu pa razdelke, ki bodo trajali enako dolgo kot ena polovinka, torej 1 sekundo. V drugem delu četrinka namreč traja 0,5 sekunde. Skratka, kljub temu, da se hitrost poveča iz 90 na 120, se obdobje med dvema razdelkoma poveča iz 0,75 sekunde na 1 sekundo. Enako, le bolj očitno, bi se zgodilo v bolj ekstremnem primeru, kjer bi iz hitrosti 99 udarcev na minuto prešli na hitrost 101 udarec na minuto, kar bi povzročilo skoraj dvakrat daljše obdobje med razdelki, kljub temu, da smo hitrost izvajanja pohitrili.

Zadnja opcija je ta, da dovolimo uporabniku, da si dolžino razdelka sam izbere. To je uporabno za napredne uporabnike, a večini začetnikov nepomembno in nerazumljivo. Naša naloga je, da tem uporabnikom pripravimo aplikacijo tako, da jim bo čimbolj uporabna takoj ob zagonu.



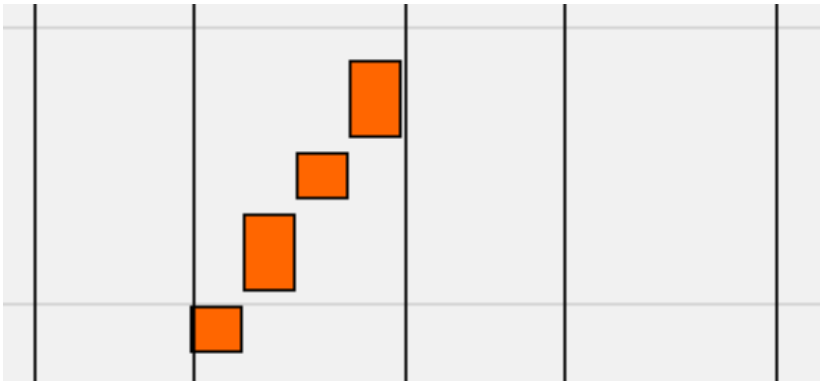
Slika 4.2: Izsek grafičnega vmesnika z razdeljenim poljem, po katerem se gibajo pravokotniki Delitev v časovna obdobja.

Med preizkušanjem aplikacije smo opazili, da so črne, vodoravne, premikajoče se črte, ki določajo trajanje dobe, zelo neprijazni za oči, zato smo jih prebarvali v svetlo sive. S tem so še vedno dovolj vidni, a ne dražijo oči.

Naslednja težava, s katero smo se spoprijeli še vedno ostaja na težavnosti prevajanja pravokotnikov v pritiske tipk na sintetizatorju, a se vrača na težavnost prepoznavanja kateri tipki ustreza nek pravokotnik. V preteklih korakih smo poskrbeli, da je, z vajo, možno določiti kateri beli tipki pripada nek pravokotnik, s tem da smo klaviaturo razdelili na območja, ki jih določajo kraji, kjer sta po dve beli tipki skupaj. S tem smo razdelili pas, po katerem se premikajo pravokotniki na območja, ki obsegajo po 3 ali 4 bele tipke. To zadostuje za identifikacijo belih tipk, a težava ostaja pri identifikaciji pravokotnikov, ki ustrezajo črnim tipkam.

V originalni zasnovi so bili pravokotniki, namenjeni črnim tipkam, le zamaknjeni za polovico širine pravokotnika, ki ustreza beli tipki. S tem smo dosegli, da se pravokotnik pojavi točno nad črno tipko, kateri ustreza, a s tem smo povzročili, da pas, ki je prej podpiral le pravokotnike, namenjene trem belim tipkam, sedaj moral podpirati pet tipk – iste tri bele kot prej, ter še dve črni, ki se nahajata med njimi. Pas, ki je pred tem ustrezal štirim belim tipkam, pa je sedaj moral podpirati skupaj sedem tipk. Težava pri temu je v tem, da mora uporabnik prepoznavati odmike od ene ali druge navpične črte, ki definira razdelek. Do trenutka, ko smo začeli upoštevati črne tipke, si je uporabnik v najtežje razpoznavnem razdelku, ki se nahaja med tonoma Fn in Hn in obsega štiri bele tipke, moral zapomniti le štiri lokacije. Te lokacije so: Tik ob levi črti ustreza tonu Fn, zamaknjen za eno širino

pravokotnika v desno od leve navpične črte ustreza tonu Gn, zamaknjen za eno širino pravokotnika v levo od desne navpične črte ustreza tonu An in pravokotnik, ki je tik ob desni črti ustreza tonu Hn. Z upoštevanjem črnih tipk pa mora uporabnik prepoznavati tudi pravokotnike, ki so zamaknjeni za polovico ene širine v desno, od leve črte, kar ustreza tonu F#n, pravokotnike, ki so v čisti sredini med obema navpičnima črtama, ki ustrezajo tonu G#n in pravokotnike, ki so zamaknjeni za polovico ene širine v levo, od desne črte, ki ustrezajo tonu H#n. Podobna, a nekoliko manj resna, je bila situacija tudi v pasu med tonoma Cn in En.



Slika 4.3: Izsek grafičnega vmesnika. Prikaz tonov Fn, Gn, An in Hn.

To je torej povečalo količino unikatno postavljenih pravokotnikov, ki se lahko pojavijo, iz 7 v vsaki oktavi na 12 v vsaki oktavi. Unikatno predstavljenih pravokotnikov bo ostalo 12 ne glede na rešitev, ki jo uporabimo, saj to število ustreza številu tipk in tonov v eni oktavi. Posledično bo, ne glede na implementacijo, učenje prevajanja pravokotnikov na pritiske tipk ostal dolgotrajen in zahteven postopek. Naša naloga pa je, da ta postopek čimbolj skrajšamo s tem, da naredimo dejanje prevajanja čimbolj enostavno in intuitivno.

Za rešitev smo pogledali na klaviaturo - v čem so očitne razlike med belimi in črnimi tipkami. Očitne razlike so tri. Prva je v višini tona, ki ga slišimo, ko tipko pritisnemo. Nobenega dobrega načina nismo našli, kako to razliko izkoristiti za poenostavljenje postopka prevajanja. Drugi dve očitni razliki pa imata tudi očitne implementacije za poenostavitev. Prva razlika je barva tipke. Do sedaj barvi pravokotnikov nismo posvečali velike pozornosti. V tem koraku smo poizkusili z implementacijo različno obarvanih pravokotnikov, ki ustrezajo barvi tipke, katero predstavljajo.

Postopek ugotavljanja barve tipke je relativno enostaven, če upoštevamo lastnosti oktav. Zaporedje tonov je namreč v vsaki oktavi enako. Po specifikaciji MIDI [9] je nota z zaporedno številko 0 tudi prva definirana nota v običajen zapisu, to je nota C0, ki je

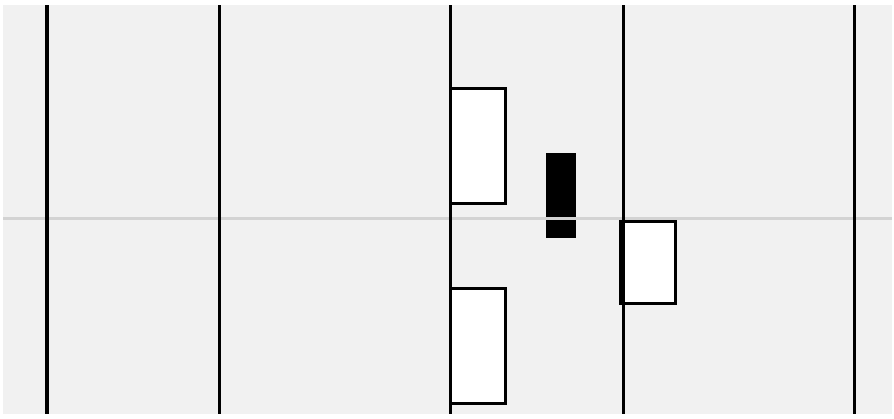
definirana s frekvenco 16,35Hz ali valovno dolžino 2,1m [12], kar je pod pragom sluha večine ljudi. C1, ki je eno oktavo višje, je definirana z zaporedno številko 12. C2 je definirana z zaporedno številko 24. Definicije tona C v višjih oktavah sledijo pravilu, da je ton C vsake naslednje oktave definiran z zaporedno številko, ki je za 12 višja od definicije tona C predhodne oktave. Enako pravilo velja tudi za vse ostale note. MIDI zapis torej izkorišča lastnost enakosti poimenovanja tonov v vsaki oktavi.

Ob prejemu sporočila iz sintetizatorja, ki se pošlje ob pritisku tipke, prejmemo tudi podatek, ki vsebuje zaporedno številko tipke, ki je bila pritisnjena ali spuščena. Ta zaporedna številka je skladna z zaporednimi številkami tonov.

Iz prejete številke lahko enostavno določimo kateri oktavi pripada in kateri ton predstavlja. Če število celoštevilčno delimo in pogledamo rezultat deljenja, nam ta predstavlja oktavo, kateri ton pripada. Če pa celoštevilčno delimo in pogledamo ostanek, nam le-ta pove kateremu tonu v oktavi ton ustreza.

Imena črnih tipk, ki se nahajajo na klaviaturi so C#, D#, F#, G# in A#. Če pritisnemo eno od teh tipk v poljubni oktavi in izvedemo celoštevilčno deljenje s številom 12, bo ostanek enak eni od sledečih vrednosti: 1, 3, 6, 8 ali 10. S tem smo izvedli algoritem, ki ugotovi ali je bila pritisnjena črna ali bela tipka. Posledično vemo ali naj pravokotnik, ki predstavlja tipko, pobarvamo črno ali belo.

Poleg že omenjenih očitnih razlik med belimi in črnimi tipkami, to so višina igranega tona ob pritisku in barva, obstaja še tretja - oblika tipke. Glavni lastnosti oblike, kateri lahko izkoristimo sta širina in pozicija tipke po vodoravni osi. Lastnost pozicije smo sicer že izkoristili, saj so pravokotniki, ki ustrezajo črni tipki, za polovico širine pravokotnika zamaknjeni v levo oz. desno, tako da se sredina pravokotnika nahaja točno nad črno tipko. Ker je v praksi dejanje pritiska na črno tipko drugačno kot dejanje pritiska na belo tipko, še posebej za začetnika, smo se odločili, da bomo drugačnost ponazorili tudi z obliko pravokotnika, namenjenega črni tipki. Ker so črne tipke na klaviaturi tanjše od belih, kar velja tudi za klaviaturo, katero imamo izrisano na zaslonu, smo se odločili, da bomo razpolovili širino pravokotnika, namenjenega črni tipki.



Slika 4.4: Izsek grafičnega vmesnika. Prikaz pravokotnikov v barvi tipke, kateri ustrezajo.

S tem smo implementirali vse načrtovane izboljšave načina prikaza pravokotnikov, katerih namen je lažje prevajanje med pravokotniki na zaslonu in pritiski tipk na sintetizatoru. Vse našteto bi lahko storili že v koraku razvoja igre, saj gre za splošne izboljšave.

Tekom razvoja in testiranja implementiranih izboljšav se je pogosto pojavljalo vprašanje obširnosti programa, kateri zadostuje za nadomestilo učitelja igranja na klavir. Program, ki omogoča igranje in avtomatsko ocenjevanje namreč ne zadostuje za nadomestilo učitelja. Posameznik, ki bi uporabljal tak program, ne da bi se posvetoval z učiteljem, bi namreč pogosto pridobil slabe navade, primer katere je prijem akordov, to je več tonov hkrati, s trdo, napeto roko. Program namreč nebi bil sposoben določiti kako posameznik dejansko igra. Sposoben je le ugotoviti, kako natančno posameznik izvaja neko pesem, ne pa na kakšen način jo izvaja in kakšne slabe navade ali prakse uporablja.

Tehnik, ki jih uporabljajo učitelji med učenem je veliko. Prav tako je veliko tehnik in prijemov, ki se jih mora igralec naučiti. Navodila za uporabo vseh teh prijemov in metod bi moral program vsebovati. Uporabniku pa bi jih moral predstaviti na smiseln način. To je izvedljivo, čeprav bi zahtevalo veliko raziskave, razvoja in testiranja. Težava pa je v merjenju in ocenjevanju izvedbe. Če je namen programa ta, da uporabnika nauči igranja na klavir, je namreč ključno, da ga zna ocenjevati glede na njegov stil igranja in ne samo pravočasnosti pritiska tipke.

Uporabnik lahko namreč C durov kvintakord, to je akord, ki ga dosežemo s simultanim pritiskom tipk C, E in G v poljubni oktavi, pritisne z uporabo različne kombinacije prstov. Lahko pritisne s kazalcem, sredincem in prstancem; lahko z palcem, kazalcem in sredincem; lahko s palcem, kazalcem in prstancem ali z kakšno drugo kombinacijo. Nekatere od teh so lastne slabim navadam in se jih je kasneje težko odvaditi.

Program pa nebi imel nobenega načina, da bi uporabnika opozoril na nevarnost slabe navade, saj nebi vedel kakšen prijem uporabnik uporablja.

Ker je namen aplikacije ta, da nadomesti učitelja igranja na klavir, je ob trenutni tehnologiji in formatu MIDI ter izbiro sintetizatorjev, ki se nahajajo na trgu, to nemogoče. Tekom razvoja smo upali, da se bo pojavila kakšna ideja, s pomočjo katere bomo lahko prešli opisane težave, a se je na koncu izkazalo, da je bil razvoj take aplikacije preveč ambiciozen, saj smo si kot cilj zastavili uporabno aplikacijo in ne le zaključeno, delujočo aplikacijo.

Razvoj, ki smo ga storili v tem koraku, je bil precej splošno usmerjen. Ves razvoj in vložen čas je bil koristno uporabljen, a razvoja, ki bi nas bližal h konceptualnim ciljem, ki smo si jih postavili v začetku koraka, ni bilo. Ta korak smo predčasno zaključili zaradi predvidenih težav, katerih smo se zavedali med razvojem drugih funkcionalnosti in izboljšav. Razvoj aplikacije v sledečem koraku bo zelo podoben temu, a se bo omejil na aplikacijo, katere namen je pomoč učenja igranja na klavir s pomočjo učitelja. Aplikacija torej ne bo poskušala učitelja zamenjati, temveč mu pomagati.

Poglavje 5

Razvoj aplikacije za pomoč učitelju klavirja

Razvoj je v predhodnem koraku potekal s konceptualno idejo izdelave aplikacije, katera bo uporabniku sposobna pomagati se naučiti igranja na klavir. Uporabnik aplikacije naj tekom učenja nebi potreboval mentorja ali učitelja. Med razvojem pa se je pojavilo ogromno vprašanj in problemov, za katere rešitev nismo našli. Posledično smo se odločili, da bo postopek učenja klavirja aplikacija še vedno olajšala, a bo tokrat zasnovana kot pomočnik, s katerega lahko učitelj uporablja pri učenju, ali ga uporablja učenec, a ob vodstvu učitelja.

Funkcionalnost branja in analize pritiskov tipk po navodilih iz MIDI datoteke bo ostala, saj je popolnoma možno, da bodo učitelji, ki bodo koristili aplikacijo, naredili svoje vaje v obliki MIDI datoteke, katere bodo nato dali svojim učencem. Učenci pa bodo vaje naložili in igrali. Te funkcionalnosti v tem koraku še nismo dokončali, saj smo vedno naleteli na težave na konceptualnem nivoju, zaradi katerih smo načrt spremenili.

Predvajanje MIDI datotek trenutno poteka tako, da se ne prikažejo pravokotniki, ki naznanijo potek še neigranega dela skladbe, marveč se izriše sled igranega. To pomeni, da se pravokotniki izrisujejo v trenutku, ko jih program tudi predvaja. Sled nam v praksi ni koristna za analizo igranja, a koristna implementacija načina zakasnitve je bila težavna.

Razvoj tehnološke implementacije algoritma, ki skrbi za zakasnitev in predvajanje zakasnjene pesmi, je šel skozi nekaj iteracij. Prvi poizkus je vključeval implementacijo knjižnice TaskScheduler [3], ki omogoča vstavitve dogodkov v urnik. Knjižnica pa nato poskrbi, da se vstavljen dogodek izvede ob trenutku v primeru, ko je izvajanje bilo predvideno. Težava z implementacijo je, da knjižnica ni namenjena za uporabo, ko želimo dogodke zakasniti za kratek čas, marveč le za daljša obdobja. Poleg tega ni dovolj natančna, da bi lahko prožila dogodke ob trenutkih, ki so do milisekunde natančno definirani. Implementacijo tega smo kmalu opustili.

Drugi poizkus je bil zasnovan na konceptu časovnikov. Časovniki so nevidni gradnik grafičnega vmesnika programa. Njihova lastnost je, da se prožijo ob določenih intervalih. Interval lahko nastavimo do milisekunde natančno. Uporabo, katero smo predvideli za rešitev problema, je vključevala enkratne časovnike. Program bi ob prvem branju sporočila, ki predstavlja začetek ali konec note, ustvaril nov časovnik, v kateremu bi shranil sporočilo. Časovniku bi nastavili interval proženja, to je interval med dvema zaporednima proženjema, na število, ki ustreza želeni zakasnitvi. Če želimo, da je med pojavitvijo pravokotnika in predvajanjem tona 3 sekunde zakasnitve, bomo časovniku nastavili interval 3000 milisekund.

Ob prvem proženju vsakega od teh časovnikov, bi koda, ki jo časovnik proži, poslala shranjeno sporočilo v predvajanje. Poleg tega pa bi tik pred zaključkom izvajanja programa, koda ustavila vsako naslednje proženje časovnika tako, da bi časovnik izključila in zbrisala.

Gre za nestandardno uporabo časovnika. Časovnik je namreč namenjen proženju v rednih intervalih. V tem primeru pa se sproži enkrat, in nato sam sebe ustavi in zbriše. Taka implementacija zakasnitve je delovala, a težava se je pojavila pri pesmih, ki vsebujejo veliko tonov. V taki situaciji se je namreč zgodilo, da se je ustvarilo toliko časovnikov, kolikor je bilo predvidenih tonov v še nepredvajanih treh sekundah pesmi. Časovniki pa so objekti, ki so relativno zahtevni pri uporabi procesorja. Posledično je aplikacija utrpela padec hitrosti in začela delovati počasi.

Zaradi težav z učinkovitostjo smo morali ponovno premisliti metodo implementacije. Tokrat smo preiskali kako je to implementirano v sami knjižnici C# MIDI Toolkit. Ugotovili smo, da v ozadju knjižnice teče časovnik, ki se proži vsako milisekundo. Ob predvajanju MIDI datoteke, se na primerna mesta v časovnik vnesejo MIDI sporočila, ki se ob pravem trenutku predvajajo.

Za doseganje višje učinkovitosti, smo se zato odločili, da bomo uporabili le en časovnik, ki se bo prožil vsakih 10 milisekund in bo predvajal vsa sporočila, ki so predvidena za tisti trenutek. Knjižnica C# MIDI Toolkit, ki se sicer nahaja v ločenem projektu, katerega vključujemo v naš projekt, ob predvajanju tona, prebranega iz MIDI datoteke, v glavnem oknu našega projekta sproži dogodek. Dogodek pa nato izvede določeno kodo.

Demonstracijski program, s katerim smo začeli, je tu predvideval le posredovanje sporočila izrisani klaviaturi, ki je nato obarvala tipko, kateri ton ustreza. V to lokacijo pa smo za implementacijo zakasnitve morali dodatki kodo, ki to zakasnitev izvede.

V programsko kodo smo implementirali seznam MIDI sporočil. Ta seznam hrani vsa sporočila, ki so trenutno v procesu zakasnitve. To so torej vsa sporočila, katera smo že prebrali iz MIDI datoteke, zanje obstajajo pravokotniki, a sama sporočila se še niso izvedla.

Vsakemu sporočilu je dodana številka, ki ustreza identifikatorju časa, ob katerem naj bi se sporočilo izvedlo.

Časovnik, ki se sproži vsakih 10 milisekund, ob vsaki sprožitvi poveča številko, ki identificira trenutke. Takoj za tem preveri, katera sporočila so predvidena za ta trenutek. To stori tako, da v seznamu MIDI sporočil preveri elemente, ki se nahajajo na začetku seznama in jih izvede. Seznam MIDI sporočil je vedno sortiran po indeksih, ki predstavljajo čas, ko naj bi se določeno sporočilo izvedlo. Časovnik, ki ta sporočila izvaja pa vedno preverja prvi element v seznamu in ga izvede, če se indeks, ki predstavlja čas, ko naj bi se sporočilo izvedlo ujema s trenutnim indeksom. Če se ujema, se sporočilo izvede in se element, ki hrani sporočilo odstrani iz seznama. S tem se spremeni prvi element seznama. Zanka preverja elemente seznama, dokler ne naleti na element, katerega indeks časa se ne ujema s trenutnim. To omogoča predvajanje več tonov v istem trenutku, kar je potrebno za izvajanje velike večine melodij.

Preizkusi delovanja so pokazali, da je taka implementacija zakasnitve mnogo bolj učinkovite od predhodne implementacije z mnogimi časovniki. Učinkovitost se je izkazala za zadostno, zato smo to implementacijo tudi obdržali.

5.1 Funkcije za pomoč učitelju

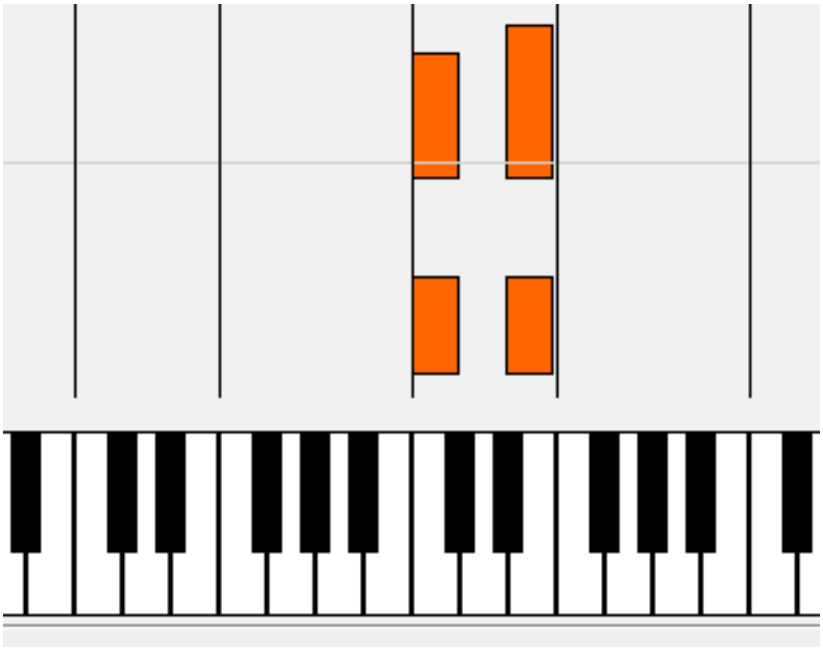
Ker je aplikacija v tem koraku namenjena pomoči učenja igranja na klavir, a s pomočjo učitelja, smo začeli z implementacijo funkcij, ki so namenjene pomoči učitelju.

Prva taka funkcija izkorišča predhodno neželeno lastnost programa, ki smo jo imeli pred implementacijo zakasnitvenega algoritma - sled igranja.

Sled igranja je izris prikaza igranja v obliki pravokotnikov, ki se izvaja istočasno kot dejansko igranje na sintetizator. Do sedaj je bila implementacija le demonstracijska po naravi, namenjena ugotavljanju zahtevnosti prevajanja med pravokotniki in tipkami na sintetizatorju. Poleg implementacije prikaza sledi igranja MIDI datotek, smo tekom razvoja implementirali tudi prikaz sledi v živo igrane melodije, katero aplikacija prejema od sintetizatorja. Izvedba tega je bila s pravokotniki, ki so se gibali od vrha proti dnu, kakor naj bi po takratnem načrtu končna aplikacija delala, a zaradi spremembe namena prikaza sledi iz demonstracije branja MIDI datotek, na vizualni prikaz igranega, se je izkazalo za bolj smiselno spremeniti smer gibanja pravokotnikov.

Vizualna analiza pravokotnikov je potencialno uporabno orodje za učitelje igranja na klavir, saj natančno prikaže vse podrobnosti igranja. A učitelji imajo pogosto sposobnosti sluha, ki omogočajo natančnejše odkrivanje napak od natančnosti, ki jo nudi aplikacija.

Pravokotniki se namreč premaknejo vsakih 50 milisekund, kar je dovolj redko, da lahko ne zaznajo neistočasnega pritiska dveh tonov in pritisk prikažejo kot istočasen, medtem ko lahko uho to jasno zazna. Take lastnosti omejujejo uporabnost te funkcije učitelju. Funkcija pa ostaja uporabna učencu. Učenec lahko med igranjem melodije opazuje zaslon, s katerega lahko jasno vidi nenatančne prijeme. Pogosto je, da lastnih nenatančnih prijemov sami ne zaznamo, oziroma jih smatramo za neopazne. Aplikacija v taki uporabi nudi nepristransko poročilo o natančnosti igranja. Cilje, katere naj bi učenec dosegel, pa bi določil učitelj.



Slika 5.1: Izsek grafičnega vmesnika, prikaz sledi igranega. Smer gibanja pravokotnikov je navzgor. Zgornja tona nista bila pritisnjena istočasno, spodnja tona pa ja.

Začetno učenje igranja na klavir vključuje veliko vaje. Učenec se mora med drugim naučiti orientacije po klavirju, pridobiti mora sposobnost samostojnosti prstov, natančnosti in pravočasnosti pritiskov tipk ter uporabo pravšnje sile pritiska tipk.

Za orientacijo po klavirju smo tekom razvoja implementirali različne izboljšave metode, ki jo za prikaz želene tipke ter časa, uporablja igra 'Guitar Hero'.

Sposobnost samostojnosti prstov pridobi učenec le ob učenju in vaji pravilnih tehnik igranja. Nesposobnost, da bi to funkcijo kakovostno implementirali je razlog, zakaj smo prešli iz izdelave aplikacije, katere namen je zamenjava učitelja, na aplikacijo, katera učitelju le pomaga v pedagoškem postopku.

Natančnost in pravočasnost pritiska tipk lahko uporabnik vidi iz prikaza igranja, ki se izriše v sledi igranega. Pravokotnika, ki predstavljata dve noti, ki naj bi bili pritisnjeni istočasno, katera se ne začneta na isti višini, sta indikator, da tona nista bila pritisnjena istočasno. S časom in vajo bo uporabnik vedno bolj sposoben istočasnega pritiska tipk.

Zadnja sposobnost, za učenje katere bo program nudil orodje, je sposobnost uporabe prave količine sile pri pritisku tipke. Hitrost in sila pritiska ustrezata glasnosti zaigranega tona. V začetnih korakih, preden uporabnik razvije to sposobnost, so glasnosti zaigranih tonov zelo nekonsistentne.

MIDI sporočilo, ki opisuje začetek tona, poleg zaporedne številke tipke, ki je bila pritisnjena, nosi tudi podatek o glasnosti, ki ustreza hitrosti pritiska. Ta podatek bomo uporabljali za prikaz glasnosti na vizualen način.

Glasnost opisuje vrednost celega števila, katerega vrednost je med 1 in 127. Glasnost 0 pomeni zaključek zaigranega tona. Glasnost 1 predstavlja izjemno počasen pritisk tipke, ki proizvede tako tih ton, da je človeku v realnem okolju neslišen. Ton, ki ustreza vrednosti nad 80 pa je glasen.

Glasbena terminologija definira določene subjektivne nivoje glasnosti. Ti nivoji nosijo italijanska imena: 'piannissimo', z oznako 'pp' pomeni 'zelo tiho'; 'piano', z oznako 'p' pomeni 'tiho'; 'mezzo piano', z oznako 'mp' pomeni 'srednje tiho'; 'mezzo forte', z oznako 'mf' pomeni 'srednje glasno'; 'forte', z oznako 'f' pomeni 'glasno' in nazadnje 'fortissimo', z oznako 'ff' pomeni 'zelo glasno'. Poleg teh obstajajo še druge, bolj opisne oznake, ki opisujejo pričakovano dinamiko, to je izvedbo, neke pesmi. Primeri teh so 'crescendo', kar pomeni 'naraščanje' in označuje del pesmi, kjer naj bi glasnost naraščala, 'diminuendo', ki pomeni 'nižanje' in pomeni obratno kot 'crescendo', 'sforzato', ki pomeni 'na silo', in označuje ton ki naj bi bil zaigran na silo. Že pogosto uporabljenih izrazov je veliko, a avtorji glasbenih zapisov lahko uporabljajo tudi svoje, s katerimi lahko bolje opisujejo želeno izvedbo svoje skladbe.

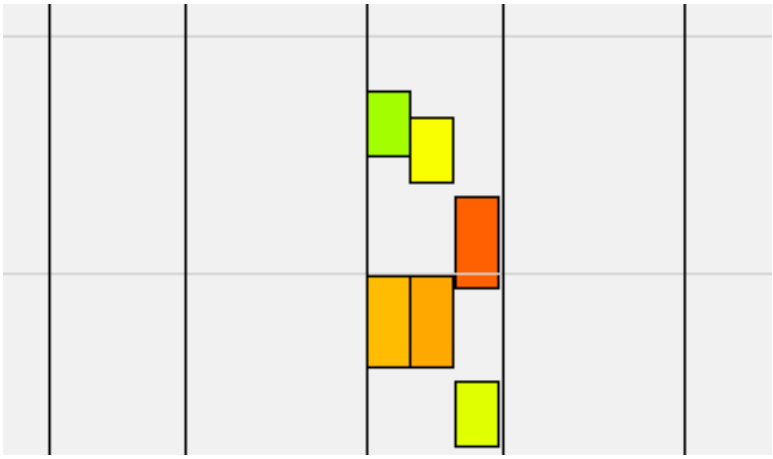
Poleg težav z jasnimi, matematičnimi definicijami opisnih izrazov, kot je 'sforzato', so težave tudi pri izrazih, katerih imena poskušajo implicirati, da bi morali imeti jasno definicijo.

Izrazi, kot so 'piano', ki pomeni 'tiho', imajo različne interpretacije v različnih pesmih. Če klarinetist igra koračnico in naleti na oznako 'piano', bo glasnost igranja še vedno mnogo višja, kot če isti klarinetist igra pesem, ki je bolj umirjena. Tudi za take izraze, ki so na videz enostavni, je težko jasno določiti primerne, matematične definicije. Posledično je nesmiselno izdelati orodje, ki bi služilo avtomatskemu ocenjevanju, saj bi bilo tako orodje slabo in bi služilo le zavajanju uporabnika, ali pa bi mu vsiljevalo nepravilen način igranja.

Aplikacija torej ne bo imela avtomatskega načina ocenjevanja ali vrednotenja dinamike. Bo pa vsebovala orodje, katerega namen je vizualizacija glasnosti.

Med končanimi funkcijami je funkcija, ki obarva pravokotnike v belo ali črno barvo v odvisnosti od tega, kakšni tipki so namenjeni. Taka funkcija je koristna, če pravokotniki predstavljajo navodila za igranje, a če predstavljajo sled igranega, ob trenutku pritiska tipke že vemo katero tipko smo pritisnili. Posledično je nekoristno, da so pravokotniki obarvani po barvi tipke. Lahko pa izkoristimo barvo za vizualizacijo glasnosti.

Barvna lestvica, ki poteka od zelene do rumene in nato rdeče, je pogosto uporabljena za prikaz podatkov, ki imajo tri pomembna območja ali stanja. V našem primeru nas zanima glasnost, katero lahko v grobem delimo na tiho, srednje glasno in glasno. Vrednost, ki določa glasnost predstavlja celo število, ki je lahko med 1 in 127. Po nekaj poskusih smo določili, da je razumna glasnost, ki ustreza srednji glasnosti okoli vrednosti 50, glasnost, ki ustreza glasnemu igranju pa okoli vrednosti 100. Točki 50 smo tako določili rumeno barvo, točki 100 pa rdečo. Zelena pa je določena za točko 0, oziroma v praksi 1. Četudi je primerna glasnost subjektivna vrednost in se lahko, kot primer, definicija besede 'tiho' zelo razlikuje v različnih pesmih, nam barvna lestvica nudi vizualizacijo glasnosti, s čimer lahko vidimo konsistentnost glasnosti v nekem delu.



Slika 5.2: Izsek grafičnega vmesnika. Barva tona prikazuje glasnost igranega od zelene (tiho) do rumene in nato rdeče (glasno).

Poleg konsistentnosti lahko učitelj učencu pokaže, kakšna barva ustreza primerni glasnosti v nekem odseku. S tem ima učenec med vajo zanesljiv način ocenjevanja primernosti glasnosti, s katero igra.

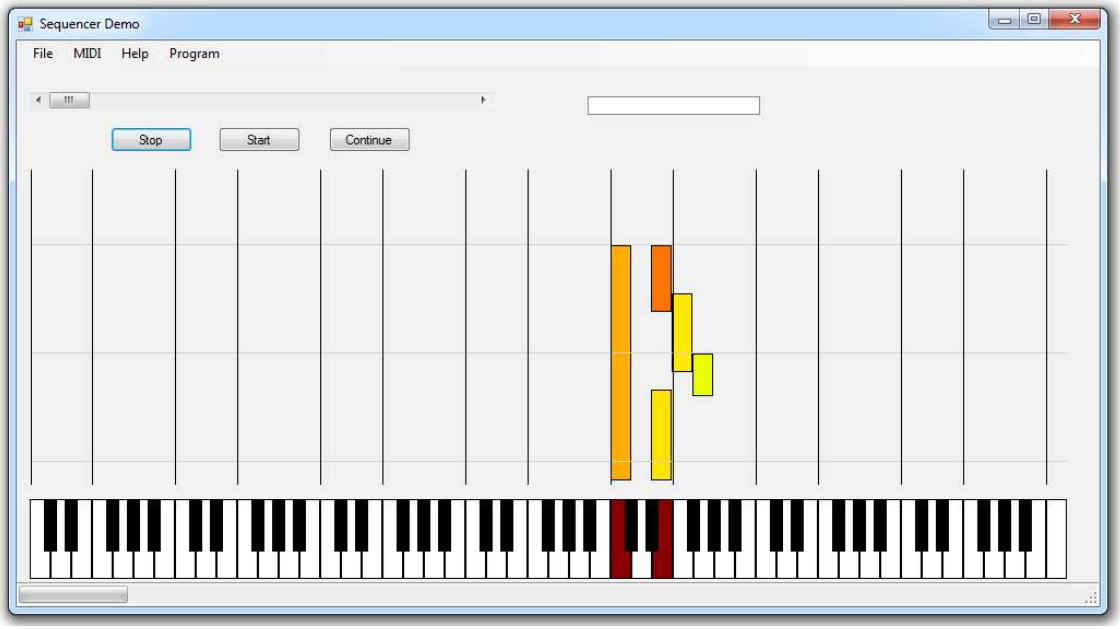
V praksi se pojavi še ena težava, ki je lastna sintetizatorjem nižjega cenovnega razreda, na katerih se učenci pogosto učijo. Težava je v tipkah, ki so pogosto narejene iz plastike in niso primerno utežene. Klavir, ki deluje mehanično, uporablja relativno težke tipke. Poleg tega pa moramo tipke pritisniti z zadostno silo, da sprožimo celoten mehanizem. Sintetizatorji tega ne potrebujejo za pravilno delovanje, zato imajo pogosto plastične tipke, ki niso utežene tako, kot so utežene tipke klasičnega klavirja. Posledično se bo posameznik, ki se želi naučiti igranja na klavir, moral ponovno naučiti primerne moči pritiska ob prehodu iz sintetizatorja na klavir. Alternativa je v uporabi kakovostnega sintetizatorja, ki ima primerno utežene tipke. Taki sintetizatorji pa so načeloma dražji od sintetizatorjev vstopne kakovosti.

Z implementacijo prikaza glasnosti pritiska tipke smo zaključili z razvojem aplikacije, katere namen je pomoč učitelju igranja na klavir pri pedagoški dejavnosti. Tekom razvoja smo pregledali primernost uporabe formule, ki se pojavi v igri 'Guitar Hero' za klavir ali sintetizator.

Poglavje 6

Kritika končne aplikacije

V začetnem načrtu je bil predvideni končni izdelek zaključena aplikacija, torej aplikacija, katera ustreza načelom izdelave dobrih aplikacij. Zaradi zapletov tekom razvoja smo načrt spreminjali na konceptualni ravni, kar je zahtevalo tehten časovni vložek, zaradi česar izdelana aplikacija dosega zgolj kakovostni razred demonstracijske aplikacije.



Slika 6.1: Grafični vmesnik zaključene aplikacije.

Grafični vmesnik aplikacije ostaja podoben začetnemu, a z mnogimi funkcionalnimi izboljšavami, ki so podrobneje opisane v opisih korakov razvoja.

Mnogo funkcionalnosti, ki so opisane v poglavju o razvoju, deluje tako, da jih lahko poljubno vključimo ali izključimo. Dostop do vseh gumbov, ki nadzorujejo vklop in izklop teh funkcij in načinov delovanja, omogoča element menija na vrhu, ki nosi ime 'Program'.

Meni 'Program' vsebuje gumb, ki omogoča dostop do obrazca, na katerem lahko preverimo zakasnitev delovanja, katerega smo razvili v začetku razvoja. Poleg tega omogoča vklop in izklop odmeva, ki vsa sporočila, ki jih prejme s sintetizatorja, predvaja tudi na računalniku; vklop in izklop zakasnitve, ki spremeni delovanje iz prikaza pravokotnikov kot navodil za igranje, v prikaz sledi igranja; spremembo smeri gibanja pravokotnikov navzgor ali navzdol; prikaz barve pravokotnikov kot črne ali bele, v odvisnosti od tega, kakšne barve je tipka, kateri ustrezajo, ali prikaz barve pravokotnikov na lestvici od zelene do rumene do rdeče, ki ustreza glasnosti tona, katerega pravokotnik predstavlja.

Naštete funkcionalnosti lahko poljubno vključujemo, izključujemo in kombiniramo. To omogoča boljšo fleksibilnost pri demonstraciji programa in njegovega delovanja, ni pa primerno za povprečnega uporabnika. V primeru dopolnitve aplikacije do nivoja, primernega za odprtokodni ali komercialni trg, bi program moral samodejno vključevati in izključevati naštete funkcije na smiseln način.

Implementacija formule 'Guitar Hero' je na prvi pogled smiselna in enostavno prevedljiva za klavir. Tekom razvoja pa smo naleteli na številne pričakovane in nepričakovane težave pri njeni implementaciji. Izvedba formule v končni aplikaciji vključuje le malo elementov, ki so lastni formuli 'Guitar Hero'.

Glavnimi element formule, ki ga aplikacija vključuje, so premikajoče se oznake, v našem primeru pravokotniki, ki označujejo tipko, ki jo moramo pritisniti in čas, ob katerem moramo to storiti.

Drugi elementi, ki so lastni formuli 'Guitar Hero', katere naša aplikacija ne vključuje, pa so deljenje izvajanja ritma na eno roko in melodije na drugo roko, kar na klavirju ni izvedljivo; abstrakcije dejanske pesmi na enostavnejša navodila za igranje, ki so sicer podobna melodiji, a ne ustrezajo dejanskim prijemom. To nebi bilo sprejemljivo v aplikaciji, katere namen je učenje igranja na inštrument. Abstrakcija navodil za igranje in namenska poenostavitev navodil nebi služila namenu učenja inštrumenta.

Naslednji element, ki ga 'Guitar Hero' vključuje, naša aplikacija pa ne more, je nastavljiva zahtevnost. Četudi pogosto obstaja več izvedb MIDI pesmi, ki so različno zahtevne, je pesem v vsakem primeru take težavnosti, kakršne je v resnici. Ker ne moremo uporabiti abstrakcije navodil, ne moremo pesmi umetno poenostavljati.

Implementacija formule 'Guitar Hero' na klavirju ali sintetizatorju je torej zelo težavna. Komercialne izvedbe, ki so to poskušale, so naletele na iste ali podobne težave, na katere smo naleteli tudi mi, le da so se izdelovalci odločili, da bodo težave ignorirali in nadaljevali z razvojem, ugotavljanje napak pa prepustili samim uporabnikom. Naš namen pri razvoju pa je vedno bil razvoj uporabne aplikacije, ki uporablja formulo 'Guitar Hero'. Dalj časa smo se poskušali čimbolj držati formule in smo zaradi neprimernosti le-te, spreminjali načrt in namen aplikacije, a na koncu smo ugotovili, da je toliko aspektov in elementov formule nesmiselnih za uporabo s klavirjem ali sintetizatorjem, da je edina smiselna rešitev izdelava aplikacije, ki vsaj daje vtis, da je navdihnjena po igri 'Guitar Hero', četudi v resnici vključuje le malo skupnih točk.

Poglavje 7

Sklepne ugotovitve

Končni dosežek projekta je konceptualni načrt za aplikacijo, katere namen je pomoč učiteljem igranja na klavir pri pedagoški dejavnosti, z demonstracijskim programom, ki bolje prikazuje funkcije, katere naj bi tak program vključeval.

Tekom razvoja demonstracijskega programa smo prišli do ugotovitev, katere definirajo primernost uporabe metod, ki jih uporablja igra 'Guitar Hero' za aplikacijo, namenjeno učenju igranja na klavir. Ugotovitve lahko služijo potencialnemu razvijalcu, ki si zastavi cilj izdelave aplikacije, katere konceptualni cilji so podobni našim.

V delu smo se omejili na aplikacijo, ki ne uporablja abstrakcije igranja, torej pričakuje od uporabnika, da bo zaigral dejansko melodijo in ne le abstrakcijo le-te, kar je nujno potrebno za aplikacijo, katere namen je učenje.

Preizkušali smo nekaj pristopov za doseganje učenja. Prvi pristop je bil razvoj igre, drugi pristop je bil razvoj aplikacije, ki nadomesti učitelja in zadnji, najuspešnejši pristop je bil razvoj aplikacije, katere namen je pomoč učitelju igranja na klavir.

S tem smo pokrili velik segment konceptualnih ciljev, za katere bi si lahko razvijalci odločili. Delo jim nudi vpogled v težave, s katerimi se bodo srečali med razvojem. Posledično lahko vnaprej ocenijo primernost razvoja zastavljene aplikacije ter predvidijo kompromise, katere bodo prisiljeni sprejeti.

LITERATURA

[1] B. Bates, »Game Design, Second Edition«, *introduction*, str. xxi, 2004

[2] C. Crowford, »Art of Computer Game Design«, *Smooth learning curve* in *The illusion of winnability*, str. 72 – 73, 1997

[3] (2012) L. Perr, »TaskScheduler«, dostopno na: http://www.codeproject.com/Articles/38553/TaskScheduler

[4] (2012) J. Portnow, D. Floyd, A. Theus »The Skinner Box«, v seriji Extra Credits, dostopno na: http://penny-arcade.com/patv/episode/the-skinner-box

[5] (2012) E. Qualls, »Guitar Hero vs. Real Guitar«, dostopno na: http://xbox.about.com/od/gamingculture/a/Guitar-Hero-Vs-Real-Guitar_2.htm

[6] (2012) L. Sanford, »C# MIDI Toolkit«, dostopno na: http://www.codeproject.com/Articles/6228/C-MIDI-Toolkit

[7] (2012) »All About the Waterfall Model«, dostopno na: http://www.waterfall-model.com/

[8] (2012) »Zaporedni ali slapovni model (waterfall model)«, dostopno na: http://www.e-studij.si/Zaporedni_ali_slapovni_model_(waterfall_model)

[9] (2012) »MIDI Event Commands«, dostopno na: http://faydoc.tripod.com/formats/mid.htm#MIDI_Event_Commands

[10] (2012) »NoteWorthy Software, Inc.«, dostopno na: http://www.noteworthysoftware.com/

[11] (2012) »A fun way to learn how to play the piano.«, dostopno na: http://www.synthesiagame.com/

[12] (2012) »Frequencies for equal-tempered scale«, dostopno na: http://www.phy.mtu.edu/~suits/notefreqs.html