SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6699

GRAFIČKA APLIKACIJA ZA UČENJE MUZICIRANJA NA BOOMWHACKER UDARALJKAMA

Vinko Benković

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6699

GRAFIČKA APLIKACIJA ZA UČENJE MUZICIRANJA NA BOOMWHACKER UDARALJKAMA

Vinko Benković

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

ZAVRŠNI ZADATAK br. 6699

Pristupnik:	**Vinko Benković (0036502621)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	izv. prof. dr. sc. Mirko Randić

Zadatak: **Grafička aplikacija za učenje muziciranja na Boomwhacker udaraljkama**

Opis zadatka:

Boomwhacker je obojena plastična cijev koja je dužinom ugođena na određenu visinu tona. Visina tona koju proizvodi boomwhacker predstavljena je njegovom bojom. To je glazbeni instrument iz obitelji udaraljki jer se zvuk proizvodi udaranjem, često o tijelo muzičara. Uobičajeno se koristi u nastavi u osnovnim školama što znači da s boomwhackerima sviraju djeca koja još ne poznaju notni zapis. Zato bi grafička aplikacija koja povezuje zvuk i njegovo trajanje s bojom boomwhackera bila korisna djeci za učenje sviranja. Vaš je zadatak da proučite i opišete kakve aplikacije postoje za učenje muziciranja bez notnog zapisa. U praktičnom dijelu trebate razviti grafičku aplikaciju koja će djeci pomagati u učenju sviranja na boomwhackerima. Aplikaciju trebate dokumentirati.

Rok za predaju rada: 12. lipnja 2020.

SADRŽAJ

Uvod

Boomwhackers su glazbeni instrumenti iz porodice udaraljki. To su lagane i šuplje plastične cijevi koje zahvaljujući svojoj karakterističnoj duljini, prilikom udaranja stvaraju određeni ton. Također, svaka cijev je obojena drugačijom bojom tako da i boja predstavlja ton.

Instrumente najčešće sviraju veće skupine izvođača tako da udaraju o vlastito tijelo, a može se svirati i udaranjem od bilo kakvu ravnu podlogu. Uporaba Boomwhackersa je zastupljena pretežito u osnovnim školama jer zbog nedostatka vremena za učenje notnog zapisa ovi obojeni instrumenti predstavljaju brži, a ujedno dobar način kako glazbu približiti djeci.

Jednostavnost sviranja ovog instrumenta leži u tome da izvođači ne moraju pratiti notni zapis kako bi odsvirali neku skladbu. Samo je potrebno složiti redoslijed boja koji odgovara notnom zapisu skladbe te nakon toga svaki izvođač treba samo čekati svoj red da odsvira note sa cijevima koje trenutno ima u rukama. Problem koji se ovdje javlja je način predočavanja redoslijeda boja koje izvođači prate.

Kako bi se doskočilo tom problemu u ovom radu opisan je postupak razvoja aplikacije koja omogućuje korisniku unos nota koje prate neku skladbu, a koja se želi odsvirati s instrumentima Boomwhackers. Nakon toga korisniku je omogućeno pokretanje animacije koja stvara prikaz boja u redoslijedu koji je prethodno unesen notama. Animacija praćena zvukom i izmjenom boja omogućuje izvođačima lakše muziciranje s instrumentima Boomwhackers.

Aplikacija je razvijena u Pythonu. Korištena je biblioteka `Tkinter` za prikaz korisničkog sučelja i grafičkih objekata te biblioteka `Pygame` za reproduciranje zvuka.

1. Popis zahtjeva i značajke korištenih tehnologija

U ovom poglavlju su opisani zahtjevi koje aplikacija mora zadovoljiti te korištene tehnologije.

1.1 Zahtjevi

Zahtjevi koje aplikacija mora zadovoljiti su unos svih nota koje se mogu proizvesti udaraljkama Boomwhackers. Također nakon što je omogućen unos nota treba omogućiti grafički prikaz tih nota specifičnim bojama i veličinama kako bi korisnik mogao razlikovati prikaz svake cijevi i kako bi se lakše snašao prilikom muziciranja. Prilikom pokretanja animacije koja prikazuje redoslijed izvođenja skladbe s udaraljkama potrebno je omogućiti reproduciranje zvuka svake note u skladu s izmjenama boje. Kako bi se omogućila izmjena tempa izvođenja skladbe potrebno je omogućiti prilagodbu brzine kojom se mijenjaju boje te u skladu s promjenom boje treba prilagoditi dužinu trajanja pojedinog zvuka.

Osim funkcionalnih zahtjeva, potrebno je omogućiti i neke nefunkcionalne zahtjeve, a to su:

- lako i jednostavno korištenje aplikacije,
- jednostavna instalacija,
- „gladak“ rad aplikacije, bez zastajkivanja.

1.2 Značajke korištenih tehnologija

1.2.1 Tkinter

`Tkinter` je standardni modul ugrađen u Python, a služi za kreiranje grafičkih korisničkih sučelja. Ime `Tkinter` je skraćeno od `Tk interface`. `Tkinter` je praktičan jer se isti kod može pokretati na različitim platformama, a to su Windows macOS i Linux. Vizualni elementi „widgets“ kao na primjer: button, entry, label… se stvaraju izgledom prilagođeni platformi na kojoj je program pokrenut. Između ostalog `Tkinter` nudi mogućnost stvaranja widgeta `Canvas` koji nam omogućuje prikaz grafičkih objekata kao što su razni geometrijski oblici i linije.

1.2.2 Pygame

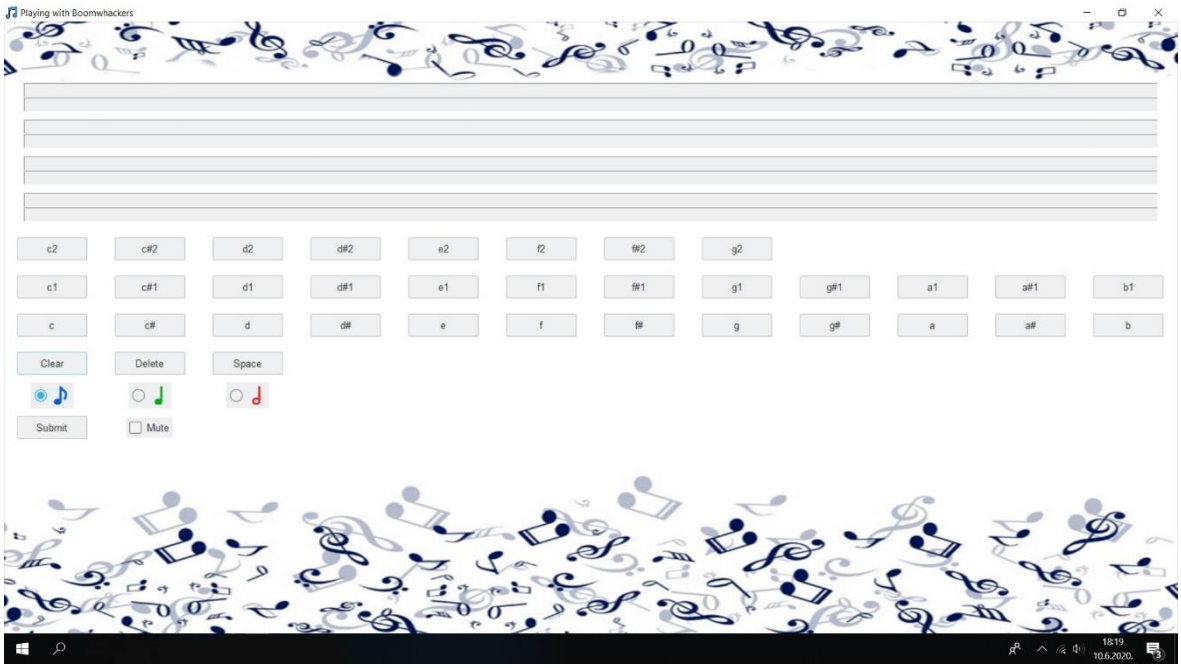
`Pygame` je skup Python-ovih modula, a dizajniran je za razvoj video igrica u Pythonu. `Pygame` se može pokretati na brojnim platformama kao što su: Windows, Linux, macOS, FreeBSD, NetBSD, Solaris, IRIX... Pygame je besplatan i pod LGPL licencom se uz pomoć `Pygame-a` može razvijati bilo kakav oblik freeware, shareware ili open source programa. Praktičan je jer za pokretanje programa razvijanog pomoću `Pygame-a` nije potrebno prethodno postaviti grafičko korisničko sučelje. Nudi brojne funkcionalnosti kao što su prikaz grafičkih modela, njihovih kretnji i međusobnih interakcija. Također podržava reproduciranje zvuka u wav formatu.

2. Implementacija programskog rješenja

U ovom poglavlju detaljno opisujemo postupak razvoja aplikacije.

2.1 Kreiranje grafičkog korisničkog sučelja

Za kreiranje grafičkog korisničkog sučelja korišten je modul `Tkinter`. Na slici 2.1.1 je prikazan izgled početnog prozora programa. Na početnom prozoru postoji osam polja(`Text widgeti`) za unos željenih nota pri čemu su prvo, treće, peto i sedmo polje povezani tako da se vrijednosti, tj. note unesene u ta polja slažu u redoslijed jedna iza druge te će se kasnije tim redoslijedom izvoditi. Isto pravilo vrijedi i za drugo, četvrto, šesto i osmo polje. Vrijednosti koje su unesene u polja (npr. prvo i drugo polje) koja se nalaze u paru, predstavljaju note koje se prilikom izvođenja sviraju u isto vrijeme. Prilikom lijevog klika miša na bilo koje polje poziva se funkcija `select` koja postavlja vrijednost zastavice `flag` na redni broj polja. Kada je zastavica postavljena za neko polje tada to polje postaje aktivno. Aktivno polje znači da će se prilikom pritiska bilo kojeg gumba vrijednost koja je napisana na gumbu unijeti u to aktivno polje.



Slika 2.1.1 Prikaz grafičkog korisničkog sučelja

Za svako stvoreno polje postavljamo dužinu na 185, širinu na 1 te font na `Fixedsys`. Korištenje fonta `Fixedsys` nam omogućuje unos znakova, pri čemu svaki znak ima jednaku širinu. To nam je bitno kako bi imali uredno poravnanje znakova koje unosimo u polja. Uz svako polje koje stvaramo je također potrebno postaviti postavke za tag-ove. Tag omogućuje promjenu fonta i boje nekih dijelova teksta iz polja. Slika 2.1.2 prikazuje primjer stvaranja jednog polja(`Text widget-a`) i popratnih akcija koje je potrebno obaviti.

```
e = Text(root, font="Fixedsys 12 ", height = 1, width = 185)
e.bind("<Button-1>", lambda event: select("0"))
e.tag_config('blue', foreground="blue")
e.tag_config('red', foreground="red")
e.tag_config('green', foreground="green")
e.grid(row = 0, column = 0, columnspan = 12, pady = (80, 0))
```

Slika 2.1.2 Stvaranje Text widget-a i popratne akcije

Kako bi omogućio unos željene skladbe kreirani su gumbi koji pokrivaju sve tonove koje instrumenti Boomwhackers mogu proizvesti. Gumbi sadrže vrijednosti osnovnih i povišenih tonova: C, C#, D, D#, E, F, F#, G, G#, A, A#, B. Svaki ton koji se može odsvirati Boomwhackers-ima se proteže kroz tri oktave. Za potrebe ovog programa odabrao sam zvukove male, prve i druge oktave. Mala oktava označava duboke tonove, dok druga oktava označava visoke tonove. Također valja napomenuti da kod Boomwhackers-a druga oktava sadrži samo tonove: C, C#, D, D#, E, F, F#, G. Iz toga razloga nema gumba za unos nota G#, A, A#, B u drugoj oktavi.

Svakom gumbu koji predstavlja notu prilikom stvaranja pridružujemo funkciju `write`. Kao argumente funkcija `write` prima vrijednost note koja se nalazi na tom gumbu te primjerak razreda `pygame.mixer.sound`. Kako bi stvorili primjerak razreda `pygame.mixer.sound` u konstruktor mu se predaje zvučna datoteka u wav formatu. U ovom programu će svaka zvučna datoteka wav formata predstavljati zvuk jednog tona koji će se predavati kao argument `write` funkcije. Funkcija `write` se poziva svaki puta kad se pritisne neki gumb za unos tona, a obavlja posao zapisivanja tonova u odabrano polje te stvaranje primjerka razreda `re`.

Prilikom upisivanja nove vrijednosti u polje treba ograničiti unos koji polje može primiti. Ograničenje je postavljeno na 185 kako broj znakova u polju ne bi bio veći od dužine samog polja. Unutar funkcije `write` se poziva funkcija `write1` koja obavlja posao pozicioniranja novih znakova koji se unose u polje. Funkcija `write1` kao argumente prima boju, objekt polja i znakove tipke koja je pritisnuta. U varijablu `t_length` se sprema dužina trenutnog sadržaja polja, te se na osnovu te dužine kreira indeks na koji umećemo novi znak unutar polja. Za umetanje novog sadržaja u polje pozivamo metodu `insert` nad poljem. U metodu `insert` kao parametar predajemo i boju koja nam omogućuje upisivanje novog sadržaja u željenoj boji.

```
def write1(color, widget, text):
    t_length = len(widget.get("1.0", "end-1c"))
    t_index = "1."+str(t_length)
    widget.insert(t_index, text, color)
```

Slika 2.1.3 Funkcija write1

U funkciji `write` se također odvija stvaranje primjerka razreda `re`. Razred `re` kao argumente u konstruktor prima vrijednost note za koju je pritisnut gumb, primjerak razreda `pygame.mixer.sound` te `integer` koji predstavlja duljinu trajanja zvuka tona. Nakon što je primjerak razreda `re` stvoren dodajemo ga u listu `lista` ako se vrijednost tona unosi u prvo, treće, peto ili sedmo polje. Dok u listu `lista1` primjerak razreda `re` dodajemo ako se vrijednost tona unosi u drugo, četvrto, šesto ili osmo polje. Slika 2.1.4 prikazuje isječak koda iz funkcije `write`.

Prethodno je spomenuta duljinu trajanja tona koji se u obliku `integera` predaje u konstruktoru razreda `re`. Kako bi svakom tonu kojeg želimo upisati u zato predviđeno polje pridijelili duljinu trajanja, aplikacija nudi opciju odabira `Radiobuttona`. Svaki `Radiobutton` uz sebe ima simbolički prikaz note koji označava njeno trajanje. Kada gledamo s lijeva na desno prvo ide osminka, četvrtinka i polovinka. Kako bi se željena duljina trajanja pridijelila nekom tonu valja voditi računa da prije odabira gumba na kojem se nalazi ton postavimo `Raddiobutton` na željenu duljinu trajanja. Na primjer ako želimo dodati ton C duljine trajanja polovinke prvo je potrebno postaviti `Radiobutton` na simbol polovinke i zatim odabrati gumb na kojem se nalazi ton C. U slučaju da se `Radiobutton` nakon pokretanja aplikacije ne mijenja, njegov položaj će biti postavljen na duljinu trajanja osminke.

Svaki od simbola za duljinu trajanja je prikazan drugačijom bojom, tj. bojama: plava, zelena i crvena. Plava boja predstavlja osminku, zelena boja predstavlja četvrtinku i crvena boja predstavlja polovinku. Razlog zašto su ovi simboli prikazani drugačijim bojama je taj što želimo da se uneseni tonovi u poljima mogu razlikovati po duljini trajanja. Tako će na primjer tonovi za koje je `Radiobutton` bio postavljen na osminku svi biti plave boje kada se budu prikazivali u polju za unos tonova.

```
if flag == "0" and len(e.get("1.0", "end-1c")) < 185:
    if r.get()==1:
        write1('blue', e, sy)
    if r.get()==2:
        write1('green', e, sy)
    if r.get()==4:
        write1('red', e, sy)
    a = re(sy, r.get(), noteSnd)
    lista.append(a)
```

Slika 2.1.4 Isječak koda iz funkcije write

U isječku koda prikazanom na slici 2.1.4 možemo vidjeti kako se ovisno o vrijednosti varijable `r Radiobuttona` poziva funkcija `write1` s drugačijom bojom kao argumentom. Boje crvena, zelena i plava nemaju nikakvu simboliku u glazbi. Jedini razlog zašto su odabrane je taj što imaju dobar kontrast i omogućuju korisniku da može lako razlikovati duljina trajanja tonova koji su uneseni u polja za unos tonova.

Kako bi odabranu vrijednost `Radiobuttona` predali konstruktoru razreda `re` potrebno je prilikom stvaranja `Radiobuttona` stvoriti i varijablu tipa `IntVar`. Tu istu varijablu tipa `IntVar` je potrebno predati svakom primjerku stvorenog `Radiouttona` kako bi omogućili da svaki `Radiobutton` može postaviti drugačiju duljinu trajanja tona. Stvaranje `Radiobutttona` te pridruživanje slikovnih elemenata uz `Radiobuttone` prikazano je na slici 2.1.5.

```
r = IntVar()
r.set(1)
image1 = PhotoImage(file="1plava.png")
image2 = PhotoImage(file="1zel.png")
image3 = PhotoImage(file="4crvena.png")
rad1 = ttk.Radiobutton(root, text="1 sec", image = image1, variable=r, value=1)
rad2 = ttk.Radiobutton(root, text="2 sec", image = image2, variable=r, value=2)
rad3 = ttk.Radiobutton(root, text="4 sec", image = image3, variable=r, value=4)

rad1.grid(row = 12, column = 0)
rad2.grid(row = 12, column = 1)
rad3.grid(row = 12, column = 2)
```

Slika 2.1.5 Stvaranje RadioButton-a

Osim tipki za unos tonova imamo još i tipku Space koja nam služi za umetanje praznine, točnije ona pri pokretanju animacije predstavlja razdoblje tijekom kojega se ne treba ništa svirati. Za gumb Space je također omogućeno postavljanje duljine trajanja pomoću `Radiobuttona` tako da možemo umetati tri različite vrste praznina. Prilikom pritiska gumba Space prikaz praznine će biti označen velikim slovom X u poljima za unos tonova te će poprimati boju koju definira duljina trajanja. Svaki puta kada se pritisne gumb Space poziva se funkcija `space_com` koja obavlja istu funkcionalnost kao što je obavljala i funkcija `write`. Jedina razlika je što se prilikom poziva funkcije `space_com` stvara primjerak razreda `re` koji kao treći parametar u konstruktoru prima vrijednost `None`, dok je u funkciji `write` na tom mjestu bio primjerak razreda `pygame.mixer.sound`. Razlog zašto se predaje `None` je taj što gumb Space treba stvarati praznine koje u kasnijoj implementaciji ne će proizvoditi nikakav zvuk.

Prilikom unosa oznake tona u polje, korisnik može napraviti pogrešku te unijeti krivu oznaku. Kako bi korisnik mogao ispraviti pogrešku koju je napravio, program sadrži dva gumba koja to omogućuju. Gumb Delete omogućuje korisniku brisanje elemenata koji se nalaze u polju za unos tonova. Svaki put kada se pritisne gumb Delete poziva se funkcija `de` koja briše jedan ton iz odabranog polja. Funkcija `de` poziva funkciju `de1` kojoj kao argument predaje polje u kojem se vrši brisanje. Funkcija `de1` obavlja brisanje pozivom metode `delete` nad poljem. Kako bi metoda `delete` obrisala samo posljednji element polja, kao prvi argument joj predajemo indeks elemenata koji se nalazi šesti od kraja polja, a kao drugi argument postavljamo oznaku `end` koja označava brisanje do kraja. Ova dva argumenta zapravo predstavljaju raspon u kojem će se obavit brisanje polja. Prvi argument smo dobili oduzimanjem dužine sadržaja polja s pet jer je svaka oznaka tona koja se unosi u polje dugačka pet znakova.

```
def de1(widget):
    d_length = len(widget.get("1.0", "end-1c")) - 5
    d_index = "1."+str(d_length)
    widget.delete(d_index, "end")
```

Slika 2.1.6 Funkcija de1

Brisanje tonova tipkom Delete je omogućeno samo za zadnje znakove polja. Na primjer ako imamo niz znakova C D F G znak D neće moći biti obrisan sve dok se prije njega ne obrišu znakovi G i F. Svaki znak koji se briše iz polja također se izbacuje i iz lista: `lista` i `lista1` ovisno o tome iz kojeg polja se znak briše. Također prije brisanja potrebno je provjeriti zastavicu `flag` preko koje saznajemo iz kojeg polja se briše podatak. Uz provjeru zastavice `flag` potrebno je provjeriti i popunjenost polja koje se nadovezuje na polje iz koje brišemo oznaku. Isto tako moramo provjeriti postoji li ikakav podatak u polju iz kojeg brišemo. Isječak koda iz funkcije `de` koji obavlja brisanje nad poljem 3 prikazan je na slici 2.1.7. Tu vidimo uvjet koji ispituje - postoji li sadržaj u polju 3 koji se može izbrisati i je li polje 5 prazno. Ako su uvjeti zadovoljeni provodi se brisanje.

```
if flag == "3" and len(e3.get("1.0", "end-1c")) >= 5 and len(e5.get("1.0", "end-1c")) == 0:
    de1(e3)
    lista1.pop()
```

Slika 2.1.7 Isječak koda funkcije de

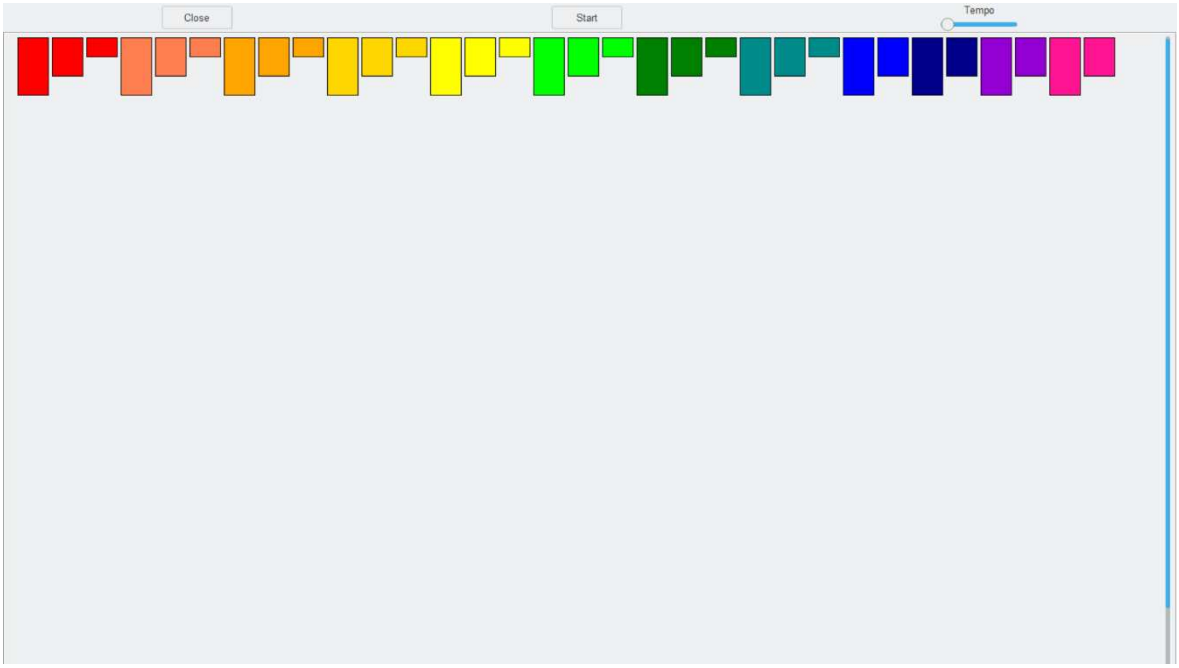
Osim gumba Delete za korekciju unosa imamo još i gumb Clear. Gumb Clear omogućuje korisniku da obriše sav sadržaj međusobno povezanih polja za unos. Na primjer ako odaberemo polje četiri i pritisnemo Clear, obrisat će se sadržaj polja četiri, ali isto tako i polja dva, šest i osam. Prilikom pritiska gumba Clear poziva se funkcija `cl` koja radi brisanje kao što je prethodno opisano, ali također uz to obavlja pražnjenje lista: `lista` i `lista1` ovisno o polju koje je odabrano. Isječak koda iz funkcije `cl` koji prikazuje brisanje je prikazan na slici 2.1.8.

```
if flag == "0" or flag == "2" or flag == "4" or flag == "6":
    e.delete("1.0", END)
    e2.delete("1.0", END)
    e4.delete("1.0", END)
    e6.delete("1.0", END)
    lista = []
```

Slika 2.1.8 Isječak koda funkcije cl

2.2 Implementacija grafičkog prikaza tonova

Nakon što smo unijeli nizove tonova u za to predviđena polja, pritiskom na tipku Submit pokrećemo funkciju `creat` koja stvara novi prozor u full screenu. Na novo stvorenom prozoru možemo uočiti dva gumba pri vrhu ekrana te jednu skalu. Ispod ova tri elementa na novom prozoru imamo još i okvir(`frame`) u kojem je stvoren widget(vizualni element) `Canvas`. Na `Canvasu` ćemo prikazivati sve prethodno unesene tonove u grafičkom obliku. Funkcija `creat` prolazi kroz liste: `lista` i `lista1` te stvara primjerke razreda `rect`. Razred `rect` u konstruktoru prima koordinate i boju pravokutnika koji se potom crta. Uz osobine pravokutnika razred `rect` u konstruktoru također prima primjerak razreda `pygame.mixer.sound` te dužinu trajanja tona, koji nam kasnije omogućuje reproduciranje zvuka određene dužine trajanja. U slučaju da se u listama: `lista` i `lista1` nalazi prazan znak X, taj znak će rezultirati stvaranjem primjerka razreda `empty` koji u konstruktor prima samo duljinu trajanja praznine, odnosno vrijeme kada se zvuk neće reproducirati. Primjerak razreda `empty` ne rezultira crtanjem nikakvih oblika te će svaki takav znak X na `Canavas-u` biti prikazan prazninom koja ima širinu jednog pravokutnika.

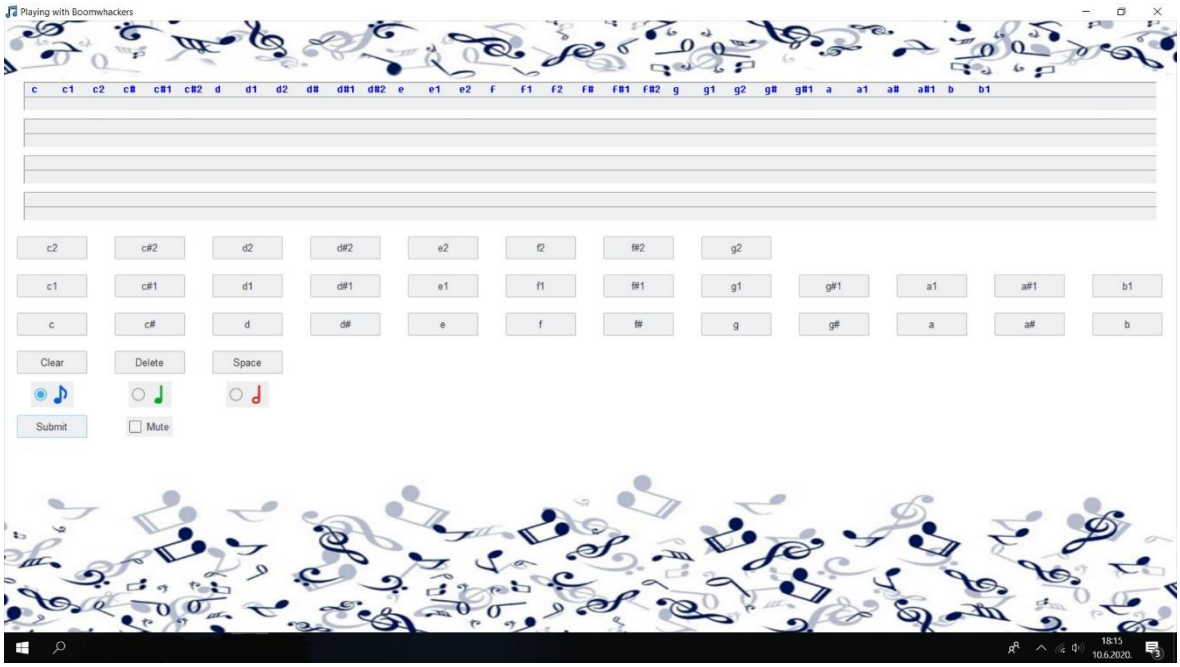


Slika 2.2.1 Prikaz svih boja tonova BoomWhackers-a

Primjerci razreda `rect` i `empty` se dodaju u dvije nove liste `boja` i `boja1` koje ćemo kasnije predati funkciji `update_sc` koja pokreće animaciju. Stvaramo dvije liste `boja` i `boja1` kako bih kasnije mogli lakše pratiti primjerke razreda `rect` i `empty` koji se izvode paralelno, odnosno u isto vrijeme. Nacrtani pravokutnici će biti prikazani u parovima, pri čemu svaki par pravokutnika predstavlja tonove koji se izvode u isto vrijeme. Također valja spomenuti da se može dogoditi da se u paru pojave praznina i pravokutnik, ali isto tako i dvije praznine. Pravokutnici mogu biti obojeni u dvanaest različitih boja pri čemu svaka boja predstavlja jedan ton. Tonovi i za njih pripadajuće boje su:

- C
- C#
- D
- D#
- E
- F
- F#
- G
- G#
- A
- A#
- B

Tonovi prilikom grafičkog prikazivanja mogu imati tri različite veličine koje su određene trajanjem pojedinog tona. Kod Boomwhackers-a duža cijev predstavlja niži ton dok kraća cijev predstavlja viši ton. Iz tog razloga je i ovaj program implementiran istom logikom tako da će tonovi najviše oktave biti prikazani pravokutnicima najmanje dužine, tonovi srednje oktave bit će prikazani pravokutnicima srednje dužine, a tonovi najniže oktave će biti prikazani pravokutnicima najveće dužine. Prikaz svih tonova koji su uneseni na slici 2.2.2 rezultira grafičkim prikazom koji je prikazan na slici 2.2.1.



Slika 2.2.2 Tonovi uneseni u polje

Na slici 2.2.1 možemo uočiti gumb Close za zatvaranje novog otvorenog prozora, a samim time uzrokuje povratak na osnovni početni prozor gdje smo unosili tonove. Kako bi omogućili tu funkcionalnost, svakim pritiskom gumba Close se poziva metoda `destroy` nad novo stvorenim prozorom.

Drugi gumb na novom prozoru je gumb Start koji nam služi za pokretanje animacije. Prilikom pritiska gumba Start pokreće se nova dretva koja obavlja funkciju `update_sc`. Razlog zašto pokrećemo novu dretvu je taj što unutar funkcije `update_sc` imamo poziv metode `pygame.time.delay` koja zaustavlja trenutnu dretvu na određeno vrijeme. Kako bi usprkos tome sve ostale funkcionalnosti na novom prozoru nesmetano radile potrebna nam je ova nova dretva koju ćemo moći pauzirati bez da to utječe na glavnu dretvu. Funkcija `update_sc` prolazi kroz liste: `boja` i `boja1` te za elemente na istom indeksu obje liste provjerava koje su boje ti elementi. Nakon što odredi koje su boje elementi na trenutnom indeksu radi se promjena boje elemenata u novu boju koja je za nijansu svjetlija od trenutne boje. Promjenu boje obavlja funkcija `change_col` koja se poziva nad primjerkom razreda `rect`. Nakon što promijenimo boje za trenutna dva elementa potrebno je ažurirati `fram-e` na kojem se pravokutnici iscrtavaju kako bi mogli uočiti promjenu boje.

Ažuriranje `frame-a` se radi pozivom metode `update` nad `frame-om` koji ažuriramo. Smisao ovih promjena boja je zapravo predočiti koje su cijevi na redu za sviranje, točnije pravokutnik na kojem se dogodi promjena boje predstavlja cijev koja se tog trena mora odsvirati. Ovakav način prikaza slijeda izvođenja olakšava izvođačima praćenje boje koja je na redu tijekom muziciranja. Nakon što je obavljena promjena boja za dva pravokutnika koja su bila na redu funkcija `update_sc` pokreće reprodukciju zvuka. Kako je svaki grafički prikazan pravokutnik predstavljen razredom `rect` koji kao privatnu varijablu sadrži primjerak razreda `pygame.mixer.sound` moguće je nad svakim pravokutnikom reproducirati njegov zvuk. Za pokretanje zvuka potrebno je pozvati naredbu:

```
pygame.mixer.find_channel().play(i[0].snd, maxtime=i[0].dur * par)
```

Kako bi se reprodukcija svakog zvuka izvodila neometano potrebno je pozvati metodu `find_channel` koja pronalazi `channel` koji je slobodan te se u njemu ne izvodi niti jedan zvuk. Nakon toga se poziva metoda `play` kojoj kao argument predajemo primjerak razreda `pygame.mixer.sound` i maksimalno vrijeme trajanja reproduciranog zvuka. Kada se pokrene reprodukcija zvuka pojedinog tona potrebno je zaustaviti dretvu u vremenskom razdoblju koje odgovara vremenu trajanja tog tona. Zaustavljanje dretve se obavlja pomoću metode `pygame.time.delay` kojoj se kao argument predaje vrijeme u milisekundama tijekom kojega je dretva zaustavljena. Ovaj proces zaustavljanja je bitan kako bi se ograničio prelazak na sljedeći par pravokutnika sve dok pravokutnici koji su trenutno aktivni reproduciraju svoj zvuk.

```
if isinstance(i[0], rect) and isinstance(i[1], rect):
    if var.get() == 0:
        pygame.mixer.find_channel().play(i[1].snd, maxtime=i[1].dur * par)
        pygame.mixer.find_channel().play(i[0].snd, maxtime=i[0].dur * par)
if isinstance(i[0], empty) and isinstance(i[1], rect):
    if var.get() == 0:
        pygame.mixer.find_channel().play(i[1].snd, maxtime=i[1].dur * par)
if isinstance(i[1], empty) and isinstance(i[0], rect):
    if var.get() == 0:
        pygame.mixer.find_channel().play(i[0].snd, maxtime=i[0].dur * par)
if i[0].dur > i[1].dur:
    pygame.time.delay(i[1].dur * par)
else:
    pygame.time.delay(i[0].dur * par)
```

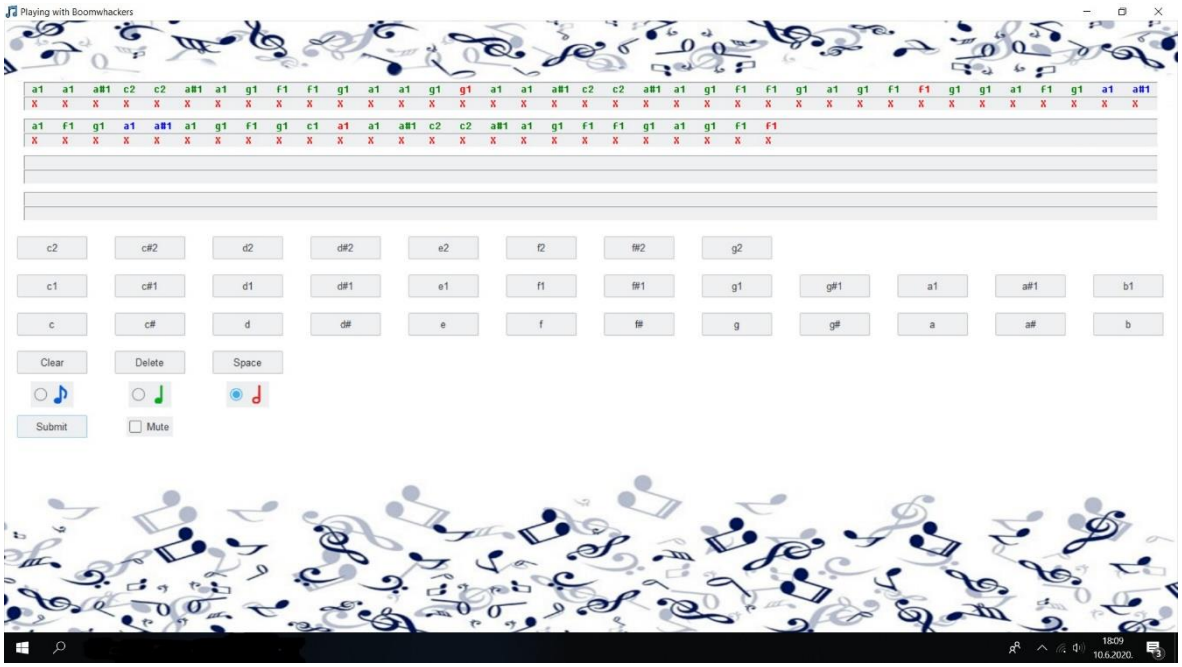
Slika 2.2.3 Isječak koda funkcije update_sc

Na slici 2.2.3 vidimo isječak koda iz funkcije `update_sc` gdje se obavlja reproduciranje zvuka i odmah zatim zaustavljanje dretve. Također možemo vidjeti da imamo tri slučaja, odnosno može se dogoditi da se u paru nađu dva pravokutnika(razred `rect`), dvije praznine(razred `empty`) te ravnina i pravokutnik. Kod zaustavljanja dretve bitno je uočiti da se dretva zaustavlja onoliko vremena koliko traje reproduciranje vremenski kraćeg tona. Prethodno navedeni uvjet je bitan jer on omogućuje pravilnu izmjenu boja pravokutnika za vrijeme pokrenute animacije. Na primjer može se dogoditi da se neki ton koji ima trajanje polovinka izvodi tijekom vremena izvođenja četiri tona koji imaju trajanje osminka.

Na novom prozoru imamo i skalu Tempo koja nam omogućuje ubrzavanje osnovnog tempa pet puta. Skalu možemo mijenjati prije pokretanja animacije ili za vrijeme izvođenja animacije. Svaki puta kada se skala pomakne poziva se funkcija `set_par` koja obavlja ažuriranje varijable tipa `Integer` na novu postavljenu vrijednost skale. Skala ima raspon od 1000 do 200, a prije bilo kakve promjene početna vrijednost skale je postavljena na 1000. Na slici 2.2.3 možemo uočiti da se prilikom pokretanja reprodukcije zvuka parametar `maxtime` koji predstavlja duljinu trajanja reprodukcije zvuka računa na način da se množi trajanje zvuka(varijabla `dur`) koje sadrži primjerak razred `rect` i varijabla `par`. Varijabla `par` predstavlja globalnu varijablu koja se ažurira pomicanjem skale u funkciji `set_par`.

Kako parametar `maxtime` prima vrijeme u milisekundama tako će promjena varijable `par` prilikom promjene od 1000 do 200 omogućiti ubrzanje tempa pet puta. Ako pogledamo sliku 2.2.3 možemo uočiti da se reprodukcija zvuka pokreće tek kada je vrijednost varijable `var` postavljena na nula. Varijabla `var` je povezana sa Checkbutton-om Mute na početnom prozoru. U slučaju kad je Mute označen varijabla `var` će se postaviti na vrijednost jedan što će rezultirati pokretanje animacije bez zvuka pritiskom na gumb Submit.

Na slici 2.2.4 imamo prikaz tonova za ulomak iz Ode radosti, a na slici 2.2.5 imamo grafički prikaz prethodno unesenih tonova.



Slika 2.2.4 Unos tonova za Odu radosti

Slika 2.2.5 Grafički prikaz tonova Ode radosti

Zaključak

Cilj ovoga završnog rada je bio napraviti grafičku aplikaciju za pomoć pri učenju sviranja instrumenata Boomwhackers. Aplikacija je razvijana u Python-u, preciznije za razvoj korisničkog grafičkog sustava i za prikaz grafičkih elemenata je korišten modul Tkinter, dok je za reproduciranje zvuka korišten modul Pygame.

Aplikacija je podijeljena u dva dijela. Prvi dio aplikacije čini prozor na kojem je prikazano korisničko sučelje. Ovaj dio aplikacije korisniku omogućuje proizvoljan unos tonova, te ispravljanje grešaka koje se dogode tijekom unosa.

Drugi dio aplikacije čini novi prozor na kojem korisnik može vidjeti grafički prikaz svih tonova koje je prethodno unosio te mu je omogućeno pokretanje animacije koja radi promjene boja na pravokutnicima kako bi se mogao pratiti redoslijed sviranja instrumenata. Aplikacije također nudi mogućnost pokretanja animacije bez zvuka te prilagodbu tempa izvođenja tonova.

Literatura:

[1] pygame.mixer, Pygame, poveznica: https://www.pygame.org/docs/ref/mixer.html

[2] Burkhard A. Meier. *Python GUI Programming Cookbook*. 2. izdanje. Birmingham: Packt Publishing, 2017.

[3] *An Introduction to Tkinter* (Work in Progress), poveznica: https://effbot.org/tkinterbook/

[4] Alan D. Moore. *Python GUI Programming with Tkinter.* 1. izdanje. Birmingham: Packt Publishing, 2018.

[5] Pygame.time, Pygame, poveznica: https://www.pygame.org/docs/ref/time.html

Sažetak

Boomwhackers su glazbeni instrument iz porodice udaraljki. Instrument se sastoji od šupljih, plastičnih, obojenih cijevi različite duljine. Dužina i boja cijevi predstavljaju ton. Cijevi su ugođene tako da mogu svirati tonove iz tri oktave, pri čemu cijevi najveće dužine proizvode najniže tonove. U završnom radu je detaljno opisan razvoj aplikacije koja omogućuje unos tonova melodije, nakon čega korisnik pokretanjem animacije dobiva prikaz tonova predstavljenih bojama. Prilikom izrade aplikacije korišten je modul Tkinter za razvoj grafičkog korisničkog sučelja te za prikaz grafičkih objekata, dok je za reproduciranje zvuka korišten modul Pygame.

Ključne riječi: Boomwhackers, boje, zvuk, Pygame, Tkinter, ton, grafičko korisničko sučelje.

Summary

Boomwhackers are musical instrument from the percussion family. The instrument consist of hollow, plastic, colored tubes of various lengths. Color and length of the tube represent tone. Tubes can play tones from three octaves, while the longest tubes produces the lowest tones. This paper describes development of application that allows you to enter tones, and run animation. Animation make representation of tones in graphic form. The application uses modul Python Tkinter for making graphical user interface, and modul Pygame for music reproduction.

Keywords: Boomwhackers, colors, sound, Pygame, Tkinter, tone, graphical user interface.