FUNTASTIC

Tehnička dokumentacija

Autor:

Matko Zainul Kurcinak

Mentor:

Snježana Mihočka

Sadržaj

Sadržaj **2**

1. UVOD **4**
1.1. O autoru 4
1.2. Ideja i smisao projekta 4

2. DETALJAN OPIS PROJEKTA **5**
2.1. Glavni način korištenja 5
2.1.1. Frame LOBBY 6
2.1.2. Frame GAME 1 6
2.1.3. Frame DOUBLE or NOTHING? 7
2.1.4. Frame GAME 2 7
2.1.5. Frame SCORE 8
2.2. Rukovanje greškama 8
2.2.1. Poteškoće unutar LOBBY 8
2.2.1.1. Nije odabran niti jedan igrač ili maksimalan broj igara 8
2.2.1.2. Igrači nisu upisani redom 9
2.2.1.3. Vrlo duga imena igrača 9
2.2.2. Poteškoće unutar GAME 1 9
2.2.2.1. Ponovno biranje istog broja 9
2.2.2.2. Prebrzo biranje brojeva 9
2.2.2.3. Biranje sljedećeg broja suprotno od naputka voditelja 10
2.2.2.4. Prilagodba zvukovnih poruka 10
2.2.3. Poteškoće unutar DOUBLE or NOTHING 11
2.2.3.1. Gašenje prozora bez tipke “Quit” 11
2.2.4. Poteškoće unutar GAME 2 11
2.2.4.1. Pogrešni odgovori na ukupno 3 pitanja 11
2.2.5. Poteškoće unutar SCORE 12
2.2.4.1. Ponovni prelazak u LOBBY 12
2.3. Upute za korištenje 12

3. TEHNIČKA DOKUMENTACIJA **12**
3.1. Glasovna obilježja programa 12
3.1.1. Generiranje glasovnih zapisa 12
3.2. Vizualna obilježja programa 13
3.2.1. Generiranje nijansi boja koje međusobno odgovaraju 13
3.3. Upravljanje procesima 14
3.3.1. Metoda wait_variable() 14
3.4. Konvencija o imenovanju 16
3.4.1. Google stil 16
3.5. Problemi prilikom upravljanja modulima 17
3.5.1. Slika kao pozadina widgeta 17
3.5.2. Pozicioniranje widgeta 17
3.5.3. Metoda playsound() 17

3.5.4. Metoda RadialProgressbar3D() 18
3.5.4. Metoda Playsound() 18
3.5.5. Python i operacijski sustavi 18
3.6. Povijest programa 18
3.6.1. Guess-A-Number (Funtastic - Alpha) 19
3.6.2. Funtastic - Beta 19
3.6.3. Funtastic - v1.0 19

4. BUDUĆNOST PROJEKTA 19

5. ZAKLJUČAK 20

6. LITERATURA 20

1. UVOD

1.1. **O autoru**

Moje ime je Matko Zainul Kurcinak i učenik sam Osnovne škole “Đuro Ester” u Koprivnici.

Volim učiti o tome kako današnji svijet komunicira, kako se tehnologija razvija, na koji način računala obrađuju podatke i time stvaraju potpuno nova znanja i vještine. Volim zamišljati moguće smjerove u kojima će se razvijati znanost u budućnosti te bih volio i sam doprinijeti nekom pozitivnijem razvoju događaja u barem jednoj od brojnih znanstvenih disciplina.

Informatika, kao nastavni predmet, moja mentorica i jedan član moje obitelji pobudili su u meni veliku znatiželju da naučim više o programiranju. Tako sam i počeo s realizacijom svojih prvih ideja u programskom jeziku Python.

Također mi je pomogao i internet; ponajviše StackOverflow, Quora, Reddit i StackExchange.

1.2. **Ideja i smisao projekta**

Mogućnosti interakcija između računala i čovjeka su brojne i međusobno vrlo različite. Otkako sam počeo eksperimentirati s programskim jezikom Python, shvatio sam kako interakcija između čovjeka i stroja uvelike ovisi o znanju i kreativnosti svakog programera. Od trenutka kada sam prvi put shvatio tu činjenicu, razmišljam o ovom programu.

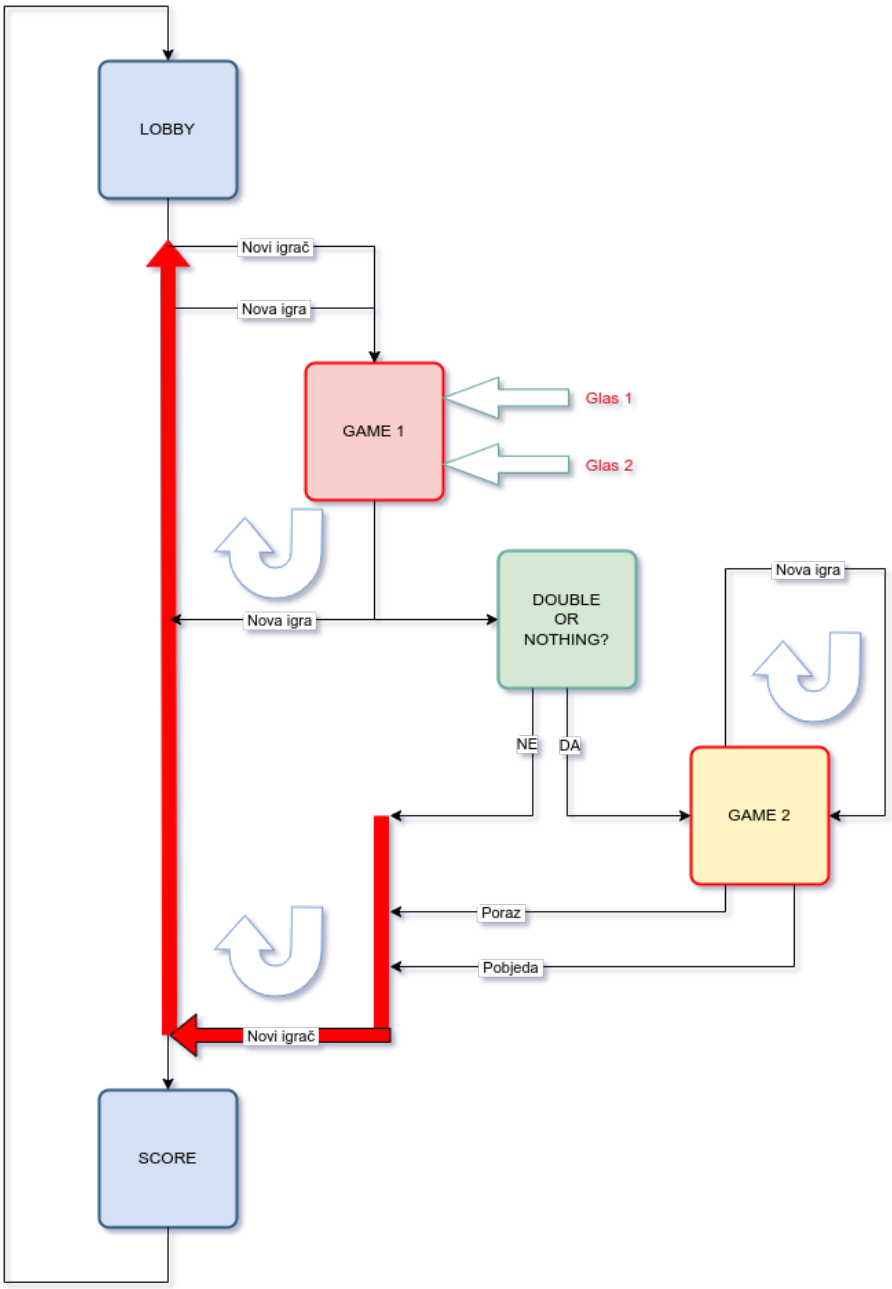
Došao sam na ideju kako bi komunikacija između računala i čovjeka mogla biti puno bliža načinu na koji i ljudi međusobno komuniciraju. Pokušao sam demonstrirati mogućnosti programskog jezika Python kroz jednostavnu igru. Pravi smisao programa bio je dočarati kako Python uz određen stupanj kreativnosti može služiti i za zabavu govornim uputama nakon svakog poteza igrača, koje su neočekivano ležerne po stilu, ali i smisleno ispravne (kroz rječnik koji je bliži svakodnevnom govoru). Sve se istovremeno odvija kroz prikaz stotinu nijansi boja koje međusobno odlično odgovaraju (i djeluju pomalo hipnotički), a iza svake se nalazi jednostavan algoritam za slaganje nijansi, kao i kroz natjecateljske elemente bodovanjem uspjeha svakog od igrača, što unosi potrebnu dinamiku. Cilj mi je bio utjecati na veći broj osjetila, eksperimentirati i učiniti interakciju kroz igru zabavnom i pomalo neočekivanom.

Smisao jedne ovakve aplikacije je zabaviti igrače. S obzirom da je veliki naglasak na posebnosti u zvukovnom odazivu aplikacije, uz odgovarajuću nadogradnju programa da podržava i **zvučne ulaze** putem mikrofona, ovo bi mogao postati i jedan zabavan način provođenja slobodnog vremena za slijepe i slabovidne osobe, što bi mi bilo izuzetno drago.

2. DETALJAN OPIS PROJEKTA

2.1. **Glavni način korištenja**

Program Funtastic čine dvije različite igre namijenjene za više igrača, koristi 2 voditeljska glasa, 200 unikatnih zvukova, 200 različitih pitanja, 800 mogućih odgovora, uz proizvoljan izbor tijeka izvođenja programa. Program je napisan u programskom jeziku Python i koristi grafičko sučelje Tkinter. Kostur programa sačinjavaju 5 *frameova* koji su prikazani na dijagramu tijeka (*Slika 1*): LOBBY se koristi za odabir maksimalnog broja igara i unosa imena svih igrača, u *frameu* GAME 1 se odvija računalna igrica pogađanja brojeva pri čemu se skupljaju osnovni bodovi, u *frameu* "DOUBLE or NOTHING?" igrač može odlučiti i želi li udvostručiti svoje bodove igranjem kviza u igri br. 2, koja se igra u idućem *frameu*, dok se u *frameu* SCORE na samom kraju igre prikazuje ukupan rezultat.



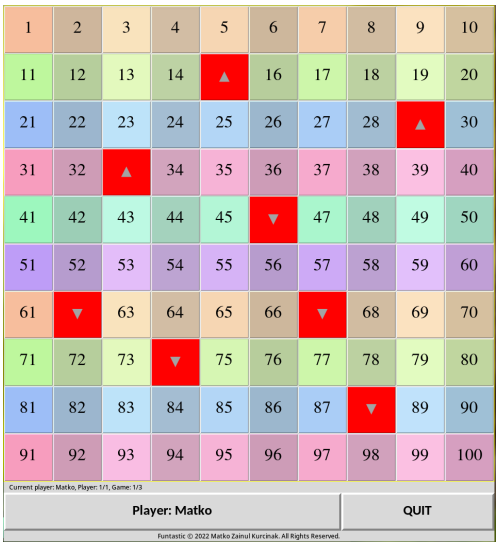
Slika 1. Dijagram tijeka

2.1.1. **Frame *LOBBY***

U pozadini ovog *framea* nalazi se slika koja se svojom dimenzijom prilagođava veličini vidljivog dijela prozora, što je praktično kod prikaza na različitim rezolucijama zaslona. U lijevom donjem kutu nalazi se prostor za upis imena igrača. Imena se mogu upisati u bilo koju rubriku, tj. ako igra samo jedan igrač, svoje ime može upisati na bilo koju poziciju. Idući odabir je maksimalan broj igranja po igraču: 3, 6 ili 9. Minimalan uvjet za prelazak u idući *frame* je postojanje barem jednog upisanog imena igrača i odabir broja igranja, nakon čega tipka "PLAY" postaje dostupna za odabir, a do tada je neaktivna. Igrač može izaći iz igrice u bilo kojem *frameu* i bilo kojem trenutku jer je gumb "QUIT" uvijek aktivan.

2.1.2. **Frame *GAME 1***

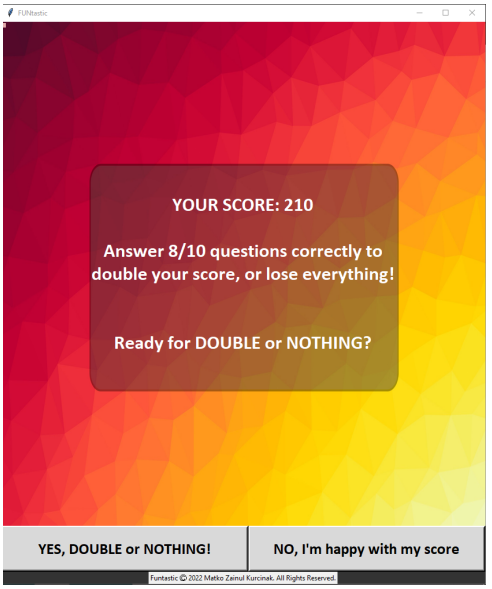
U ovom *frameu* odvija se veći dio programa. Na samome početku igra generira nasumično odabranu cjelobrojnu vrijednost između 1 i 100, zatim iscrtava polje tipki u formatu 10x10 koje se prilagođavaju ukupnoj dimenziji prozora. Svaki igrač ima najviše 9 pokušaja da pogodi prethodno generirani broj, a računalo vrednuje rezultat pogađanja i daje zvučni komentar ovisno o kontekstu razvoja situacije. Na primjer, ukoliko je traženi broj manji ili veći od pritisnutog, program daje vizualnu i zvučnu obavijest o tome je li broj manji, odnosno veći. Pravi "*show*" nastaje u trenucima kada igrač previše puta pokušava pogoditi, ili ako se, npr. ne obazire na instrukcije aplikacije i, umjesto manjeg, odabere veći broj ili obrnuto. Audio komentarima sam pokušao oživjeti karaktere dviju zamišljenih voditeljica: jedna od njih ima dobroćudan karakter, dok je druga sasvim suprotna te ima zajedljive komentare, kako bih ciljano stvorio komičnu situaciju, oraspoložio igrače, ili ih barem potaknuo da se zainate i osmisle bolju strategiju za pogađanje broja. Zamisao mi je bila da se generirani komentari što rjeđe ponavljaju i da budu prirodniji ili barem ne baš "tipično robotski".



Brojevi za odabir napravljeni su u obliku tipki te svaki ima prilagođenu pozadinu, kako bi se sve nijanse u cijelosti doimale skladno i harmonično. Nakon svakog pokušaja pritisnuti broj više se ne može ponovno odabrati, tj. gumb postaje neaktivan, sve do novog ciklusa pogađanja. Kraća statistika o trenutnom igraču, stalno je vidljiva na zaslonu (pri dnu ekrana), kako ne bi došlo do zabune oko toga tko je na redu.

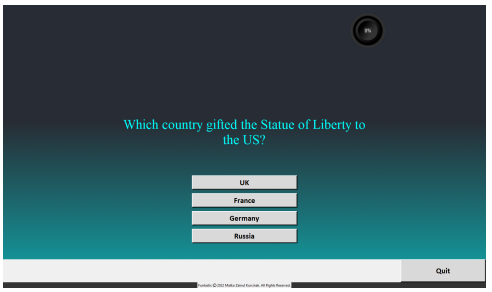
2.1.3. **Frame *DOUBLE or NOTHING?***

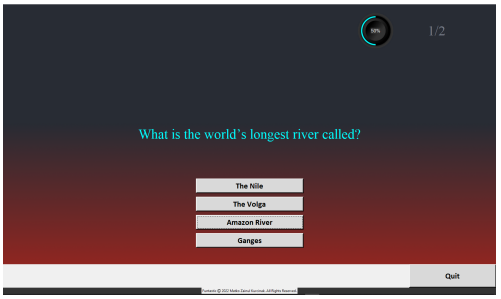
Do ovog *framea* se dolazi nakon što je igrač završio s igrom 1 (GAME 1). *Frame* služi kao upitnik, tj. kako bi korisnik mogao odlučiti želi li pokušati udvostručiti svoje bodove igrajući novu igru - kviz od 10 pitanja. Ako točno odgovori na barem 80% pitanja (8 od 10 ili više), bodovi će mu se udvostručiti, no čim pogriješi na 3 pitanja, kviz se automatski prekida i dodjeljuje mu se 0 bodova. Ako korisnik odabere da želi pokušati odigrati kviz, odabir "YES, DOUBLE or NOTHING!" otvara se *frame* ***GAME 2***, no ako ipak odluči zadržati postojeće bodove, treba kliknuti na "NO, I'm happy with my score", nakon čega igru nastavlja sljedeći igrač. Ukoliko nema drugih igrača, nakon klika na ovaj gumb, otvara se frame ***SCORE.***



2.1.4. **Frame *GAME 2***

Frame ***GAME 2*** je kratak kviz od 10 pitanja općeg znanja. *Frame* ***GAME 2*** odvija se generiranjem nasumičnog pitanja koja sam prethodno unio u program. Točnije, upisao sam 200 mogućih pitanja i pridružio im 800 mogućih odgovora! Dodao sam tzv. "*Progress Bar*" (vidi u ***LITERATURA***) koji korisnika obavještava o trenutnom postotku uspješno odgovorenih pitanja. Nakon svakog odabira ponuđenog odgovora, korisnik može saznati je li njegov odgovor točan ili netočan i prema zvučnom signalu, ali i prema boji *framea*.



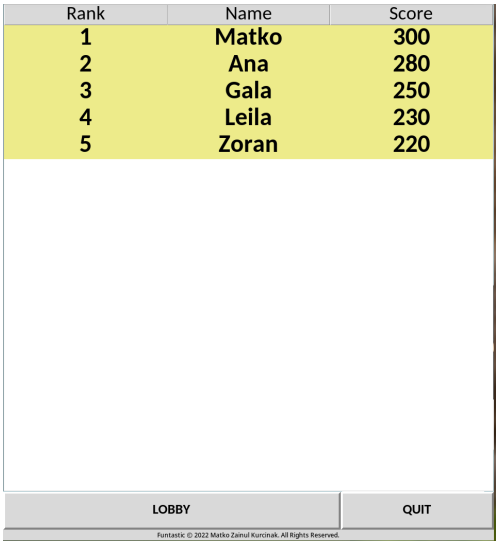


Ovdje treba napomenuti kako se sva iskorištena pitanja zapisuju u posebnu listu radi toga da se pitanja ne bi tako često ponavljala. Program prvo izvuče jedno nasumično pitanje iz liste svih pitanja, zatim provjeri je li se to pitanje već postavilo, a ako jest, izvlači novo nasumično pitanje. Ukoliko lista izvučenih pitanja bude jednaka listi svih pitanja, lista izvučenih pitanja se prazni, i tako u krug, kako bi se smanjila mogućnost da se pitanja ponavljaju.

Čim korisnik ne odgovori točno na 3 pitanja, gubi sve bodove te program odmah otvara *frame* **GAME 1** i poziva idućeg igrača, ili ako više nema igrača na čekanju, prikazuje se *frame* ***SCORE***. Nakon zadnjeg odgovora posljednjeg igrača, bio točan ili netočan, pokreće se zvučni zapis koji igrača pozdravlja i najavljuje kraj igre.

2.1.5. **Frame *SCORE***

Ovaj *frame* prikazuje pobjednika igre, tj. popis igrača i osvojenih bodova. Tablica je sortirana i prikazuje igrače od osvojenog prvog do zadnjeg mjesta. Iako je ovo zadnji *frame*, igraču se pruža mogućnost da pritisne gumb “LOBBY” i započne igru ispočetka. Ukoliko igrač pritisne “LOBBY”, iscrtava mu se prvi *frame* s već zapamćenim imenima igrača i izborom maksimalnog broja igara i gumb “PLAY” će mu biti odmah aktivan. Igraču se nudi mogućnost da promijeni svoje postavke i da u novi krug uđe s potpuno drugačijim postavkama. Svakim novim početkom kruga igre briše se sva prethodna statistika, kako se postignuti rezultati međusobno ne bi pomiješali.



2.2. Rukovanje greškama

Aplikacija je osmišljena kako bi bila što jednostavnija za korištenje, bez previše otvorenih mogućnosti da korisnik napravi grešku. Postoji svega par poteškoća do kojih korisnik može doći tijekom korištenja programa, koje navodim u nastavku.

2.2.1. Poteškoće unutar LOBBY

2.2.1.1. Nije odabran niti jedan igrač ili maksimalan broj igara

LOBBY predstavlja početno sučelje programa, gdje korisnik upisuje imena igrača i odabire maksimalan broj igara. Ukoliko nešto od spomenutog nedostaje, igra ne može započeti. Umjesto poruke greške za svaku od mogućih nedozvoljenih kombinacija, odlučio sam iskoristiti mogućnost onemogućavanja tipke PLAY pomoću atributa *state*, koji je u samome početku generiranja stranice postavljen na vrijednost “disabled”:

```
frame1_btn = ttk.Button(frame1, text='PLAY', state="disabled",
style="TButton", command=lambda:[OnTkraise()])
```

Čim se zadovolji minimalan uvjet za početak igre (postojanje barem jednog imena igrača i odabir maksimalnog broja igara), tipka PLAY postaje aktivna. Programsko prebacivanje iz statusa aktivno u neaktivno i obrnuto odvija se putem posebnih funkcija *EnableButton()* i *DisableButton()*. Svako nedozvoljeno stanje onemogućuje igraču početak igre, a izbacivanje poruke greške sam namjerno izostavio, jer su postavke dovoljno pregledne pa bi poruke samo ometale igrače prilikom unosa vrijednosti.

2.2.1.2. Igrači nisu upisani redom

Za ovaj problem sam, također, izbjegao obavještavanje korisnika o mogućoj greški pa sam predvidio i ovakvo moguće stanje, koje je za korisnika nebitno, hoće li, npr. ime jedinog igrača upisati na prvom, trećem ili zadnjem mjestu. Sve moguće opcije su dozvoljene, bitno je samo da bude upisano barem jedno ime dulje od jednoga znaka, jer pritiskom na PLAY program dodaje redom upisana imena na listu uz pomoć metode *append()*.

2.2.1.3. Vrlo duga imena igrača

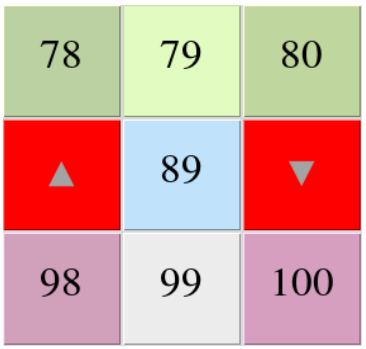
Iduća poteškoća pri korištenju programa bila bi kada bi korisnik upisao vrlo dugo ime (npr. preko 100 znakova), što bi rezultiralo pomicanjem dimenzije prozora preko maksimalne širine ekrana. Umjesto poruke greške, program automatski reže svako ime dulje od 20 znakova na točno 20 znakova. U protivnom ime ostaje nepromijenjeno:

```
PLAYERNAME.append(var1.get()[0:20])
```

2.2.2. Poteškoće unutar GAME 1

2.2.2.1. Ponovno biranje istog broja

Iako ovo ne spada pod tehničku grešku, već logičku - želio bih napomenuti kako je ova situacija također spriječena time da se svaka iskorištena tipka u trenutnom ciklusu pogađanja broja jednostavno učini neaktivnom (*disabled*). S obzirom da se ta tipka u cijelom tom jednom ciklusu pogađanja više ne može koristiti, iskoristio sam je za obavijesti o tome je li traženi broj manji ili veći. Na zaslonu se prikazuje crvena tipka sa strelicom prema gore ili prema dolje, ovisno o tome je li traženi broj manji ili veći. Primjer na slici je za traženi broj 89: crvena polja su neaktivna (ne mogu se pritisnuti).



2.2.2.2. Prebrzo biranje brojeva

Ovaj problem mi je bilo teško riješiti, budući da Python rješava izvršenje koda na specifičan način kada je u pitanju metoda *playsound()*. Sve prethodne akcije (i unosi od strane korisnika i ispisi na zaslon) su na čekanju, dok se ne izvrši *playsound()*. Postojale su dvije opcije: jedna je da se reprodukcije zvuka izvode kao pozadinski procesi, tako da se unese *False* kao posljednji parametar:

```
from playsound import playsound
playsound(PATH + "soundfile.mp3", False)
```

Ovaj pristup bi uzrokovao nesklad u izvršavanju ostalih vezanih zvukova, koji se vrlo često trebaju izvoditi slijedno, jedan za drugim.

Drugi pristup rješavanja ovog problema je obavijestiti korisnika putem skočnog prozora na samom početku pokretanja programa i zamoliti ga da pričeka s odabirom idućeg broja i da to obavi kada se završi svaka zvukovna poruka koja je u tijeku:

```
from tkinter import messagebox
messagebox.showinfo('Info', 'The game demonstrates casual audio response
from your computer. Please be patient and wait for my comments to finish
before start clicking.')
```

Ovakav pristup je dosta agresivan za krajnjeg korisnika pa sam izbjegao tu opciju, iako sam i to isprobao, ali reakcija korisnika rezultirala je brisanjem ove opcije.

Ekstremniji primjer bio bi učiniti sve *widgete* neaktivnima za vrijeme obavljanja reprodukcije zvuka, ali mi se ta ideja u startu nije svidjela, jer čak i da korisnik klikne zaredom na dva broja, oba klika će se slijedno izvršiti, tako da sam ovaj problem prepustio korisniku i njegovom odabiru.

2.2.2.3. Biranje sljedećeg broja suprotno od naputka voditelja

Ovo je jedan sasvim konkretan primjer poruke greške, ali je poruka greške zvukovna. Ako igrač prvo odabere broj 50, zatim dobije informaciju kako je broj manji - ukoliko se ogluši na ovaj naputak i svejedno odabere veći broj, pojavit će se zvukovna poruka greške. Ove zvukovne poruke su dizajnirane tako da oponašaju jednu realnu situaciju pa će u tim situacijama voditeljica reći nešto poput sljedećeg (ovisno o karakteru voditeljice koja nam se obraća). Npr. ako je voditeljica br.1 u pitanju, obratit će nam se s: "*Možda me nisi čuo, rekla sam da je broj manji. Pokušaj ponovo.*", i sl., a ako nam se obraća voditeljica br.2, reći će nešto poput: "*Manji, manji… rekla sam manji, ne veći! Razumiješ?*", i sl.

2.2.2.4. Prilagodba zvukovnih poruka

Program prati slijed svake igre i aktivno prilagođava reprodukciju glasovnih zapisa. Nije svejedno koje glasovne poruke će se reproducirati na samome početku, u sredini, ili pri kraju svakog ciklusa pogađanja. Ovakve prilagodbe programa će se čuti isključivo glasovnim porukama, a način ovih prilagodbi može se konkretno vidjeti u primjeru ispod, za slučaj kada je traženi broj manji od odabranog:

```
elif broj > num:  # ako je traženi broj manji
   if i == 1 or guesses[i-2] < num:
      if RND_ID=='2':  # dodjeljivanje glasova prema ID voditeljice
          less_list=['less1-1.mp3', 'less1-2.mp3']
      else:
          less_list=['less1-3.mp3', 'less1-4.mp3']
   elif guesses[i-2] > num and guesses[i-2] > guesses[i-1]:
      if RND_ID=='2':  # dodjeljivanje glasova prema ID voditeljice
          less_list=['less2-1.mp3', 'less2-2.mp3']
      else:
          less_list=['less2-3.mp3', 'less2-4.mp3']
   elif guesses[i-2] > num and guesses[i-2] <= guesses[i-1]:
```

```
    if RND_ID=='2':  # dodjeljivanje glasova prema ID voditeljice
        less_list=['less3-1.mp3', 'less3-2.mp3']
    else:
        less_list=['less3-3.mp3', 'less3-4.mp3']
  # Reproduciraj prilagođenu glasovnu poruku da je broj MANJI
  less = choice(less_list)
  playsound(PATH + less)
```

Vrijednost *num* predstavlja cjelobrojni traženi broj, a u varijabli *broj* je pospremljen trenutni pokušaj. U listu *guesses* upisuju se svi pokušaji pogađanja broja. Svaki naziv datoteke zvukovnog zapisa je šifriran, tako da unaprijed znam koja *mp3* datoteka pripada kojem voditeljskom glasu i za koji je dio ciklusa pogađanja taj zapis namijenjen.

2.2.3. Poteškoće unutar DOUBLE or NOTHING

2.2.3.1. Gašenje prozora bez tipke “Quit”

Jedina meni poznata poteškoća u ovom *frameu* je izlazak iz programa, ukoliko se ne koristi tipka “Quit”, ali ova poteškoća zapravo vrijedi za svaki *frame* osim *framea* LOBBY, i to na samom početku igre. Svaki preostali *frame* kreira nove procese koji se automatski generiraju pozivom metode *wait_variable()*. Gašenjem aktivnog prozora pritiskom na tipku “x” u gornjem desnom uglu zaobilazi moju pripremljenu funkciju “Quit()”, koja rješava ovaj problem. Izlaskom iz programa bez pritiska tipke “Quit” rezultira time da svi postojeći procesi “djeca” postaju “*ghost procesi*”, tj. “*zombie procesi*”.
Opširnije o ovom problemu u rubrici ***3.1.1. Metoda wait_variable()***.

2.2.4. Poteškoće unutar GAME 2

2.2.4.1. Pogrešni odgovori na ukupno 3 pitanja

S obzirom da je pravilo igre br. 2 - pokušati sakupiti barem 8 točnih odgovora od 10, jedan od problema koji se nameće je: “Što učiniti ako igrač odgovori pogrešno na 3 pitanja u bilo kojem dijelu druge igre?” Umjesto poruke greške, odlučio sam ovaj slučaj automatizirati, tako da program vrlo jasno obavijesti novom zvučnom porukom kako igrač nije zadovoljio i ne treba dalje igrati, te glasovna poruka poziva novog igrača da počne s igrom br. 1.

```
# ako je igrač već pogriješio 3 puta, onda je za njega igra gotova
  if TOTAL2-SCORE2>2:
      TOTAL2=max_games2
      percent=0
```

Varijabla *TOTAL2* bilježi koliko je puta igrač odigrao drugu igru, varijabla *SCORE2* bilježi koliko puta je korisnik odgovorio točno, a ako *TOTAL2* izjednačimo s maksimalnim brojem pokušaja (10), tada za njega igra automatski završava. S obzirom da se igra na sve ili ništa, varijabla *percent* se, koja označava njegov uspjeh, postavlja na nulu.

2.2.5. Poteškoće unutar SCORE

2.2.4.1. Ponovni prelazak u LOBBY

Ovo je problem koji sam, također, trebao automatizirati. Prije povratka na frame LOBBY, potrebno je resetirati sve varijable na početne vrijednosti, kako ne bi došlo do neke greške u novom krugu igre, što sam omogućio pozivom funkcije *ResetScore()*. Nakon toga igrač može nastaviti novi ciklus igara, bez obzira na to je li ostavio iste postavke u *frameu* LOBBY ili ih je potpuno promijenio.

```
# funkcija resetira potrebne varijable na početno stanje
def ResetScore():
   global BODOVI, PLAYER
   BODOVI = [-1, -1, -1, -1, -1]    # bodovi svih igrača
   PLAYER = 0                       # ID trenutnog igrača
   ShowFrame(frame1)
```

Vrijednost **-1** znači da igrač na toj poziciji nije upisan na popisu imena igrača, tj. da trenutno još ne postoji.

2.3. **Upute za korištenje**

Korištenje programa je vrlo jednostavno, u početku samo upišite (korisnička) imena igrača ili samo svoje ako igrate sami (nije bitno u koju rubriku unosite igrače), odaberite maksimalni broj igara i igra počinje pritiskom na PLAY. Sve što trebate nakon toga je kliknuti na broj za koji mislite da je program odabrao za pogađanje, čekajte između odabira - da završi trenutna glasovna poruka, kako biste imali bolje iskustvo igrajući Funtastic. Nakon što svi igrači završe, pojavit će se i rezultati. Važno je napomenuti da, iako, na prvi pogled izgleda kako nije potrebna neka posebna strategija i kako je samo sreća u pitanju - razmislite još jednom! Neka onaj s najviše iskustva pobijedi!

3. TEHNIČKA DOKUMENTACIJA

U prethodnoj rubrici nalazi se dokumentacija vezana uz neke riješene poteškoće do kojih su igrači mogli doći korištenjem programa. U ovoj rubrici bih rekao nešto o ostalim tehničkim rješenjima, koji ne ovise o korisničkom upravljanju programom.

3.1. **Glasovna obilježja programa**

3.1.1. Generiranje glasovnih zapisa

U samom početku pokušao sam napraviti program koji će u potpunosti podržavati hrvatski jezik, što bi značilo da će generirati i glasovne zapise na hrvatskom jeziku. Pokušao sam

različitim postavkama modula *pyttsx3*, ali je, nažalost, za hrvatski jezik ovaj modul gotovo potpuno neupotrebljiv jer zvuči monotono i djeluje naporno već nakon prvog slušanja.

Nakon duljeg traženja dostupnih mogućnosti, naišao sam na fantastičan modul *gTTS* (*Google Text-to-Speech*) koji pretvara proizvoljni tekst u glasovne poruke i to radi na vrlo realističan način. Nažalost, ovaj modul trenutno najbolje zvuči samo na engleskom jeziku, ali s obzirom da sam bio oduševljen njegovom kvalitetom, odlučio sam napisati program čije je i korisničko sučelje na engleskom jeziku.

Za potrebe ovog programa generirao sam oko 200 glasovnih poruka, čije sam nazive datoteka unaprijed šifrirao na određeni način, kako bih kasnije u kodu točno znao koji zvuk dohvaćam i za koju konkretnu situaciju. Pritom sam izradio pomoćni program u Pythonu koji mi je poslužio samo za ovu namjenu:

```
import gtts
from playsound import playsound

filename="guess6-7.mp3"

# zatraži generiranje teksta u glasovnu poruku
tts = gtts.gTTS("Well, are we going to just stand here?", lang="en-us")
# snimi glasovnu poruku
tts.save(filename)
# generiraj glasovnu poruku
playsound(filename)
```

3.2. **Vizualna obilježja programa**

3.2.1. Generiranje nijansi boja koje međusobno odgovaraju

S obzirom da sam želio prikazati 100 widgeta kao izbornik prilikom pogađanja broja, trebale su mi i boje jer mi standardna siva nije bila prihvatljiva opcija. Pokušavajući se dosjetiti koje boje bi mogle biti prihvatljive, odjednom sam naišao na problem: neke boje u potpunosti nisu odgovarale, kada bi se rasporedile jedna pokraj druge, pa sam malo proučio teoriju vezanu uz slaganje boja. Pošto se svaka nijansa boje može odrediti po zastupljenosti crvene, zelene i plave nijanse (RGB), ovaj način označavanja ima i neka pravila.

Veće vrijednosti u svim komponentama označavaju svjetlije nijanse, dok niže vrijednosti označavaju tamnije. Za potrebe ovoga programa sam se odlučio za svjetlije nijanse, ali s tim da je najzastupljenija komponenta (nazovimo je X) obavezno vrijednost veća od decimalnih 200, a udaljena od najmanje zastupljene (nazovimo je Z) za određenu decimalnu vrijednost (npr. 50), kako bih spriječio stvaranje sivih nijansi. Preostalu komponentu nazovimo Y. Zatim sam dodao pravilo da je razmak između Y i Z komponente obavezno veći od 20, kako bih spriječio smeđe nijanse. Generirao sam 10 brojeva koje zadovoljavaju gore navedeno pravilo i dobio niz od 10 decimalnih RGB komponenti:

```
rgb=[[190, 157, 247], [182, 156, 206], [227, 189, 249], [189, 164, 210],
[214, 179, 246], [183, 156, 205], [205, 170, 246], [188, 161, 209],
[226, 192, 251], [192, 159, 214]]
```

Sve ove brojčane vrijednosti su zastupljene u svim bojama koje se prikazuju u onih 100 komponenti u formatu 10x10, tako da je u prvom redu pravilo XYZ za svaku od ovih komponenti, u drugom redu ZYX, u trećem ZXY, itd. Nakon što na ovaj način popunimo 6 redova, mijenjajući samo položaje zasebnih RGB komponenti, ovisno o njihovim međusobnim vrijednostima, doći ćemo do toga da moramo ponoviti kombinacije nijansi iz prva 4 retka, kako bismo popunili redove nijansama sve do desetog retka.

```
1.  XYZ
2.  YZX
3.  ZXY
4.  YXZ
5.  XZY
6.  ZYX
7.  XYZ  ← ponovljena kombinacija iz 1. retka
8.  YZX  ← ponovljena kombinacija iz 2. retka
9.  ZXY  ← ponovljena kombinacija iz 3. retka
10. YXZ  ← ponovljena kombinacija iz 4. retka
```

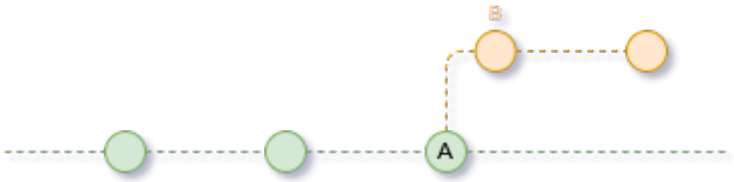
Na ovaj način sam dobio sve nijanse boja koje imaju jednak intenzitet i koje si međusobno savršeno pristaju. Vidi sliku u ***2.1.2. Frame GAME 1***.

3.3. **Upravljanje procesima**

3.3.1. Metoda **wait_variable()**

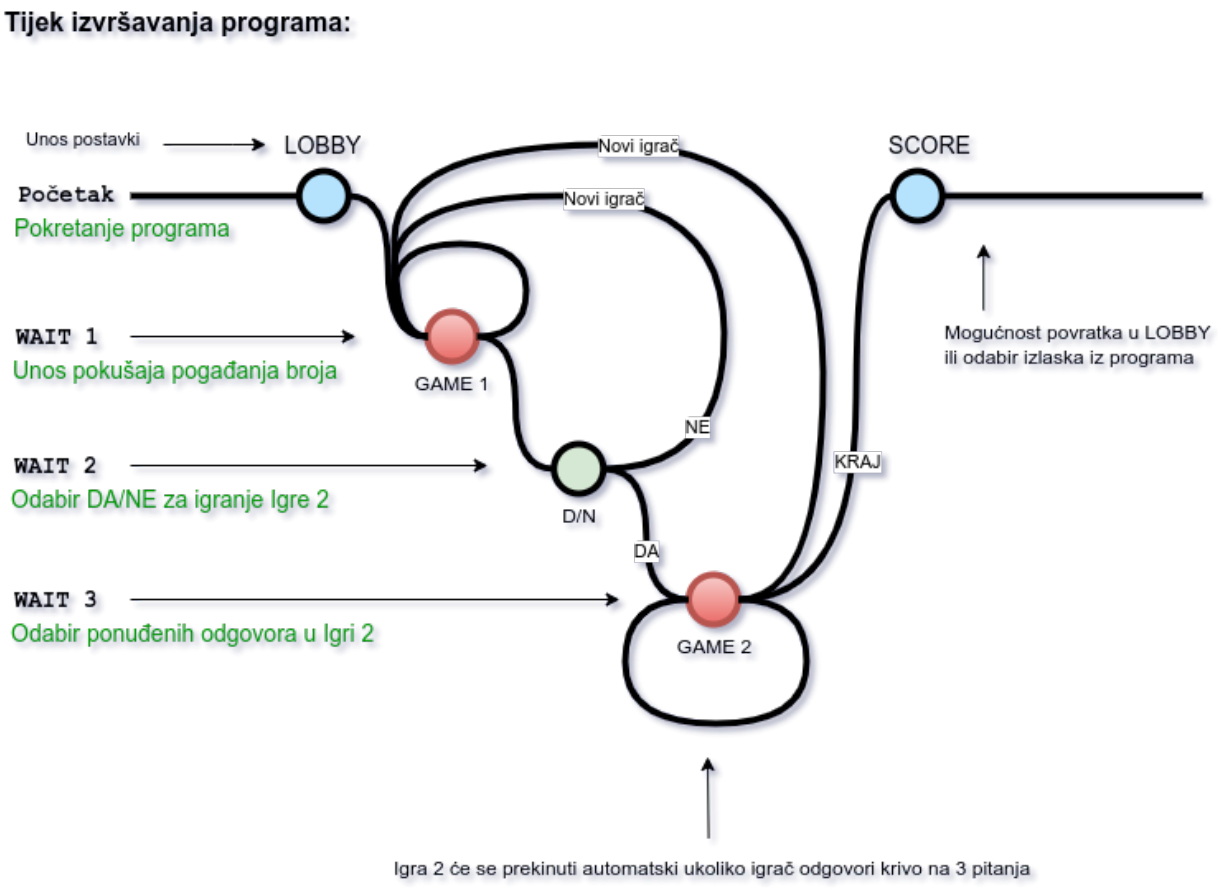
Upravljanje procesima za mene je bilo jedno vatreno iskustvo, jer sam naučio neke osnovne stvari na teži način. S obzirom da ovo područje nije pokriveno nastavnim gradivom sedmog razreda OŠ, kasno sam shvatio da metoda wait_variable(VAR) zapravo stvara dodatan proces "dijete" prilikom izvršavanja i čeka na promjenu varijable VAR, što se neće promijeniti izlaskom iz programa. Taj proces će i dalje ostati čekati promjenu varijable VAR, čak i ako je osnovni program zatvoren i time će se stvoriti "ghost/zombie proces".

Da pojasnim: Python izvršava naredbe slijedno, jednu po jednu programsku liniju, sve dok ne naiđe na metodu *wait_variable(A)* (vidi sliku 2). Nakon toga se stvara novi proces (žuta isprekidana linija), a prethodni proces je na čekanju, sve dok se ne realizira promjena varijable A koja bi trebala nastaviti njezino izvršavanje (zelena isprekidana linija).



Slika 2. wait_variable(A)

Bez korištenja ove metode Python, jednostavno, ne bi čekao na korisnički ulaz klikanjem miša u Tkinter sučelju niti na jednome mjestu, a kako bih pojasnio na kojim sve mjestima koristim ovu metodu, najbolje je prikazati grafički (vidi sliku 3, oznake “WAIT”):



Slika 3. Upravljanje procesima

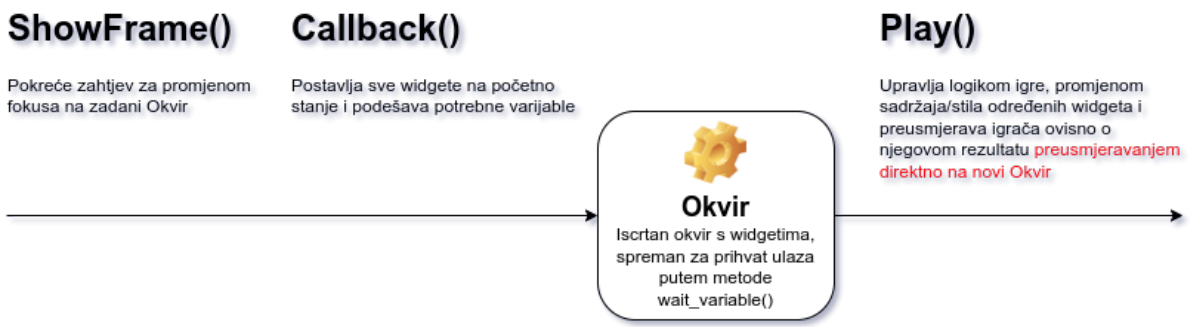
Metoda wait_variable() koristi se na 3 mjesta u programu, a s obzirom da postoje 4 petlje, iskoristio sam različite vrijednosti za ovu varijablu, kao šifriranu obavijest procesu “roditelj”, koji mora znati što se zapravo desilo u procesu “dijete”, npr.:

- 0 = igra je tek počela,
- 1 = povratak iz igre 2 bez igranja, (korisnik se odlučio zadržati bodove)
- 2 = povratak nakon odigrane igre 2,
- 3 = pokrenut izlazak iz programa pomoću Quit()

Za razumijevanje ovakvog sustava potrošio sam najviše vremena. Pokušao sam sve odraditi samostalno i naučio još jednu pouku: program u Pythonu sasvim je moguće odraditi i bez promjena svih varijabli koje očekuje metoda *wait_variable*. Program će donekle i raditi, a sustav vam neće signalizirati kako ste u zabludi, sve dok ga ne preopteretite.

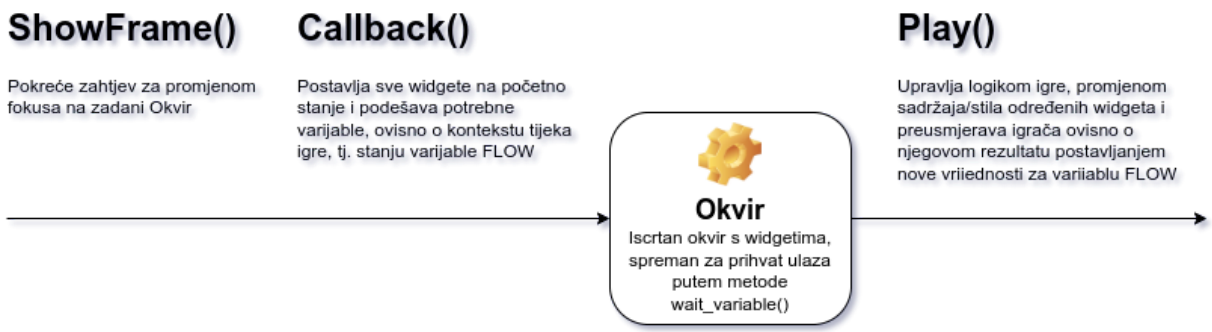
Jedini način da saznate kako ova metoda nije baš najbolji izbor je da nastavite igrati sve dok vam se program ne “sruši”, zbog preopterećenosti, jer ste stvorili previše procesa “djece” i vjerojatno pospremili previše podataka u svaki od stvorenih procesa.

Nakon što mi se ovo desilo, počeo sam postavljati prava pitanja i otkrio da je ovo bio glavni uzrok problema:



Program postavljen na način koji je prikazan iznad garantirano dovodi do problema. Drugim riječima, nakon pozvane funkcije ne bismo smjeli nastaviti pozivati nove *frameove* (okvire), već treba prvo zatvorili prethodni proces postavljanjem nove vrijednosti za varijablu koju očekuje prethodno pokrenuta metoda **wait_variable**.

Ukratko, tako je izgledala BETA verzija programa Funtastic, pa čim sam otkrio problem, morao sam presložiti sve pozive funkcija i promijeniti logiku na nekoliko mjesta, kako bih došao do finalnog i sigurnog rješenja, tj. kako bih došao do verzije 1.0. Dakle, umjesto gore navedenog načina, sustav koji koristi **wait_variable** trebao bi slijediti sljedeći obrazac:



3.4. **Konvencija o imenovanju**

3.4.1. Google stil

U želji da učinim izvorni kod što preglednijim, pronašao sam informaciju kako to radi Google i pokušao primijeniti njihovu postojeću konvenciju o imenovanju Python programskog koda:

- **GLOBALNE_VARIJABLE**: velikim tiskanim slovima, gdje su riječi odvojene s "_"
- **lokalne_varijable**: malim slovima, gdje su riječi odvojene s "_"
- **NaziviFunkcija**: *UpperCamelCase format*

3.5. **Problemi prilikom upravljanja modulima**

3.5.1. Slika kao pozadina *widgeta*

Način na koji Tkinter prikazuje pozadinske slike u widgetima je još jedno za mene neugodno iskustvo. Nisam znao da veće slike (iako su u pozadini) zapravo istiskuju element koji su iscrtani unutar tog elementa sa slikovnom pozadinom. Potrošio sam puno vremena na ovom problemu, dok nisam shvatio o čemu je riječ. Za rješenje ovog problema poslužio sam se iskustvima Python programera na *StackOverflow*-u i nakon što sam implementirao ponuđeno rješenje, mogao sam nastaviti rješavati ostale probleme prilikom izrade svog programa.

```
# funkcija OneResize(): izvršava se prilikom promjene dimenzije prozora
def OnResize(event):
    # promijeni veličinu pozadinske slike na veličinu Label-a
    image = bgimg.resize((event.width, event.height), Image.ANTIALIAS)
    # osvježi sliku widgeta Label
    frame1_title.image = ImageTk.PhotoImage(image)
    frame1_title.config(image=frame1_title.image)

bgimg = Image.open('images/lobby.jpg') # učitaj pozadinsku sliku

frame1_title = Label(frame1, text='FUNTASTIC',
compound='center',font='calibri 72 bold', fg='#fff', image=img)
frame1_title.pack(fill='both', expand=True)
frame1_title.bind('<Configure>', OnResize)
```

Nisam našao rješenje za postavljanje pozadine unutar tablice u *frameu* SCORE, koje koristi *Treeview* metodu.

3.5.2. Pozicioniranje *widgeta*

Tkinter ima velikih problema kod prikazivanja widgeta kao **grid** i kao **pack**. Kojiput je zgodno kombinirati ove metode, ali u tom trenutku vam sigurno stižu nevolje. Za neke situacije je zgodnije koristiti prednosti *grida*, ali u nekima ne. Kod jednostavnijih sučelja dovoljno je koristiti **pack**, ali kod malo složenijih sučelja, vrlo je komplicirano graditi vizualno sučelje, npr. to sam osjetio gradeći rastezljivo sučelje pri izradi ovog programa, kod pozicioniranja komponenti. Dodao bih ovome i zaključke iz rubrike ***3.5.5. Python i operacijski sustavi***, što dodatno komplicira proces izgradnje sučelja.

Također, želio sam pozicionirati PNG sličicu s prozirnom pozadinom na postojeće sučelje, ali *Tkinter* mi nije nimalo olakšao tu namjeru.

3.5.3. Metoda **playsound()**

Vidi rubriku ***2.2.2.2. Prebrzo biranje brojeva***.

3.5.4. Metoda **RadialProgressbar3D()**

Tražeći rješenje za implementaciju trake za napredak (progress bar), naišao sam na postojeće Open Source rješenje. S obzirom na to da se ne radi o vitalnom dijelu programa, odlučio sam ga integrirati. Modul se zove awesometkinter, a metoda kojom sam se poslužio zove se RadialProgressbar3d(). Nažalost, ova metoda radi odlično na linux sustavima, ali iz nekog razloga i ne baš najbolje na Windows operacijskim sustavima. S obzirom da mi se svidjela, pokušao sam naći rješenje povremenog nestajanja s ekrana, ali sam primjetio i da se neki drugi dijelovi sučelja na Windows sustavima ponašaju slično. U razgovoru s mentoricom zaključio sam da problem nije u mom kodu, već se vjerojatno radi o problemu u verziji nekog modula s podrškom za Windows operacijski sustav.

3.5.4. Metoda **Playsound()**

Sličan primjer problematične metode iz prethodnog primjera odnosi se i na modul *playsound*. Ukoliko ne pročitate upute za instalaciju, koje dolaze uz moj programski kod, mogli biste instalirati najnovije module potrebne za izvršavanje ovog programa, što će vrlo vjerojatno rezultirati greškom. Modul playsound ima kritičnu grešku u novijim verzijama i potrebno je instalirati jednu od starijih verzija, kako bi radila:

```
py -m pip install playsound==1.2.2
```

3.5.5. Python i operacijski sustavi

Veliki dio koda programskog jezika Python može se izvršavati na više različitih operacijskih sustava bez ikakve promjene. Međutim, postoje neke metode koje zahtijevaju dublju integraciju s radnom okolinom pojedinog operacijskog sustava. Primjetio sam kako moj program radi gotovo besprijekorno na linux sustavima, dok se na Windows sustavima dešavaju manje nepravilnosti, što sam već objasnio u prethodnim rubrikama. Vjerojatno je krivac preobilno korištenje modula, zbog vlastitih ambicija da napravim čim bolji program, što je moglo rezultirati i ovakvim promjenama u izvođenju, no sve je ovo dio procesa učenja i mogu slobodno reći da sam puno toga naučio iz svega gore navedenog.

3.6. **Povijest programa**

Evolucija ideje za izradom ovakvog programa počela je još prije dvije godine, kada sam iz znatiželje pokušao isprobati mogućnosti Pythona. Uz metodu *random.choice()* bilo je vrlo lako napraviti algoritam nasumičnog odabira iz neke liste, no, naravno, s vremenom sam proširivao logiku same aplikacije, a prvo čega sam se dosjetio je implementirati gTTS uz taj program za komandnu liniju. Imao sam veliku pomoć kod tih prvih koraka pa sam od samog početka na pravi način počeo shvaćati kako programiranje funkcionira i na što sve treba obratiti posebnu pažnju. Naveo bih slijed ove evolucije u kraćim crtama:

3.6.1. Guess-A-Number (Funtastic - Alpha)

Izradom ovog programa sam počeo u 5. razredu osnovne škole. U tom periodu program nije imao mnogo koda. Napravljen je tako da se izvršava u komandnoj liniji (CLI), prvo bez zvučnih zapisa i grafičkog sučelja, no bio je potpuno funkcionalan. Ubrzo sam na taj osnovni program ubacio i par generiranih mp3 datoteka pomoću gTTS-a, koje bih reproducirao putem metode *playsound*, što me oduševilo. Naveo bih da je iz tog razdoblja i ostao onaj *drugačiji* "voditeljski glas br.2", kako ga navodim u ovoj dokumentaciji. Naime, radi se samo o *Googleovoj* starijoj verziji generatora glasa, koji danas više ne postoji, a koji mi je bio dobro došao kao "drugačiji karakter" i unio malo dodatne dinamike.

Program je radio samo osnovne stvari, poput generiranja nasumičnog broja, i dopuštao je unos za pogađanje broja od strane korisnika u jednoj FOR petlji, a prilikom svakog pokušaja, generirao bi se glasovni komentar o tome je li broj manji, veći ili je igrač pogodio broj. Program je bio bio vrlo jednostavan, no najvažnije je to što mi je bio vrlo koristan za upoznavanje s Pythonom i načinom na koji radi. Kao što sam spomenuo, iz tog razdoblja ostale su mi i neke starije *mp3* datoteke, kojima se i danas koristim u novijim verzijama.

3.6.2. Funtastic - Beta

Prvo grafičko sučelje predstavlja i ovu BETA verziju programa. Glavne osobine ove verzije programa i osnovna razlika između ove i finalne verzije dobro je objašnjena u rubrici ***3.3.1. Metoda wait_variable()***.

3.6.3. Funtastic - v1.0

Ovo je prva stabilna verzija koja ima i grafičko sučelje i koje radi ispravno, ima proširenu kolekciju glasovnih zapisa i pročišćen kod. Trenutna verzija programa.

4. **BUDUĆNOST PROJEKTA**

U budućnosti bih volio proširiti ideju komunikacije s računalom uz dodavanje većeg broja unikatnih i zabavnih igrica koje bi koristile slično sučelje, a program bi imao još više dovitljivih komentara i zabavnih elemenata, kako bi cjelokupan dojam bio još više "Funtastic".

Idealna verzija ovakvog tipa programa u nekoj daljoj budućnosti bila bi uporaba umjetne inteligencije i njen prolaz na *Turingovom testu*. Učinjeno je mnogo fascinantnih projekata u kojima se pokušava olakšati ljudima raditi, učiti i stvarati. Moja ideja na prvi pogled može se doimati vrlo jednostavnom, no cilj mi je bio napraviti prototip, koji bi ljude razveselio i nasmijao oponašajući prirodne ljudske reakcije, a mislim da je smijeh vrlo bitan i svima nama potreban. Jedan takav program bi u budućnosti mogao imati dodatak zvučnog unosa putem mikrofona, a uz dodatne *Funtastic* igrice uz šaljive primjedbe i komentare, bio bi prava poslastica za ljude koji žele odmak od monotonih reakcija stroja. Možda bi se svidio i onima koji imaju problem s vidom pa su osuđeni na ovakav tip igrica s glasovnom interakcijom, pogotovo ako im još k tome igra ne bi bila dosadna. U toj namjeri bi mi puno pomoglo i dodatno znanje iz psihologije.

5. **ZAKLJUČAK**

Iako program na prvi pogled možda djeluje banalno i ne previše zahtjevno, bilo je tu puno elemenata za spoticanje. Rješavanje problema na dnevnoj razini mi je pomoglo da počnem otkrivati svijet programiranja i na pravi način razmišljati. Tek kada sam došao na ovu razinu, shvatio sam koliko toga još moram naučiti. Rad s Pythonom mi je bio puno ugodniji, no iako *Tkinter* ima puno elemenata koji ne olakšavaju ugodan rad u kreiranju grafičkog sučelja, vizualan doživljaj informacije je bitan, pa je i njegova uloga opravdana. Volio bih se u budućnosti okušati u stvaranju nekog drugog grafičkog programa, na nekoj drugoj platformi, koja bi mi pružila više mogućnosti od *Tkintera*.

Što se zaključka o rezultatima izrade ovoga programa tiče, mislim da je moja najveća korist već postignuta, a to je iskustvo koje sam pokušao do u detalje opisati u rubrici ***3.3. Upravljanje procesima***, zbog čega se već, što se mene tiče, isplatio sav moj trud.

Na kraju bih želio zahvaliti svojoj mentorici Snježani Mihočki na korisnim uputama prilikom svake pojave nedoumice, kao i mojoj obitelji, jer je imala razumijevanja za moje vrlo intenzivno razdoblje, tijekom izrade ovoga projekta.

6. **LITERATURA**

1. https://www.w3schools.com/python/
2. https://docs.python.org/3/library/tkinter.html
3. https://www.quora.com/What-is-tkinter-in-Python
4. https://www.reddit.com/r/Tkinter/
5. https://stackoverflow.com/questions/tagged/tkinter
6. https://codereview.stackexchange.com/questions/tagged/tkinter
7. https://github.com/Aboghazala/AwesomeTkinter
8. https://soundbible.com/
9. https://www.bensound.com/