Qubic — 3D križić-kružić

Završni projekt - Multimedijski sustavi

Ivan Krcivoj, Antonela Bogdanić

Zagreb, 2. ožujka 2022.

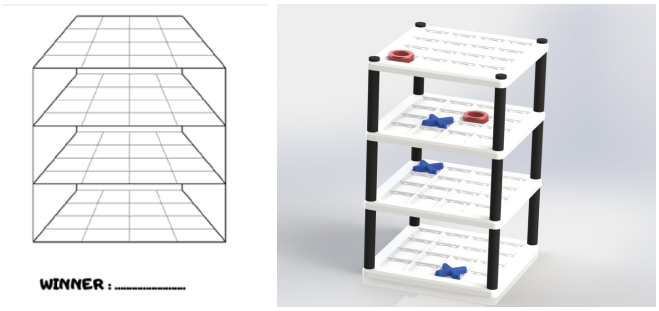
Sadržaj

1. Uvod

Jedna od najpoznatijih igara na svijetu zasigurno je križić-kružić, engl. *tic-tac-toe*. Možda je upravo zbog svoje jednostavnosti pravila, "opreme za igru" i strategije stekao takvu slavu među ljudima. Kao završni projekt iz kolegija Multimedijski sustavi odlučili smo se implementirati malo kompliciraniju varijantu ove popularne igre.

Križić-kružić u tri dimenzije — Qubic nije toliko popularan među igračima kao njegov prethodnik, no često je predmet proučavanja znanstvenika. Igra nudi mnoštvo strategija i posjeduje neka svojstva koja su veoma zanimljiva za proučavanje. Naša je ideja bila približiti "naizgled" tešku igru korisnicima i omogućiti igranje međusobno ili protiv računala.

Prije opisivanja igre, aplikacije za igranje i detalja samog programskog koda, iznijet ćemo kratku povijest Qubica. Parker Brothers predstavili su igru i krenuli prodavati set za igru davne 1964. 1972. prestavljaju novi moderniji izgled igre. Oba izdanja imala su naziv *Parker Brother 3D Tic Tac Toe Game*. Set je uključivao prozirnu kocku za igranje i oznake igrača u bojama, no igra se može igrati i na papiru(Slika 1 b)). Tako su se kasnije na tržištu pojavili i prazni papirnati predlošci za igranje Qubica(Slika 1 a)). Paralelno uz popularizaciju igre za šire pučanstvo, razvijao se i interes znanstvenika koji su htjeli pronaći najbolju strategiju za garantiranu pobjedu prvog igrača.



Slika 1: a) igra na papiru, b) igra na kocki

Ovu dokumentaciju smo podijelili na dva dijela. Prvi dio se odnosi na korisničku dokumentaciju, u kojoj ćemo opisati igru, navesti pravila igre i detaljno prikazati kako koristiti aplikaciju. U drugom dijelu bavit ćemo se detaljima implementacije, opisat ćemo pomoćne klase vezane uz logiku igre i objasniti dijelove koda. Nadamo se da će opisani dijelovi pomoći studentima prilikom unaprjeđenja našeg projekta, ali i korisnicima za što lakše korištenje.

Dio I.
Korisnička dokumentacija

2. Uvodno o igri

Qubic je varijanta križić-kružića u tri dimenzije s ukupno 64 polja na koja dva igrača postavljaju znakove X ili O. Umjesto na ploči s 9 polja kao u "klasičnoj " varijanti igre, igrači u Qubicu postavljaju svoje oznake na kocki koja sadrži četiri horizontalne ploče sa 16 kvadratića, odnosno igra se na kocki dimenzija `4x4x4` kvadratića. Također, igru je moguće igrati i na kocki manjih dimenzija, konkretno `3x3x3`, tada kocka sadrži 27 polja za igru raspoređena na tri horizontalne ploče.

Igra se unatoč trodimenzionalnosti može igrati i na papiru, no može se nabaviti i posebna kocka za igranje. Mi smo odlučili ploču za igru prikazati što jednostavnije kako bi korisnicima bilo lakše vizualizirati kocku.

Pravila igre ne razlikuju se previše od križić-kružića na koji smo navikli. Za pobjedu je potrebno da jedan od igrača uspije složiti 4 (ili 3) svoja znaka u nizu. Igrač može na više načina složiti tzv. pobjedničke linije, engl. *winning lines*. Igrač koji na bilo kojoj horizontalnoj ploči u retku, stupcu ili dijagonali složi svoje znakove pobijedio je. Nadalje, isto vrijedi i za vertikalne ravnine/ploče kocke. Posljednji način na koji je moguće pobijediti je tako da igrač složi znakove na neku od 4 glavne dijagonale kocke.

Igrači naizmjenice igraju, tako da postavljaju svoje oznake na ploču, počevši od onog igrača koji ima oznaku X. U daljnjem ćemo radu prvog igrača, odnosno onoga koji postavlja X označavati kraće kao X igrač ili samo X, analogno za igrača O. Igra završava pobjedom nekog od igrača ili remijem ukoliko je cijela ploča ispunjena, a nema pobjedničkih linija.

Nakon što smo opisali pravila, u sljedećim ćemo Vas točkama provesti kroz korištenje aplikacije i pokazati mogućnosti koje onda nudi. Nadamo se da ćemo primjerima i opisima olakšati korištenje navedene aplikacije i da će Vam igranje biti što ugodnije.

3. Pokretanje i početak igre

Unutar repozitorija u kojem se nalazi ova dokumentacija, unutar podmape *Qubic* možete pronaći datoteku naziva *Qubic.pde*. Ona se pokreće koristeći *Processing*(mi smo za pisanje i pokretanje koristili verziju *4.0b2*.), a sama igra se pokreće pritiskom na gumb `Run`.

Nakon pokretanja na ekranu bi se trebao pojaviti prozor kao na Slici 2. To je početni prozor aplikacije u kojem je potrebno odabrati vrstu igre i upisati imena igrača. Najprije se upisuje ime X igrača, vidimo da je on posebno označen na početku. Kada je korisnik unio željeno ime pritisne tipku `Enter`, tada je posebno označen O znak i po završetku pritiskom na `Enter` započinje igra.

Potrebno je napomenuti kako su imena igrača ograničena na sveukupno 10 znakova te

Slika 2: Početni zaslon aplikacije

da su neki simboli onemogućeni za korištenje jer služe kao poziv dodatnih funkcionalnosti u aplikaciji. Na primjer, nije moguće koristiti simbole: `%,$,&,#,?`. Također, sve dok nije upisano ime, pritiskom na `Enter` ništa se neće promijeniti.

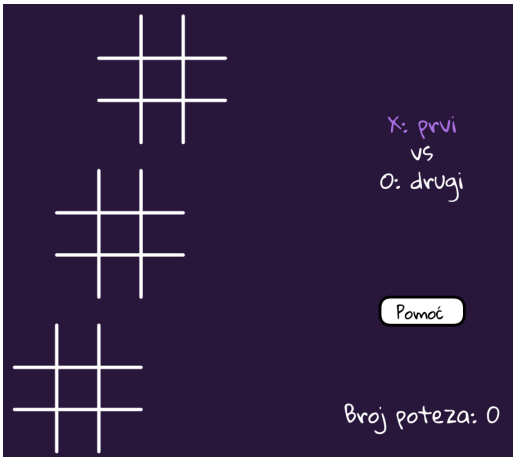
Na početnom se zaslonu nalazi i mogućnost odabira vrste igre, korisnik može odabrati želi li igrati na ploči dimenzija `3x3x3` ili `4x4x4`. Kao zadana vrsta igre označena je igra na ploči `3x3x3`, a to lako promjeni klikom na željeni gumb.

Također, na početnom zaslonu vidimo i gumb `Pravila`. Klikom na njega otvorit će se poseban prozor u kojem su ispisana pravila igre, ali o tome više u 6. Napomena na dnu obavještava korisnike o mogućnosti korištenja *help prozora* u kojem mogu pronaći popis dodatnih funkcionalnosti, kao i načina kako ih pokrenuti. Korištenje *help* prozora klikom na tipku `?` omogućeno je tijekom cijelog korištenja aplikacije.

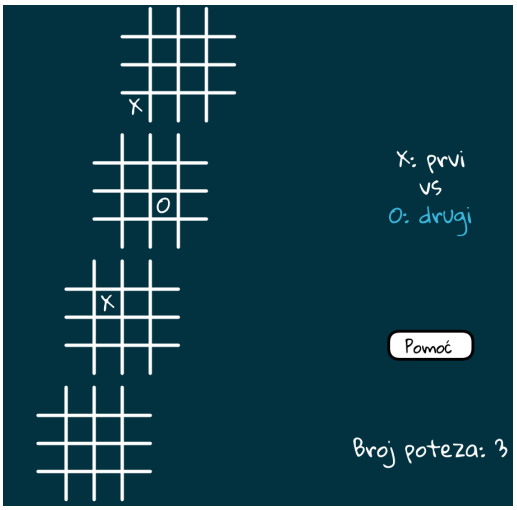
Iako je igra namijenjena za dva igrača, omogućili smo igranje protiv računala. Ako želite igrati protiv računala, potrebno je napisati `racunalo` kao ime igrača kojeg želite da igra računalo.

4. Tijek igre

Nakon unosa imena i odabira vrste igre pojavljuje se glavni zaslon aplikacije. Na njemu se vidi prikaz odabrane ploče kao i detalja same igre, kao na Slikama 3 i 4. Posebnom bojom označeno je ime igrača koji je na potezu, na početku smo rekli da je to igrač X. Svakim potezom promijenit će se oznaka igrača koji je na potezu, promijenit će se broj poteza koji piše na dnu prozora kao i stanje na ploči, primjer vidimo na Slici 4.



Slika 3: Ploča za igru `3x3x3`.



Slika 4: Ploča za igru `4x4x4`.

Igrači klikom na željeno polje na ploči odigraju svoj potez. Potez je ispravan kada je igrač kliknuo unutar nekog polja na ploči i pritom je to polje neodigrano, odnosno prazno. Zvuk koji se pritom čuje moguće je isključiti pritiskom na tipku `#`, ali o tome u 6. Ukoliko je potrez neispravan, prikazat će se odgovarajuća poruka kao na Slici 5.

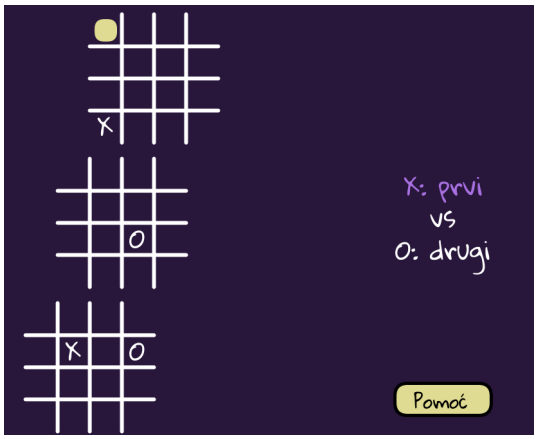
Slika 5: Poruka o neispravnom potezu.

Na tom se prozoru još nalazi i gumb `Pomoć`. On služi kao pomoć igračima prilikom igranja igre. S obzirom na složenost traženja najboljeg poteza zbog velikog broja mogućnosti, izračunavanje poteza može potrajati. Ako je potez izračunat, prelaskom miša po gumbu `Pomoć` gumb će se pobojati posebnom bojom koja ovisi o odabiru teme(više u 6), kao na Slici 6 a). U slučaju da se potez još računa prelaskom miša po gumbu `Pomoć` on će izgledati kao na Slici 6 b), posebno naglašen sivom bojom uz prateću poruku.

Slika 6: Primjer prelaska miša preko gumba:
a) kada je potez izračunat, b) kada računanje još traje

Kada je potez izračunat, klikom na gumb `Pomoć` na ploči se posebnom bojom označi potez koji računalo predlaže kao najbolji potez. Primjer možete vidjeti na Slici 7.

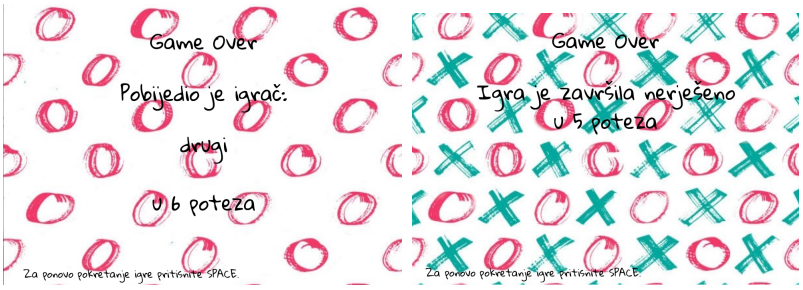
Igra s računalom odvija se na isti način kao u slučaju dva igrača. Jedina je razlika što dok je računalo na potezu korisnik ne može igrati poteze, nego mora pričekati da računalo odigra svoj potez.



Slika 7: Prikaz hinta na ploči.

5. Kraj igre

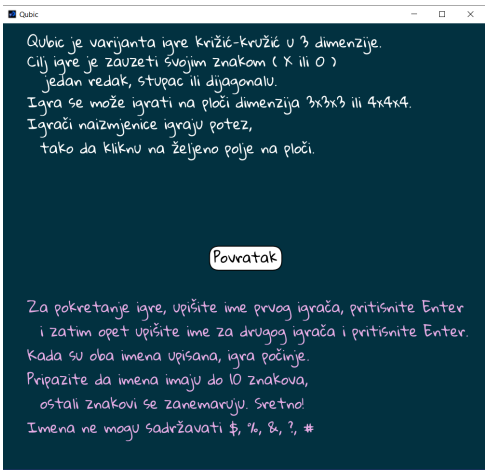
Igra završava pobjedom nekog igrača nakon što složi tzv. pobjedničku liniju ili remijem ako je cijela ploča popunjena. Završni zaslon u pozadini ima oznaku igrača koji je pobijedio ili zajedničkih oznaka u slučaju remija, primjer na Slici 8. Ispisuje se i pripadna poruka o pobjedi igrača ili remija u određenom broju poteza. Izgled pozadine osim o pobjedi X ili O igrača ovisi i o temi aplikacije koju je korisnik odabrao, a ako je korisnik omogućio zvuk čut će se i prigodni zvuk pobjede. Rezultati igre spremaju se automatski u datoteku rezultata. Omogućen je povratak na početni zaslon pritiskom na tipku `SPACE`.



Slika 8: Primjer završnog zaslona u slučaju: a) pobjede O igrača b) remija

6. Dodatne funkcionalnosti

U ovoj ćemo točki objasniti nekoliko dodatnih funkcionalnosti i objasniti kako se koriste.

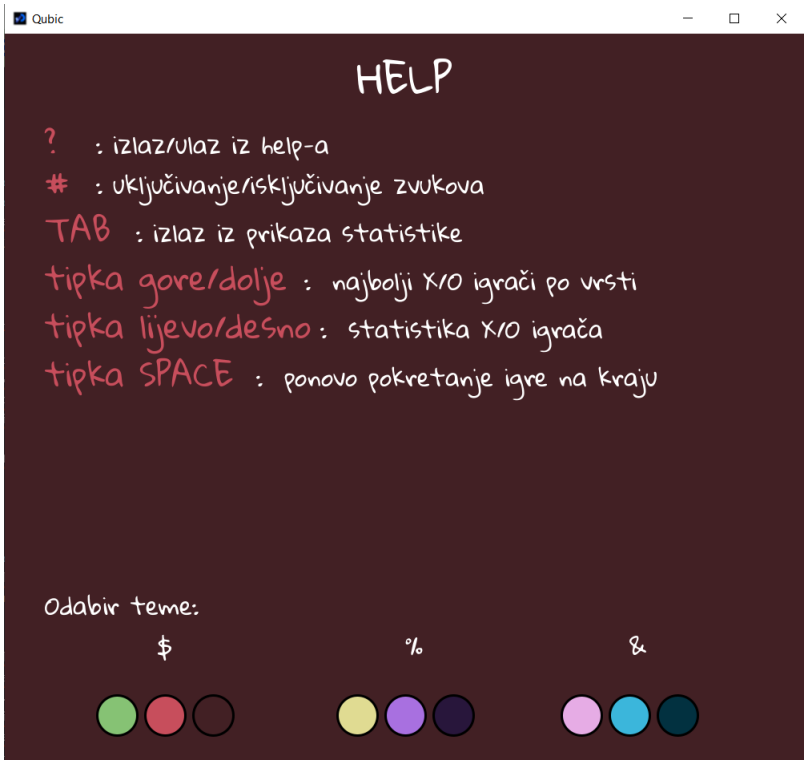


Slika 9: Pravila igre

Za početak prisjetimo se gumba `Pravila` koji se nalazio na početnom zaslonu aplikacije, Slika 2. Klikom na njega prikazuje se prozor kao na Slici 9. Kratko su ispisana pravila igre kao i upute o pokretanju igre uz dodatne napomene. Klikom na gumb

`Povratak` vraćamo se na početni zaslon. Prelazak mišem preko oba gumba posebnom će bojom označiti gumb.

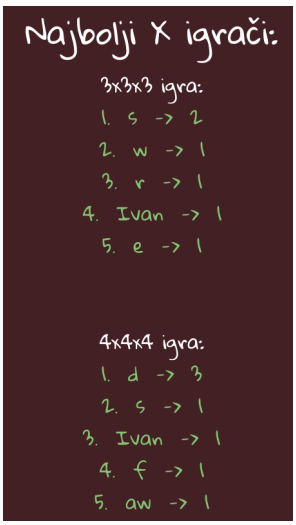
Sljedeća funkcionalnost koja je omogućena je pozivanje *help* izbornika. On se poziva klikom na tipku `?`. Na Slici 10 možete vidjeti kako izgleda taj zaslon. Osim što vidimo opise i načine korištenja svih funkcionalnosti, vidimo i prikaz tema koje je moguće koristiti u aplikaciji. Teme su konkretno palete boja koje se koriste u aplikaciji. Mi smo omogućili 3 različite palete — teme, a one se lako mijenjaju pritiskom na tipke: `$,%,&`.



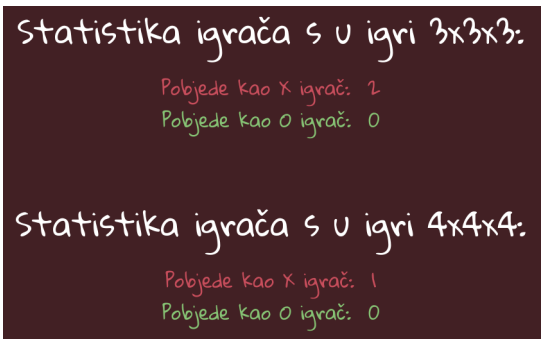
Slika 10: *Help* izbornik

Opišimo redom preostale funkcionalnosti koje vidimo u *help* izborniku. Tipkom `#` uključujemo ili isključujemo zvukove u aplikaciji, a to su zvukovi igranja poteza i zvuk na kraju igre.

Sljedeća funkcionalnost koja je omogućena, i već spomenuta u točki 5, je povratak na početni zaslon tipkom `SPACE`. Taj je povratak omogućen samo ukoliko smo na završnom zaslonu, odnosno završili s igrom.



Slika 11: Prikaz 5 najboljih X igrača po vrsti igre



Slika 12: Prikaz statistike trenutnog igrača

Zadnja funkcionalnost koju je preostalo opisati je prikaz statistike. Postoji više različitih prikaza statistike koje je moguće pregledati u aplikaciji, a svi se pozivaju klikom na tipke strelica. Strelica gore poziva ispis pet najboljih X igrača ovisno o vrsti igre(Slika 11, dok strelica dolje prikazuje 5 najboljih O igrača. Strelica lijevo poziva prikaz osobne statistike X igrača koji trenutno igra Qubic, dok strelica desno prikazuje osobne rezultate trenutnog O igrača, primjer na Slici 12. Ukoliko se osobna statistika poziva na početku igre i imena igrača još nisu upisana, tada se ispiše poruka: `Upišite ime igrača pa pokušajte ponovo vidjeti statistiku!`. Izlaz iz prikaza statistike događa se pritiskom na tipku `TAB`.

Dio II.
Tehnička dokumentacija

7. Opis projekta

U ovom ćemo dijelu dokumentacije detaljno objasniti programski kod i detalje implementacije kako bi kolegama koji će nadograđivati našu aplikaciju što više olakšali posao. Za početak objasnit ćemo okruženje u kojem smo stvarali kod i navesti materijale koji su dio projekta.

Sve materijale možete pronaći na: `https://github.com/antonelab/Qubic-Processing`. U dokumentu *Opis projekta.docx* nalaze se detalji našeg stvaranja projekta, poput podjele poslova, tijeka rada i popis poslova koje smatramo da još treba obaviti. Datoteka naziva *Qubic.pdf* je upravo dokumentacija koju čitate, a materijali za stvaranje `.pdf` datoteke možete pronaći u podmapi *dokumentacija*.

Programski kod u kojem smo implementirali igru i napravili aplikaciju nalazi se u podmapi *Qubic*. Unutar nje nalazi se *Qubic.pde* u kojem se nalazi glavni dio programskog koda, a koji se otvara i pokreće koristeći *Processing*(mi smo koristili verziju *4.0b2*). U podmapi *code* nalaze se pomoćne klase u Javi i datoteka *Qubic.jar*, dok se u podmapi *data* nalaze se potrebni materijali za rad aplikacije poput fonta, zvuka i pozadina.

Kao pomoć pri pisanju projekta, konkretno klasa u Javi, koristili smo *NetBeans* u kojem smo stvorili projekt i provjeravali dijelove koda. Također, htjeli bi napomenuti da treba biti oprezan pri spajanju pomoćnih klasa s `.pde` datotekom, mi smo to uspijeli postići na način na koji je programski kod sada organiziran. Zbog toga smo dodali liniju `import qubic.*;`. Odgovarajuću `.jar` datoteku dobili smo upravo koristeći *NetBeans*.

Prije objašnjavanja koda potrebno je ispisati dodatne biblioteke koje smo koristili u *Processingu* prilikom izrade aplikacije. Potrebno je instalirati i *importati* biblioteku `Sound` i `Minim` koje nam služe za korištenje zvukova. Također, zbog dodatnih *Javinih* funkcija koje koristimo prilikom spremanja rezultata u datoteku potrebno je *importati* i `java.io.FileWriter`.

8. Logika igre

Logika igre većinski je smještena unutar pomoćnih klasa u podmapi *code*. Ukratko ćemo proći kroz dijelove datoteka obzirom na to da su oni dobro komentirani, no bitno je reći gdje se što nalazi kako bi se mogli lakše snaći u datotekama.

8.1. `Cube`, `Cube3` i `Cube4`

`Cube` je *interface* koji predstavlja ploču za igru. U njoj se nalaze važne funkcije koje su potrebne za rad s kockom, a njih ćemo implementirati u klasama `Cube3` i `Cube4`.

`Cube3` predstavlja kocku `3x3x3`, a klasa `Cube4` kocku `4x4x4`. Obzirom na to da su klase međusobno jako slične, mi ćemo malo detaljnije opisati `Cube4`.

Redom, `Cube4` je konstruktor kocke i stvara praznu kocku tako što poziva `clear`. Funkcija `clear` postavlja sva polja ploče na ' ' što predstavlja da je to polje prazno i postavlja `mNumber` na 0 obzirom da nema oznaka na polju na početku igre. Kako bi mogli znati što se nalazi na određenom polju kocke funkciji `value` dajemo 3 integera, odnosno oznaku pozicije. Intovi i, j, k su brojevi od 0 to 3, npr. trojka (1,2,2) predstavlja poziciju na kocku: prvi nivo (druga ploha od gore), drugi redak (na toj plohi) i drugi stupac (drugi stupac gledajući s lijeva na desno), za primjer pogledati Sliku 4 i poziciju O.

Funkcija `result` vraća broj koji predstavlja pobjedu, remi ili trajanje igre. Prilikom provjere pobjede nekog od igrača zovemo `winning_line` koja nam vraća oznaku pobjedničkog igrača. Ako pobjeđuje X vrijednost završnog stanja je 500, a za O -500, ali ako funkcija nije vratila oznaku X ili O, a nema slobodnih polja na kocki jer je `mNumber` maksimalan, onda je završno stanje neriješeno i vrijednost 0. Funkcija `winning_line` sadrži petlje koje prolaze po svim mogućim pobjedničkim linijama (ukupno 76) i provjeravaju postoje li 4 ista znaka u toj liniji te ako da vrate oznaku kojoj pripadaju.

Slijede četiri jednostavne funkcije, a to su redom: `generate_moves` koja prolazi cijelom kockom i provjerava je li polje prazno (tj. ' '). Ako je polje prazno, moguće je odigrati potez na toj poziciji pa stvorimo `Move` objekt koji dodamo u vektor koji pohranjuje sve moguće poteze na kocki. Move klasa opisana je 8.2.

Zatim slijedi funkcija `isValid` koja provjera je li potez valjan. Potez je valjan ako su sve varijable članice postavljene na 0,1,2 ili 3 te je polje (i, j, k) na ploči prazno tj. ' '. Nadalje, funkcija `play` koja odigra potez move s oznakom igrača tako što najprije provjeri je li potez valjan s funkcijom `isValid`.

Ako je potez valjan treba "odigrati" potez stavljajući oznaku igrača na poziciju na ploči koja odgovara potezu. Vraća se *true* ili *false* vrijednost ovisno o tome je li potez bio valjan. Osim što možemo odigrati potez, možemo ga i "vratiti", odnosno u samom algoritmu nam kasnije treba *undo* poteza pa to radi upravo funkcija `unPlay`. Način na koji se to izvodi je da vrati oznaku praznog polja ' ' na mjesto koje odgovara potezu `move` i broj odigranih poteza `mNumber` smanji se za jedan.

8.2. `Move`

`Move` je klasa koja predstavlja potez. Kako se igranje igra odvija u tri dimenzije, `Move` ima tri koordinate `mLevel`, `mRow`, `mColumn`, koje predstavljaju razinu, red i stupac. Objekt klase `Move` možemo konstruirati na tri načina. Defaultni konstruktor sve vrijednosti postavlja na nulu, zatim imamo konstruktor koji prima redom vrijednosti razine, retka i stupca. Posljednji konstruktor prima *string* koji se treba sastojati od tri broja koji su međusobno odvojeni zarezom.

Pomoću odgovarajućeg *gettera* možemo pročitati željenu koordinatu poteza. *Setteri* ne postoje jer nije predviđeno da se vrijednosti komponenti mijenjaju.

Postoje nadjačane metode `equals` i `toString` koje uspoređuju dva poteza, odnosno pretvaraju potez u *string* kako bi ga bilo moguće ispisati.

8.3. `Pair`

Klasa `Pair` je parametrizirana klasa koja se sastoji od dva atributa koji mogu biti različitih tipova. Koristimo ju kada želimo omogućiti funkciji da vrati dvije vrijednosti.

8.4. `Player`

Igrača predstavlja klasa `Player` koja pamti ime i simbol igrača. Simbol kojeg igrač koristi postavlja se u konstruktoru i kasnije se ne može mijenjati.

`minMax` je funkcija koja predstavlja stablo odluke s alfa-beta podrezivanjem. Funkcija vraća par koji se sastoji od heurističke vrijednosti najpovoljnijeg poteza te koji je najpovoljniji potez. Za to izračunavanje funkciji su potrebni redom: trenutno stanje na kocki, niz dopustivih poteza, oznaka igrača na potezu, vrijednost parametara alfa i beta te maksimalnu dubinu. Kako je broj mogućnosti velik koristimo alfa-beta podrezivanje kako ne bi morali provjeravati situacije u kojima znamo da ne možemo dobiti bolje rješenje. Međutim, kako broj stanja koji trebamo provjeriti i dalje može biti prevelik koristimo potragu do zadane dubine kako igra ne bi predugo trajala.

Funkcija `hint` prima stanje kocke i pomoću funkcije `minMax` računa optimalni potez. Kako izračunavanje pomoći može potrajati izvodit ćemo ga u zasebnoj dretvi pa nakon što izračunamo hint provjeravamo je li u međuvremenu korisnik odigrao sljedeći potez. U tom slučaju *hint* nam više nije koristan pa ga odbacujemo podizanjem `interrupt` izuzetka. Ukoliko sljedeći potez nije odigran, hint je valjan pa ga prosljeđujemo dalje kao povratnu vrijednost.

8.5. `Hint`

`Hint` je klasa koja implementira sučelje `Runnable` i služi za računanje *hinta* u posebnoj dretvi.

8.6. `QubicGame`

`QubicGame` je klasa u kojoj je implementirana logika igre. U konstruktoru prima tip igre i dva igrača koji međusobno igraju. Tip igre je broj 3 ili 4 i određuje koje će biti dimenzije kocke za igru.

Funkcija `play` je najvažnija funkcija u ovom dijelu izlaganja. Pozivanjem te funkcije započinje izvršavanje same igre. Sastoji se od petlje koja čeka idući potez sve dok ne dođe do završnog stanja igre. Kada se odigra novi potez naprave se sve potrebne promjene kako bi igra funkcionirala. Po dolasku u završno stanje igre određuje se tko je pobijedio te time funkcija završava svoje izvođenje.

Kako čekanje na potez traje, radi izbjegavanja blokiranja igre, klasa `QubicGame` implementira sučelje `Runnable`. Nadjačana funkcija `run` omogućava da se funkcija `play` izvodi u zasebnoj dretvi.

9. Qubic.pde

U ovoj se datoteci nalaze dijelovi koda koji povezuju logiku igre s aplikacijom i određuju izgled i funkcionalnosti aplikacije. Obzirom na količinu i raspored programskog koda odlučili smo ovaj dio opisati na način da objasnimo globalne varijable,a zatim svaku pojedinu funkciju.

9.1. Globalne varijable

Zbog velikog broja globalnih varijabli koje koristimo prikazat ćemo ih sažeto i pregledno u nastavku, dok ćemo u kasnijim točkama spomenuti na kojim ih mjestima koristimo.

- `type` - predstavlja vrstu igre, može biti 3 ili 4
- `showHint` - pomoćna varijabla koja određuje treba li se prikazati/iscrtati *hint* na ploči
- `written` - pomoćna varijabla koja služi kako bi se samo jednom upisali rezultati igre u statistiku
- `mess` - pomoćna varijabla koja govori imamo li poruku o grešci koju treba prikazati na glavnom prozoru
- `name` - globalna varijabla koja simbolizira stanje u kojem je naša aplikacija, zadana vrijednost je 1 i označava da X igrač unosi svoje ime, 2 označava da ime postavlja O igrač, a ako je vrijednost 0 to znači da su oba igrača unijeli svoje ime
- `info` - globalna varijabla koja govori o tome treba li iscrtati zaslon s pravilima, ako je 1 zaslon se iscrtava
- `help` - globalna varijabla koja govori o tome treba li iscrtati zaslon s naredbama, ako je 1 zaslon se iscrtava
- `stat` - globalna varijabla koja služi za prikazivanje statistike, ovisno o vrijednosti prikazuju se različite vrste statistika, zadana vrijednost je 0 jer se statistika ne prikazuje, ako je 1 ili 2 prikazuju se najbolji igrači X ili 0, ako je 3 prikazuju se osobne statistike
- `mute` - pomoćna varijabla koja služi za isključivanje ili uključivanje zvuka, ako je 0 zvuk je omogućen
- `newGame` - pomoćna varijabla koja služi na kraju igre kako bi znali trebamo li ponovo iscrtati početni zaslon, zadana vrijednost je 0, a kada je 1 iscrtat ćemo ponovo početni zaslon

- `player1` i `player2` - spremamo objekte tipa `Player` koji predstavljaju naše igrače
- `game` - predstavlja objekt `QubicGame` koji predstavlja igru

- `gameThread` - dretva u kojoj se čeka da korisnik odigra potez
- `font` - spremamo font koji koristimo u cijeloj aplikaciji
- `move_sound` i `win_sound` - varijable koje spremaju imena datoteka zvukova
- `minim, minim2` i `audio_win, audio` - varijable potrebne za korištenje zvuka
- `winners3` i`winners4` - služe za spremanje podataka o statistici kako bi se ispisali na zaslonu za statistiku
- `label_color, name_color, bg_color, bg_theme` - varijable koje postavljaju temu aplikacije, određuju boju teksta, naglašenog teksta i dijelova aplikacije, boju pozadine i boju, konkretno sliku, završnog zaslona
- `PImage` varijable u kojima se učitavaju slike koje koristimo u aplikaciji

9.2. `setup()`

Kao što znamo, u *Processingu* se funkcija `setup()` poziva jednom, na početku, i služi za postavljanje svih važnih svojstava aplikacije. Mi smo najprije postavili veličinu zaslona na kojem će se aplikacija iscrtati kao i omogućili promjenu veličine zaslona. Nadalje, učitali smo font koji ćemo koristiti u cijeloj aplikaciji i učitali sve slike koje su nam potrebne za završni zaslon igre. Također, potrebno je učitati i zvukove odigravanja poteza i potez pobjede koje ćemo koristiti u nastavku. Za kraj pozivamo funkciju `init()` koju ćemo detaljnije opisati u 9.7.

9.3. `draw()`

Druga važna funkcija u *Processingu* je funkcija `draw()` koja se vrti beskonačno, sve dok je aplikacija pokrenuta, a služi za iscrtavanje zaslona. Ona je doista složenija, a opisivat ćemo njezine dijelove koje ovise o stanjima varijabli koje smo u točki 9.1 opisali.

Prva provjera određuje jesu li imena oboje igrača upisana i ako jesu iscrtava se glavni zaslon za igru, odnosno zaslon s pločom. Kako bi znali koju ploču iscrtati moramo provjeriti stanje varijable `type`. Ovisno o tome je li igra na ploči veće ili manje dimenzije koristeći linije iscrtamo ploču, odnosno polja za igru. Nakon toga je potrebno ispisati stanje kocke/ploče čije je stanje spremljeno u `game.cube`. Također, koristimo i pomoćnu varijablu `showHint` da provjerimo treba li posebno označiti polje koje je računalo pronašlo kao najbolji potez. Polje označavamo tako da iscrtamo pravokutnik posebne boje na određenim koordinatama, odnosno unutar određenog polja. Nakon toga potrebno je iscrtati detalje igre, ispisat ćemo imena igrača i broj poteza. Dio nakon toga vezan je uz prikaz gumba `Pomoć`. Ovisno o stanju pomoćne funkcije `hover()` i varijable `showHint` drugačije ispunjavamo pravokutnik koji predstavlja gumb. Svi gumbovi u aplikaciji su zapravo kombinacija pravokutnika, s posebnim uređenjem i tekstom. Na kraju provjeravamo trebali li ispisati poruku o grešci, konkretno nepravilnom potezu.

U tom je djelu koda potrebno provjeriti i je li igra možda završila. Informacija o tome pohranjena je u `game.winner`. Ako zvuk nije onemogućen u varijabli `mute` pokrećemo zvuk pobjede, a ovisno o tome tko je pobijedio ispiše se prikladni tekst. Također, provjere `bg_theme` u kodu određuju koju ćemo pozadinu iscrtati ovisno o temi koja je odabrana. Na kraju ako rezultat još nije spremljen to učinimo koristeći `PrinterWriter` i `File`. Za kraj, provjeravamo treba li pozvati funkciju `init()`, pomoću stanja varijable `newGame`, koja će ponovo pokrenuti igru.

Druga važna provjera je `info == 0` koja će u slučaju da je uvjet točan iscrtati početni zaslon. Cijeli dio koda odnosi se na formatiranje ispisa teksta i pripadnog uređenja ispisa. Također, dodali smo i gumb koji uređujemo koristeći ponovo funkciju `hover()`, o kojoj više u 9.6. Ukoliko je uvjet bio kriv pokreće se ispisivanje zaslona s pravilima. On sadrži i gumb za povratak na početnu stranicu.

Sljedeća važna provjera je vezana uz to je li potrebno iscrtavati zaslon statistike. Ovisno o vrijednosti varijable `stat` koristimo drugačija oblikovanja teksta. Nećemo ulaziti u detalje, bitno je samo da za ispis koristimo `winners3` i `winners4`.

Zadnja provjera je vezana uz stanje varijable `help`. Ako je njezina vrijednost 1 ispisat ćemo *help* zaslon. On se ponovo sastoji od teksta koji je posebno uređen, uz iscrtavanje krugova različite boje koji pomažu pri vizualizaciji paleta boja.

9.4. `mousePressed()`

U ovoj ćemo točki objasniti dijelove koda koji se događaju kada korisnik klikne mišem po ekranu. Prvi dio vezan je uz klikanje miša ukoliko se nalazimo na glavnom zaslonu, tj. u igri. Tada je veoma bitno ovisno o vrsti igre provjeriti je li korisnik kliknuo na mjesto koje predstavlja polje i nakon toga stvoriti odgovarajući potez, što se postiže pozivajući funkciju `move()`, koja je objašnjena u 9.8. Također, provjerava se je li korisnik kliknuo na "gumb" `Pomoć`, ako je korisnik kliknuo i *hint* je izračunat postavlja se zastavica koja će iscrtati *hint*.

Druge provjere provjeravaju je li korisnik kliknuo na gumbove kojima se može birati vrsta igre, kao i je li kliknuo na pravila ili povratak. Ponovo se koristi pomoćna funkcija `hover()`.

9.5. `keyPressed()`

Sljedeća važna funkcija vezana je uz korisnikov unos s tipkovnice. Moramo provjeriti je li korisnik htio upisati ime (kada je u stanju igre u kojem se ime upisuje) i vrijednosti spremati na određeno mjesto. Ako korisnik klikne `Enter` ili mijenjamo stanje `name` ili pokrećemo igre ili moramo ignorirati pritisak tipke. Ukoliko se igra pokreće stvorimo novi `QubicGame` objekt, kao i pokrenemo novi `Thread` u kojem čekamo da korisnik unese/odigra potez.

Ostale se provjere odnose na pozivanje posebnih funkcionalnosti na način da samo promijenimo stanje određenih varijabli. Skrenut ćemo pažnju na promjenu tema, ukoliko netko želi promijeniti izbor boja, to može učiniti na ovome mjestu. Korisno je detaljnije opisati što se događa prilikom klika na strelice. Koristeći rječnike, ovisno o podacima

koje iščitavamo iz datoteke `results.txt` spremamo vrijednosti, kako bi ih mogli kasnije ispisati, u već spomenute `winners3` i `winners4`. Čitamo liniju po liniju datoteke u kojoj se spremaju rezultati i ažuriramo rječnike, ovisno o provjerama.

9.6. `hover()` i `removeLastChar()`

U ovoj ćemo točki opisati dvije jednostavne, no korisne funkcije. Funkcija `hover()` koristi se često, a sastoji se od provjere pozicije miša na temelju proslijeđenih vrijednosti. Ako se miš nalazi unutar određenih granica vraća se `true`, inače `false`.

Funkcija `removeLastChar()` poziva se prilikom korisnikova klika na tipku `BACKSPACE` i služi kako bi se izbrisali znakovi dok korisnik utipkava svoje ime.

9.7. `init()`

Ova je funkcija veoma jednostavna, no važna u kodu. Služi za postavljanje početka igre, potrebne se varijable inicijaliziraju na početne vrijednosti. Također, potrebno je određena stanja igre postaviti na zadane vrijednosti. Pogotovo treba paziti da se omogući ponovo pokretanje igre nakon završetka igre.

9.8. `move()`

Zadnja funkcija vezana je uz odigravanje poteza. Vrijednosti koje njoj prosljeđujemo predstavljaju oznake polja na ploči gdje korisnik želi odigrati potez. No prije odigravanja poteza treba provjeriti je li potez valjan, zato pozivamo funkciju `game.cube.isValid(move)` i ukoliko je potez valjan isti odigramo, ako smo na potezu. Uz to pokrećemo zvuk odigravanja poteza ako je zvuk omogućen i postavljamo `hint` na početnu vrijednost 0. Ako potez nije valjan postavljamo varijablu `mess` pomoću koje će se kasnije ispisati poruka o neispravnom potezu.

Literatura

- Rubinoff, A., *3D Tic Tac Toe*, University of Rochester, dostupno na: `https://www.cs.rochester.edu/u/brown/242/assts/studprojs/ttt10.pdf`
- Patashnik, O., *Qubic: 4 x 4 x 4 Tic-Tac-Toe*, MathematicsMagazine, dostupno na: `https://www.jstor.org/stable/2689613?read-now=1&seq=1`