**Logika implementacije I faze projekta Triggle**

Ovaj dokument obuhvata celokupno objašnjenje logike koja je primenjena za implementaciju I faze projekta Triggle. U prvoj fazi je bilo potrebno odraditi sledeće funkcionalnosti :

1. Omogućiti izbor ko igra prvi (čovek ili računar).

2. Omogućiti izbor simbola igrača koji igra prvi (X ili O).

3. Unos dimenzija table i validacija unosa.

4. Generisanje početnog stanja igre na osnovu unosa.

5. Prikaz trenutnog stanja table sa pozicijama čvorova i poteza.

6. Unos poteza sa validacijom unosa.

7. Provera kraja igre na osnovu popunjenosti table ili broja zauzetih trouglića.

8. Definisati način za predstavljanje stanja problema (igre)

* Predstavljanje pozicija razvučenih gumica i zauzetih trouglića.

9. Napisati funkciju za postavljanje početnog stanja

* Definiše se na osnovu zadate veličine table.

10. Napisati funkcije za proveru kraja igre

* Tabla je popunjena razvučenim gumicama ili je igrač zauzeo više od polovine

Trouglića.

U nastavku dokumenta sledi opis ideje koja je primenjena za svaku od funkcionalnosti.

Inicijalizacija table za igru Triggle

U našoj implementaciji, funkcija **draw\_hexagon(side\_length)** generiše matricu koja predstavlja heksagonalnu tablu. Korisnik na početku bira tj. unosi dimenziju table n, nakon čega se kreira matrica sa tačkama (stubovima u igri) označenim kao '●' .

Na početku smo izračunavale visinu i širinu matrice. Visina se sastoji od "gornjeg" i "donjeg" dela šestougla, gde svaki deo ima (2 \* side\_length - 2) redova. Faktor 3 označava visinu svakog reda tačaka, dok +1 osigurava centralni red. Širina svake linije je proporcionalna broju tačaka u redu, gde svaka tačka zahteva prostor od 6 karaktera. Pre for petlje u ovoj funkciji, definisale smo dictionary nodes u kome se čuvaju x i y koordinate svake tačke. Pomoću for petlje smo obezbedile da svaki red sadrži tačke na pravilnim razmacima, a tačke se postavljaju horizontalno počev od start kolone pa do end kolone. Druga for petlja služi da centralni red popuni tačkama tj. stubovima. Funkcija vraća iscrtanu tablu kao i sve čvorove table.

Prikaz table

Funkcija **print\_board(matrix, nodes)** prikazuje trenutno stanje table na osnovu matrice matrix. Na početku smo kreirale red first\_row, kako bi omogućile da brojevi kolona počinju sa leve strane ali da budu poravnati sa elementima matrice. Ti brojevi se zatim dodaju na svakoj šestoj poziciji i tako omogućujemo da se odgovarajuć broj kolone nadje u liniji sa odgovarajućom kolonom. Nakon toga, u funkciji se prolazi kroz svaki red matrice i ukoliko je on deljiv sa 3 dodaje oznaku reda kao veliko slovo azbuke (dobijeno pomoću ASCII vrednosti, dodavanje u svakom trećem redu jer su tu čvorovi). Pre ispisivanja smo sve elemente pretvorili u stringove zbog korišćenja join funkcije.

Unos poteza igrača i validacija poteza

Unos poteza igrača omogućujemo kroz funkciju **play\_move(matrix, nodes, start, direction)**. Potez uključuje početnu poziciju (npr. A1) i smer razvlačenja gumice (D - desno, DD - dijagonalno desno, DL - dijagonalno levo). Na početku funkcije definisani su svi validni pravci i izvršena je provera da ukoliko nije unet dozvoljen pravac ili ukoliko stub oko kog želimo da krenemo razvlačenje gumice nije u nodes, potez je nevalidan. U positions listi se čuvaju sve pozicije matrice koje treba ažurirati potezom. U zavisnosti od direction-a izvršava se odredjeni deo koda i u position listu se dodaje pozicija. Na kraju pre nego da se potez ažurira, vršimo proveru da li su sve pozicije validne (funkcija **is\_valid\_move(matrix, x, y)**, proverava da li je koordinata x,y unutar granica matrice i da li postoji čvor). Ukoliko jesu, dodaje se poteg u matrici, a ukoliko nisu onda je nevalidan potez.

Zauzimanje trouglića (tj. x ili o) na tabli i provera za kraj igre

Ovaj deo zadatka smo implementirale pomoću funkcije **draw\_triangle(matrix, symbol, count,max\_triangles)**. Prolazimo kroz svaku poziciju matrice pomoću dvostruke petlje i na početku proveravamo da li je broj nacrtanih trouglova jednak ili veći od maksimalnog broja trouglova, pa samim tim proveravamo da li je došlo do završetka igre. Ako je trenutna pozicija u matrici prazna, onda se prvo vrši provera da li je moguće formirati trougao sa – na vrhu(pozicija gore-desno (i-1, j+1) mora biti validna i sadržati "\", pozicija gore-levo (i-1, j-1) mora biti validna i sadržati "/", pozicija dole (i+1, j) mora biti validna i sadržati "-") i ako jeste dodajemo ga i označavamo simbolom koji predstavlja igrača (x ili o) i povećavamo count. Ako nije moguće, proveravamo za trougao sa – na dnu. Funkcija vraća broj nacrtanih trouglova.

Promena igrača ( deo druge faze?)

Pomoću funkcije **switch\_player(current\_player)**, omogućeno je da se menja koji igrač je trenutno na potezu (računar ili čovek).

Izbor koji igrač igra prvi (čovek ili računar)

Ovu funkcionalnost smo implementirale u Main delu programa. Od korisnika se zahteva da unese ko igra prvi (čovek ili računar) i da ukoliko navede neku drugu opciju od te dve se ponovo vrati i izabere jednu od dve moguće.

Izbor simbola igrača koji igra prvi (X ili O)

I ovu funkcionalnost smo implementirale u Main delu programa. Od korisnika se zahteva da unese koji simbol bira i da ukoliko navede neku drugu opciju od te dve se ponovo vrati i izabere jednu od dve moguće ( X ili O). Nakon njegovog odabira za drugog igrača se postavlja suprotan simbol i pravi se dictionary symbols za čuvanje informacija o simbolima.

Prikaz trenutnog stanja tabele na osnovu prosledjenog poteza

-uraditi-