Projekat - *Turn Snake*

Tema projektnog zadatka *Turn Snake* je bila napraviti igricu koja će kombinovati rad klasične *Snake* igrice i planiranja strategije na bazi poteza igrača.

1. Kako se igra?

Realizovanu igricu mogu igrati od 2-4 igrača, pri čemu svaki od njih poseduje jednu do tri zmije, naravno svaki igrač će imati isti broj zmija. Na početku igre se svaka zmija može pomeriti najviše 3 koraka, a igrač koji je na potezu ima određeno vreme da pomeri jednu ili sve svoje zmije, za jedan ili maksimalan broj koraka. Zmije se pomeraju upotrebom tastera strelice na gore, dole, levo ili desno, a biraju pritiskom na taster razmaka. Ukoliko igrač ne pomeri ni jednu svoju zmiju 3 runde gubi partiju i biva izbačen. Svakako cilj svakog igrača je da zarobi suparničke zmije i onemogući im dalje kretanje.

Kako bi se realizacija cilja svakog igrača ubrzala, postoje dodatni elementi među kojima su:

1. hrana – kada se pojede povećava se telo zmije i broj koraka, a ukoliko se ne pojede, nakon runde se pomera pravolinijski od 1-3 koraka,
2. neočekivana sila – kada se pojede smanjuje ili povećava broj koraka.

Važno je napomenuti da zmije umiru kada igrači pokušaju da ih pomere izvan granica table na kojoj se igra, kao i kad se sudare u svoje telo ili telo suparničkih zmija.

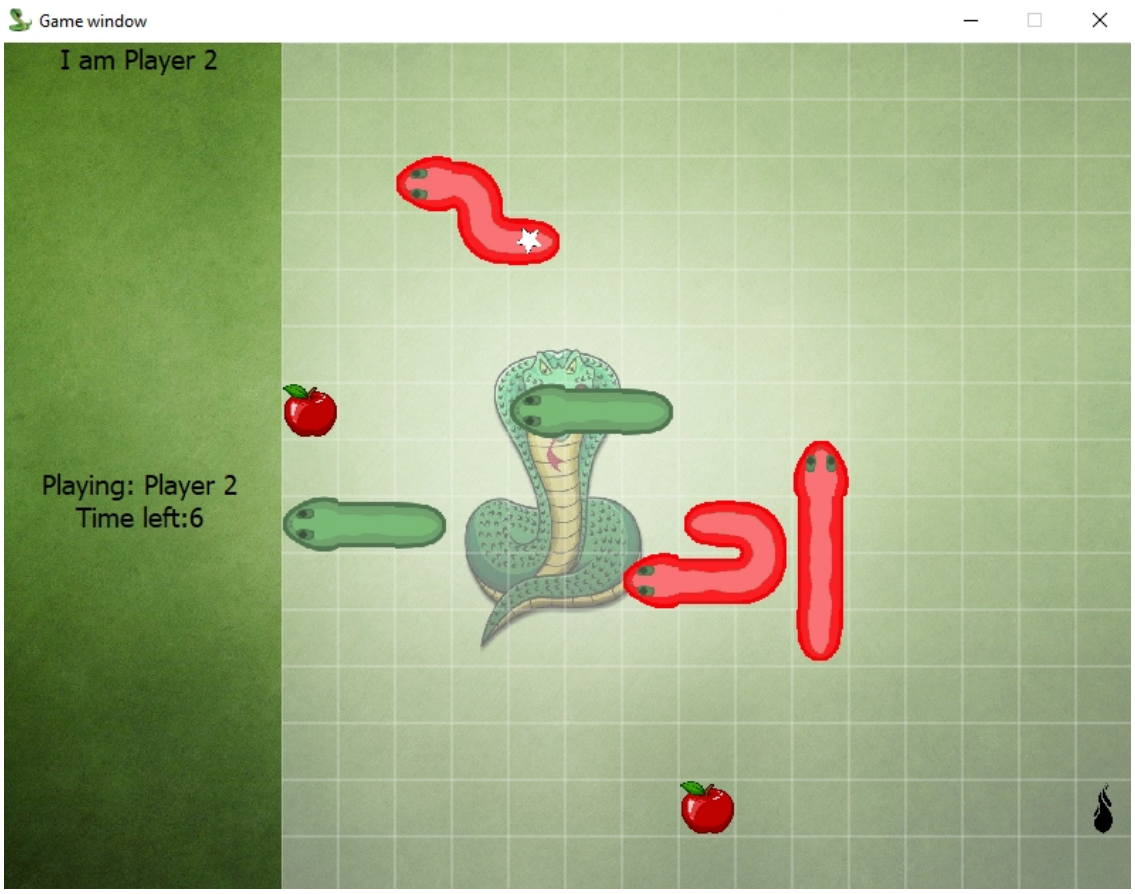
2. Softverska implementacija

Zasniva se na klijent-server arhitekturi upotrebom TCP komunikacije. Igrač koji će biti domaćin, podiže server na svom računaru gde takođe definiše koliko igrača se očekuje u partiji i koliko će svaki od njih imati zmija. Zatim se čeka da se svi igrači konektuju na server i tada igra može da počne.

Kako bismo bolje opisali funkcionalnost, opisaćemo poslove pojedinačnih komponenti klijent-server arhitekture.

1. Klijent – GUI aplikacija na kojoj su prikazane zmije svih igrača, svaki igrač iz svog prozora aplikacije ima mogućnost da upravlja samo svojim zmijama, a nakon odigranog poteza serveru se šalje komanda koju je igrač primenio. Naravno pored prikazanih zmija, iscrtane su i hrana, neočekivana sila i vreme preostalo za potez igrača.
2. Server – aplikacija koja prima sve informacije o potezima od klijenta i prosleđuje ih ostalima, kako bi u svakom trenutku imali ažurirano stanje o pozicijama svih entiteta. Pored toga, server obavestava i svakog klijenta ko je trenutno na potezu, kako bi se vreme preostalo za potez azuriralo i kako bi svaki klijent znao od koga pristižu podaci za ažuriranje.

Izgled jedne klijentske aplikacije prikazan je na slici ispod (*Slika 1*).



Slika 1: *Snimak ekrana jedne GUI aplikacije*

3. Prednosti i mane

Uočene prednosti i mane prilikom implementacije projekta ćemo sagledati iz tri segmenta koje čine upotreba *Python* programskog jezika, *PyQt5* okvira i paralelizacije rada.

1. Upotreba *Python* programskog jezika

Python je programski jezik visokog nivoa opšte namene. Podržava imperativni, funkcionalni i objektno orijentisani stil programiranja. Nije strogo tipiziran i izvršava ga interpreter, a uz interpreter se isporučuje i velika standardna biblioteka, što ga čini veoma jednostavnim za upotrebu i pisanje malih programa. Zbog njegove interpreterske prirode programi pisani u *Python*-u su kompatibilni sa ostalim platformama, bez obriza na kojima su pisani. U drugim jezicima je bitno formatirati kod, kako bi bio čitljiviji, dok je kod *Python*-a formatiranje koda njegov sastavni deo, odnosno obaveže programera da mu se prilagodi, tako da su programi uvek čitljivi. Što se tiče njegovih mana, najviše možemo zameriti njegovoj interpreterskoj prirodi što ga čini sporijim od programskih jezika koji se kompajliraju. Pored toga, to što nije tipiziran jezik ga čini teškim za upotrebu u velikim projektima, a i okruženja često ne mogu da predlože automatsko dopunjavanje koda, jer je tip promenljivih veoma često nepoznat.

2. Upotreba *PyQt5* okvira

PyQt5 je *Python* biblioteka koja uokviruje *Qt* interfejs pisan u *C++* koji je prilagođen *Python* progamskom jeziku za kreiranje višeplatformskih aplikacija. Omogućava nam jednostavno pisanje grafičkih aplikacija, a pruža nam i sepcijalizovane *threading* biblioteke koje olakšavaju paralelni rad. Kao mane okvira možemo navesti nedostatak dizajnera koji bi nam olakšao kreiranje izgleda aplikacije, kao i što je izgled aplikacije usko povezan sa kodom, jer se preko koda definiše raspored komponenti.

3. Paralelizacija rada u *Python*-u

Python nam omogućava upotrebu kako više-nitne tako i više-procesne paralelizacije rada uz pomoć svojih biblioteka, kao i komunikaciju i sinhronizaciju između niti i procesa u zavisnosti od toga za šta se opredelimo. Ali ukoliko hoćemo pravu paralelizaciju moramo iskoristiti više-procesnu, ukoliko imamo dovoljno procesora na raspolaganju, jer se više-nitna paralelizacija zasniva na upotrebi iste sistemske niti kao i glavni program, tako da se ostali procesori ne upošljavaju iako ih posedujemo.

Za kraj bi valjalo istaći da je upotreba *PyQt5* okvira bila pogrešan izbor za implementaciju GUI aplikacije, jer je stvarala puno poteškoća prilikom pisanja, a na kraju nije pružila dovoljno dobre performanse. A što se tiče upotrebe *Python* programskog jezika u ovom projektu, kao dobre stvari mu prepisujemo što je čitljiv, a kao loše što je slabo tipiziran, pa bi kolege koje su učestvovale u izradi imali problema sa razumevanjem tuđih implementacija.

Python kao programski jezik je dobar za pisanje manjih programa ili skripti zbog svih navedenih prednosti, a kada su u pitanju veliki projekti, na kojima radi veći broj ljudi, trebao bi se izbeći.

Na projektu radio tim MIVD.

Miroslav Radin PR19/2017

Ivan Gajić PR8/2017

Vladimir Sapundžić PR6/2017

Darie Čolak PR46/2017