Šolski center Celje
Splošna in strokovna gimnazija Lava

Spletna igra enka

(Raziskovalna naloga)

Mentor:
Borut Slemenšek, univ. dipl. inž

Avtor**:**
Matej Jakop, GL-4. F

Celje, marec 2007

Povzetek

Raziskovalna naloga obsega program za igranje igre Enka in spletno stran-portal, ki služi podpori igre. Celoten sistem je napisan v kombinaciji večih jezikov, in sicer PHP, MySQL, Turbo Delphi, HTML in CSS. Igranje igre poteka preko spleta z drugimi igralci ali pa kar proti računalniku. Celotna komunikacija preko spleta poteka s protokolom TCP/IP znotraj katerega sem razvil svoj način komunikacije. V raziskovalni nalogi je prikazan del poteka izdelave, predstavljenih nekaj rešitev določenih problemov in princip delovanja celotnega sistema.

Kazalo

Kazalo slik

Kazalo tabel

1 Uvod

1.1 Predstavitev raziskovalnega problema

Za raziskovalno nalogo sem hotel izdelati spletno verzijo priljubljene namizne igre Karte ena-Enka z mojim nazivom Spletna igra enka. Omogočala naj bi igranje preko spleta z drugimi igralci ali proti računalniku z večimi težavnostnimi stopnjami, ki bi jih bilo mogoče nastavljati preko posebnih datotek.

1.2 Hipoteza in cilji

Po dolgem razmišljanju sem se odločil, da bom naredil spletno verzijo, saj je namizna, ki je bila sicer »zastarela«, že obstajala na internetu. Vsak uporabnik se naj bi prijavil v igro in ustvaril svojo lastno sobo v kateri bi igral proti računalniku ali drugim igralcem. Za igranje bi bila potrebna registracija uporabniškega imena na spletni strani, kjer bi bile še nekatere dodatne funkcije kot sta npr. forum in predstavitev uporabnika.

1.3 Raziskovalne metode

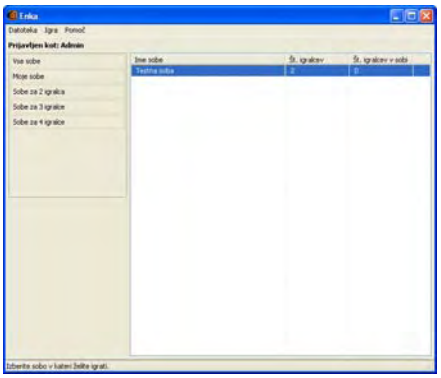
Raziskovalna naloga je bila izdelana v večih programskih jezikih. Sama igra je narejena v vizualnem okolju Turbo Delphi. Jezik tega okolja je pascal, ki sem se ga naučil sam pred leti in ga precej obvladam. Spletna stran je zasnovana v PHP, javascript (AJAX) in pa HTML v kombinacij s CSS. Na strani je uporabljen smarty template pogon, ki omogoča ločevanje kode (PHP) od oblike (HTML, CSS in javascript). To pripomore k večji preglednosti kode in lažjemu izvajanju sprememb na portalu (dodajanje dodatnih funkcij ipd.).

2 Izdelava

Pred samo izdelavo igre sem si zasnoval načrt kako naj bi celotna stvar delovala. Glavni problem je bil zasnovata sistema, da centralni strežnik, na katerem bi potekale vse igre, ne bi bil potreben, saj bi za strežnik igre (program) težko našel primernega ponudnika (Windows gostovanje tujih strežnikov za ugodno ceno). Prišel sem do rešitve, da vsak igralec, ki ustvari sobo to tudi gostuje. Se pravi princip decentraliziranega igranja s centralno prijavo. Ta princip precej spominja na p2p omrežja (torrent). Po končanem načrtovanju sem začel sistem (igra+portal) tudi izdelovati. Sama izdelava ni bila čisto ločena na dva dela(program in portal) ampak je bila predvsem sprva zelo povezana, zato sem moral večino časa preklapljati med večimi jeziki programiranja.

2.1 Program

Pri izdelavi programa sem se najprej lotil same prijave v sistem. Prijavo sem predhodno spisal na portalu, in sicer tako da mi »skripta« oziroma program na portalu vrne datoteko s podatki, ki so ločeni s TAB znakom. Na podlagi tega ve program ali je prijava uspela ali ne. Po končani izdelavi prijave sem se lotil programiranja priprave sob in pridruževanja v le-te. Ko uporabnik ustvari sobo, lahko jih tudi več, se na centralni strežnik prenesejo podatki o tej sobi in IP naslov lastnika ter vrata na katerih prisluškuje njegov strežnik. Soba se pojavi na seznamu sob in drugi uporabniki se lahko pridružijo vanjo. Ko sem dokončal ta del programa sem se lotil programiranja samega igranja. Še pred tem pa sem si moral narisati slike kart. Za to sem uporabil program Inkscape, ki je odprtokoden ter tako brezplačen za uporabo. Nakar je sledilo pisanje jedra programa. Celotno jedro je spisano v obliki večnitnega programa, kar omogoča hitrejše delovanje na večjedrnih procesorjih in pa nezatikajoč uporabniški vmesnik pri velikem številu sob in uporabnikov. Če tega ne bi storil na ta način, bi lahko prihajalo do motenj delovanja igre. V jedru so zapisani ukazi, ki popolnoma ustrezajo pravilom igre in so zasnovani tako, da se jih lahko preprosto spreminja, razširja ipd. Ko mi je uspelo spisati dovolj kode, da je igra že solidno delovala, sem jo začel testirati preko spleta.



Slika 1: Sobe

Slika 3:Prijava

Slika 2: Igra

Tukaj mi je na pomoč priskočil sošolec s katerim sem odigral par iger. Sprva se je pojavljalo kar nekaj težav. To so bile predvsem težave z ohranjanjem povezave. Kako sem ta problem rešil je zapisano pod naslovom Mehanizmi ohranjanja in preverjanja povezave med igralci in strežnikom (5. poglavje). Sedaj je prišla na vrsto vpeljava statistike igre in določanje zmagovalca. Program je sposoben voditi statistiko za vse odigrane igre v sobi vse dokler so isti člani v sobi in ne pride do razdora sobe in s tem uničenja trenutne igre. Soba se razdre v primeru, ko eden ali več igralcev zapusti sobo in ostane premalo igralcev v sobi za nadaljnje igranje. Večina statistike se vodi na portalu. Nanj pošilja program v obliki TAB ločenega besedila podatke o številu kazenskih točk, o zmagovalcu oziroma o poraženčih. Po prvih beta testiranjih sem se lotil pisanja umetne inteligence oziroma možnosti igranja proti računalniku, saj je bilo težko najti soigralca. Sistem umetne inteligence sem zasnoval tako, da se lahko dodajo osebe s pomočjo konfiguracijskih datotek in tako omogočil neomejene

kombinacije različnih parametrov, ki vplivajo na igranje. Več o umetni inteligenci je razloženo pod naslovom Umetna inteligenca (6. poglavje). Ko je bila igra skoraj končana, sem dodal še avtomatsko preverjanje, ali ima uporabnik nameščeno zadnjo verzijo igre. Na koncu sem igro dobro pretestiral in jo objavil na spletu skupaj s stranjo na naslovu enka.si.

2.2 Portal

Vzporedno s pisanjem programa sem delal tudi spletno stran, saj so bile nekatere stvari nujno potrebne v kombinaciji s programom. Tako sem najprej izdelal registracijo uporabnikov in prijavo. Prijava na portal je mogoča na dva načina:

- stalna prijava, ki poteče v roku enega dne in
- sejna prijava, ki poteče po zaprtju brskalnika

Po končani pripravi prijave, sem začel pisati administracijski del. Tukaj sem najprej naredil administracijo uporabnikov. Preko nje lahko uporabnike dodajam, brišem, spreminjam ipd. Po končani administraciji uporabnikov sem se lotil pisanja administracije sob. Tukaj lahko sobe brišem ali si ogledam njene podatke. Ne morem pa recimo spreminjat naslova sob ipd., saj to ne bi imelo učinka na uporabnika, ki sobo gostuje. Ker nobena stran ne more zaživeti, če ni nekega stika z uporabniki sem, dodal še nekaj funkcij v administracijo. To so: administracija kontaktov (pregled kontaktov s strani uporabnikov), administracija novic (objavljanje informacij o igri, novosti…), administracijo podstrani (celotno besedilo lahko uredim kar preko spletnega vmesnika), administracijo prenosa datotek (dodajanje map, datotek ipd.), administracijo anket in administracijo modula povej naprej. Ko je bila administracija končana sem se lotil izdelave vmesnika za bodoče uporabnike. Vmesnik sem naredil za vse funkcije, ki se jih je dalo urejati v administraciji. Dodal sem še klepet (deluje na principu AJAX tehnologije), forum (PHPBB) in pa urejanje lastnih podatkov-profila. Sama izdelava portala mi ni delala pretiranih težav. Le-te so se pojavile pri izdelavi klepeta, saj sem z AJAX tehnologijo delal prvič. Tudi ta problem sem s pomočjo člankov na internetu kar hitro rešil.

Slika 4: Portal uporabniki

3 Objekti

V igri je veliko podatkov, ki morajo biti združeni v celoto. Slednja predstavlja točno nek objekt npr. karto. Za pripravo objektov je idealno objektno programiranje. Princip takšnega načina programiranja je preprost. Imamo objekt. Ta objekt ima neke lastnosti, vrednosti in metode oziroma funkcije, ki se lahko izvajajo nad njim. Lastnosti so v bistvu spremenljivke, ki jih uporabljamo za shranjevanje različnih podatkov, ki se tičejo tega objekta. Objekti se ponavadi, če jih je več enakega tipa in pripadajo isti skupini npr. igralcev, nahajajo na za to ustreznem seznamu preko katerega enostavno dostopamo do njih.

Za uspešno delovanje igre sem potreboval kar precej objektov in seznamov zato bom naštel in opisal samo nekaj najpomembnejših.

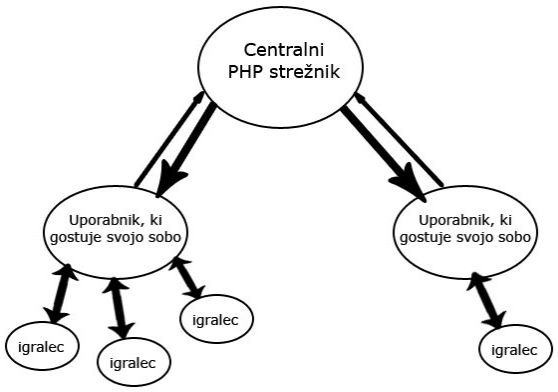
- TIgralec: TIgralec je tip objekta, ki v igri predstavlja »realnega« igralca. S pomočjo njega je definiran vsak igralec, ki je povezan na strežnik in igra v kateri izmed sob. V njem so shranjeni naslednji podatki: podatki o njegovih kartah, njegovo ime, njegov IP naslov, vrata na katerih posluša, podatki o njegovi zadnji igri in statistika le-te in podatek o sobi, v kateri se nahaja (kazalec na objekt tipa TSoba).
- TSoba: TSoba je najbolj obširen objekt znotraj igre. Predstavlja kot že ime pove »realno« sobo, ki zaključuje neko združbo igralcev, ki igrajo skupaj. Najbolj obširen objekt je zato, ker se v njem odvija vsa logika igre. Tu gre za logiko dodajanja igralcev v sobo, obveščanja igralcev o ukazih, podatkih, vodenje statistike, obdelava igre in drugo. Ta objekt zavzema tudi nekaj pomembnih lastnosti, ki so potrebne za samo delovanje igre. Naj jih naštejem samo nekaj: št. igralcev, seznam igralcev, vrhnja karta, kupček kart, odvržene karte, smer poteka igre… Metode za pošiljanje ukazov uporabnikom so zasnovane tako, da se vedno preden se pošlje naslednji ukaz, pošlje vsem uporabnikom trenutni ukaz. Tako je rešen problem s sinhronizacijo in mi tako ni bilo potrebno posebej dodajati metod za reševanje problema sinhronizacije. Težava pri takšnem načinu pa je ta, da igra deluje tako hitro kot deluje najpočasnejši med njimi. Se pravi, če ima nekdo visok ping (čas potovanja paketka) se to pozna pri vseh igralcih. Vendar v času hitrih povezav, ta problem ni preveč pereč. V sobi je shranjen tudi seznam kart, ki tej sobi pripadajo. Karte se ob nastanku sobe same zgenerirajo in premešajo. Za mešanje je uporabljena funkcija naključnih števil (random), in sicer tako, da je naključno mesto s katerim bomo neko karto zamenjali, kot število ponovitev celotnega postopka premika kart po kupu. Takšen način omogoča, da se začne igra vedno s popolnoma drugačnim zaporedjem kart. Karte se tako kot v realnosti jemljejo z vrha kupa oziroma v programu iz seznama. Ko igralec dobi karto, se ta prenese iz glavnega seznama na uporabnikov interni seznam kart. Potem pa, ko igralec karto odloži gre objekt na seznam odvrženih kart, ki se ohranja vse dokler na glavnem kupu ne zmanjka kart. Potem se kup odvrženih kart obrne in prenese na glavni kup. Na takšen način karte venomer krožijo po sobi, tako kot v realnosti.
- TKarta: TKarta je tip objekta, ki predstavlja »realno« karto. Objekt je zgrajen iz naslednjih lastnosti: vrednost, barva in tip. Vsaka izmed teh lastnosti predstavlja število. Vrednost je lahko številka od 0 do 8. Pri posebnih tipih kart je vrednost enaka vrednosti karte (npr. vzemi 4 ima vrednost 50). Barva lahko zavzema 4 vrednosti, ki so od 1 do 4. Tip pa lahko zavzema 8 vrednosti, saj je toliko število tipov kart. Za vsako sobo se ustvari svoj komplet kart. Teh kart je skupaj 100 vendar kljub temu ne zasedejo veliko prostora v glavnem pomnilniku, saj sem za njihovo predstavitev uporabil podatkovni tip shortint, ki za eno vrednost zavzame le 1B, kar pa ni veliko če pomislimo, da navadno celoštevilo (integer) v Delphiju zavzame 4B. Torej je ta način zelo primeren.

Tabela 1: Seznam kart

Tip karte	Število kart
Navadna karta(št. 0 do 8)	72
STOP karta	4
Sprememba smeri	8
Sprememba barve	4
Vzemi 2(vrzi modro ali rdečo)	4
Vzemi 2(vrzi rumeno ali zeleno)	4
Vzemi 4	3
Vzemi 5	1
SKUPAJ:	**100**

4 Osnovni principi komunikacije

Igra Enka je bila mišljena kot igra za igranje proti drugim igralcem preko spleta. Teh igralcev naj bi bilo možno neskončno oziroma omejitev bi bila strojna oprema. Enako velja za sobe. V vsaki sobi pa naj bi bilo 2-8 igralcev kot zahtevajo pravila igre. Komunikacija med klientom in strežnikov mora biti dvosmerna v smislu, da lahko klient kadarkoli kontaktira strežnik in obratno. Torej je bilo potrebno program spisati tako, da bi bili vsi ti elementi zajeti. Odločil sem se za naslednji način. Klient se poskuša povezati na strežnik. Da bo povezava uspešna mora klient poslat poseben ključ, ki pove, da gre res za pravega klienta in ne morebiti za kak drug program, ki bi se rad povezal na vrata, ki so odprta s strani strežnika. Po uspešnem sprejetju ključa strežnik od klienta zahteva, da pove svoje ime. To ime zaradi delovanja sistema ob registraciji (nobeno ime ni enako) enolično določa vsakega klienta. Če strežnik ugotovi, da je že povezan s klientom, ki ima takšno ime, novo povezavo zavrne. V nasprotnem primeru od klienta še zahteva vrata na katerih klient prisluškuje ukazom strežnika. Po pridobitvi podatka se strežnik poskuša povezati s klientom. V primeru, da je povezava neuspešna javi klientu, da ni povezljiv od zunaj. Do tega pride zaradi napačne nastavitve požarnega zidu ali usmerjevalnika. Če se povezava le vzpostavi se na strežniku ustvari objekt TIgralec, kateremu se nastavijo vse lastnosti, ki so potrebne za njegovo »življenje«. Te lastnosti so npr. ime oziroma vzdevek, IP naslov, vrata na katerih prisluškuje, podatek o sobi v kateri se nahaja (na začetku se še ne nahaja nikjer), ustvari se seznam njegovih kart, ki pa je sprva prazen ipd. Po uspešni povezavi je spet na vrsti klient, ki mora povedat v katero sobo želi. To poda z njenim imenom, ki ga je pridobil s centralnega strežnika (portala). Strežnik mora, preden ga vključi v sobo, preveriti ali ta soba obstaja in če ni klient slučajno že v tej sobi. Če soba ne obstaja javi klientu in povezava se podre. Drugače se v seznam igralcev neke sobe doda kazalec na objekt igralca. Sedaj preveri še število igralcev v sobi in igra se lahko začne. V primeru, da je



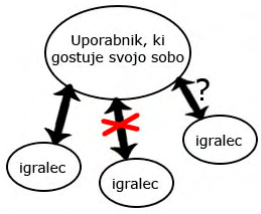
Slika 5: Shema komunikacije

temu res tako, strežnik obvesti vse ostale igralce(kliente), da se je igra začela in določi tistega, ki začne.

Celotna komunikacija med klienti in strežnikom poteka preko TCP/IP protokola. Sama povezava ni šifrirana recimo s SSL, čeprav bi tudi to bilo mogoče enostavno dodati. Vendar sem se za začetek odločil, da to še ni potrebno. Vsako sporočilo, ki si ga klient in strežnik izmenjata, je sestavljeno na sledeči način: šifra_ukaza+parametri. Šifre ukaza so nizi znakov od 00001, 00002 in tako naprej. Pred samim nizom znakov pa je še dodana predpona npr. za klientove ukaze strežniku IU. Večino ukazov zahteva povratni odziv. Leta je lahko niz znakov od 0000, 0001 in tako naprej s to razliko, da ne vsebuje predpone. Ker sobe gostujejo uporabniki se lahko zgodi, da ravno takrat, ko nekdo igra v eni izmed sob zapre program in tako prekine gostovanje, kar povzroči zmedo. V tem primeru sem predvidel delovanje tako, da se klientu najprej sporoči, da se povezava prekinja. Tako je uporabnik obveščen zakaj je temu tako. Podobno se zgodi če uporabnik zapre program, ker noče več igrati. V tem primeru klient sporoči strežniku, da noče več igrati. Strežnik podre igro in obvesti ostale kliente, da je igre konec ter da se čaka na novega igralca. Vseskozi pa se v ozadju igre in čakanja izvajajo mehanizmi za preverjanje in ohranjanje povezave med igralci in strežnikom.

5 Mehanizmi ohranjanja in preverjanja povezave med igralci in strežnikom

Igranje igre je primarno namenjeno igranju preko interneta. Zaradi same narave interneta in morebitnega prekinjanja povezave z internetom prihaja do motenj pri delovanju igre. Motnje se kažejo v tem, da se drugi igralci ne odzivajo na ukaze, ki jim jih pošilja strežnik in da tudi igralec sam ne more nič poslati strežniku. Zato sem bil primoran razviti način kako rešit sistem iz takih situacij. Samo preverjanje podrtja povezava se opravi tako pri klientu (igralcu) kot tudi s strani strežnika. Klient preverja ali je strežnik še dostopen, če ni slučajno izpadel z interneta ipd. V primeru, da ugotovi napako se poskuša ponovno povezati nanj. Če poskus povezovanja ni uspešen, javi uporabniku, da je strežnik izpadel. V tem primeru mora igralec izbrati drugo sobo, ki jo gostuje nek drug uporabnik in tako lahko nadaljuje z novo igro. Strežnik skrbi za preverjanje povezave za vse uporabnike, ki so se kdaj povezali na strežnik. Na to nimajo vpliva posamezne sobe za igro. Strežnik preverja povezljivost klientov vsakih 5 sekund. V primeru, da se klient ne odziva na njegovo preverjanje se poskuša ponovno povezati z njim in ga v primeru neuspeha označi, da se ni odzval. V vsakem intervalu se postopek ponovi vse dokler klient ni spet dosegljiv ali da število neodzivov doseže število tri. Če klient postane ponovno dosegljiv se igra lahko nadaljuje, v nasprotnem primeru se ga odstrani iz strežnika (zapis z njegovimi podatki) in igra se konča.



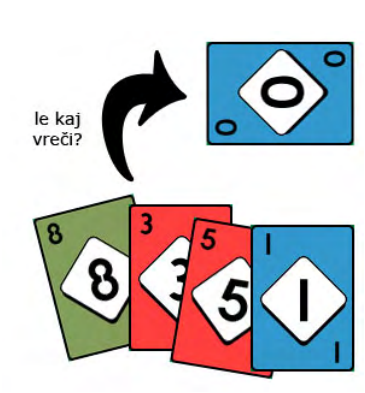
Slika 6: Preverjanje povezav

6 Umetna inteligenca

Umetna inteligenca je poskus, da bi stroj posnemal inteligentno bitje npr. človeka. Stroj je do tega težko pripraviti, še posebno če gre za kompleksen problem kot je npr. lastno razmišljanje o svojem življenju, dejanjih (tega si tudi v bistvu nihče ne želi, sploh po ogledu kakšnega filma kjer vladajo roboti). Pri izdelavi umetne inteligence obstaja več pristopov kot so npr. nevronske mreže, vendar ker po prebranem na wikipediji nisem našel najboljšega načina kako to implementirati v mojo igro, sem se odločil za lasten pristop. Odločil sem se, da bom posnemal moje razmišljanje, kako sam igram igro. To mi je kar v

veliki meri uspelo, saj igra računalnika predstavlja kar trd oreh za nekatere igralce kar opažam po statistiki na spletu.

Odločil sem se, da bom naredil umetno inteligenco na način, da bo v igri več osebkov, vsak s specifičnimi lastnostmi, ki se jih bo lahko nastavilo v posebni datoteki, ki bi bila za vsakega igralca posebej. Naredil sem tako, da datoteko preprosto dodaš v to predvideno mesto in igralec bi se naj pokazal v igri. V tej datoteki je zapisano ime igralca, opis igralca in njegova odličnost v igri. Odličnost v igri se lahko nastavlja preko treh parametrov: hitrost, doslednost in znanje. Vse te vrednosti so lahko številke od 0 do 100 ter imajo odločilen vpliv na igralčevo obnašanje v igri. Odločajo o verjetnosti, da se bo neka stvar zgodila oziroma, da bo igralec (računalnik) preveril določeno stvar npr. da ima možnost odvreči več kart hkrati). Iz številke, ki je podana pri enem izmed parametrov računalnik izračuna ali naj nekaj stori ali ne na sledeč način (x naj bo vrednost parametra):



Slika 7: Umetna inteligenca

- če je x enak vrednosti 100 potem vedno opravi nalogo
- če je x enak vrednosti 0 potem nikoli ne opravi naloge
- drugače se program zateče k funkciji naključnih števil in generirano število deli s 101 in primerja ali je x večji od te vrednosti. Če je potem se naloga opravi, drugače ne.

Posamezne funkcije, ki predstavljajo dele realnega razmišljanja (npr. iskanje barve, ki se največkrat ponovi), se pred izvršitvijo preverijo z načinom opisanim zgoraj. Takšen način daje na videz pridih razmišljanja, čeprav v resnici ni tako. Je pa dovolj učinkovito, da prepriča igralce. Ob izvajanju »razmišljanja« se izvedejo naslednji postopki, če seveda verjetnost izračuna to narekuje:

- iskanje ustreznih kart,
- izločanje kart zaradi nedoslednosti (če pride po verjetnosti do tega),
- če ima premalo znanja se izločijo največje karte,
- če ima veliko znanja si ustvari seznam povezavnih kart (pri igri za 2 igralca, to so karte s katerimi je on vedno na vrsti…to so vzemi 2, vzemi 4, obrni),
- če ima malo znanja vrže eno izmed kart naključno ali
- če ima veliko znanja izbere največjo oziroma tisto, ki njemu najbolj ustreza.

Zavedam se, da tak pristop ni med najboljšimi, vendar za pojme te igre popolnoma zadostuje. Sploh pa je malo težje napisati »zmagovalno« umetno inteligenco za to igro, ker se ne da v naprej predvidevati vseh potez kot npr. pri šahu.

7 Možnosti izboljšav

Za igro je možno še pripraviti precej izboljšav, ki jih nisem mogel vpeljati zaradi pomanjkanja časa. Najbolj potrebna bi bila izboljšava uporabniškega vmesnika v stilu animacij kart in samega izgleda. Izgled oken (form) je sedaj pravokoten, v mislih pa imam izdelavo malo bolj zaobljenih in očem prijaznejših oblik. Tako bo igra še bolj pritegnila igralce. Druga izboljšava bi bila uvedba pasivnega načina igranja. To bi rešilo težave z nastavitvami usmerjevalnikov, saj jih ne bi bilo treba posebej nastavljati. Odpreti bi bilo potrebno le vrata. Pasivni način bi bil narejen po principu, da klient venomer sprašuje strežnik ali je on na vrsti, ali je zanj kakšna karta ipd. Tretja izboljšava, ki jo imam v mislih je razširitev umetne inteligence v smislu, da bi vsak igralec v umetni inteligenci imel svoj

nabor povedi s katerimi bi v primeru poraza malo potožil ali se v primeru zmage veselil in tako jezil nasprotnika. Pri umetni inteligenci imam tudi namen narediti za vse igralce umetne inteligence možnost svoje slike. Podobno je to že narejeno za »realne« igralce. Dobrodošla bi bila tudi možnost prehitevanja in odprava omejitve igralcev na 4, kar je uporabljeno zaradi lažjega začetnega ugotavljanja napak, teoretično pa je igralcev lahko neomejeno. Realno to ni možno zaradi pomanjkanja kart!

Na spletu pride v poštev še več sprememb kot v samem programu. Če želim ustvariti nekakšno spletno skupnost moram ponuditi več kot samo igro in forum. Zato imam v mislih narediti sistem glasovanja za uporabnike igre, objave do aktualnih spletnih vsebin in podobno za privabljanje ljudi.

8 Nekaj statističnih podatkov

Sama izdelava celotnega sistema (brez pisanja tega besedila) je trajala dobra 2 meseca. V tem času nisem delal čisto vsak dan, temveč toliko, kolikor sem imel časa. Če približno ocenim porabljene ure bi rekel okoli 180 do 220 ur. Celoten program ima okoli 7000 vrstic, portal pa okoli 6800. Skupaj to pomeni okoli 13800 vrstic. Iz tega lahko nadalje sklepamo, da pišem dokaj počasi, komaj 69 vrstic na uro. Igra je bila na spletu dobra 2 meseca. Prijavilo se je okoli 600 registriranih uporabnikov, kar je kar veliko glede na to, da nisem preveč oglaševal strani (samo na dveh forumih sem podal povezavo). Ti uporabniki so odigrali skupaj okoli 3000 iger. Upam, da bo še nadaljnja rast strani tako uspešna, sploh po uvedbi dodatnih funkcij in možnosti zabave ter večji promociji spletne strani.

9 Zaključek

S pomočjo raziskovalne naloge sem še malo poglobil moje dosedanje znanje programiranja. Največ novega mi je pomenila umetna inteligenca, nekaj malega pa tudi komunikacija preko spleta, saj prej še nikoli nisem delal tako obširnega programa preko spleta, pri katerem bi sam skrbel za celoten protokol. Z raziskovalno nalogo sem potrdil svoj cilj, to je izdelati delujočo igro in portal. Po mojem mnenju tudi kar dobro.

10 Viri

- Pomoč v Turbo Delphi
- Mrhar, P. (2000). Delphi zvijače in nasveti. Nova gorica: Flamingo.
- Zandstra, M. in S. Klemen. (2004). Naučite se PHP v 24 urah. Ljubljana: Pasadena.
- AI. [Online]. [Citirano 13. januarja 2007; 17:10]. Dostopno na spletnem naslovu: http://en.wikipedia.org/wiki/AI
- PHP Manual. [Online]. [Citirano 26. decembra 2006]. Dostopno na spletnem naslovu: http://www.php.net/manual/en/

11 Zahvala

Zahvaljujem se mentorju Borutu Slemenšku za obveščanje o datumih in drugo pomoč, ki se tiče izvedbe raziskovalne naloge.