Jamajka

Završni izvještaj projekta kolegija “Umrežene igre”

Eugen Preglej
Duje Huljev
Dominik Juršić
Anteo Vukasović

Uvod

Proizvodnja videoigara je u posljednjim desetljećima postala ozbiljna industrija s velikom količinom novaca koja se godišnje uloži, ali i uprihodi. Procjenjuje se da je sveukupna vrijednost industrije dostigla gotovo 300 milijardi američkih dolara [1]. Logično bi onda bilo za zaključiti da je broj korisnika videoigara samim time velik. Zaključak bi bio ispravan, s obzirom da se pretpostavlja da preko 3 milijarde ljudi igra videoigre [1]. Osim što je brojka sama po sebi ogromna, nema naznake da bi se u skorije vrijeme mogla smanjiti; prije samo 9 godina ta brojka je bila na 2 milijarde, a svake godine u vremenskom periodu od tada do sada broj korisnika se povećao za bar 3% u odnosu na prethodnu godinu [1].

Gledajući kakve žanrove videoigara ljudi vole, prema istraživanju platforme Newzoo vidljivo je kako su najpopularnije igre pucačke igre (engl. shooters). Iza njih slijede redom avanture, igre uloga (eng. role playing games), strateške i sportske igre [2]. Pogledom na ovu listu, ali i ponukani vlastitim iskustvima u svijetu igara, shvatili smo da klasične društvene igre nisu baš popularne u svijetu video igara. Tranzicija društvene igre koja se igra uživo u umreženu videoigru na poteze nije pretjerano izazovan zadatak, no ipak ljudima to jednostavno nije toliko privlačno za igrati. Postoje popularne društvene igre čije inačice u svijetu videoigara su već razvijene i objavljene; neki od primjera su Catan, Monopoly i Uno. No, kada pogledamo brojke aktivnih igrača tih igara na Steam-u, najpopularnijoj platformi za preuzimanje, kupovinu i igranje igara na svijetu, shvatimo da one nisu ni približne dominantnim silama na tržištu igara. Od ove 3 igre najpopularnija je Catan sa maksimumom od 1800 aktivnih igrača u posljednjih tjedan dana, što je gotovo ništa u usporedbi s najpopularnijim igrama na tržištu čije brojke nerijetko sežu u milijune aktivnih igrača u jednom trenutku [3].

Uzimajući sve to u obzir, odlučili smo napraviti umreženu virtualnu verziju neke društvene igre. Htjeli smo obraditi igru čije relativno popularno komercijalno rješenje ne postoji, već bi mi bili među pionirima. Konačno, odluka je pala na društvenu igru Jamajka.

U idućim poglavljima će biti prezentiran kratki opis postojeće društvene igre Jamajka. Zatim će biti dan opis naše implementacije, s posebnim naglaskom na umrežene aspekte igre. Na samom kraju će biti vidljivi konačni rezultati našeg rada.

O društvenoj igri “Jamajka”

Jamajka je društvena igra u kojoj se igrači natječu da dođu što prije do cilja, ali da pritom skupe što više resursa koji su na putu do cilja dostupni. Igru su osmislili i 2007. izdali Malcolm Braff, Bruno Cathala i Sebastien Pauchon [4]. Igru može igrati 2-6 igrača, a prikladna je za sve dobne uzraste.

Ova igra ima piratsku temu; svaki igrač je jedan od pirata koji sudjeluje u Velikom Izazovu piratske legende Henrya Morgana. Veliki Izazov je utrka oko otoka Jamajke, no u ovoj utrci nije važno samo doći do cilja prvi već je važno i prikupiti dovoljno zlata po putu.

Slika 1*: Otok Jamajka u igri, preuzeto s [5]*

Osnovni elementi ove igre su igraće kockice, akcijske kartice, spremišta, blaga i resursi. Igraće kockice se bacaju dva puta istovremenu po potezu, a vrijednosti na njima određuju vrijednosti akcija koje igrači mogu poduzeti. Kada su kockice bačene, svaki igrač baca odigrava svoju akcijsku karticu koja određuje na koji način će iskoristiti vrijednosti na kockicama. Akcijske kartice mogu igraču omogućiti da se pomakne naprijed po ploči ili da se napuni jednom od 3 vrste resursa; zlato, hrana i barut. Svaki od resursa ima svoju važnost i svoj način na koji se koristi.

Zlato je važno na kraju jer ono određuje pobjednika, no osim toga, svaki put kada se igrač nađe u jednoj od luka koje se nalaze na putu, on mora dati određenu količinu zlatnika kako bi mogao nastaviti igru. Na isti način, kada se nađe na otvorenom moru, igrač mora iskoristiti određenu količinu hrane. Količine zlata i hrane koje se koriste na nekom polju su označene na mapi, a uglavnom variraju od 1 do 4. Barut se koristi u slučaju kada se 2 igrača nađu na istom polju. Tada se bacaju kockice kako bi se

odredilo tko pobjeđuje, a svaki igrač iznosu na svojoj kockici može dodati proizvoljnu količinu baruta kojeg posjeduju kako bi poboljšali šanse za pobjedu i tako izbjegli pomicanje unazad.

Resursi se pohranjuju u spremištima koje igrači imaju. Svaki igrač ima 5 skladišta, a količina resursa koja se u njima može pohraniti je ograničena, stoga igrači moraju biti pažljivi koliko resursa koje vrste spremaju. Na putu do cilja postoje i posebne pećine sa skrivenim blagom. Blago je u stvari posebna kartica koja igraču daje neke dodatne mogućnosti. Blago može igraču pomoći na kraju igre i dodati zlatnike na njegov ukupni broj, može mu ih oduzeti ukoliko nema sreće, a postoje i posebna blaga koja modificiraju neke druge aspekte igre, primjerice daje igraču dodatno spremište ili slično.

Igra završava kada jedan od igrača obavi krug oko otoka i dođe do cilja. U tom trenutku se za svakog igrača računa njegov konačni rezultat. Rezultat se dobiva kao kombinacija broja zlatnika koje igrač posjeduje, blaga kojeg je igrač tokom igre skupio (bilo pozitivnog ili negativnog) i njegove udaljenosti od cilja u trenutku kraja igre (igrač koji je došao do cilja dobiva najveći broj bodova, a ostali dobivaju bodove na temelju njihove udaljenosti od cilja).

U sklopu ovog projekta društvena igra Jamajka je rekreirana na način da pruži što sličnije iskustvo kao što bi ono bilo da se igra u stvarnom život, na stolu s prijateljima. Velika većina mogućnosti navedenih u ovom poglavlju je implementirana u igri, no više informacija o tome možete pronaći u idućim poglavljima. Za više informacija o igri možete provjeriti službeni priručnik igre objavljen na stranici World of Board Games [6].

Opis razvijene igre

U prethodnom poglavlju je opisana stvarna društvena Jamajka, dok će u ovom poglavlju biti dan opis igre razvijene u sklopu ovog projekta. Biti će navedeni korišteni alati, dijagram toka igranja te osnovni detalji implementacije. Valja napomenuti kako u detaljima implementacije neće biti pokriveni umreženi aspekti nego će svi detalji vezani za umrežavanje biti pokriveni u zasebnom poglavlju iza ovog.

Korišteni alati

Unity

Unity je razvojni softver korišten za izradu aplikacija ili igara u 2D ili 3D okruženju [7]. Dostupan za upotrebu postao je 2005. godine, a kreiran je od strane kompanije "Unity Technologies". Unity nudi razvoj na više platformi kao što su Windows, Android, iOS, Linux i slične. Sami razvoj aplikacija u Unity-u temelji se na izgradnji scena pomoci raznih objekata koji se organiziraju u određenu hijerarhiju. Svaki objekt se može nadograđivati dodavanjem proizvoljnih gotovih komponenti te uređivanjem zadanih parametara svake komponente. Mnogo posla u Unity-u se može napraviti na drag-and-drop principu, primjerice dodavanje objekata u scenu, razmještaj objekata, dodavanje novih komponenti i slično. Za neke detaljnije i naprednije funkcionalnosti moguće je pisati vlastite skripte u jeziku C#. Unity također nudi "Asset Store", biblioteku raznih materijala i gotovih komponenti razvijenih od strane Unity Technologies, ali i drugih korisnika ovog softvera. Sa Asset Store-a moguće je kupovati ili preuzimati besplatne materijale i koristiti ih u vlastitim Unity projektima.

Microsoft Visual Studio

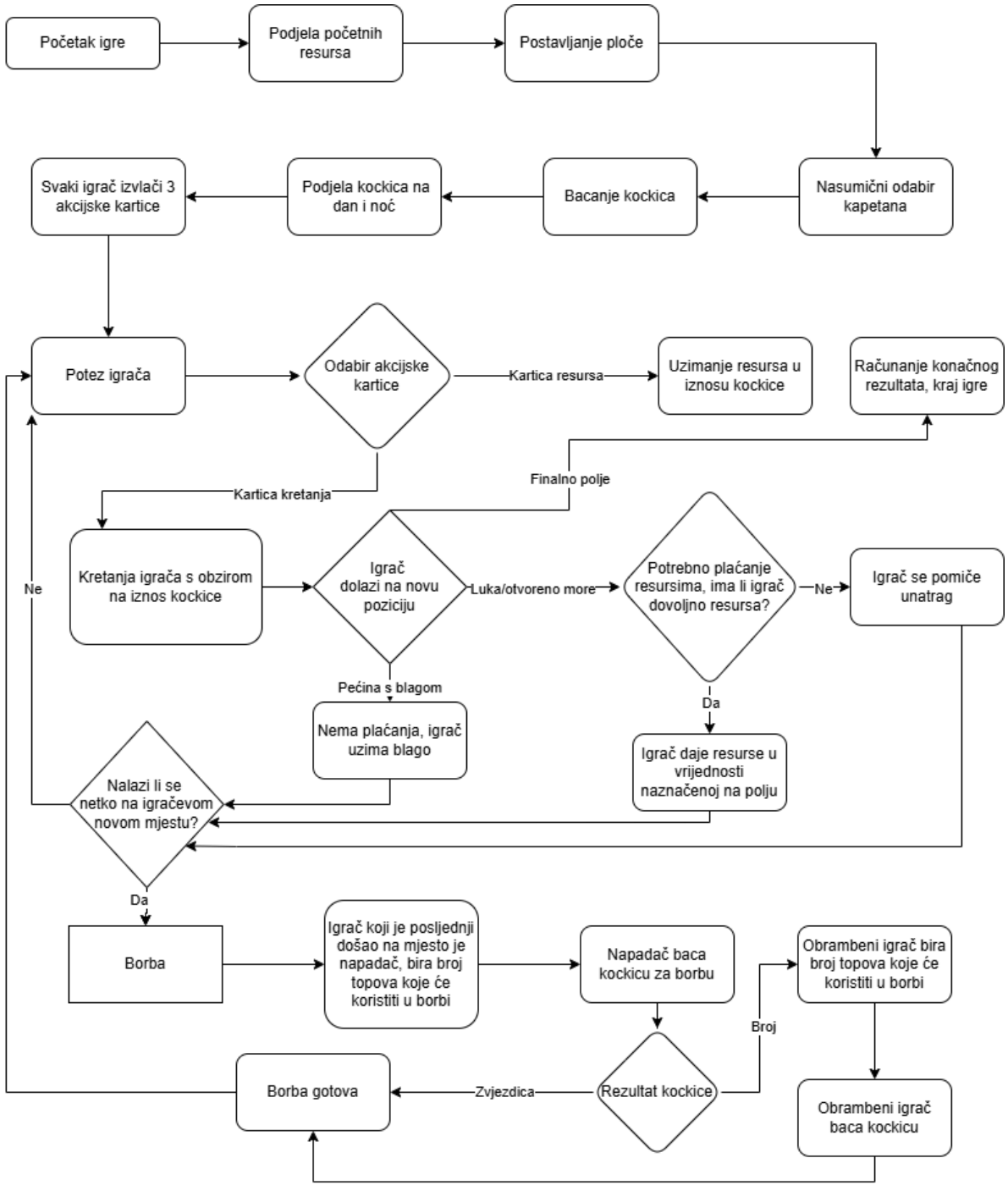
Microsoft Visual Studio je integrirano razvojno okruženje razvijeno od strane tvrtke Microsoft [8]. Koristi se za razvoj raznih web aplikacija, web usluga, mobilnih aplikacija i slično, a podržava rad u programskim jezicima kao što su C++, C#, JavaScript, TypeScript, Python i brojni drugi. Za potrebe razvoja ove aplikacije Microsoft Visual Studio korišten je za pisanje vlastitih skripti u programskom jeziku C#. Te skripte su korištene za ostvarivanje dodatnih, naprednijih funkcionalnosti u softveru Unity.

Git/Github

Git je distribuirani sistem za kontrolu verzija programskog koda prikladan za aplikacije koje razvija više ljudi istovremeno [9]. GitHub je platforma koja koristi Git kako bi programerima omogućila pohranjivanje, upravljanje i dijeljenje programskog koda i povezanih datoteka [10].

Dijagram toka igre

Dijagram toka igre je način vizualizacije tijeka jednog tipičnog igranja igre Jamajka. Dijagram toka razvijene igre vidljiv je na slici 2.



Slika 2: *Dijagram toka igre Jamajka*

Detalji implementacije

U ovom potpoglavlju biti će iznesen opis posla koji je napravljen za implementaciju ove igre. Navesti će se korišteni materijali te navesti i kratko opisati skripte korištene za implementaciju ove igre. Implementacija se može podijeliti na tri dijela:

- virtualno okruženje,
- korisničko sučelje te
- izvođenje tijeka igre.

Virtualno okruženje

U ovu kategoriju spada izrada virtualnog okruženja unutar kojeg se izvodi igra. Virtualno okruženje je izrađeno tako da što vjernije reprezentira stvarnu društvenu igru Jamajka. U toj igri, ploča na kojoj igrači igraju sadrži veliki središnji otok okružen manjim otocima. Na otocima se nalaze razne luke sa svojim pripadnim naseljima, plaže, stijene i pećine s blagom, a većina otoka je prekrivena zelenilom.

U našoj implementaciji igre čitavo virtualno okruženje izrađeno je korištenjem paketa POLYGON – Pirate Pack [11]. Korišteni su već gotovi elementi, točnije prefabovi, koji su po principu drag-and-drop postavljani u scenu. Konačni izgled poprilično vjerno replicira ploču igre Jamajka, a vidljiv je na slici 3.

Slika 3: *Izgled virtualnog okruženja igre*

U sklopu virtualnog okruženja potrebno je bilo i implementirati polja po kojima se igrači kreću tokom igranja. Svako polje implementirano je uz pomoć četiri square objekta te jednog canvas objekta. Square objekti su okviri polja te su tu za estetski dojam, a na canvas objektu su slikom prikazani tip polja te broj resursa koje to polje zahtijeva, ukoliko se radi o luci ili otvorenom moru. Primjer jednog polja, otvorenog mora koje zahtijeva 4 jedinice hrane kada se stane njega, vidljiv je na slici 4.

Slika 4: *Primjer jednog polja*

Osim vizualnog izgleda, svakom polju su potrebne i skripte kako bi se osiguralo pravilno funkcioniranje same igre. Skripta *Square* je postavljena na svako polje te daje o informacije o poziciji polja u redoslijedu obilaska igre, tipu polja, vrijednosti resursa koja se odbija od igrača kada stane na polje te indeksi svih igrača koji se na tom polju nalaze. Polja mogu biti po tipu otvoreno more, ili pećina s blagom, a ukoliko je polje pećina s blagom u skripti je pohranjena i kartica s blagom koja je različita za svako polje tog tipa.

U igri se nalazi 31 polje raspoređeno u jedan veliki krug oko otoka. Za razliku od prave igre, u našoj implementaciji zbog jednostavnosti nije implementirano grananje putova. No i bez toga doživljaj igranja bi trebao biti isti. Izgled virtualnog okruženja sa svim poljima vidljiv je na slici 5.

Slika 5: *Razmještaj polja*

Sva polja su hijerarhijski grupirana u prazni objekt koji sadrži skriptu *SquareManager*. Ova skripta sadrži listu svih polja u igri, te posebne oznake za prvo i posljednje polje koje se koriste u njenim metodama. Važne funkcionalnosti koje omogućuju metode ove skripte su dohvaćanje polja na kojem se nalazi određeni igrač, dohvaćanje igrača koji se nalazi na određenom polju, dohvaćanje tipa određenog polja te pronalaženje najbližeg polja nekog tipa.

Korisničko sučelje

Implementacija korisničkog sučelja može se podijeliti na izradu glavnog izbornika, akcijskih kartica, spremnika za resurse te izradu korisničkog sučelja za bacanje kockice i borbe između 2 igrača koja su se našla na istom polju. Korisničko sučelje u ovoj igri uglavnom je izrađeno korištenjem postojećih Unity canvas elemenata kao što su Button, Text, Image i slično. Za uređivanje elemenata korišteni su materijali paketa Dark Fantasy GUI preuzetog sa Unity Asset Store-a [12].

Slike za akcijske kartice korištene u ovoj igri su dobivene skeniranjem akcijskih kartica fizičke verzije igre Jamajka. Skenirane slike su uvezene u Unity projekt gdje su od njih napravljeni spriteovi korišteni u izradi akcijskih kartica unutar igre. Svaka akcijska kartica unutar igre je implementirana jednim objektom tipa Scriptable Object. Svaki taj objekt sadrži sliku kartice, točnije sprite, jedinstveni brojčani identifikator te 2 tipa akcije koja igrač poduzima ukoliko odigra tu karticu. Tipovi akcije su kretanje unaprijed, kretanje unazad, uzimanje hrane, uzimanje zlatnika, uzimanje baruta i nepoduzimanje nikakve akcije. Primjer akcijskih kartica vidljiv je na slici 6. Osim akcijskih kartica, izgled resursa i kartica s blagom je također dobiven skeniranjem fizičke verzije igre Jamajka.

Slika 6: *Akcijske kartice*

Za potrebe izrade elemenata korisničkog sučelja izrađene su i skripte koje svakom od elemenata korisničkog sučelja omogućuje interaktivnost s korisnikom te izvođenje radnji koje su važne za pravilno izvođenje igre. Te skripte su:

- *ActionCarSidebarScript,*
- *ActionCardsUIScript,*
- *CombatUIScript,*
- *DiceUIScript,*
- *MainMenuUIScript* te
- *HoldUIScript.*

Sve navedene skripte imaju zajedničku funkcije određivanja i postavljanja elemenata koji se koriste u prezentiranju informacija o igri korisniku. No svaka skripta ima svoje posebnosti koje će ovdje biti navedene.

Skripte *ActionCarSidebarScript* i *ActionCardsUIScript* omogućuju prikaz akcijskih kartica koje korisnik ima u svom posjedu. Ukoliko igrač posjeduje posebnu karticu s blagom *Morgan’s* , ove skripte omogućuju da igrač istovremeno posjeduje 4 akcijske kartice umjesto standardnih 3. Također, u ovim skriptama je omogućen odabir kartice koju igrač namjerava iskoristiti u nekom potezu.

Skripta *CombatUIScript* omogućuje korisniku prikaz svih informacija vezanih za borbu. Te informacije uključuju naziv igrača protiv kojeg se bori, odabir broja jedinica baruta korištenog u borbi, odabir broja jedinica baruta korištenog u borbi od strane protivnika, bacanje kockice za borbu te rezultate bacanja i u konačnici rezultat borbe. Također, u slučaju da igrač posjeduje karticu s blagom *Saran’s Saber* ova skripta mu omogućuje ponovno bacanje kockice ukoliko nije zadovoljan inicijalnim rezultatom.

Skripta *DiceUIScript* omogućuje igraču koji je kapetan za taj potez da zamijeni redoslijed kockica, odnosno promjeni koja je kockica namijenjena za dan, a koja za noć.

Skripta *MainMenuScript* omogućuje priključivanje igrača u igru pritiskom na Button objekte. Priključivanje je moguće u host i client načinu radu, no više o tome će biti u idućem poglavlju. Također, omogućuje prikazivanje popisa igrača priključenih u igru koji se osvježava s obzirom na to koliko se igrača pridružilo.

Skripta *HoldUIScript* omogućuje promjenu resursa u spremnicima. Ove promjene se odvijaju kada igrač stane na polje luke i otvorenog mora ili kada puni spremnike resursima na temelju akcijske kartice koje je odigrao. Također, u ovoj skripti je omogućeno prikazivanje razlike u broju resursa prije i poslije određene akcije.

Izvođenje tijeka igre

Za tijek igre glavnu odgovornost preuzimaju skripte *GameManager* i *PlayerGameScript. GameManager* osigurava ispravnu izvođenje tijeka igre, prati stanje igre u svakom trenutku te poziva vlastite ali i metode drugih skripti kako bi osigurala pravovremeno izvršavanje svih dijelova igre onako kako je navedeno na dijagramu toka igre (slika 2). *PlayerGameScript* je u stvari reprezentacija jednog igrača i njegovog stanja u svakom trenutku igre.

GameManager skripta interno sadrži 4 enumeracije koje služe označavanju određenih aspekata igre. Te enumeracije su sljedeće:

- TreasureCard,
- TokenType,
- ActionCardOption te
- SquareType.

TreasureCard označava posebne kartice s blagom kojim igrač može dobiti, ali i izgubiti bodove kojima se na kraju računa pobjednik igre. Postoji 6 ovakvih kartica, ali i posebne 4 kartice s blagom; Morgan’s Map, Saban’s Sabre, Lady Beth i 6th Hold. Morgan’s Map svom vlasniku omogućuje posjedovanje 4 akcijske kartice u ruci umjesto standardne 3. Saban’s Sabre i Lady Beth su povezane uz borbu igrača na istim poljima. Prva kartica vlasniku dopušta ponovno bacanje kockice za borbu, a druga automatski dodaje 2 boda na vrijednost dobivenu u procesu borbe koji će biti objašnjen nešto kasnije. 6th Hold korisniku daje dodatni spremnik za pohranjivanje resursa. Izgled kartica s blagom vidljiv je na slici 7.

Slika 7: *Izgled kartica s blagom*

Enumeracije SquareType i TokenType odnose se na tip polja na kojem se igrač može nalaziti te tip resursa koje na tom polju mora koristiti ukoliko želi nastaviti igru normalnim tijekom, o čemu je više bilo riječ u opisivanju skripte *Square*. Uz njih, postoji i token tipa Cannon koji se koristi u borbama. O njemu će biti biše riječi u nastavku ovog potpoglavlja. ActionCardOption sadrži tipove akcija koji se mogu nalaziti na akcijskim karticama, a koji su opisani u prijašnjem potpoglavlju.

Izvođenje tijeka igre započinje pozivom metode StartGame() unutar koje se postavlja lista svih igrača te poziva metoda DistributeStartingResources(). Ta metoda služi za dodavanje početnog broja resursa svakom igraču te postavljanje kartica s blagom na polja koja su po tipu pećine s blagom. Tijek igre se dalje ostvaruje naizmjeničnim pozivima metoda StartGameCycle() i EndGameCycle(). Jedan ciklus pozivanja ovih metoda je u stvari jedan potez unutar igre.

Ciklus počinje pozivanjem metode ThrowDice() koja daje 2 nasumična broja od 1 do 6 koja će se služiti kao vrijednosti igračevih akcija. Jedan broj pripada “noćnoj” kockici, a drugi “dnevnoj”. Svaki igrač zatim odabire akcijsku kartu koju za taj potez i brojke sa kockica želi odigrati. Metoda ExecutePlayerAction() obavlja akcije koje se nalaze na akcijskoj kartici igrača. Za akcije koje se odnose na nadopunjavanje resursa, poziva se metoda LoadResourceToPlayerHold(). Za akcije koje se odnose na kretanje igrača unaprijed ili unatrag korištena je pomoćna skripta *PlayerMovement*. Zadaća ove skripte je pomicanje igrača polje po polje onoliko puta koliko je to zadano u nekoj drugoj skripti koja poziva metode *PlayerMovement* skripte. Kretanje je moguće unaprijed i unatrag, a kretanje je realizirano interpolacijom položaja i rotacije između 2 polja. Na kraju svake kretnje provjeravaju se dvije važne stvari; na kojem tipu polja je igrač stao te nalazi se na tom polju već neki igrač. Kako se igrač kreće tako i kamera, kojom igrač vidi stanje na ploči, mijenja svoju poziciju i usmjerenja kako bi pratila brod koji putuje od polja do polja. Ovo je realizirano pomoćnom skriptom *CameraController.*

Prvo se metodama već ranije spomenute skripte *SquareManager* provjerava stoji li na istom polju više igrača te ukoliko je to slučaj počinje faza borbe. U svakoj borbi igračima je omogućeno biranje koliko jedinica resursa topova žele uložiti u borbu. Na to se još zbraja rezultat bacanja posebne kockice za borbu. Ova kockica može poprimiti 2, 4, 6, 8, 10 i posebnu vrijednost označenu zvjezdicom. Pobjednik se određuje većim zbrojem brojeva s kockice i uloženih topova, a zvjezdica označava automatsku pobjedu. Korištenje mogućnosti sa kartica s blagom Saban’s Sabre i Lady Beth je omogućeno za vrijeme borbe. Nakon borbe pobjednik od gubitnika može ukrasti sve resurse iz jednog od njegovih spremnika ili ukrasti jednu od njegovih kartica s blagom.

Ukoliko nije došlo do borbe dolazi do provjere tipa polja na kojem se igrač nalazi. Ukoliko se igrač nalazi na polju pećine s blagom dodjeljuje mu se kartica s blagom koja se na tom polju nalazi. U slučaju otvorenog mora, igrač mora platiti onoliko jedinica resursa hrane koliko je na tom polju označeno, a slično se dešava i sa lukama i zlatnicima. Ukoliko igrač u svojim spremnicima nema dovoljno potrebnih resursa on ponovno baca kockicu za borbu koja ovog puta određuje koliko polja unazad se igrač pomiče. Vrijednosti 2 i 4 igrač se vraća u najbližu luku, 6 i 8 ga vraćaju na najbliže otvoreno more, a 10 ga vraća u najbližu pećinu. Ukoliko igrač dobije zvjezdicu ostaje na svom mjestu.

Nakon što je svaki od igrača odradio svoje akcije te su obavljena sva potencijalna plaćanja i borbe, obavlja se provjera o tome jeli došlo do kraja igre u metodi

EndGameCycle(). Do kraja igre se dolazi kada jedan od igrača obavi cijeli krug oko ploče i dođe ponovno na start/cilj. Igrač koji prvi obavi krug dobiva 15 bodova. Igrači koji su završili 7 ili manje polja udaljeni od cilja dobivaju bodove na sljedeći način; igrač koji je završio jedno mjesto od cilja dobiva 10 bodova, igrač koji je završio 2 mjesta od cilja dobiva 9 bodova i tako dalje sve do igrača koji je završio 7 mjesta od cilja i dobiva 3 boda. Na ove bodove se zbrajaju ili oduzimaju bodovi označeni na karticama s blagom koje su igrači skupili kroz igru. Na samom kraju prikazuju se konačni rezultat igre i proglašava pobjednik.

Skripta *PlayerGameScript* u sebi pohranjuje stanje i imanje nekog igrača u svakom danom trenutku. Unutar ove skripte se nalaze informacije o broju kartica s blagom koje igrač posjeduje. Također navedene su i akcijske kartice koje igrač može iskoristiti. Unutar ove skripte su također implementirane metode ShuffleDeck() i ReshuffleDeck() koje miješaju špil akcijskih karata te metoda UpdateActionCardsInHand() koja igraču dodjeljuje novu akcijsku kartu nakon što iskoristi jednu koju već posjeduje. Unutar ove skripte su također pohranjene i informacije o broju resursa unutar spremnika svakog igrača. Svaki igrač posjeduje 5 spremnika unutar koje može pohranjivati resurse po želji. Jedan spremnik može u sebi sadržavati resurse koje su dobiveni iz samo jedne akcije sa akcijske kartice. Ukoliko igrač napuni sve spremnike nudi mu se opcija uklanjanja resursa iz jednog od spremnika te zamjena s novim resursima.

Umreženi aspekti igre

U prijašnjem poglavlju opisana je implementacija igre i izvođenje tijeka igre, no jedan važan dio je u potpunosti preskočen. S obzirom da je ovaj projekt razvijen za kolegij “Umrežene igre” sasvim logično je da je opisana igra umrežena, što znači da je mogu igrati igrači bez obzira na kojoj lokaciji ili mreži se nalaze. U našem projektu nije realizirana globalna umreženost već je potrebno da se igrači nalaze u istoj lokalnoj mreži. U ovom poglavlju će biti opisano što je korišteno kako bi se ta umreženost realizirala.

Za umrežavanje igre korišten je Netcode for GameObjects (skr. NGO)[13]. Radi se o biblioteci izgrađenoj specifično za razvoj umreženih igara u programu Unity što je upravo ono što nama treba. Ova biblioteka apstrahira umrežavanje i sav posao koji se obavlja u protokolima niže razine. Koristeći ovu biblioteku korisnik može uspostaviti sjednicu s više igrača koji mogu međusobno izmjenjivati informacije i slati razne podatke preko mreže bez da se zamara što se događa “ispod haube” i na koji se način to sve skupa realizira što uvelike olakšava posao razvijanja igre. Na ovaj način se manje vremena može posvetiti umrežavanju, a više razvijanju značajki same igre.

Ova biblioteka nudi mnoštvo gotovih objekata, skripata i metoda spremnih za korištenje, no ovdje će se izdvojiti ono najvažnije korišteno za realizaciju ovog projekta. To je sljedeće:

- NetworkManager,
- NetworkObject,
- NetworkVariable te
- RPC.

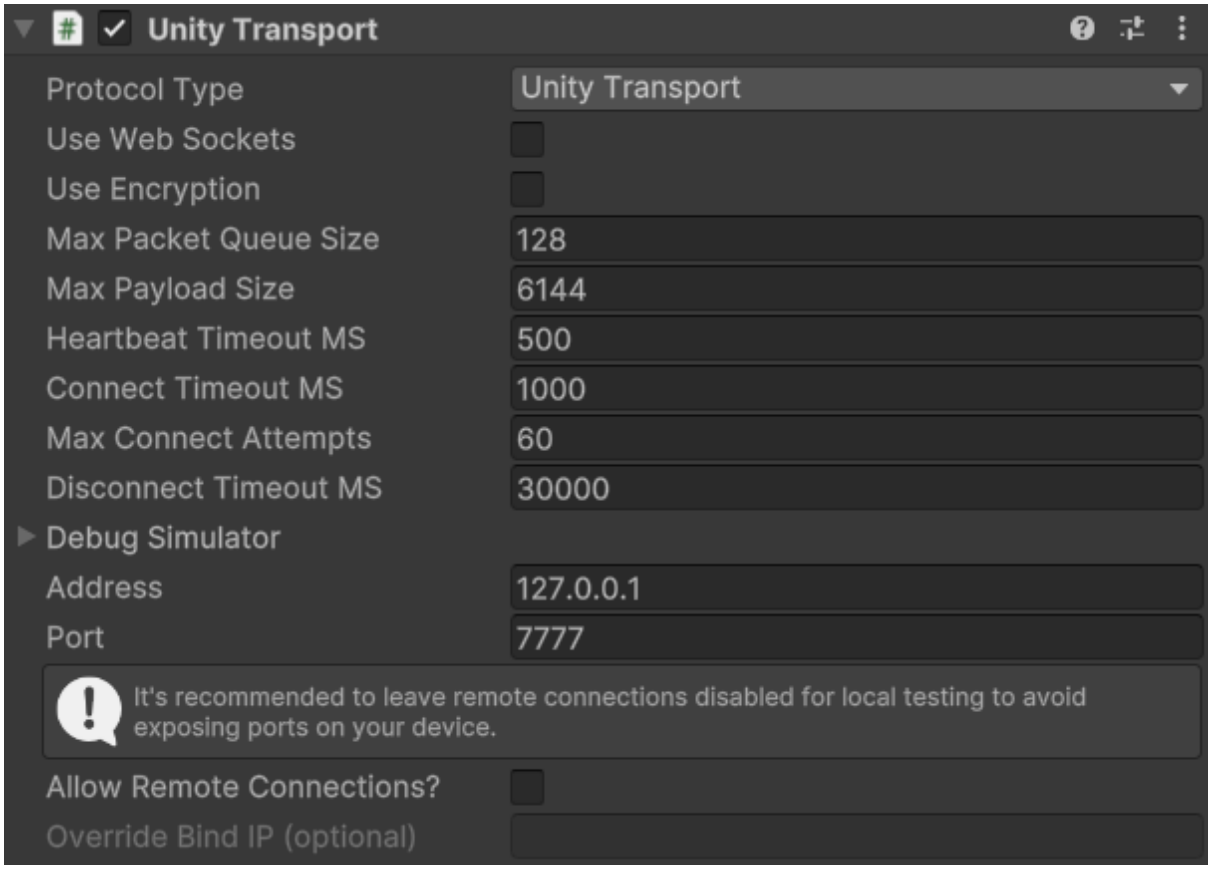
U idućim potpoglavljima biti će opisano kako je svaka od navedenih komponenti iskorištena i implementirana u našem projektu.

NetworkManager

NetworkManager je obavezna komponenta u svakoj implementacija koja koristi NGO biblioteku [13]. Ovu komponentu možemo smatrati centrom za upravljanje svega što ima veze s umrežavanjem. Pomoću NetworkManager-a se uspostavlja komunikacija igrača u igri te se održava sve dok igra traje i dok su igrači priključeni na mrežu. NetworkManager je u stvari skripta koja je u našem projektu dodana na prazni objekt istog imena. Podešavanjem parametara skripte moguće je podesiti način na koji se ostvaruje umreženost igre (slika 10). Parametri koji su nama bili bitni za postavljanje su NetworkTransport, NetworkTopology te DefaultPlayerPrefab.

Parametar NetworkTransport određuje transportni sloj kojim se prenose podatci preko mreže od igrača do igrača. Za našu implementaciju odlučili smo se za

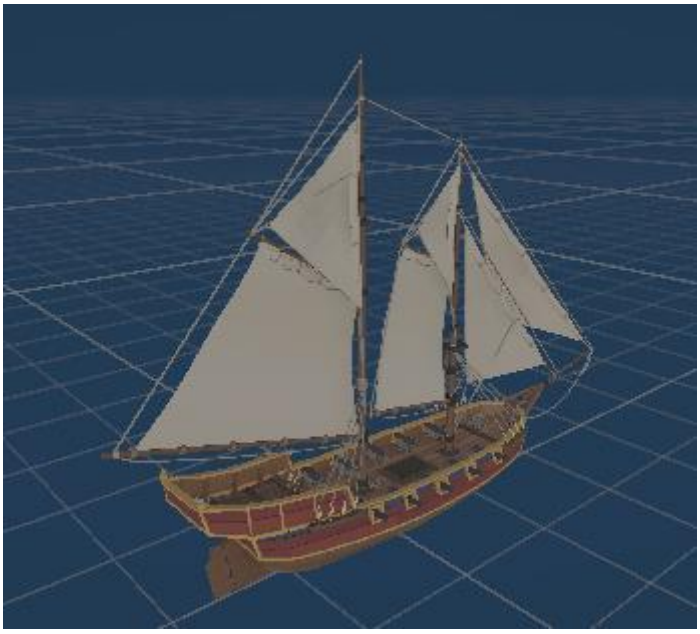
UnityTransport [14]. To je podrazumijevani transportni sloj koji je posebno prilagođen projektima izrađeni u Unity-u, a može ga se koristiti i neovisno o NGO biblioteci. UnityTransport se dodaje kao zasebna komponenta NetworkManager komponenti te ona sama po sebi ima niz parametara koji se mogu po volji podešavati (slika 8).



Slika 8: *Parametri UnityTransport komponente*

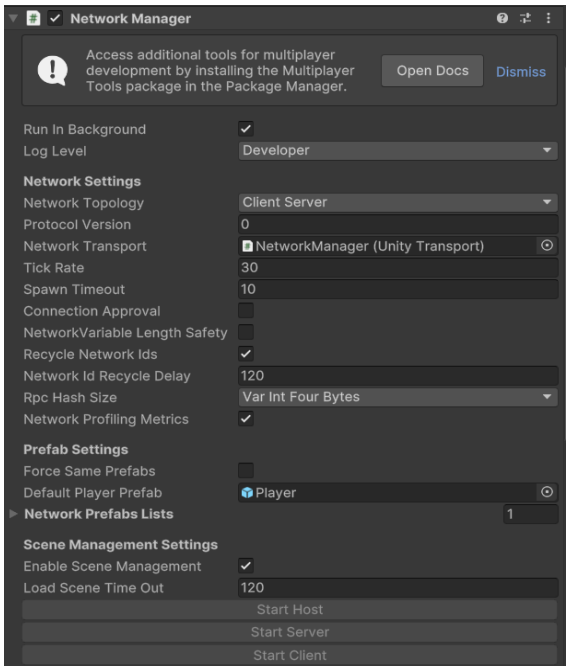
NetworkTopology određuje arhitekturu mreže korištene za povezivanje više igrača. U našem projektu odabrana je opcija Client Server. U ovoj arhitekturi sva komunikacija i svi izračuni potrebni za izvođenje igre vrše se na jednom mjesto, računalu koji se naziva server. Tom serveru se pridružuje proizvoljan broj klijenata koji cijelo vrijeme komuniciraju sa serverom. Konekcija se uspostavlja već u glavnom izborniku odabiranjem opcije za pokretanje igre kada se poziva jedna od dvije metode; StartHost() ili StartClient(). Prvi igrač koji se priključi u igru dobiva host ulogu koja u stvari znači da je taj igrač istovremeno i poslužitelj i klijent, dok svi igrači iza njega poprimaju ulogu klijenta.

DefaultPlayerPrefab omogućava instanciranje jednog prefaba koji će služiti kao središnji objekt koji pripada jednom od igrača, reprezentacija igrača unutar igre. Za naš projekt odabran je model broda iz paketa POLYGON - Pirate Pack [11]. Prefab je nazvan jednostavnim imenom Player, a odabrani model vidljiv je na slici 9.



Slika 9: *Model broda korišten za prefab Player*

Postoji još niz parametara komponente NetworkManager koji su mogu podešavati no ovo su najbitniji za naš projekt. Izgled konačno definirane komponente u Inspector kartici vidljiv je na slici 10.



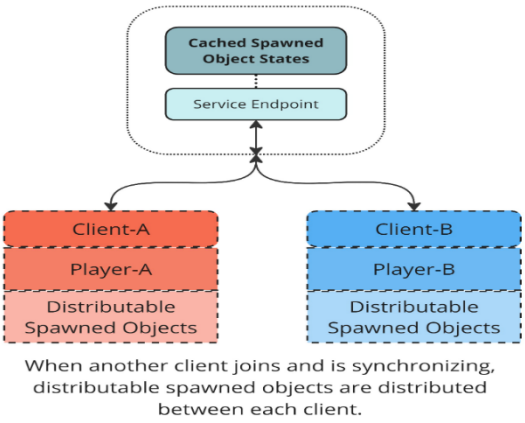
Slika 10: *Parametri komponente NetworkManager*

NetworkObject

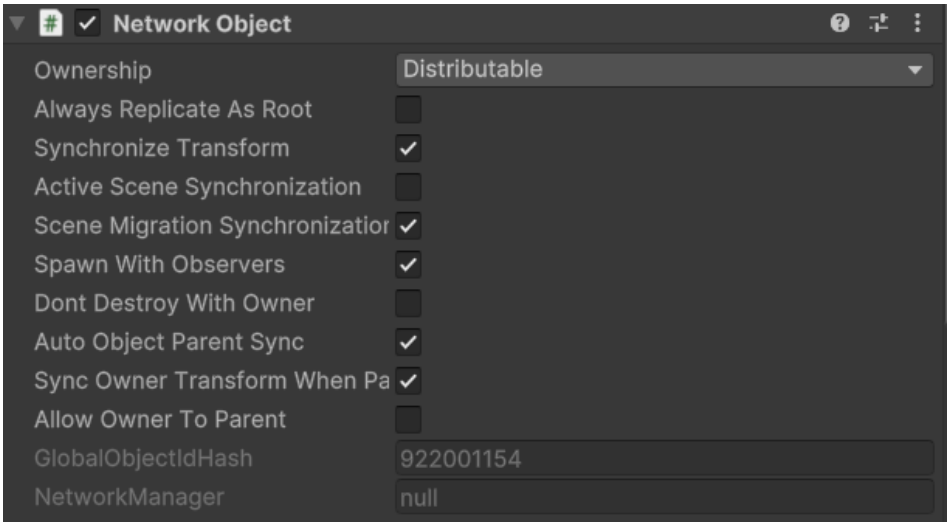
NetworkObject je komponenta koja se može dodati na bilo koji objekt unutar igre te mu omogućiti da “postane” umreženi objekt [13]. Na taj način se omogućuje objektu da surađuje s ostalim NGO komponentama te se sinkronizira i razmjenjuje poruke i podatke s drugim objektima na udaljenim mjestima u mreži. Osim dodavanja same NetworkObject komponente, za postizanje umreženosti objekta potrebno je i postojanje barem jedne NetworkBehaviour komponente priključene na taj objekt. Ovo se može postići dodavanjem skripte koja nasljeđuje klasu NetworkBehaviour umjesto standarnog i podrazumijevanog nasljeđivana klase MonoBehaviour. Unutar tih skripti moguće je koristiti NGO koncepte kao što su pozivi udaljenih procedure (eng. *Remote Procedure Calls, skr. RPC*) i NetworkVariable koji su već spomenuti i o kojima će više riječi biti u nastavku ovog poglavlja. Skripte koje nasljeđuju NetworkBehaviour u našem projektu su:

- *GameManager,*
- *PlayerGameScript,*
- *PlayerMovement,*
- *CombatUIScript,*
- *SquareManager,*
- *Square,*
- *MainMenuUIScript* te
- *CameraController.*

Svaka NetworkObject komponenta kao i NetworkManager i UnityTransport imaju niz parametara koji se mogu podešavati, no najvažniji za nas je parametar Ownership. Ovaj parametar određuje tko od sudionika u mreži ima autoritet nad nekim umreženim objektom. U našem projektu za vrijednost parametra odabrana je opcija Distributable. Ova opcija omogućuje distribuciju vlasništva nad nekim umreženim objektom između klijenata pridruženih u igru. Ilustracija ovog koncepta vidljiva je na slici 11 koja je preuzeta s [13]. Vrijednosti drugih parametara na primjeru NetworkObject komponente prefaba Player vidljiva je na slici 12.



Slika 11: *Ilustracija koncepta Distributable vlasništva nad umreženim objektima*



Slika 12: *NetworkObject parametri*

NetworkVariable

NetworkVariable je način sinkroniziranja svojstava između sudionika mreže koja održava perzistentnu povezanost i razmjenu informacija o stanju svake varijable [13]. Ove varijable su u stvari omotači proizvoljnog tipa podataka <T>, a vrijednosti podatka koja se ovim načinom sinkronizira može se pristupiti svojstvom NetworkVariable.Value. Vrijednosti u ovim varijablama između klijenata se sinkroniziraju u trenutcima dodavanja novih klijenata u mrežu ili u trenutcima kada se mijenja vrijednost unutar same varijable. Moguće je podešavanje pristupa varijabli, odnosno dozvole čitanja varijable i unosa novih vrijednosti. U našem projektu su korištene podrazumijevane dozvole; server ima pravo čitanja i unosa dok klijenti imaju samo pravo čitanja.

U našem projektu NetworkVariable tip podataka korišten je u 2 skripte, *GameManager* i *PlayerGameScript.* U skripti *GameManager* NetworkVariable koristi se za sinkronizaciju isključivo cijelih brojeva, a oni predstavljaju sljedeće elemente:

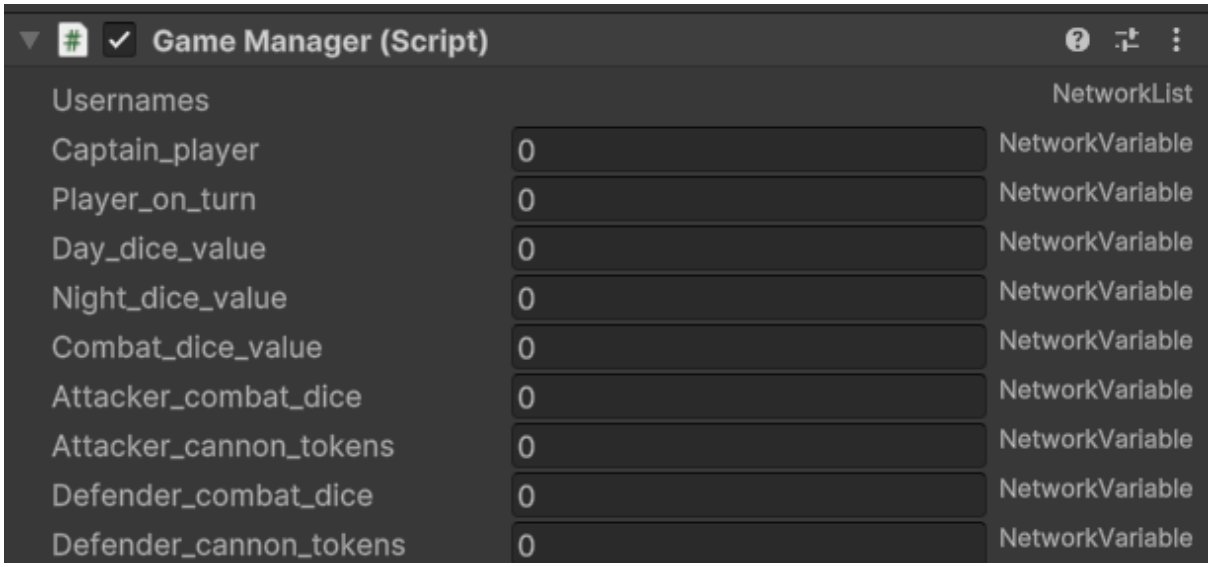
- indeks kapetana trenutnog ciklusa,
- indeks igrača trenutno na potezu,
- vrijednost kockica za akcijske kartice (dan i noć),
- vrijednost kockica za borbu te
- količina topova dodanih od strane oba igrača u borbi.

Što se tiče skripte *PlayerGameScript* u njoj se ovim tipom podataka sinkroniziraju idući elementi:

- indeks igrača (int),
- identifikacijska oznaka polja na kojem se igrač nalazi (int),

- broj bodova (int) te
- ime igrača (FixedString32Bytes).

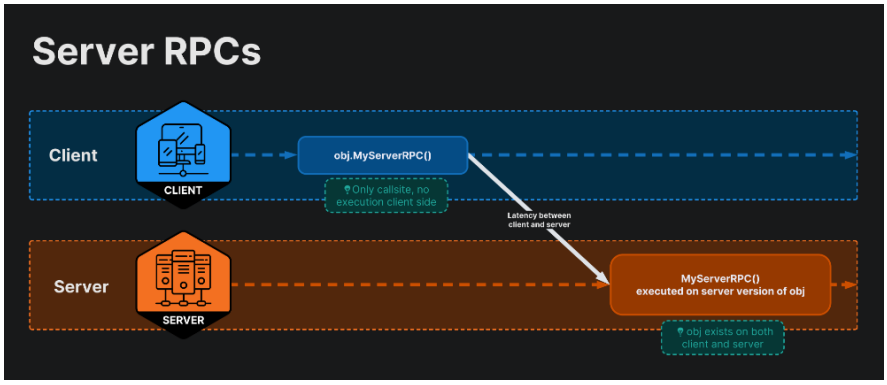
Također, valja napomenuti da postoji i NetworkedList koji funkcionira na sličan način kao i NetworkVariable no umjesto da omotava jedan tip podataka čini to nad cijelom listom. U skripti *GameManager* korišten je taj tip podataka za sinkronizaciju liste igrača unutar igre. Početne vrijednosti ovog tipa podataka moguće je postaviti unutar samog koda ali i unutar Inspector kartice projekta. Primjer Inspector kartice fokusiran na same NetworkVariable vrijednosti može se vidjeti na slici 13.



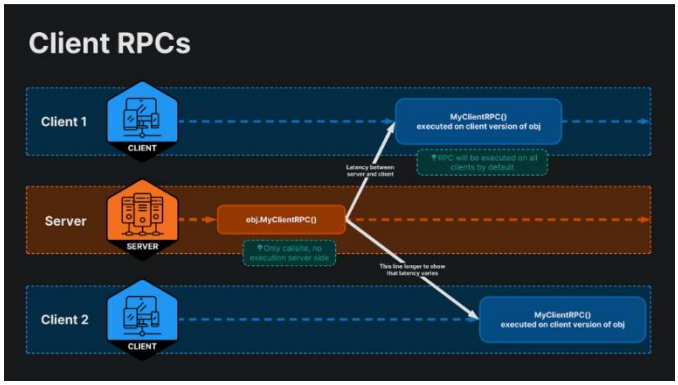
Slika 13: *NetworkVariable podatci skripte GameManager*

RPC

RPC pozivi su česti način usklađivanja stanja između klijenata i servera [13]. Oni služe za poziv metoda koje se ne nalaze na istom uređaju/adresi unutar mreže. Za razliku od NetworkVariable RPC ne održava perzistentnu sinkronizaciju podataka već se podatci usklađuju samo u trenutku samog poziva udaljene metode. Pozive mogu slati i klijenti i server, a mogu biti usmjereni specifično klijentima, serveru ili svima. Postoji više opcija za izvođenje RPC, no nama su bitna samo ova 3. Primjeri pozivanja RPC prema serveru i prema klijentu prikazani su na slikama 14 i 15.



Slika 14: *RPC klijenta prema serveru, preuzeto s [13]*



Slika 15: *RPC servera prema klijentima, preuzeto s [13]*

RPC metode se pišu slično kao i obične metode, no postoje određene razlike. Prije svake metode potrebnu je u uglatim zagradama naznačiti da se radi o RPC metodi. Unutar uglatih zagrada se navodi jeli RPC upućen serveru, klijentima ili svim dionicima mreže. Imena ovih metoda moraju završavati nizom slova “rpc”, a ove metode moguće je pisati isključivo u skriptama koje nasljeđuju ranije spomenutu NetworkBehaviour klasu. Primjer jedne ispravno napisane RPC metode je vidljiv na slici 16.

```
[Rpc(SendTo.Server)]
public void SetDefenderCannonTokensServerRpc(int ammount)
{
    defender_cannon_tokens.Value = ammount;
}
```

Slika 16: *Primjer ispravno definirane RPC metode*

Pozivanje RPC metoda je primarno sredstvo sinkronizacije stanja igre i vrijednosti određenih podataka u našem projektu. Svaka skripta (osim *SquareManager*) koja je u potpoglavlju NetworkVariable navedena u popisu skripti koje nasljeđuju klasu NetworkBehaviour sadrži barem jednu RPC metodu. Primjerice usklađivanje toka igre navedenog u opisu skripte *GameManager* obavljeno je pomoću RPC metode. U toj skripti RPC metode služe se za: dodavanje novih igrača u igru, postavljanje jednakih

vrijednosti kockica (dan/noć) svim igračima, usklađivanje vrijednosti kockica za borbu i korištenih topova u borbama igrača, proglašenje pobjednika borbe i uzimanje “ratnog plijena” te za izračun konačnog broja bodova i prikaz pobjednika.Unutar skripte *PlayerGameScript* RPC metode se koriste za usklađivanje otvaranja potrebnih elemenata korisničkog sučelja, slanje informacije o posjedovanju određenih kartica s blagom kao i postavljanje novih kartica s blagom u posjed nekog igrača(bilo krađom nakon bitke ili dolaskom na pećinu s blagom), ažuriranja broja resursa u spremnicima te mješanje akcijskih karata i njihov odabir na svakom potezu.

Square skripta koristi RPC za postavljanje kartice s blagom na polje koje je po tipu pećina s blagom, ali i za dodavanje i oduzimanje indeksa iz liste igrača koje se nalaze na tom polju. Kretanje brodova koji predstavljaju igrače na ploči je također usklađeno pomoću RPC metoda skripte *PlayerMovement,* a usklađenost kretanja kamere koja prati igrača je također realizirana RPC metodama skripte CameraController.

Ostalo

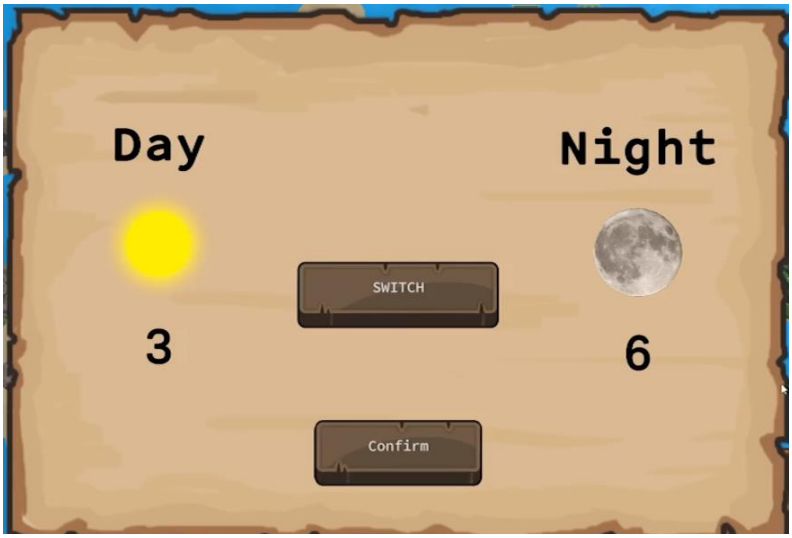
Dosad navedene komponente NGO biblioteke su odgovorne za veliku većinu umrežavanja ove igre te im je zbog toga posvećena iznimna pozornost u pisanju ovog izvještaja. No postoji i još jedan aspekt NGO biblioteke koji je korišten u ovom projektu, a to je serijalizacija. Ona je iskorištena za strukturu akcijskih kartica u ovoj igri. Struktura u sebi sadrži jednu vrijednost, jedinstveni cjelobrojni identifikator akcijske kartice, a serijalizacija strukture omogućena je implementacijom INetworkSerializable sučelja. Serijalizacija se obavlja metodom NetworkSerialize implementiranog sučelja, a konačna struktura se na kraju omotava NetworkVariable tipom.

Konačni proizvod

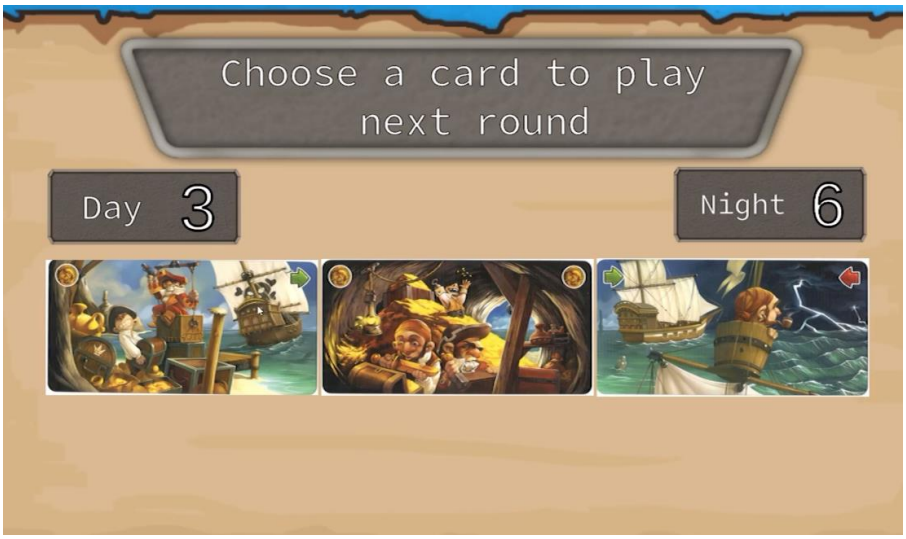
Rezultat rada na ovom projektu je gotova igra koja se može igrati na lokalnoj mreži, a kojoj može istovremeno pristupiti do 4 igrača. Izgrađen je i build igre koji omogućuje pokretanje igre van Unity pogonskog sustava, a igra je izgrađena za Windows OS. Cijeli projekt zajedno s izgrađenom verzijom nalazi se na našem GitHub repozitoriju [16]. Na repozitoriju se uz to nalazi i kratki video koji demonstrira jedan tipični tijek igranja, a u nastavku ovog poglavlja priložene su fotografije koje prikazuju neke detalje konačnog proizvoda.

Slika 17: *Čekanje na igrače u izborniku*

Slika 18: *Izgled ploče i korisničko sučelje spremnika*



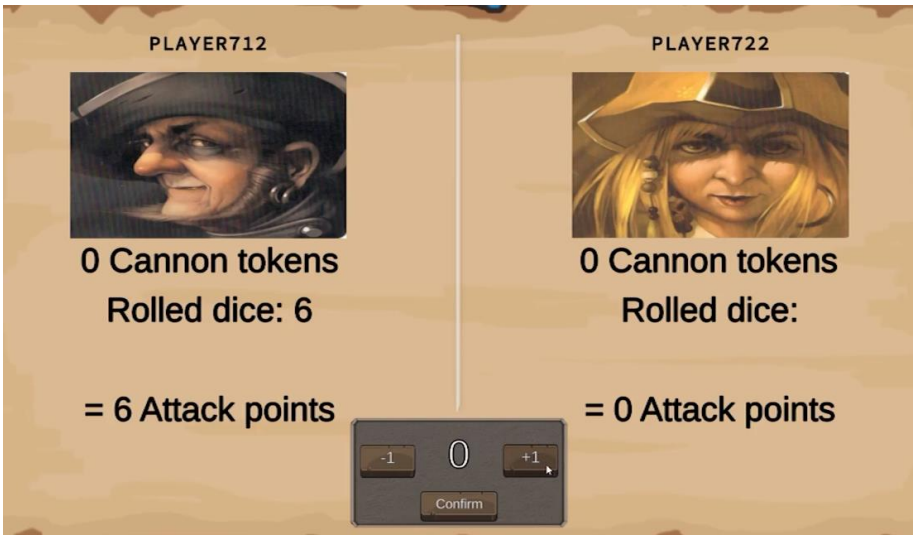
Slika 19*: Korisničko sučelje bacanja kockice*



Slika 20: *Korisničko sučelje odabira akcijske kartice*

Slika 21: *Dodavanje resursa u pune spremnike s prikladnim korisničkim sučeljem*

Slika 22: *Pomicanje unatrag zbog nedovoljnog broja resursa*



Slika 23: *Korisničko sučelje borbe*

Zaključak

Cilj ovog projekta je bio timskim radom doći do funkcionalne umrežene igre. Kao tip igre odabrali smo društvenu igru koja nema komercijalno razvijenu verziju za umreženo igranje putem računala. Naša želja je bila što vjernije rekreirati društvenu igru Jamajka i u tom naumu smo u velikoj mjeri i uspjeli. Jedine ozbiljnije razlike od stvarne igre su one koje ne utječu na iskustvo igranja igre. Raspored otoka i općenito cijelo virtualno okruženje nije baš najvjernije replicirano s obzirom na original. Također u igri ne postoji račvanje puteva kao što postoji u stvarnoj verziji.

No ipak, iako je rezultat ovog projekta funkcionalna umrežena igra koja prilično vjerno rekreira fizičku društvenu igru, postoje i neke stvari na kojima se još da poraditi. Korisničko sučelje je stavka na kojoj je potrebno malo poraditi. Najvažnija stvar koja nedostaje je sekcija s uputama za nove igrače. Na ovaj način igra je prilagođena samo onim igračima koji su već upoznati s Jamajkom i znaju kako je igrati. Ova preinaka bi našu igru približila širem broju potencijalnih igrača. Osim toga korištenjem boljih materijala moguće je dodatno poboljšati izgled nekih dijelova korisničkog sučelja, primjerice glavnog izbornika. Bolji materijali bi mogli biti i iskorišteni u izgradnji virtualnog okruženja unutar kojeg je igra smještena. U ovom projektu korišteni su modeli s malim brojem poligona što je za naše potrebe sasvim dovoljno, no definitivno ostavlja prostora za napredak. Također umjesto statičnih “ploča” vodene površine moguće je i ubaciti potpuno animirano more za bolji igrački doživljaj. Što se tiče umreženih aspekata jedina ozbiljnija preinaka koja bi bila potrebna je mogućnost igranja igračima koji se ne nalaze u istoj lokalnoj mreži.

Literatura

[1] Fabio Duarte, “How many gamers are there?”, 22.1.2025. URL: https://explodingtopics.com/blog/number-of-gamers

[2] Rocket Brush Studio, “Most popular video game genres in 2025: revenue, statistics”, URL: https://rocketbrush.com/blog/most-popular-video-game-genres-in-2024-revenue-statistics-genres-overview

[3] SteamDB, URL: https://steamdb.info/charts/

[4] BoardGameGeek, Jamaica, URL: https://boardgamegeek.com/boardgame/28023/jamaica

[5] BoardGameGeek, Jamaica, URL: https://boardgamegeek.com/image/275442/jamaica

[6] WorldOfBoardGames, Jamaica Rulebook, URL: https://world-of-board-games.com.sg/docs/Jamaica.pdf

[7] Unity, URL: https://unity.com/

[8] Microsoft Visual Studio, URL: https://visualstudio.microsoft.com/

[9] Git, URL: https://git-scm.com/

[10] Github, URL: https://github.com/

[11] Synty Store, POLYGON – Pirate Pack, URL: https://syntystore.com/products/polygon-pirate-pack

[12] Unity Asset Store, Dark Fantasy GUI/UI Kit, URL: https://assetstore.unity.com/packages/2d/gui/dark-fantasy-gui-ui-kit-68828?srsltid=AfmBOorUHu-guqlG6eE4plXCniq1juu67xstrcC2-uIobwTU2UdSMMGj

[13] Netcode for GameObjects, URL: https://docs-multiplayer.unity3d.com/netcode/current/about/

[14] UnityTransport, URL: https://docs.unity3d.com/Packages/com.unity.transport@2.0/manual/index.html

[15] GitHub repozitorij projekta, URL: https://github.com/epreglej/Jamajka