Implementacija mrežne višekorisničke video igre borbe robota u Unreal radnom okviru

Implementation of a network multiplayer robot fighting video game in Unreal engine

Student:
Nikola Ilić
2017/0640

Mentor:
prof. dr Marko Mišić

Sadržaj

Uvod

Unreal Engine je savremeni i svestrani alat za izradu video igara koji korisnicima omogućuje potpuni i nezavisni razvoj video igre u svim domenima i nivoima. Predstavlja snažnu bazu za kreiranje visoko optimizovanih, kompleksnih igara i danas je industrijski standard. Njegova implementacija sadrži podršku za sve standardne zahteve sadašnjih video igara, kao što su 3D i 2D grafičko renderovanje, simulacija fizike, sistem za izradu korisničkog interfejsa, audio sistem, umrežavanje više igrača, registrovanje ulaza od strane igrača i ostale. Unreal Engine je softver otvorenog koda koji se svakodnevno unapređuje, te sadrži i podršku za najnovije tehnologije poput veštačke inteligencije, praćenje zraka svetlosti (ray tracing) kao i voksele (3D alternativa pikselu). Razlog njegove popularnosti i među visokobudžetnim kompanijama i nezavisnim korisnicima jeste upravo javni pristup njegovom kodu, te ga korisnici mogu neograničeno i bez ograničenja prilagoditi svojim potrebama. Okuplja široku zajednicu koja razvija alatke i dodatke, i danas Unreal Engine sa svojim širokim spektrom mogućnosti nije samo softver za programere, već za i sve ostale učesnike u kreiranju jedne igre: dizajnere, skulptore, crtače, arhitekte, animatore, dizajnere zvuka, producente, analitičare.

Ovaj projekat predstavlja istraživanje svih aspekata izrade jedne višekorisničke 3D video igre, sa naglaskom na samo umrežavanje igrača. Tematika video igre je međusobna borba robota na točkovima, što je i popularna disciplina u stvarnosti. Cilj je potpuno funkcionalna igra gde korisnik bira robota i uz jednostavan interfejs se konektuje u sesiju sa drugim igračima i otpočinje borbu. Unreal Engine je izabran kao radni okvir zbog svoje izvanredne podrške za realizaciju 3D igre: jednostavni uvoz Blender 3D modela u igru, podrška za animaciju modela, implementirana fizika za vozila i koliziju, jednostavna izrada terena i mape i ugrađena podrška za umrežavanje igrača. Zahvaljujući Unreal Engine sistemu za umrežavanje, preko daljinski pozivanih procedura (RPCs), repliciranih svojstava i upravljanju sesijama, rezultat je međusobna sinhronizacija između različitih igrača uz minimalno vreme kašnjenja.

Summary

Unreal Engine is modern and versatile tool for developing video games that allows users to fully and independently develop video games in all domains and levels. It represents strong base for creating highly optimized, complex games and today it is industry standard. Its implementation consists of support for all the standard requirements for video games, such as 3D and 2D graphics rendering, physics simulation, system for developing user interfaces, audio system, multiplayer networking, handling inputs and others. Unreal engine is open source software that is continuously evolved, so it contains support for new, cutting edge technologies, such as artificial intelligence, ray tracing, voxels etc. The reason for its popularity among high-budget studios and independent users is the public access to its code, to users can adapt it to their needs indefinitely and without restrictions. It attracts a wide community that develops tools and plugins, and today, Unreal Engine is not only software for game programmers, but also for all other participants in the creation of a game: designers, sculptors, artists, architects, animators, sound designers, producers, analysts.

This project represents the research of all aspects of creating a multiplayer 3D video game, with an emphasis on the networking. The theme of the video game is mutual fight of wheeled robots, which

is also a popular discipline in real world. The goal is a fully functional game where the user chooses a robot to play with and with a simple interface connects to a session with other players and starts to battle. Unreal Engine was chosen as the framework because of its outstanding support for 3D game development: easy import of Blender 3D models, support for animating models, existing physics for vehicles and collisions, easy creation of terrain and maps, and built-it support for player networking. Thanks to the Unreal Engine networking system, via Remote Procedure Calls (RPCs), replicated properties and session management, the result is synchronization between different players with minimal time delay and a stable connection.

Opis problema sa pregledom postojecih resenja

Umrežavanje

U zavisnosti od igre, arhitektura umrežavanja se razlikuje, i na programeru je da koristeći Unreal Engine API prilagodi mrežni sistem svojim potrebama. Korisnički zahtevi vezani za umrežavanje su konkretno: mogućnost klijenta da prvenstveno vrši pretragu postojećih sesija i konektuje se na izabranu, ukoliko nema mogućih sesija kreira novu i preuzima ulogu servera, održava sinhronizaciju u realnom vremenu o svojstvima i akcijama robota. Iako Unreal Engine sadrži arhitekturu umrežavanja, to ne znači da je umrežavanje klijentskih procesa brz i jednostavan način bez dodatnih konfiguracija i algoritama. Procesi nižeg nivoa, kao što su kreiranje i održavanje klijentskih konekcija, slanje paketa, serijalizacija, rutiranje, load-balancing, već si implementirani na nivou radnog okvira, te klijent, odnosno programer, kreira logiku umrežavanja na nivou veće apstrakcije [2]. Rezultat ovog projekta jeste i mrežna arhitektura koja može služiti kao osnova za druge igre sličnih karakteristika.

Modeli

U igri sa malom i jednostavnom mapom, sa konkretnim i očekivanim akcijama i strogo definisanim pravilima, glavni utisak na igrača ostavlja model svog virtuelnog karaktera. Suprotno od estetike modela, potrebno je voditi računa i o njegovoj kompleksnosti. Model robota mora biti optimizovan jer će svaki klijent u svakom trenutku prikazivati i modele svih ostalih igrača u sesiji. Takođe različiti tipovi karaktera moraju biti i dovoljno vizuelno različiti, kako bi igrač jednostavno mogao da prepozna tipove protivnika u žaru borbe. Model mora poštovati i logičke karakteristike karaktera kome pripada, i njihove različite uloge.

Animacije i vizuelni efekti

Zajedno sa modelima, animacije i vizuelni efekti predstavljaju jedan od najupečatljivih detalja u igri. Igrači očekuju realistične animacije u odnosu na zahtevane akcije. Upravljanje vozilom treba se jasno uočiti na samom modelu vozila. Ubrzavanje, kočenje, i biranje smera treba reflektovati na točkove vozila, kao i na samo vozilo po zakonima inercije. Glavna odlika koja razlikuje različite tipova robota jeste njihovo oružje, zbog čega je jako bitno ispravno animirati pokrete aktiviranja oružja. Sve spomenute animacije potrebno je izvršiti u realnom vremenu kako bi korisnik imao potpuni doživljaj kontrole robota. U okruženju neprestane borbe bitno je da animacije aktivacija različitih tipova oružja budu jasno vidljive i razlikujuće kako bi igrač imao šanse da izbegne

konkretne napade. Unreal Engine sadrži dobru podršku za animiranje dinamičkih modela, gde se model deli na sitnije logičke celine koje je moguće zasebno manipulisati.

Pored animacija, još jedan način kako igrači vizuelizuju događaje u igri su specijalni efekti, koji se aktiviraju kao posledica nekog događaja koji je od izuzetnog značaja. U ovom projektu vizuelni efekti su posledica uspešnog napada jednog robota na drugi. Nepravilno rukovanje specijalnim efektima jedan su od najčešćih uzroka problema optimizacije. Smanjenje njihovog broja je danas često rešenje kada želimo da ostvarimo manje iskorišćenje resursa računara.

Fizika vozila

Igrači očekuju da pored ispravnog vizuelnog pokreta točkova i oružja, ti pokreti jasno oslikavaju i fizičku realnost u igri. U tim slučajevima, zapravo je fizička logika ta koja upravlja animacijama. Neki od glavnih izazova pri razvijanju vozila su odnos vozila prema različitom terenu, preprekama, brzini, uglu točkova itd. Zahvaljujući određenim dodacima, u Unreal Engine se većina fizičke logike konfiguriše parametrima. Bitni parametri vozila koje je moguće podesiti su sam oblik fizičkog tela (koje je različito od vizuelnog modela), konfiguracija točkova, trenja, proklizavanja, motora, težište i masa objekta.

Poseban problem predstavlja definisanje oblika fizičkog tela, koje osim što utiče na ostale fizičke pojave poput inercije, ima i ulogu detektovanja kolizije. Detektovanje kolizije predstavlja i srž same definicije igre, jer je krucijalna za ispravno funkcionisanje celokupne igre. U brzoj igri borbe robota, u maloj areni, kolizije se dešavaju često i haotično. Ignorisanjem ili pogrešnim detektovanjem kolizije dolazimo do situacija prolaska robota jedne kroz druge kao i pogrešnim apliciranjem štete od oružja.

Grafički korisnički interfejs

Informacije i stanje igre koje igrači ne mogu zaključiti iz animacija i efekata predstavlja se putem korisničkog interfejsa. U video igrama dva su osnovna tipa korisničkog interfejsa sa drugačijim zahtevima, HUD (Head-up display) i meniji.

HUD predstavlja sve tekstualne i 2D informacije prikazane igraču tokom 3D simulacije igre. Glavni zahtevi su jednostavnost, laka uočljivost i razaznalost podataka i prikaz svih potrebnih informacija bez zatrpavanja ekrana suvišnim. U višekorisničkoj igri često podrazumeva i prikazivanje informacija o ostalim igračima u realnom vremenu. HUD treba da ima sve potrebne podatke, a ujedno i da ne ometa i oduzima fokus igračima od same igre.

Meniji predstavljaju zasebni, nezavisniji deo video igre i prate ih druga pravila i problemi. Osim što služe kao izvor informacija, takođe omogućuju i konkretne akcije. Glavni zahtev je intuitivnost. Igrač treba bez puno razmišljanja i sa lakoćom da dođe do željene akcije. Kod konkretne video igre, meni se prikazuje tik pred ulazak u igru, i služi da korisnik pre konekcije na server izabere željenog karaktera i započne igru. Jedan od glavnih izazova u projektu jeste ispuniti spomenute zahteve za reševanje problema odabira karaktera.

Dizajn i pravila igre

Kako bi zainteresovala i zadržala igrače, video igra mora da ima dovoljno balansirana pravila i parametre. Potreban je cilj, kao i jasne povratne informacije koliko smo mu blizu. Ključna mehanika jeste borba robotima, haotična, brza, nepredvidiva, dinamična. Igrači očekuju fer i balansirano iskustvo, gde svaki tip robota ima podjednake šanse za pobedu, bez očiglednih favorita. Stil igre, umeće i kreativnost trebaju biti jedini parametri koji predviđaju pobedu. Podesivi parametri su brzina, maksimalni životni poeni, manevarske sposobnosti, efikasnost oružja. Vremenom, iskustvom i testiranjem, potrebno je svakodnevno podešavati ove parametre kako bi dostigli balansiranu igru. Takođe je podjednako važno redovno ažurirati igru, ubacivanjem novih tipova robota kao i novih mapa.

Cilj video igre je zabava. To je ujedno konačni izazov pri razvijanju igre, učiniti je zabavnom za ostale igrače. Svi navedeni podnaslovi ponaosob utiču na celokupno iskustvo.

Opis korišćenih tehnologija

Unreal Engine

Unreal Engine je platforma i radni okvir za razvoj video igara napravljena i i dalje razvijana od strane kompanije Epic Games. Prvi put je objavljen 1998. godine kao deo video igre Unreal. Inicijalno je napravljen za igre pucanja oružjem iz prvog lica, a vremenom se proširio i na sve ostale žanrove 3D video igara. Trenutno aktivno izdanje je Unreal Engine 5, koji je objavljen u Aprilu 2022. godine, kao dugo iščekivano izdanje koje je uvelo mnoštvo inovativnih sistema kao i podršku za konzole nove generacije i moderni hardver. Glavne nove tehnologije su Nanite – virtuelni geometrijski sistem koji koristi nove formate za grafičko predstavljanje da prikaže visokodetaljne strukture na scenama sa puno objekata[4], Lumen – dinamični globalni sistem osvetljenja i refleksije koji je dizajniran za moderne kompjuterske sisteme[5], World Partition sistem – podela velikih svetova na mrežu manjih ćelija koje se automatski učitavaju i briši iz memorije po potrebi [6]. Verzija korišćena u ovom projektu je Unreal Engine 5.4, što predstavlja poslednje stabilno izdanje. Izabrao sam Unreal Engine jer predstavlja vrhunac razvoja radnih okvira za razvoj igara kao i svojih savremenih mogućnosti. Sam otvoreni kod omogućuje dublje razumevanje celokupne arhitekture i pruža neograničene mogućnosti daljeg razvijanja. Takođe postoji velika zajednica Unreal Engine entuzijasta i vrlo brzo se mogu naći odgovori na sva potencijalna pitanja i probleme.

C++ i Blueprint sistem

Logika u Unreal Engine se ostvaruje na dva načina. Radni okvir napisan je u programskom jeziku C++, što je ujedno i skriptni jezik u sistemu. To omogućuje visoke performanse, fleksibilnost i potpunu kontrolu nad resursima. Budući da u igri želimo maksimalne perfomanse i najoptimalniju iskorišćenost kompjuterskih resursa, kao i maksimalno precizna izvršavanja, C++ predstavlja idealno rešenje. Uz sve prednosti programskog jezika niskog nivoa, bliskog hardveru, C++ je ujedno i objektno orijentisani jezik sa visokim nivoom apstrakcije. Zahvaljujući toj kombinaciji, C++ omogućuje brz i efikasan, precizan kod, uz poznate, standardizovane i moderne tehnike pisanja koda. Još jedna moderna opcija u današnjoj verziji Unreal Engine jeste uživo kodiranje (Live Coding). To je proces prevođenja i umetanja u izvršni proces novih i izmenjenih dinamičkih biblioteka dok je

Unreal Engine alat pokrenut i izvršava se. Još su u ranijim verzijama postojale slične alatke pod nazivom Hot Reload, međutim uz razne greške, slabe performanse i prljanje drugih zavisnih objekata, nije bio deo standardnog radnog okvira i korisnici su ga izebagavali. Od verzije 4.22 uveden je novi, standardizovani sistem pod nazivom Live Coding koji u velikoj većini slučajeva radi ispravno i znatno skraćuje vreme programiranja, jer nije potrebno ugasiti i ponovo pokrenuti proces, izbegavajući ponovno učitavanje svih zavisnih fajlova, baza podataka i konfiguracija.[6]

Drugi način za definisanje logike u Unreal Engine je Blueprint sistem. Predstavlja visokofuncionalni vizuelni skriptni sistem koji omogućuje i programerima i ne-programerima da definišu logiku izvršavanja brzo, efikasno i fleksibilno, kreirajući grafove izvršavanja. Čak i pored Live Coding procesa, iteriranje, testiranje i eksperimentisanje u Blueprint sistemu je i dalje mnogo brže nego u C++. Mane ovog sistema su performanse: blueprint logika se prevodi na C++ međukod, te dodaje još jedan nivo apstrakcije što utiče na performanse, pogotovo kod visoko kompleksnih sistema sa puno grananja. Što dovodi i do dužeg vremena prevođenja koda. Takođe ima lošu skalabilnost: kako se projekat razvija, Blueprint logika postaje kompleksnija i konfuznija, graf postaje težak za razumevanje, i pronalaženje grešaka i debagovanje postaje sve kompleksnije do trenutka kada je praktično nemoguće.

Za kvalitetan, efikasan, održiv i skalabilan projekat video igre, potrebno je koristiti i C++ i Blueprint sistem, zajedno ih kombinovati i iskoristiti najbolje iz obe opcije. Jednostavno pravilo kojim se vodim jeste da kompleksni logički kod koji utiče direktno na igru pišem u C++, dok ostale jednostavnije stvari koje su podložnije promenama pišem kao Blueprint graf, uglavnom aktiviranje i logika vizuelnih komponenti.

The Unreal Editor

Tehnički, Unreal Engine predstavlja programsku osnovu koju nadograđujemo stvarajući našu video igru. Editor predstavlja alatku za samo nadograđivanje i manipulisanje te baze. Sadrži sve potrebne alatke. Pokreće se zajedno sa igrom i tokom razvoja igre služi kao glavni softver. Krajnjem korisniku, igraču, dostavlja se igra bez editora, kao izvršni fajl, u zavisnosti od platforme. U ovom dokumentu, Unreal Engine kao nadogradivi softver nad kojim je ugrađena naša logika što predstavlja samu igru, i Unreal Engine kao alatka su izjednačene i misli se na obe stvari, jer zajedno čine nerazdvojivi Unreal Engine ekosistem.

Chaos Vehicle Plugin

Chaos je naziv za podrazumevani sistem fizike i razaranja okruženja razvijen od strane Epic Games[7]. Moguće je jednostavno u svoju igru ubaciti i posebni modul za podršku za vozila. On omogućuje korisniku razvijanja realističnih vozila, uz konfiguraciju svi standardnih svojstava za vozila kao u realnosti: amortizera, diferencijala, menjača, motora, kvačila. Moguće je podesiti po svojoj želji i aerodinamiku, krivu ubrzanja, broj obrtaja. Detaljnije, nudi i zasebnu konfiguraciju točkova: širinu i prečnik, masu, trenje, okretnost, kao i ponašanje u slučaju aktiviranja kočnice i ručne kočnice. Veoma je detaljno i realno osmišljen da postoji i opcija dodavanja sistema protiv blokiranja točkova (ABS). Zbog svojih širokih mogućnosti Chaos Vehicle Plugin trenutno nema alternativu pri razvoju vozila u Unreal Engine i predstavlja očigledni izbor.

Niagara VFX sistem

Niagara sistem je osnovni i standardizovani alat za simulaciju vizuelnih efekata unutar Unreal Engine. Sadrži ugrađenu podršku za najkorišćenije efekte poput vatre, dima, eksplozija, varnica, raznih efekata vode. Uz sistem za vizuelno skriptovanje sličan Blueprint sistemu, omogućuje potpunu kontrolu nad najmanjim elementima efekta, česticama. U konkretnom projektu nije bilo potrebe sa puno različitih efekata, glavni efekat predstavlja varničenje pri uspešnom napadu, i to je omogućeno uz malo truda i maksimalne mogućnosti.

Enhanced Input System

Jedan od noviteta Unreal Engine od verzije 5 jeste poboljšano rukovanje ulaza igrača. Za razliku od starog sistema koji je zavisio od fiksno ugrađenog mapiranja ulaza i akcija, novi sistem predstavlja fleksibilnije i više održivo rešenje. Takođe pojednostavljuje mogućnost igrača da kroz igru menja na koji konkretan unos se aktivira koja akcija.

Microsoft Visual Studio

Podrazumevani alat za pisanje koda za Unreal Engine je Microsoft Visual Studio. Moguće je koristiti i druga popularna razvojna okruženja poput JetBrains Rider, ali je potrebno uraditi nekoliko dodatnih koraka. Visual Studio možemo pokrenuti direktno iz editora, dok je Unreal Engine aktivan, a možemo i samostalno pokrenuti program i kao rezultat prevođenja dobiti našu igru u Unreal Engine okruženju. U projektnom folderu generisanom od strane radnog okvira, dobijamo i skripte za generisanje Visual Studia projekta, pa se korišćenje ovog programa samo nameće. Instalacija svih potrebnih modula za ispravno funkcionisanje u Visual Studiu je pravolinijska i svodi se na pokretanje posebnog programa Visual Studio Installer. Pod posebnom kategorijom razvoja igara, jednostavnim štikliranjem možemo uvesti i instalirati sve neophodne komponente. Dok kod ostalih razvojnih okruženja ovaj posao bi morali raditi ručno, često uz mnogo muka dok pokušavamo iz konzolnog izlaza da saznamo zašto se projekat ne prevodi. Visual Studio sadrži i jako bogat izbor različitih ekstrenzija koje znatno olakšavaju svakodnevno programiranje. Uz instaliranje Unreal Engine komponenata dobijamo i potrebne ekstenzije za isticanje sinktakse, automatsko dopunjavanje koda i predloge koda za Unreal Engine specifične ključne reči.

Jedna od najkorisnijih ekstenzija tokom izrade projekta bila mi je VSChromium ekstenzija, koja sadrži kolekciju alatki za izmenu koda i navigaciju kroz kod. Konkretno, koristio sam njegovu pretragu koda, koja je nekoliko puta brža i preciznija od podrazumevane. Dobro skalira sa veličinom projekta, što je u ovom slučaju nekoliko desetina hiljada fajlove (uglavnom Unreal Engine otvoreni kod) i može prikazati rezultate pretrage za manje od 0.1 sekunde za 100,000+ fajlova.[8]

Debagovanje Unreal Engine programa u Visual Studiu nudi raznovrsne opcije i informacije, omogućavajući programeru da brzo identifikuje i ukloni probleme u kodu. Sadrži stalne i uslovne tačke prekida (breakpoints), mogućnost prekida programa u bilo kom trenutku, pristup vrednostima varijabla i memoriji, kao i steku poziva funkcija. Takođe, konzolni izlaz u Visual Studiu povezan je sa Unreal Engine izlazom i u realnom vremenu nam daje informacije, upozorenja i greške. Sve navedene mogućnosti predstavljaju sve što je potrebno za razvoj i održavanje kvalitetnog koda bez grešaka.

Za izradu projekta korišćena je najnovija verzija, Microsoft Visual Studio Enterprise 2022.

Blender

Unreal Engine poseduje i svoje ugrađene alate za kreiranje modela. Ovi alati su danas dosta ograničeni i ne pružaju dovoljnu slobodu i mogućnosti za potrebe većine igara. Uglavnom služe za prototipiranje i izradu jednostavnih statičkih modela. Za kompleksnije, dinamičke modele koji podržavaju animacije, koriste se eksterni programi. U velikim profesionalnim studijima koriste se uglavnom plaćeni softveri poput Maya, 3D Max, ZBrush koji omogućuju visoke performanse i mogućnosti. Blender je program otvorenog koda koji je realni rival mnogim drugim plaćenim rešenjima u svetu 3D modelinga. Za potrebe izrade modela robota korišćen je program Blender 4.1. U njemu su urađeni svi aspekti vizuelne reprezentacije robota: model, tekstura i pripremanje modela za potrebe animacija. Sam model je urađen u stilu malog broja poligona, kao skup tačaka, ivica i površina. Tekstura je urađena u alatu u okviru Blender programa za direktno bojenje modela, koja posle eksportuje teksturu kao 2D sliku. Pripremanje modela za potrebe animacije predstavlja odvajanje logičnih celina modela u posebne skupove zvane kosti (eng. bones) i konfigurisanje njihove međusobne zavisnosti. Ove celine predstavljaju skup poligona koji se mogu zajedno, nezavisno ili zavisno animirati. Uglavnom se menja njihova pozicija u svetu, rotacija, veličina.

Upravljanje izvornim kodom

Radi lakšeg razvoja i praćenja promena i napretka, korišćen je git sistem za upravljanje izvornim kodom. U konkretnom projektu zanemarena je glavna odlika git sistema – kolaboracija više ljudi pri izradi softverskih rešenja, već je git korišćen za logičko razdvajanje različitih perioda u razvoju softvera. Udaljeni repozitorijum na platformi github napravljen je za osiguranje projekta od mogućih havarija na lokalnom računaru. Kao korisnički interfejs za upravljanje git sistemom izabran je program GitKraken, koji predstavlja dobru vizuelizaciju svih stanja na lokalnom i udaljenom direktorijumu i svih mogućih operacija.

Realizacija sistema

Realizacija

Opis rada sistema

Opis

Zaključak

Zaključak

Literatura

https://continuebreak.com/creations/low-poly-ue4-vehicle-project-files/

[2]https://cedric-neukirchen.net/docs/multiplayer-compendium/network-in-unreal

https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-4-documentation

[4] https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine

[5] https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine

[6] https://unrealcommunity.wiki/live-compiling-in-unreal-projects-tp14jcgs

[7] https://unrealcommunity.wiki/glossary-dmdvfflc

[8] https://chromium.github.io/vs-chromium/