Razvoj simulacijske igre "Bouncer: Chapter 1" u Unity okruženju

Koštro, Eugen

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Eugen Koštro

Razvoj simulacijske igre „Bouncer: Chapter 1“ u Unity okruženju

Završni rad

Pula, rujan 2024. godine

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Eugen Koštro

Razvoj simulacijske igre „Bouncer: Chapter 1“ u Unity okruženju

Završni rad

JMBAG: 0303103405

Studijski smjer: Informatika

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Kolegij: Dizajn i programiranje računalnih igara

Mentor: izv. prof. dr. sc. Tihomir Orehovački

Pula, rujan 2024. godine

SADRŽAJ

1. UVOD

Korištenje besplatnih tehnologija, poput Unity alata, uvelike olakšava učenje i prakticiranje razvoja igara na različitim platformama. Ovako veliki alati imaju velike zajednice ljudi sa sličnim idejama, koji su voljni podijeliti svoje znanje i resurse sa ostalima, omogućujući jednostavno korištenje tih istih resursa u vlastitim projektima. Zbog takve velike dostupnosti resursa, u Unity alatu, moguće je razviti igre koje mogu konkurirati igrama razvijenim od strane profesionalnih timova.

Industrija igara širi se velikom brzinom i omogućuje programerima i dizajnerima da uspješno realiziraju svoje ideje. Ovakav rast daje veliki poticaj manjim timovima da kreiraju atraktivne ideje uz pomoć Unity alata, kao i drugih koji su kompatibilni sa Unityem. U ovom završnom radu detaljno će biti prikazan razvoj simulacijske igre „Bouncer: Chapter 1“ u Unity razvojnom okruženju.

Unity alat je odabran za razvoj zbog svoje fleksibilnosti, bogatih resursa i snažne zajednice [1]. Alat sam po sebi pojednostavljuje proces razvoja igara s već integriranim sustavima za upravljanjem animacijama, fizikom i korisničkim sučeljem. Također je za kreiranje vlastitih materijala korišten Aseprite alat koji omogućuje kreiranje 2D slika i animacija koje mogu biti integrirane u Unity razvojno okruženje.

Sama motivacija za razvoj igre „Bouncer: Chapter1“ dolazi iz osobne strasti prema igrama, kreativnosti i bezbroj mogućih načina na koji se jedna igra može razviti. Fascinacija prema dinamici igre, dizajniranja igre, programiranja, umjetnosti i zvuku uvijek je bila prisutna, kako bi se bolje razumio i savladao razvoj igre. Zato je razvoj ove igre omogućio priliku za dublje istraživanje svih aspekata game developmenta koje uključuje kompleksnost dizajniranja likova, kreiranja skripti i optimiziranja igri.

Cilj je prikazati proces planiranja, izradu prototipa igre i implementiranjem ključnih elemenata kao što su NPC-evi, skripte za njihove objekte koji određuju ponašanje istih, kreiranje i postavljanje redoslijeda scena, dizajniranje vlastitih materijala poput pozadine, likova, animacija te razvoj glavnog protagonista i antagonista. Sami cilj igre je kreirati okruženje gdje su odluke igrača važne za korektni tijek igre kako bi igrač uspješno odradio posao za kriminalnu grupu koja vlada podzemljem.

2. INSPIRACIJA

Slične igre koje su inspirirale ovaj projekt su „Contraband Police“ [2] i „Papers Please“ [3]. Obje igrice rade na slični princip kao i igra „Bouncer: Chapter 1“ i koriste glavnu mehaniku za provjeravanje dokumenata od raznih NPC-eva.

„Contraband Police“ igra je u 3D okruženju gdje glavni igrač preuzima ulogu graničnog policajca kojem je zadatak zaustavljati vozila, pregledavati ih i pregledavati dokumente. Ukoliko uspije pronaći neispravne dokumente ili resurse koji nisu dozvoljeni, ima opcije poput zatvaranja NPC-a u pritvor ili uzeti mito. Sličnu mehaniku ima i ovaj projekt, a to je pregledavanje dokumenata i donošenje odluka o propuštanju ili zabrani ulaska NPC-a u klub. Izgled igre prikazan je na slici 1.

SLIKA 1. PRIKAZ "CONTRABAND POLICE " IGRE

„Papers Please“ je igra u 2D okruženju u kojoj glavni igrač također preuzima ulogu graničnog policajca koji mora donositi teške moralne odluke tijekom cijele priče, odnosno mora birati hoće li ili neće dopustiti izbjeglicama, političarima i sumnjivim ljudima ulaz u imaginarnu sovjetsku zemlju, gdje svaki od likova ima duboku priču iza sebe. Inspiraciju koju je dala igra „Papers Please“ je samo 2D okruženje i dizajn, te mehanike provjere dokumenata. Prikaz igre vidljiv je na slici 2.

SLIKA 2. PRIKAZ "PAPERS PLEASE" IGRE

3. RAZVOJNO OKRUŽENJE

Unity je jedno od najpoznatijih i najčešće korištenih razvojnih alata za izradu videoigara, pa čak i animacija koje mogu biti vizualizirane u 2D i 3D okruženju. Razvijen od strane tima Unity Technologies [4], omogućuje velikoj skupini ljudi poput programera, dizajnera i kreatorima sadržaja da stvaraju aplikacije za različite platforme poput računala, konzola i mobilnih uređaja. Zbog svoje rasprostranjenosti jednostavan je i najbolji za početnike, a također i za profesionalce.

Važno je napomenuti da Unity nudi niz karakteristika koje ga čini fleksibilnim i prilagodljivim alatom za razvoj igara [5]:

1. Višestruka platforma: Unity dopušta razvoj i samo distribuciju aplikacija na preko 20 platformi poput Windows, Linux, PlayStation , Xbox [6], Oculus Rift [7], HTC Vive [8] i mnoge druge.
2. Pristupačnost i fleksibilnost: Unity je razvijen tako da bude prilagodljiv svim razinama iskustva. Zajednica nudi mnogi broj materijala, predavanja i foruma za česta pitanja i problema, a veliki dio te zajednice sastoji se od indie developera zbog svog Unity Personal plana i programa.
3. Sustav skripti: Glavni programski jezik koji se koristi je C# i namjena mu je za pisanje skripti za kontrole i interakcije. Integracija sa Visual Studiom [9] omogućuje lagano pisanje i debbugiranje koda, kreiranje različitih komponenti i upravljanje ponašanjem videoigre ili animacije.
4. Engine: Unity koristi fizički engine za jednostavno simuliranje kretanja, kolizija i različitih fizičkih interakcija. Za fizičke interakcije i simulacije koristi se Nvidia PhysX engine [10] koji daje realistično ponašanje objekata u 3D okruženju i sadrži jednostavno dodavanje komponenata kao što su sila, gravitacija, mase i ostalih.
5. Grafički alati: Unity podržava predivnu grafiku kroz alate poput Shadera, Render Pipeline i Post-procesiranje. Shaderi omogućavaju kreiranje vizualnih efekata poput osvjetljenja, sjena i refleksija. Render Pipeline nudi više mogućnosti renderiranja poput URP-a i HDRP-a koji stvaraju visokokvalitetne grafike za sve platforme. Alat za post-procesiranje daje efekte poput zamućivanje pokreta, dubine polja i mnogih drugih vizualnih poboljšanja koje često koriste profesionalni programeri i dizajneri.

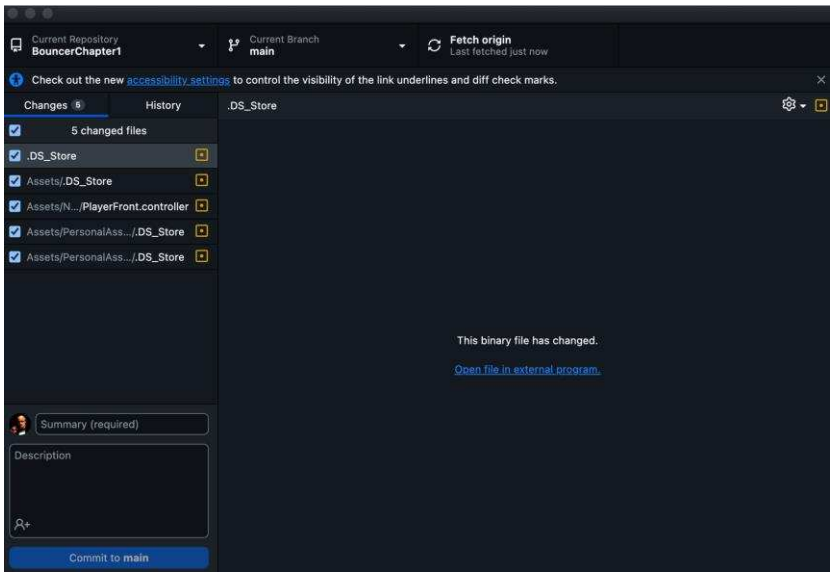
6. 2D i 3D razvoj: za izradu igara u 2D okruženju koristi se alat za spriteove koji omogućuje ubacivanje, uređivanje i animiranje 2D objekata, likova i kompleksnih animacija. Na drugu stranu za 3D okruženje Unity pruža podršku za izradu svih trodimenzionalnih objekata i likova sa mogućnošću upravljanja svjetlom i sjenom.
7. Animator: Alat za animiranje jednostavan je i moćan sustav za upravljanje animacijama i objektima. Animator zato omogućuje jednostavno kreiranje animacijskih stabala i povezivanje animacije u različitim prijelazima.
8. Upravljanje scenama: Kombiniranje više scena bilo je potrebno za lagano povezivanje na desetine scena kako bi se pravilno izvršavalo spremanje, učitavanje i prebacivanje scena. Ovakvo višestruko dijeljenje scena u videoigrama pomaže tijekom razvoja i optimizacije.
9. Unity Asset Store [11]: Online trgovina je mjesto gdje zajednica kupuje, prodaje, pa čak i dijeli besplatno razne resurse poput izgleda likova, zvukova, različitih ikona za korisnička sučelja, objekata, pozadina i terena za 2D i 3D okruženja.
10. Multiplayer podrška: Unity nudi osnovne alate za mrežne videoigre preko Unity Network [12] sustava uz dopuštenje korištenja platformi treće strane za naprednije mogućnosti poput servera za čuvanje stanja igre i drugo.
11. Aseprite: Alat za kreiranje 2D likova, animacija i objekata aktivno se koristi za prebacivanje i uređivanje svega navedenog. Aseprite je također namijenjen i za početnike i za profesionalne dizajnere uključene i van razvojnog svijeta videoigara.

4. PLANIRANJE RAZVOJA PROJEKTA

Za planiranje razvoja videoigre bilo je potrebno odrediti tijek igre. Igra je zamišljena u kasnim 70-ima u talijanskoj četvrti u gradu sličnom New Yorku punom talijanskih mafijaških obitelji, gdje je glavni lik ove priče ostao dužan veliku svotu novaca i prisiljen je raditi na ulazu lokalnog kluba koji služi kao mjesto za pranje novca. U zamjenu za život glavni lik, Toni, 7 večeri mora paziti i provjeravati dokumente raznih posjetitelja te paziti da ti isti imaju korektne informacije na osobnim iskaznicama. Svaka greška skupo košta i ukoliko, Toni, napravi više pogrešaka može ostati taj dan gladan, žedan, pa čak može i „nestati“, sve dok mu za vratom diše član talijanske mafije zvan Paulie.

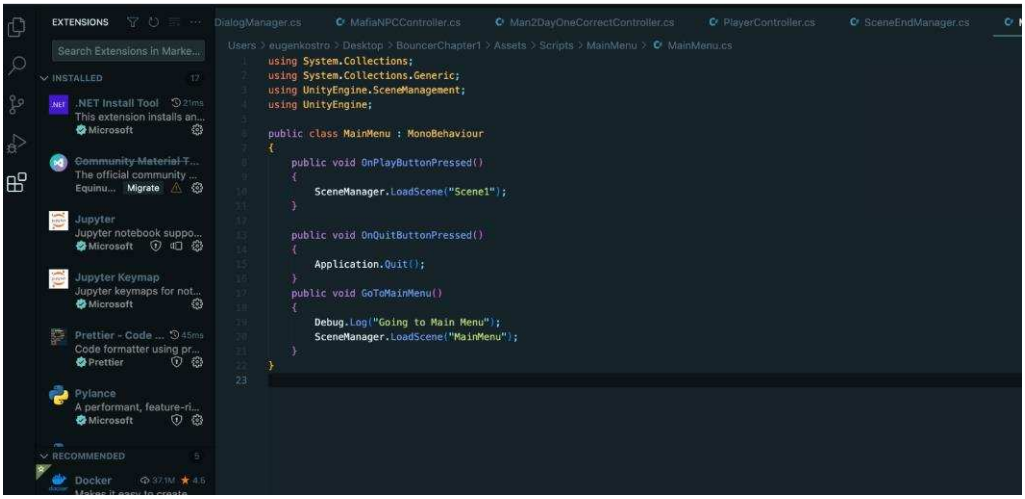
Sljedeći korak je bio određivanje tehnologija i alata koji će se koristiti tijekom razvoja. Uključene tehnologije su bile: Github [13], Github Desktop [14], Visual Studio Code [15], Aseprite [16] i Unity.

Github i Github Desktop omogućili su spremanje igre na server. Github je u ovom projektu korišten minimalno i to samo tijekom kreiranja repozitorija u kojem je igra spremljena. Github Desktop služio je kao spremanje svih promjena nakon završetka jedne sesije razvoja. Također su zapisane promjene na svakoj sesiji, brisanja i dodavanja novih elemenata, objekata ili slika svakim završetkom jedne razvojne sesije. Prikaz GitHub Desktop aplikacije vidljiv je na slici 3.



SLIKA 3. PRIKAZ GITHUB DESKTOP APLIKACIJE

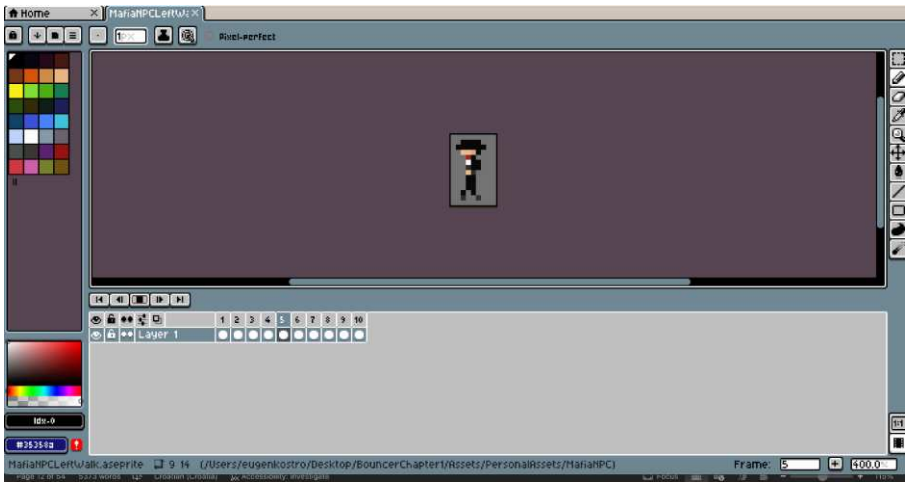
Visual Studio Code korišten je za kreiranje svih skripti koje utječu na ponašanje igrača, likova, dijaloga, funkcioniranja gumbova i spremanja potrebnih objekata za naredne scene. Jezik koji je korišten je C# zbog njegove kompatibilnosti i jednostavnosti u razvoju videoigara. Rad u alatu prikazan je na slici 4.



SLIKA 4. PRIKAZ RADA U VISUAL STUDIO CODE ALATU

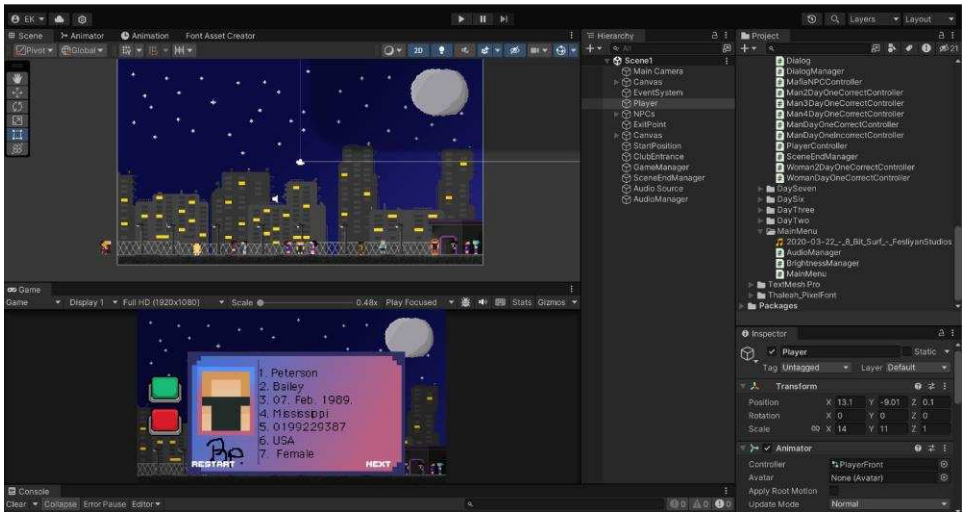
Aseprite alat korišten je za kreiranje svih potrebnih animacija i slika, te su svi materijali ručno rađeni i korišteni u razvoju ove videoigre.

Zbog manjka optimizacije između Asepritea i Unitya, nastaje greška tijekom prebacivanja animacija. U Animation prozoru potrebno je ručno postavljat slike od uvezenih animacija, zato su sve potrebne animacije duplirane, ali ručno sliku po sliku. Primjer rada u Aseprite alatu prikazano je na slici 5.



SLIKA 5. PRIKAZ RADA U ASEPRITE ALATU

Unity alat je bio glavni alat namijenjen za razvoj ove igre. U njemu su korištene mnoge karakteristike i elementi poput hijerarhije, gdje se mogu vidjeti svi objekti za svaku scenu posebno, project prozor, korišten za jednostavan pregled svih materijala i resursa u mapi gdje se nalazi igra i u koju se spremaju sve promjene, animator prozor, koji je korišten za kreiranje animacija za pomjeranje gumbova, dokumenata, likova i ostalih, scene prozor, korišten za postavljanje objekata u sceni na željenu poziciju i animation prozor za snimanje animacija sa povezivanjem slika i postavljanjem početnih pozicija za animacije. Prikaz rada u Unity razvojnom okruženju vidljiv je na sljedećoj slici 6.

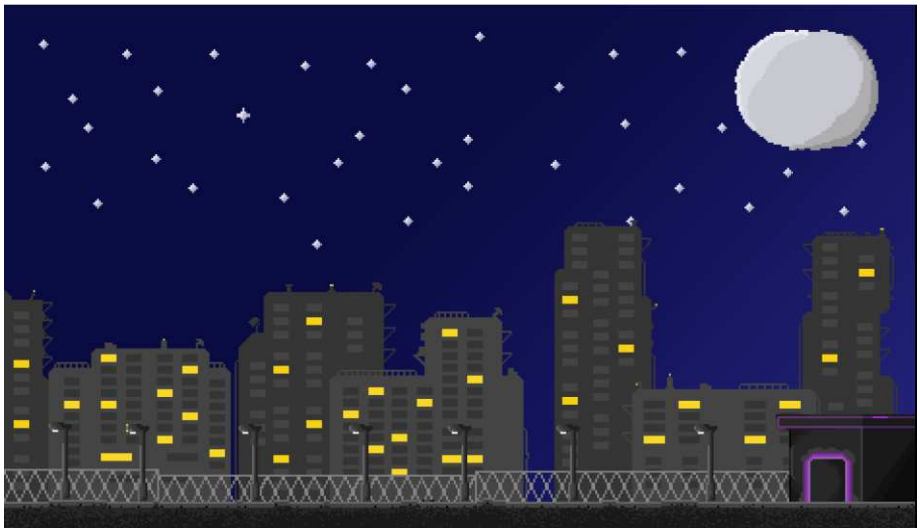


SLIKA 6. PRIKAZ RADA U UNITY ALATU

Za daljnje planiranje razvoja bilo je potrebno odabrati stilska obilježja igre poput rezolucije ekrana za koju će se igra razvijati, stil 2D videoigre, za koju je izabran pixel art, stil likova i rezolucija istih.

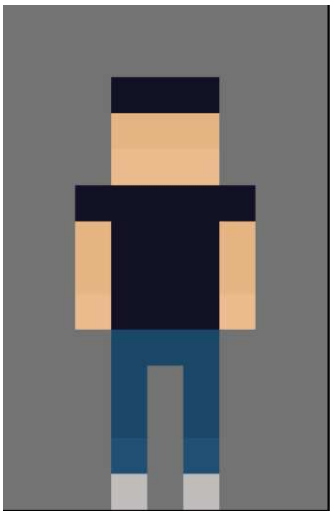
5. PROTOTIP

U prototipu je ideja bila napraviti pozadinu, glavnog lika i NPC-eve. U Unity su postavljeni objekti poput Canvasa na koji je dodana slika pozadine. Pozadina se može vidjeti na slici 7.



SLIKA 7. PRIKAZ POZADINE U PROTOTIPU

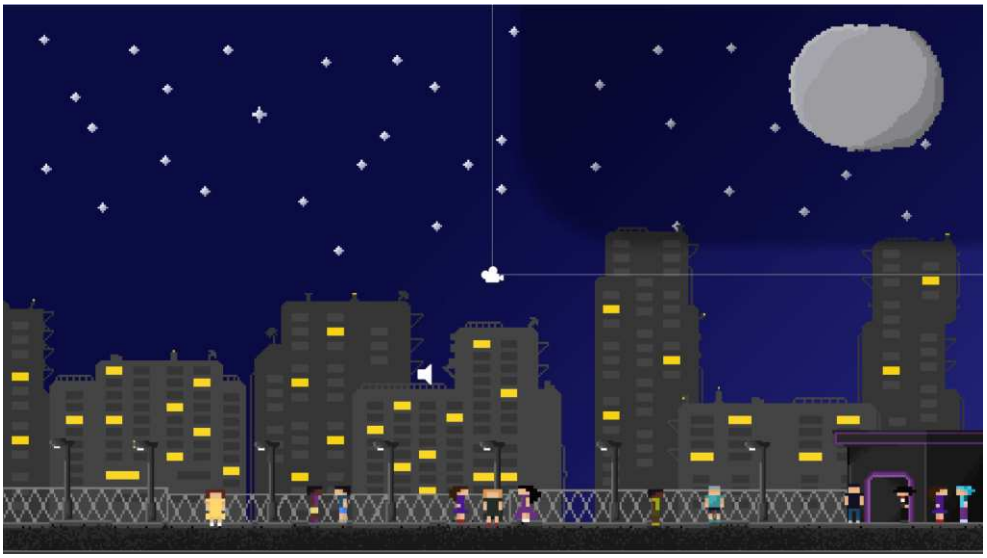
Dodan je i objekt za glavnog igrača koji na sebi ima Sprite Renderer koji sadrži Sprite od glavnog lika. Izgled glavnog lika vidljiv je na slici 8.



SLIKA 8. IZGLED PLAYERA

Nakon toga potrebno je bilo dodati nekoliko likova pod objekt pod nazivom „NPC“. Također kao i glavni lik, odnosno objekt „Player“, likovi imaju svoje Spriteove.

Uz sve navedeno prototip od kojeg je zadržan prvobitni izgled za ostatak igre, a primjer možete vidjeti na slici 9.



SLIKA 9. ZAVRŠNI PRIKAZ PROTOTIPA

6. MEHANIKA IGRE

U ovom dijelu biti će detaljno opisani svi elementi igre kao i svi objekti i skripte nužne za pravilno ponašanje i funkcioniranje videoigre.

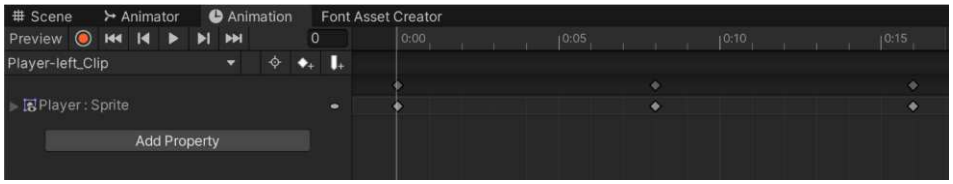
6.1 LIK KOJIM SE UPRAVLJA U IGRI

Igrač sam po sebi kao objekt nema mogućnosti bilo kakvog kretanja već upravljanje dijalogom sa glavnim antagonistom i funkcionalnost interakcija sa dva gumba na koji se dopušta ili odbija ulazak NPC-evima nakon provjere dokumenata.

Izgled igrača sastoji se od 3 animacije i 9 slika sklopljenih u te iste animacije. Nazivi tih animacija su PlayerFront, odnosno prikazivanje igrača sa prednje strane, Player-right, animacija koja se prikazuje samo u slučajevima sa razgovorom sa glavnim antagonistom, odnosno MafiaNPC i Player-left animacija koja se aktivira nakon dijaloga, pa sve do kraja scene. Primjer je vidljiv na slici 10.

SLIKA 10. PRIKAZ PLAYERFRONTIDLE IZGLEDA

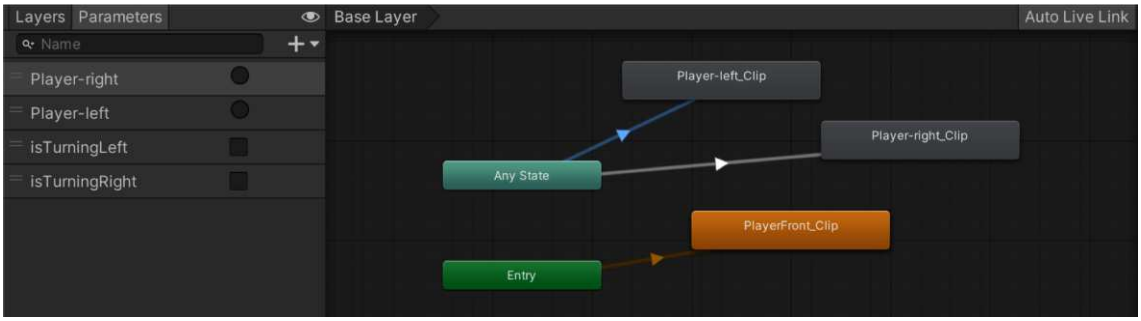
Zbog već navedenih problema sa Aseprite alatom, animacije su rađene ručno u Animation prozoru sa jednakim odvojenim vremenskim razmakom kako bi animacije izgledale ispravno. Navedeni primjer je prikazan na slici 11.



SLIKA 11. ANIMATION PROZOR

Kao i za Player-left_Clip animaciju, koristi se jednaki vremenski razmak kao i za ostale dvije animacije PlayerFront_Clip i Player-right_Clip, samo sa različitim slikama odnosno spriteovima.

Sve te animacije ne bi bile korisne bez pravilnog povezivanja jednih sa drugima i dodavanjem korektnih parametara na veze između istih. Animator PlayerFront_Clip ima vezu sa Entry stanjem, a veza između njih je iz Entry stanja u PlayerFront_Clip. Any State stanje povezano je prema Player-left_Clip i Player-right_Clip animatorima. Između Any State stanja i Player-left_Clip sadrži parametar isTurningLeft bool postavljen na right, isto vrijedi i za Player-right_Clip samo što je na vezi parametar isTurningRight bool postavljen na true. Izgled Player animatora vidljiv je na slici 12.



SLIKA 12. PLAYER ANIMATOR PROZOR

Na Player objekt je također dodana skripta PlayerController, koja služi za ponašanje objekta u videoigri tijekom različitih interakcija sa drugim objektima tako da mijenja stanja po potrebi.

Klasa PlayerController sadrži jednostavan kod koji dohvaća animacije iz Animator prozora, odnosno animatora (linija 11) u funkciji Start(). Uz Start() funkciju imamo i dvije glavne funkcije pod nazivom TurnLeft() i TurnRight(). U TurnLeft() se parametri, prethodno navedeni, postavljaju na true i false, odnosno bool parametar isTurningLeft se postavlja na true, a isTurningRight na false. Ista situacija je i kod funkcije TurnRight() sa obrnutom logikom.

6.2 KREIRANJE VLASTITIH MATERIJALA

Materijali su, kao što je već napomenuto, kreirani u Aseprite alatu u stilu 2D pixel arta kako bi odgovarali 2D okruženju u kojem je videoigra zamišljena. U Aseprite alatu kreirane su sve animacije i svi spriteovi korišteni u ovoj igri, pa čak i dodatne animacije koje nisu bile korištene jer nije bilo potrebe za postavljanjem istih.

6.2.1 NPC

NPC materijali ne koriste nikakvu vrstu slojeva pošto nisu kompleksni i rezolucije su 9 * 14 piksela kako bi omjer između likova i pozadine bio uredan, odnosno koliko toliko realne veličine kao i u stvarnom svijetu.

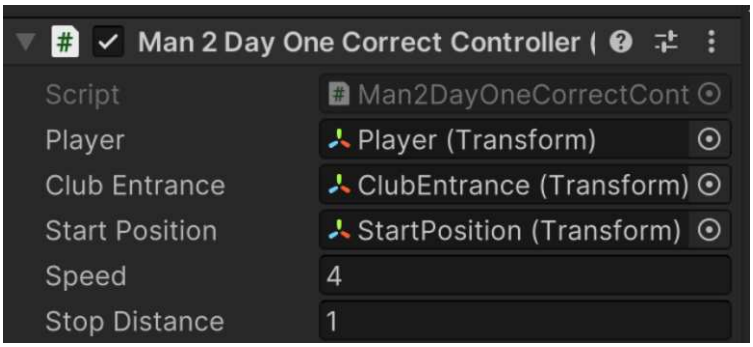
Player objekt, kao i NPC objekti, sadrže svaki posebno 5 animacija kao što su: FrontIdle, LeftIdle, RightIdle, LeftWalk i RightWalk. Svaki objekt, individualno, sadrži 31 sliku, od kojih hodajuće animacije imaju po 10 slika, a sve ostale po 3 slike. Svaka slika crtana je ručno i kombinirana da se sklopi u potrebne animacije. Primjer animacije vidljiv je na slici 13.

SLIKA 13. PRIMJER LEFTWALK ANIMACIJE

Zbog problema sa kompatibilnošću između Unity i Aseprite alata potrebne animacije bile su duplicirane i rađene posebno u Animation prozoru sa određenim vremenskim

razmakom kako bi animacija ostala primjerena brzini hodanja od početne točke, odnosno StartPosition objekta i ClubEntrance objekta.

Za svakog NPC objekta postoje reference u skripti, kasnije objašnjenoj, koje dopuštaju postavljanje ovih objekata na NPC objekt kako bi se ti NPC-evi pravilno ponašali. Primjer referenci za NPC-eve prikazan je na slici 14.

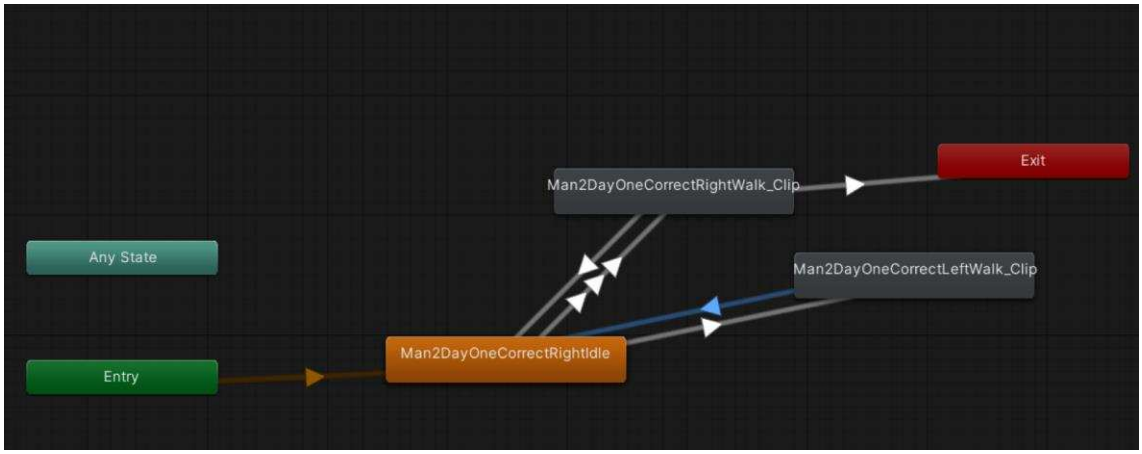


SLIKA 14. PRIKAZ REFERENCI KOD NPC OBJEKTA

Svaki NPC objekt u sebi sadrži kontroler, odnosno animator, u kojem su sve animacije postavljene kako bi odgovarale ponašanju tijeka igre. U animatorima se nalaze i parametri od kojih su: hasReachedPlayer tipa trigger, TurnBack tipa trigger, hasReachedStart tipa trigger, hasReachedClub tipa trigger, MoveToPlayer tipa trigger i isMovingToPlayer tipa bool.

Svaki animator je napravljen isto i jednako funkcionira na djelovanje NPC-a. Entry stanje povezano je sa RightIdle animacijom bez parametara pošto je to početno stanje u kojem se nalaze NPC-evi prilikom pokretanja videoigre. Veza između RightIdle i RightWalk stanja sadrži dva parametra isMovingToPlayer tipa bool, postavljen na true i MoveToClub tipa trigger. Ta veza je kreirana da bi NPC „hodao“ do Playera i od Playera do „ulaza u klub“ kako bi kretanje bilo postavljeno po tijeku videoigre. Također je tu i obrnuta veza odnosno iz RightWalk u RightIdle stanje koja sadrži parametar hasReachedPlayer tipa trigger, kako bi se NPC zaustavio ispred Playera nakon kojeg se pokreće animacija prikazivanja NPC dokumenta i dva gumba koja su dodatno objašnjena. Za RightWalk stanje postoji i veza sa Exit stanjem koje se pokreće nakon što se NPC-u „odobri ulaz“ u klub. Ta veza ima parametar hasReachedClub tipa trigger. Zadnje dvije veze su između RightIdle i LeftWalk stanja, odnosno iz LeftWalk u RightIdle stanja. Prva veza iz RightIdle u LeftWalk stanja sadrži parametar TurnBack tipa trigger ukoliko Player „odbije“ NPC-a, te se NPC vrati nazad na StartPosition. Zadnja veza iz LeftWalk u

RightIdle stanje sadrži parametar hasReachedStart tipa trigger kada se NPC vrati na početno stanje, odnosno StartPosition, u RightIdle stanje. Na slici 15 vidljiv je animator za NPC-eve.



SLIKA 15. PRIMJER ANIMATORA ZA NPC-EVE

Važno je napomenuti da svaki NPC animator controller sadrži svoje posebne animacije na isti princip zato što je ponašanje svakog NPC-a jednako kako bi se pravilno održao tijek videoigre.

Od svih NPC-eva jedan je poseban i on je glavni protagonist igre imenom Paulie. Razlika između njega i ostalih NPC-eva je u tome što on sadrži dijalog koji vodi sa Playerom i svaka nova scena sadrži novi dijalog i time prati priču i odnos sa Playerom. Na slici 16 vidljiv je prikaz dijaloga.

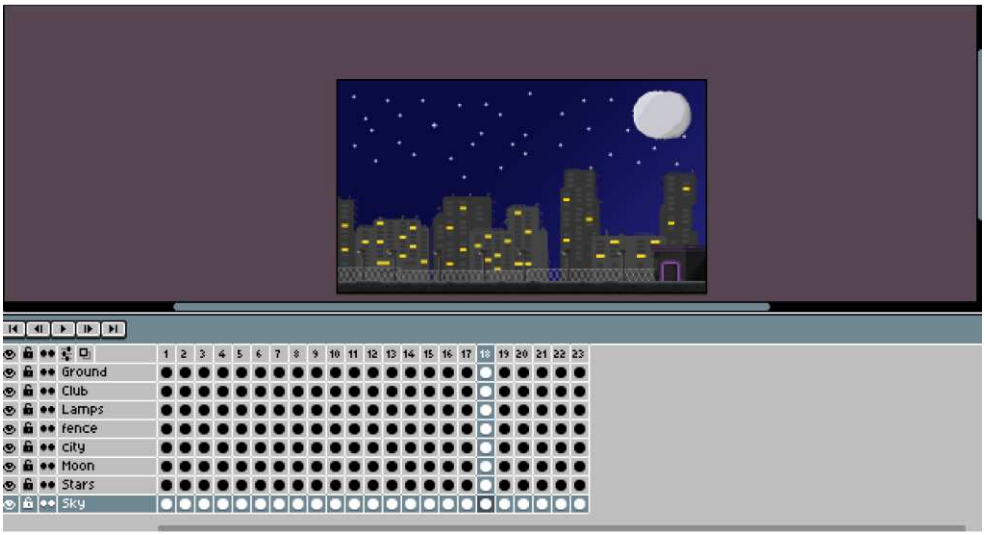
SLIKA 16. PRIKAZ DIJALOGA IZMEĐU MAFIANPC-A I PLAYERA

Animacije su iste kod MafiaNPC-a kao i kod drugih NPC-eva. Što se tiče animatora, MafiaNPC je drugačiji i sadrži različite parametre koji su: MafiaNPCLeftWalk tipa bool, MafiaNPCRightWalk tipa bool, MafiaNPCLeftIdle tipa trigger i MafiaNPCRightIdle tipa trigger.

Entry stanje povezano je sa LeftWalk stanjem i ne sadrži nikakve parametre pošto se to stanje pokreće prilikom pokretanja videoigre. LeftWalk i LeftIdle stanje sadrže dvije veze iz jedne u drugu. Veza između LeftWalk i LeftIdle sadrži MafiaNPCLeftWalk parametar i postavljen je na false jer se pokreće kada MafiaNPC dođe do Playera. Veza iz LeftIdle u LeftWalk sadrži parametar MafiaNPCLeftWalk postavljen na true. Veza između Any State i RightWalk stanja sadržava parametar MafiaNPCRightWalk postavljen na true, koji se pali kada dijalog završi. RightWalk stanje sadrži vezu sa RightIdle na kojem je parametar MafiaNPCRightWalk postavljen na true kada se dijalog završi. Zadnja veza je iz RightIdle u RightWalk gdje je parametar MafiaNPCRightWalk postavljen na true kada MafiaNPC dođe do ExitPointa gdje nestane.

6.2.2 POZADINSKE ANIMACIJE

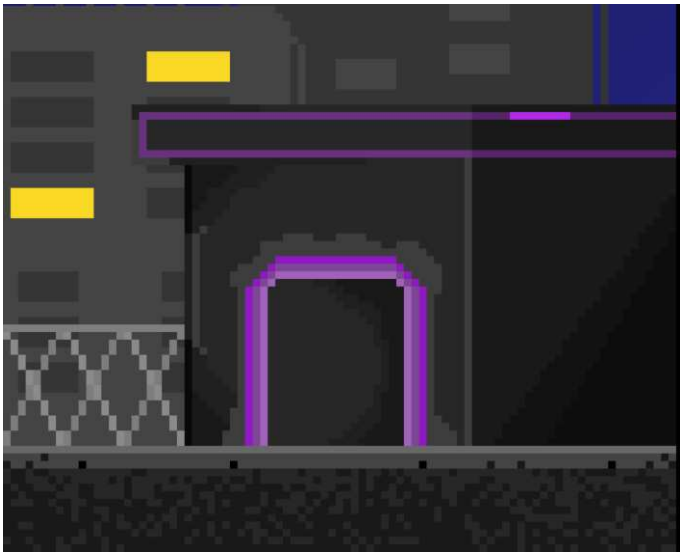
U samom prototipu igre kreirana je pozadina koja je korištena tijekom cijelog razvoja videoigre. Pozadina se sastoji od više slojeva koji uključuju: Ground, Club, Lamps, fence, city, Moon, Stars, Sky. Razlog iz kojeg je pozadina razvrstana u toliko slojeva je radi lakšeg i jednostavnijeg razvoja, nešto slično metodi „Podijeli pa vladaj“. Prikaz slojeva vidi se na slici 17.



SLIKA 17. PRIKAZ SLOJEVA ZA POZADINU

Cijela pozadina, uključujući i animacije, sastoji se od 184 slike koje tvore cjelokupnu animiranu pozadinu rezolucije 470 * 270 piksela.

Samo određeni slojevi pozadine su animirani, odnosno samo Club sloj kako bi se dao doživljaj kako je unutar kluba velika zabava. Na vrata ulaza postavljeno je svjetlo koje se pojavljuje svaku drugu sliku i na vrhu kluba se prikazuje LED svijetlo koje se vrti u krug. Za animaciju LED svijetla na svakoj slici je pomaknut mali dio piksela u svjetlijoj boji da se dočara doživljaj. Prikaz ulaza u klub vidljiv je na slici 18.



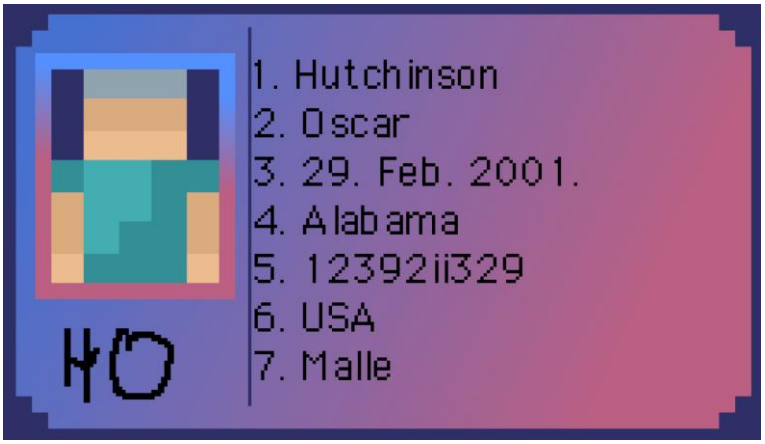
SLIKA 18. PRIKAZ ULAZA U KLUB

6.2.3 NPC DOKUMENTI

Svaki NPCId objekt sadrži svoj sprite, odnosno svoju sliku na kojoj se nalaze sve informacije za igrača kako bi odlučio može li dopustiti ulaz NPC-u u klub ili ga odbiti, ovisno o točnosti informacija.

Svaki osobni dokument sadrži osnovne informacije kao što su ime, prezime, datum rođenja, savezna država, broj dokumenta, država, rod, potpis i slika. Na početku igre, u dijalogu sa MafiaNPC-em, Player dobije instrukcije o tome kako provjeravati dokumente. Pravila su sljedeća, po redu prvo ide prezime, zatim ime, potom ide datum rođenja u formatu DD/Mon./YYYY, naziv savezne države u Sjedinjenim Američkim Državama, broj dokumenta koji sadrži samo deset brojeva, naziv države, odnosno USA, rod, potipis i slika koja mora odgovarati izgledu NPC-a. Na slici 19 prikazan je ispravan primjer dokumenta, a na slici 20 primjer pogrešnog dokumenta.

SLIKA 19. PRIMJER ISPRAVNOG DOKUMENTA



SLIKA 20. PRIMJER ISPRAVNOG DOKUMENTA

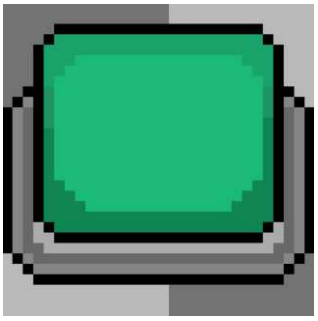
Svaki dokument sadrži svoje animacije i svoj animator. Postoje dvije animacije u kojima je dokument u Idle stanju i kada se dokument podigne na sredinu ekrana, odnosno Rise animacija. U animatoru postoji samo jedan parametar, odnosno ShowId tipa trigger koji se pokrene kada NPC dođe do Playera i on se nalazi na vezi sa Idle u Rise animaciji. Primjer dokumenta prikazanog u igri vidljiv je na slici 21.

SLIKA 21. PRIMJER DOKUMENTA U IGRI

6.2.4 INTERAKTIVNI GUMBI

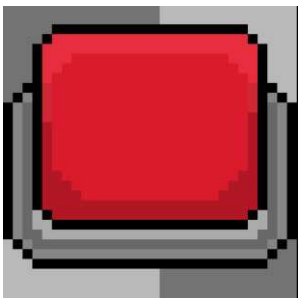
U videoigri nalazi se sedam gumbova od kojih su pet ključni za tijek igre i dva u glavnom izborniku. Dva glavna gumba koji imaju najviše utjecaja na igru su AllowEntranceButton i BanButton.

AllowEntranceButton služi za dopuštanje ulaza NPC-u u klub ako su mu dokumenti ispravni ili ako je igrač propustio grešku na dokumentu. Ukoliko se stisne na AllowEntranceButton, dodat će se dobra ili loša odluka ovisno o ispravnosti dokumenta. Primjer AllowEntranceButton gumba vidljiv je na slici 22.



SLIKA 22. PRIKAZ ALLOWENTRANCEBUTTON GUMBA

BanButton radi na isti princip kao i AllowEntranceButton, jedina razlika je u tome što se dogodi kada igrač pritisne BanButton, NPC-u je zabranjen ulaz u klub i tu se pokreće animacija hodanja koja NPC-a vodi van scene na početnu poziciju, odnosno StartPosition. Izgled BanButton gumba nalazi se na slici 23.



SLIKA 23. PRIKAZ BANBUTTON GUMBA

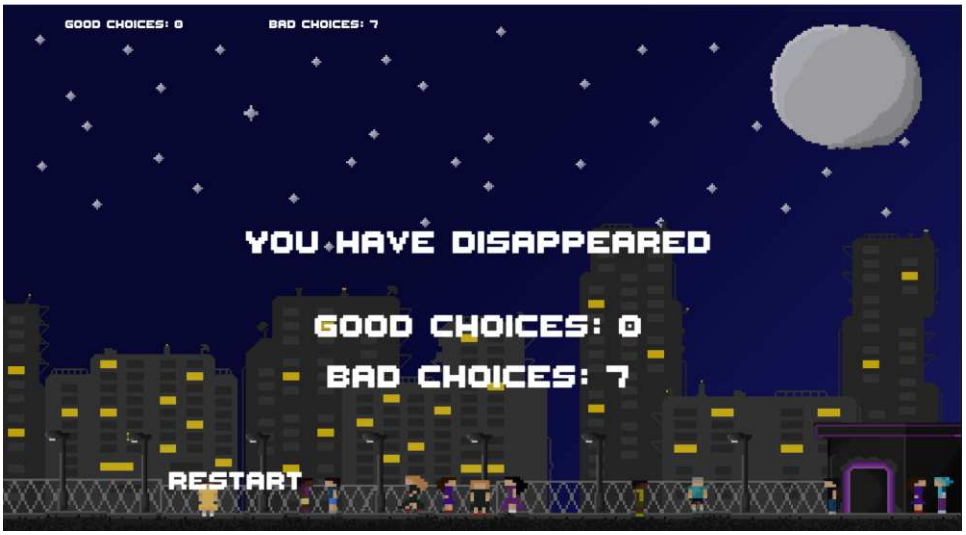
Ova dva gumba također sadrže svoje animatore sa vlastitim parametrima koji su: ShowButton tipa trigger i ButtonPressed tipa trigger. ShowButton parametar nalazi se na vezi Idle prema Rise stanju. Taj parametar omogućuje da se gumb podigne na ekranu kada NPC dođe do Playera i radi na isti princip kao i prikazivanje dokumenta. Još jedna veza koja sadrži parametar ButtonPressed je iz Rise u Clip te taj parametar omogućuje pokretanje animacije kada se gumb pritisne. Jedina razlika između ova dva gumba jest u različitim nazivima.

Sljedeća tri gumba su NextButton, RestartButton i MainMenuButton. Ova tri gumba prikazuju se samo na kraju scena, odnosno RestartButton i NextButton pojavljuju se na kraju svake scene osim u zadnjoj sceni gdje se NextButton mijenja sa MainMenuButton pošto je ta scena posljednja i tu je kraj videoigre.

Gumb NextButton omogućuje Playeru da, nakon uspješno završene scene, ima mogućnost prelaska na sljedeću scenu i nastavi tijek priče. Ukoliko je igrač neuspješno završio scenu, odnosno ako je donio previše loših odluka, gumb će biti nedostupan te je jedina opcija RestartButton gumb.

RestartButton gumb omogućuje igraču ponovni pokušaj prelaska određene scene ukoliko nije bio zadovoljan svojim rezultatom ili ako je donio previše loših odluka. Na slici 24 prikazan je dobar ishod na kraju scene, a na slici 25 primjer sa lošim ishodom.

SLIKA 24. PRIMJER DOBROG ISHODA



SLIKA 25. PRIMJER LOŠEG ISHODA

Pri završetku videoigre kada se prikaže MainMenuButton igrač ima mogućnost vratiti se na glavni izbornik gdje postoje još dva gumba PlayButton i QuitButton. PlayButton daje mogućnost ponovnog igranja cijele priče ukoliko nije bio zadovoljan prošlim prelaskom, dok QuitButton omogućuje izlazak i prestanak prelaska videoigre. Izgled glavnog izbornika prikazan je pronaći na slici 26.

SLIKA 26. IZGLED GLAVNOG IZBORNIKA

7. IMPLEMENTACIJA

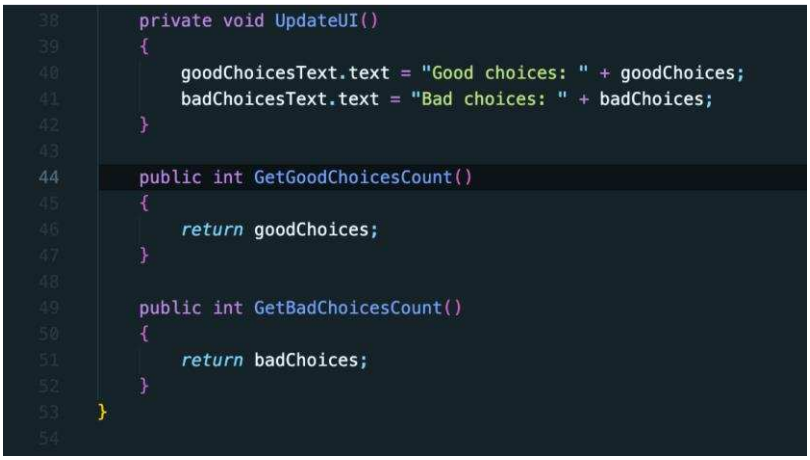
Većina skripti su iste kada su u slučaju NPC-evi pošto svi imaju isto ponašanje osim prvog NPC-a i posljednjeg u svakoj sceni, dok su ostale skripte jedinstvene, odnosno ne koriste se više puta pod različitim nazivom.

7.1 CHOICE MANAGER SKRIPTA

ChoiceManager skripta omogućuje igraču praćenje i bilježenje svih njegovih dobrih i loših odluka. U Awake() metodi postavljeno je to da se jedna instanca koristi tijekom cijele igre, a ukoliko već postoji jedan, uništava ga. Za praćenje izbora kreirane su dvije varijable goodChoices i badChoices koje se zasebno nalaze u metodama IncrementGoodChoices() i IncrementBadChoices(), a za dohvaćanje rezultata koriste se GetGoodChoicesCount() i GetbadChoicesCount() metode koje ažuriraju odluke u bilo kojem trenutku. ChoiceManager skripta povezana je na GameManager objekt u kojem se nalaze reference na Good Choices Text i Bad Choices Text. Metoda Awake(), IncrementGoodChoices() i IncrementBadChoices() prikazane su na slici 27, a UpdateUI(), GetGoodChoicesCount() i GetBadChoicesCount() na slici 28.

```
using UnityEngine;
using TMPro;

public class ChoiceManager : MonoBehaviour
{
    public static ChoiceManager Instance;

    public TextMeshProUGUI goodChoicesText;
    public TextMeshProUGUI badChoicesText;

    private int goodChoices = 0;
    private int badChoices = 0;

    void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
            Destroy(gameObject);
        }
    }

    public void IncrementGoodChoices()
    {
        goodChoices++;
        UpdateUI();
    }

    public void IncrementBadChoices()
    {
        badChoices++;
        UpdateUI();
    }
```

SLIKA 27. PRVI DIO CHOICEMANAGER KODA



```
    private void UpdateUI()
    {
        goodChoicesText.text = "Good choices: " + goodChoices;
        badChoicesText.text = "Bad choices: " + badChoices;
    }

    public int GetGoodChoicesCount()
    {
        return goodChoices;
    }

    public int GetBadChoicesCount()
    {
        return badChoices;
    }
}
```

SLIKA 28. DRUGI DIO CHOICEMANAGER. KODA

7.2 DIALOG SKRIPTA

Dialog skripta koristi se za pohranjivanje dijaloga u igri između Playera i MafiaNPC-a. Ova klasa, preko ScriptableObjecta, omogućuje definiranje dijaloga koji služi za pohranjivanje dijaloga neovisno o sceni. U skripti se nalaze dva polja speaker i sentence,

gdje sentence koristi TextArea za unošenje većih rečenica. Dialog skripta vidljiva je na slici 29.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
[CreateAssetMenu(fileName = "New Dialog", menuName = "Dialog")]
public class Dialog : ScriptableObject
{
    [System.Serializable]
    public struct DialogLine
    {
        public string speaker;
        [TextArea(3, 10)]
        public string sentence;
    }

    public DialogLine[] dialogLines;
}
```

SLIKA 29. DIALOG KOD

7.3 DIALOG MANAGER SKRIPTA

DialogManager skripta služi za upravljanje dijalozima između Playera i MafiaNPC-a na početku scena. U skriptu je dodan TextMeshProUGUI element kako bi se na ekranu prikazivao tekst.

StartDialog() metoda omogućuje pokretanje dijaloga između Playera i MafiaNPC-a i provjerava postoji li i je li MafiaNPC aktivan. Ako je sve u redu, prijašnje linije dijaloga se brišu i puni se nova linija iz Dialog objekta. DisplayNextSentence() je metoda koja prikazuje sljedeću rečenicu iz reda, a ako su se sve linije dijaloga prikazale onda se poziva metoda EndDialog(). Dio koda sa navedenim metodama prikazan je na slici 30.

```
public void StartDialog(Dialog dialog, MafiaNPCController npc)
{
    if (npc == null || !npc.gameObject.activeInHierarchy)
    {
        Debug.LogError("MafiaNPCController nije pronađen ili je deaktiviran!");
        return;
    }

    dialogLines.Clear();

    foreach (var line in dialog.dialogLines)
    {
        dialogLines.Enqueue(line);
    }

    DisplayNextSentence(npc);
}

public void DisplayNextSentence(MafiaNPCController npc)
{
    if (dialogLines.Count == 0)
    {
        EndDialog(npc);
        return;
    }

    var dialogLine = dialogLines.Dequeue();
    dialogText.text = dialogLine.sentence;
    Debug.Log($"Displaying sentence: {dialogLine.speaker}: {dialogLine.sentence}");
}
```

SLIKA 30. PRVI DIO DIALOGMANAGER KODA

EndDialog() metoda završava dijalog sa MafiaNPC-om gdje se taj isti krene vraćati nazad, odnosno pokreće se njegova akcija, a tekst se sakriva. Nakon izvršenja pokreće se sljedeći NPC i Playerova animacija se mijenja. Update() metoda provjerava je li tipka Space pritisnuta, a ako je pokreće se DisplayNextSentence() kako bi se prikazala sljedeća linija dijaloga, a taj dio koda vidljiv je na slici 31.

```
public void EndDialog(MafiaNPCController npc)
{
    if (npc == null || !npc.gameObject.activeInHierarchy)
    {
        Debug.LogError("MafiaNPCController nije pronađen ili je deaktiviran!");
        return;
    }

    npc.EndDialog();

    dialogText.gameObject.SetActive(false);

    PlayerController playerController = FindObjectOfType<PlayerController>();
    if (playerController != null)
    {
        playerController.TurnLeft();
    }

    if (man2NPC != null)
    {
        man2NPC.ActivateNPC();
    }
}

void Update()
{
    MafiaNPCController npc = FindObjectOfType<MafiaNPCController>();
    if (Input.GetKeyDown(KeyCode.Space))
    {
        DisplayNextSentence(npc);
    }
}
```

SLIKA 31. DRUGI DIO DIALOGMANAGER KODA

7.4 MAFIANPC CONTROLLER SKRIPTA

Ova skripta namijenjena je za dodavanje svih potrebnih referenci kako bi se MafiaNPC ponašao prema tijeku videoigre uz sve metode unutar te iste skripte. Metoda MoveToExitPoint() pomiče NPC-a prema izlaznoj točki, odnosno exitPoint. Kada MafiaNPC dođe dovoljno blizu izlaza, prestaje se kretati i animacija hodanja prestaje sa radom. MoveToPlayer() metoda pomiče MafiaNPC-a dok ne dođe dovoljno blizu Playera i pokreće Idle animaciju te pokreće dijalog sa Playerom i pokreće DialogManager. Kod sa tim metodama prikazan je na slici 32.

```
void MoveToExitPoint()
{
    Vector2 currentPosition = new Vector2(transform.position.x, initialYPosition);
    Vector2 targetPosition = new Vector2(exitPoint.position.x, initialYPosition);

    float distanceToExit = Vector2.Distance(currentPosition, targetPosition);

    if (distanceToExit > 0.1f)
    {
        transform.position = new Vector3(
            Mathf.MoveTowards(transform.position.x, exitPoint.position.x, speed * Time.deltaTime),
            initialYPosition,
            0);

        animator.SetBool("MafiaNPCLeftWalk", true);
    }
    else
    {
        hasExitedBuilding = true;
        animator.SetBool("MafiaNPCLeftWalk", false);
    }
}

void MoveToPlayer()
{
    Vector2 currentPosition = new Vector2(transform.position.x, initialYPosition);
    Vector2 playerPosition = new Vector2(player.position.x, initialYPosition);

    float distanceToPlayer = Vector2.Distance(currentPosition, playerPosition);

    if (distanceToPlayer > stopDistance)
    {
        transform.position = new Vector3(
            Mathf.MoveTowards(transform.position.x, player.position.x, speed * Time.deltaTime),
            initialYPosition,
            0);

        animator.SetBool("MafiaNPCLeftWalk", true);
    }
    else
    {
        animator.SetBool("MafiaNPCLeftWalk", false);
        animator.SetTrigger("MafiaNPCLeftIdle");

        if (!dialogStarted)
        {
            dialogStarted = true;
            player.GetComponent<PlayerController>().TurnRight();
            FindObjectOfType<DialogManager>().StartDialog(dialog, this);
        }
    }
}
```

SLIKA 32. PRVI DIO MAFIANPCCONTROLLER KODA

EndDialog() metoda poziva se na kraju dijaloga i dopušta MafiaNPC-u kretanje prema izlazu.

MoveToExitAfterDialog() metoda pomiče MafiaNPC-a prema izlazu nakon dijaloga koristeći isti princip kao i MoveToExitPoint() metoda. Kada dođe do izlaza MafiaNPC se deaktivira, a te metode vidljive su na slici 33.

```
public void EndDialog()
{
    dialogFinished = true;
    animator.SetTrigger("MafiaNPCRightIdle");
}

void MoveToExitAfterDialog()
{
    Vector2 currentPosition = new Vector2(transform.position.x, initialYPosition);
    Vector2 exitPosition = new Vector2(exitPoint.position.x, initialYPosition);

    float distanceToExit = Vector2.Distance(currentPosition, exitPosition);

    if (distanceToExit > 0.1f)
    {
        transform.position = new Vector3(
            Mathf.MoveTowards(transform.position.x, exitPoint.position.x, speed * Time.deltaTime),
            initialYPosition,
            0);

        animator.SetBool("MafiaNPCRightWalk", true);
    }
    else
    {
        animator.SetBool("MafiaNPCRightWalk", false);
        gameObject.SetActive(false);
    }
}
```

SLIKA 33. TREĆI DIO MAFIANPCCONTROLLER KODA

7.5 MAN2DAYONE CORRECT CONTROLLER SKRIPTA

Skripta Man2DayOneCorrectController je samo primjer za jednog NPC-a; odnosno svi NPC-evi dijele istu skriptu, ali sa različitim nazivima. U ovom primjeru se skripta odnosi na prvog NPC-a u prvoj sceni.

Start() metoda postavlja animator i početnu Y poziciju, a animator uz to mijenja i animacije NPC-a tijekom interakcije i kretanja.

Update() metoda provjerava NPC-evo stanje, odnosno kreće li se NPC prema igraču, klubu ili na početnu poziciju i poziva metode za kretanje.

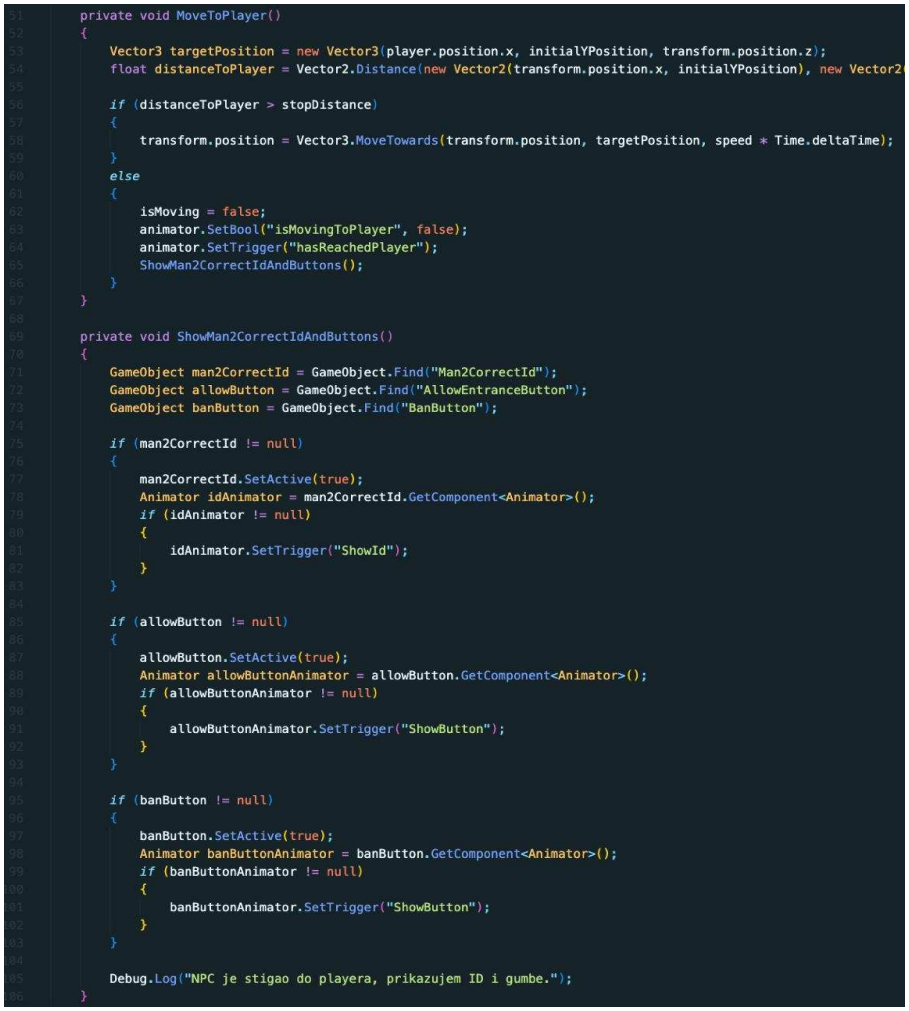
ActivateNPC() aktivira kretanje prema Playeru tako da postavi varijablu isMoving na true. Također pokreće animaciju NPC da se kreće prema Playeru pomoću animatora. Ove metode prikazane su na slici 34.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class Man2DayOneCorrectController : MonoBehaviour
{
    public Transform player;
    public Transform clubEntrance;
    public Transform startPosition;
    public float speed = 10.0f;
    public float stopDistance = 1.0f;
    private float initialYPosition;

    private Animator animator;
    private bool isMoving = false;
    private bool isReturning = false;
    private bool moveToClub = false;

    public TextMeshProUGUI goodChoicesText;
    public TextMeshProUGUI badChoicesText;

    void Start()
    {
        animator = GetComponent<Animator>();
        initialYPosition = transform.position.y;
    }

    void Update()
    {
        if (isMoving)
        {
            MoveToPlayer();
        }
        else if (moveToClub)
        {
            MoveToClub();
        }
        else if (isReturning)
        {
            MoveBackToStart();
        }
    }

    public void ActivateNPC()
    {
        isMoving = true;
        animator.SetBool("isMovingToPlayer", true);
    }
```

SLIKA 34. PRVI DIO MAN2DAYONECORRECTCONTROLLER KODA

MoveToPlayer() metoda omogućuje kretanje prema Playeru koristeći MoveTowards, a kad se NPC dovoljno približi Playeru, zaustavlja se i prikazuje sve UI elemente poput dokumenta i gumbova.

ShowMan2CorrectIdAndButtons() se koristi za prikazivanje navedenih UI elemenata, a ukoliko se ti elementi ne pronađu, neće se prikazati. Na slici 35 prikazane su navedene metode.



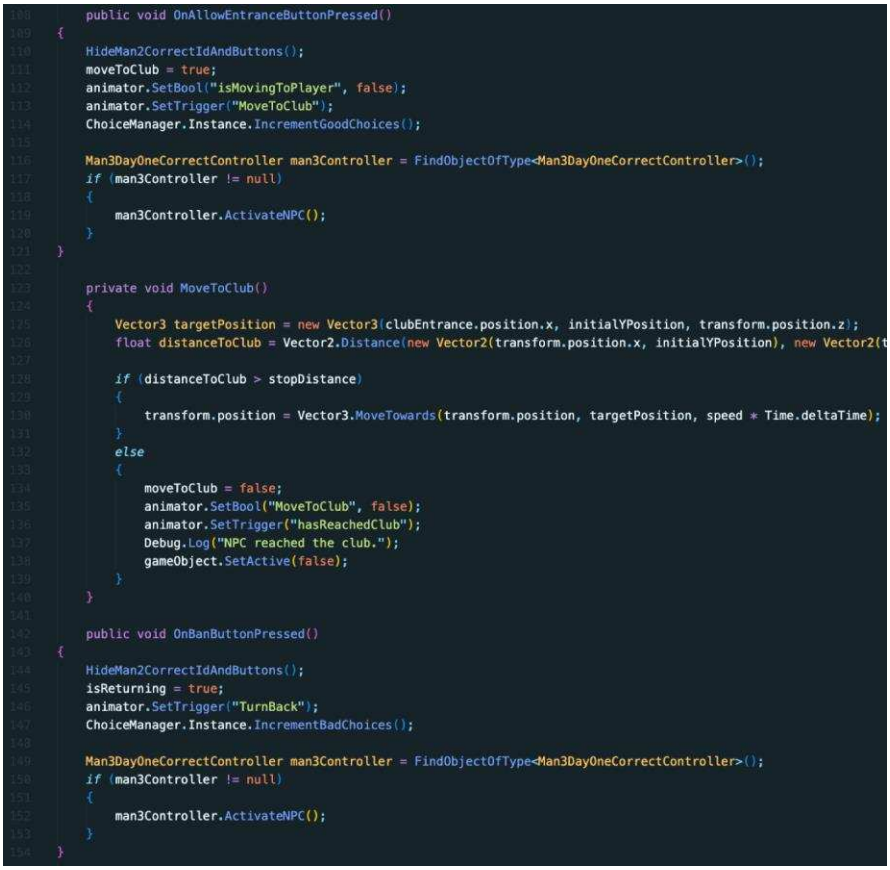
```
private void MoveToPlayer()
{
    Vector3 targetPosition = new Vector3(player.position.x, initialYPosition, transform.position.z);
    float distanceToPlayer = Vector2.Distance(new Vector2(transform.position.x, initialYPosition), new Vector2(

    if (distanceToPlayer > stopDistance)
    {
        transform.position = Vector3.MoveTowards(transform.position, targetPosition, speed * Time.deltaTime);
    }
    else
    {
        isMoving = false;
        animator.SetBool("isMovingToPlayer", false);
        animator.SetTrigger("hasReachedPlayer");
        ShowMan2CorrectIdAndButtons();
    }
}

private void ShowMan2CorrectIdAndButtons()
{
    GameObject man2CorrectId = GameObject.Find("Man2CorrectId");
    GameObject allowButton = GameObject.Find("AllowEntranceButton");
    GameObject banButton = GameObject.Find("BanButton");

    if (man2CorrectId != null)
    {
        man2CorrectId.SetActive(true);
        Animator idAnimator = man2CorrectId.GetComponent<Animator>();
        if (idAnimator != null)
        {
            idAnimator.SetTrigger("ShowId");
        }
    }

    if (allowButton != null)
    {
        allowButton.SetActive(true);
        Animator allowButtonAnimator = allowButton.GetComponent<Animator>();
        if (allowButtonAnimator != null)
        {
            allowButtonAnimator.SetTrigger("ShowButton");
        }
    }

    if (banButton != null)
    {
        banButton.SetActive(true);
        Animator banButtonAnimator = banButton.GetComponent<Animator>();
        if (banButtonAnimator != null)
        {
            banButtonAnimator.SetTrigger("ShowButton");
        }
    }

    Debug.Log("NPC je stigao do playera, prikazujem ID i gumbe.");
}
```

SLIKA 35. DRUGI DIO MAN2DAYONECORRECT KODA

MoveToClub() metoda omogućuje NPC-u da se pomiče prema klubu. Kada stigne do kluba, postavlja moveToClub na false i deaktivira NPC-a.

OnAllowEntranceButtonPressed() metoda omogućuje NPC-u da krene prema klubu i uklanja gumbove i dokument NPC-a. Također dodaje „good choice“ u brojač dobrih odluka sa ChoiceManager.Instance.IncrementGoodChoices() i aktivira sljedećeg NPC-a.

OnBanButtonPressed() radi na istom principu kao i OnAllowEntranceButtonPressed() osim što dodaje „bad choice“ u brojač loših odluka, no i to može varirati od ispravnosti dokumenta NPC-a. Navedene metode su prikazane na slici 36.



```
    public void OnAllowEntranceButtonPressed()
{
    HideMan2CorrectIdAndButtons();
    moveToClub = true;
    animator.SetBool("isMovingToPlayer", false);
    animator.SetTrigger("MoveToClub");
    ChoiceManager.Instance.IncrementGoodChoices();

    Man3DayOneCorrectController man3Controller = FindObjectOfType<Man3DayOneCorrectController>();
    if (man3Controller != null)
    {
        man3Controller.ActivateNPC();
    }
}


    private void MoveToClub()
    {
        Vector3 targetPosition = new Vector3(clubEntrance.position.x, initialYPosition, transform.position.z);
        float distanceToClub = Vector2.Distance(new Vector2(transform.position.x, initialYPosition), new Vector2(t

        if (distanceToClub > stopDistance)
        {
            transform.position = Vector3.MoveTowards(transform.position, targetPosition, speed * Time.deltaTime);
        }
        else
        {
            moveToClub = false;
            animator.SetBool("MoveToClub", false);
            animator.SetTrigger("hasReachedClub");
            Debug.Log("NPC reached the club.");
            gameObject.SetActive(false);
        }
    }

    public void OnBanButtonPressed()
{
    HideMan2CorrectIdAndButtons();
    isReturning = true;
    animator.SetTrigger("TurnBack");
    ChoiceManager.Instance.IncrementBadChoices();

    Man3DayOneCorrectController man3Controller = FindObjectOfType<Man3DayOneCorrectController>();
    if (man3Controller != null)
    {
        man3Controller.ActivateNPC();
    }
}
```

SLIKA 36. TREĆI DIO MAN2DAYONECORRECTCONTROLLER KODA

MoveBackToStart() metoda vraća NPC-a na početnu poziciju tako što koristi MoveTowards i postavlja animaciju za hodanje prema nazad.

HideMan2CorrectIdAndButtons() metoda skriva UI elemente nakon što igrač donese odluku. Na slici 37 vidljive su opisane metode.

```
    private void MoveBackToStart()
    {
        Vector3 targetPosition = new Vector3(startPosition.position.x, initialYPosition, transform.position.z);
        float distanceToStart = Vector2.Distance(new Vector2(transform.position.x, initialYPosition), new Vector2(targetPosition.x, initialYPosition));

        if (distanceToStart > stopDistance)
        {
            transform.position = Vector3.MoveTowards(transform.position, targetPosition, speed * Time.deltaTime);
        }
        else
        {
            isReturning = false;
            animator.SetTrigger("hasReachedStart");
        }
    }

    private void HideMan2CorrectIdAndButtons()
    {
        GameObject man2CorrectId = GameObject.Find("Man2CorrectId");
        GameObject allowButton = GameObject.Find("AllowEntranceButton");
        GameObject banButton = GameObject.Find("BanButton");

        if (man2CorrectId != null)
            man2CorrectId.SetActive(false);

        if (allowButton != null)
            allowButton.SetActive(false);

        if (banButton != null)
            banButton.SetActive(false);
    }
}
```

SLIKA 37. ČETVRTI DIO MAN2DAYONECORRECTCONTROLLER KODA

7.6 PLAYER CONTROLLER SKRIPTA

PlayerController skripta određuje ponašanje Player objekta. Metoda Start() inicijalizira animator pomoću pridružene komponente u objektu gdje se nalazi skripta.

TurnLeft() metoda postavlja vrijednosti Playerovog animatora za okretanje ulijevo.

TurnRight() metoda postavlja vrijednosti Playerovog animatora za okretanje udesno.

Navedene metode prikazane su na slici 38.

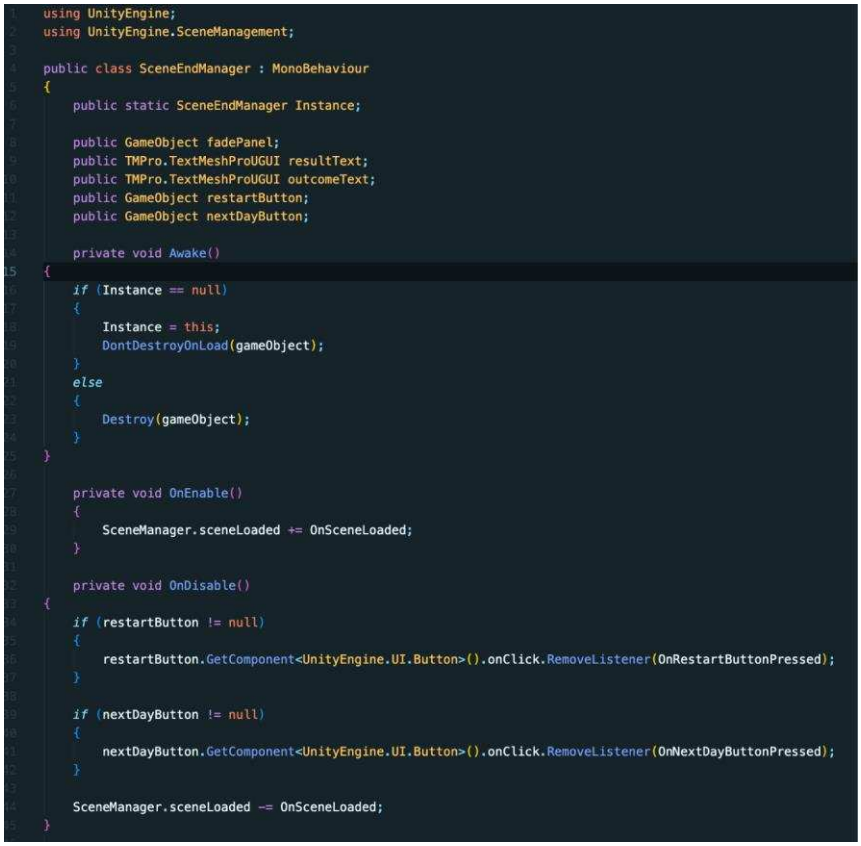
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    private Animator animator;

    void Start()
    {
        animator = GetComponent<Animator>();
    }

    public void TurnLeft()
    {
        animator.SetBool("isTurningLeft", true);
        animator.SetBool("isTurningRight", false);
    }

    public void TurnRight()
    {
        animator.SetBool("isTurningRight", true);
        animator.SetBool("isTurningLeft", false);
    }
}
```

SLIKA 38. PLAYERCONTROLLER KOD

7.7 SCENE END MANAGER SKRIPTA

Ova skripta upravlja završetkom scene i prikazuje rezultate i služi za navigiranje u sljedeću scenu. Awake() metoda osigurava da postoji samo jedna instanca SceneEndManager skripte i zadržava je tijekom izmjene scena sa DontDestroyOnLoad metodom.

OnEnable() i OnDisable() metode dodaju/uklanjaju event handler za učitavanje sljedećih scena. Opisane metode vidljive su na slici 39.



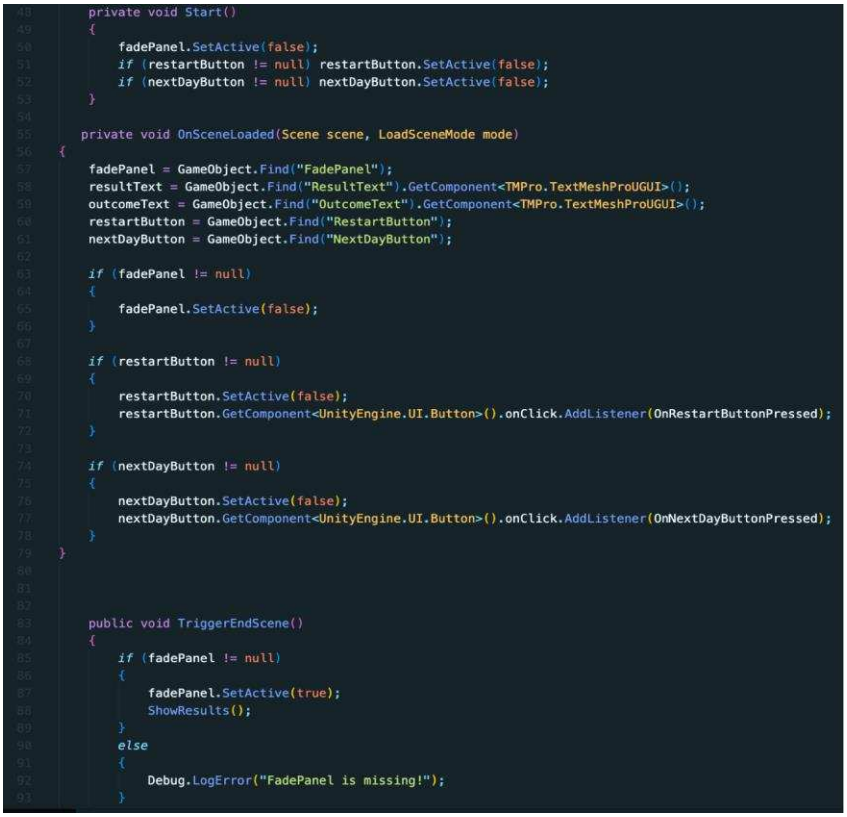
```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneEndManager : MonoBehaviour
{
    public static SceneEndManager Instance;

    public GameObject fadePanel;
    public TMPro.TextMeshProUGUI resultText;
    public TMPro.TextMeshProUGUI outcomeText;
    public GameObject restartButton;
    public GameObject nextDayButton;

    private void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
    }
}

    private void OnEnable()
    {
        SceneManager.sceneLoaded += OnSceneLoaded;
    }

    private void OnDisable()
{
    if (restartButton != null)
    {
        restartButton.GetComponent<UnityEngine.UI.Button>().onClick.RemoveListener(OnRestartButtonPressed);
    }

    if (nextDayButton != null)
    {
        nextDayButton.GetComponent<UnityEngine.UI.Button>().onClick.RemoveListener(OnNextDayButtonPressed);
    }

    SceneManager.sceneLoaded -= OnSceneLoaded;
}
```

SLIKA 39. PRVI DIO SCENEENDMANAGER KODA

Start() postavlja UI elemente na početno stanje, odnosno u neaktivno stanje.

OnSceneLoaded() metoda postavlja UI elemente na početna svojstva.

TriggerEndScene() metoda aktivira fade panel i prikazuje rezultate izbora koje je Player napravio. Metode su prikazane na slici 40.



```
    private void Start()
    {
        fadePanel.SetActive(false);
        if (restartButton != null) restartButton.SetActive(false);
        if (nextDayButton != null) nextDayButton.SetActive(false);
    }

    private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
{
    fadePanel = GameObject.Find("FadePanel");
    resultText = GameObject.Find("ResultText").GetComponent<TMPro.TextMeshProUGUI>();
    outcomeText = GameObject.Find("OutcomeText").GetComponent<TMPro.TextMeshProUGUI>();
    restartButton = GameObject.Find("RestartButton");
    nextDayButton = GameObject.Find("NextDayButton");

    if (fadePanel != null)
    {
        fadePanel.SetActive(false);
    }

    if (restartButton != null)
    {
        restartButton.SetActive(false);
        restartButton.GetComponent<UnityEngine.UI.Button>().onClick.AddListener(OnRestartButtonPressed);
    }

    if (nextDayButton != null)
    {
        nextDayButton.SetActive(false);
        nextDayButton.GetComponent<UnityEngine.UI.Button>().onClick.AddListener(OnNextDayButtonPressed);
    }
}



    public void TriggerEndScene()
    {
        if (fadePanel != null)
        {
            fadePanel.SetActive(true);
            ShowResults();
        }
        else
        {
            Debug.LogError("FadePanel is missing!");
        }
```

SLIKA 40. DRUGI DIO SCENEENDMANAGER KODA

ShowResults() metoda, ovisno o broju dobrih i loših odluka, prikazuje odgovarajući tekst kao rezultat scene.

OnRestartbuttonPressed() omogućuje ponovno učitavanje trenutne scene. Navedene metode prikazane su na slici 41.

```
    public void ShowResults()
    {
        int goodChoices = ChoiceManager.Instance.GetGoodChoicesCount();
        int badChoices = ChoiceManager.Instance.GetBadChoicesCount();

        if (resultText != null && outcomeText != null)
        {
            resultText.text = $"Good choices: {goodChoices}\nBad choices: {badChoices}";

            if (badChoices < 2)
            {
                outcomeText.text = "You did good tonight";
            }
            else if (badChoices >= 2 && badChoices <= 4)
            {
                outcomeText.text = "You did not eat today";
            }
            else if (badChoices == 5 || badChoices == 6)
            {
                outcomeText.text = "You did not have enough money for food and drink";
            }
            else if (badChoices >= 7)
            {
                outcomeText.text = "You have disappeared";
                if (nextDayButton != null)
                {
                    nextDayButton.SetActive(false);
                }
            }

            if (restartButton != null) restartButton.SetActive(true);
            if (nextDayButton != null) nextDayButton.SetActive(badChoices < 7);
        }
        else
        {
            Debug.LogError("ResultText or OutcomeText is missing!");
        }
    }

    public void OnRestartButtonPressed()
{
    fadePanel.SetActive(false);

    int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;

    SceneManager.LoadScene(currentSceneIndex);
}
```

SLIKA 41. TREĆI DIO SCENEENDMANAGER KODA

OnNextDayButtonPressed() metoda omogućuje prebacivanje igrača na sljedeću scenu u nizu.

GoToMainMenu() metoda prebacuje igrača na glavni izbornik.

Na slici 42 prikazane su zadnje dvije metode SceneEndmanagera.

```
    public void OnNextDayButtonPressed()
{
    fadePanel.SetActive(false);

    int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;

    SceneManager.LoadScene(currentSceneIndex + 1);
}
public void GoToMainMenu()
{
    Debug.Log("Go to Main Menu called");
    SceneManager.LoadScene("MainMenu");
}

}
```

SLIKA 42. ČETVRTI DIO SCENEENDMANAGER KODA

7.8 MAIN MENU SKRIPTA

MainMenu skripta upravlja osnovnim funkcijama glavnog izbornika sa mogućnostima pokretanja i izlaza iz videoigre.

OnPlayButtonPressed() metoda preusmjerava igrača na prvu scenu koristeći LoadScene.

OnQuitButtonPressed() metoda zatvara igru pomoću Application.Quit() funkcije.

GoToMainMenu() metoda prebacuje igrača na glavni izbornik i šalje potvrdu o tome u Debug.Log().

Metode MainMenu skripte prikazane su na slici 43.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine;

public class MainMenu : MonoBehaviour
{
    public void OnPlayButtonPressed()
    {
        SceneManager.LoadScene("Scene1");
    }

    public void OnQuitButtonPressed()
    {
        Application.Quit();
    }
    public void GoToMainMenu()
    {
        Debug.Log("Going to Main Menu");
        SceneManager.LoadScene("MainMenu");
    }
}
```

SLIKA 43. MAINMENU KOD

ZAKLJUČAK

U narednih nekoliko godina, industrija razvoja igara, uključujući individualne projekte, potrošila je preko milijarde dolara na kompletnu izradu nečega što će zauvijek promijeniti virtualni svijet, koristeći sve potrebne resurse i alate.

Razvoj igara zahtijeva pomno planiranje, dizajniranje i implementaciju različitih sustava, zbog čega je Unity okruženje, s obzirom na to da je besplatan i ima veliku zajednicu koja dijeli razne resurse, jedan od najboljih i najčešće korištenih alata za razvoj videoigara.

Cilj ovog projekta bio je razviti videoigru „Bouncer: Chapter 1“ u 2D okruženju koja igračima pruža iskustvo preživljavanja i donošenja važnih odluka u svijetu gdje je kriminal široko rasprostranjen. Igra sadrži interakcije između NPC-eva i igrača, što zahtijeva sustave poput dijaloga, provjere njihovih dokumenata i praćenje odluka igrača koje su vrlo važne za pravilan tijek priče. Kroz ovaj projekt prikazano je kako se Unity može koristiti za izradu tehnički naprednih i dizajnom privlačnih igara, te stvaranje interaktivnih iskustava za igrače.

Prednosti ovog projekta leže u njegovoj jednostavnosti, mehanici i mogućnostima poboljšanja. „Bouncer: Chapter 1“ usredotočen je na donošenje odluka, testiranja vještina zapažanja i sposobnosti prepoznavanja lažnih dokumenata. Zbog tih jednostavnih mehanika Unity alat je bio savršen za razvoj sustava za interakciju sa NPC-evima, animacija i dijaloga.

Međutim, unatoč prednostima koje ima ova igra, uvijek postoji prostora za unapređenje mehanika i načina implementacija iste. Igra također ima puno prostora za unapređenje vizualnog dijela, poput dodavanja novih, bolje razvijenih animacija, bogatijih vizualnih efekata, animiranje cijele pozadine, pa čak i uvođenja ciklusa za dan i noć. Napredak je dobrodošao i u dijelu priče, tako da se glavni lik upozna više sa glavnim antagonistom i razlogom zbog kojeg mu taj isti zagorčava posao svakodnevno, dodavanjem površne priče sa svakim NPC-em pomoću novih dijaloga, poput dodavanja razgovora gdje NPC nagovara glavnog igrača za dopuštenje ulaska u klub.

„Bouncer: Chapter 1“ prikazuje ogroman potencijal koji, ovako jednostavna igra, može postići korištenjem istih alata za razvoj kao i za ovu igru. Proširenje igre samo su neki od koraka koji bi mogli unaprijediti projekt u budućim fazama razvoja.

LITERATURA

[1]. Unity [Internet]. Unity Technologies; 2024. Dostupno na: https://unity.com/download

[2]. MP Games, PlayWay. Contraband Police [Internet]. PlayWay S.A.; 2024. Dostupno na: https://store.steampowered.com/app/756800/Contraband_Police/

[3]. 3909 LLC. Papers, Please [Internet]. Lucas Pope; 2024. Dostupno na: https://store.steampowered.com/app/239030/Papers_Please/

[4]. Unity Technologies [Internet]. Unity Technologies; 2024. Dostupno na: https://unity.com/

[5]. Games R. What are the key features of unity game development? [Internet]. Medium; 2024. Dostupno na: https://medium.com/@ropstam_games/what-are-the-key-features-of-unity-game-development-858e87ba6f8c

[6]. Unity Technologies. Platform development [Internet]. Unity3d.com; 2024. Dostupno na: https://docs.unity3d.com/2020.2/Documentation/Manual/PlatformSpecific.html

[7]. Meta Platforms. Oculus Rift [Internet]. Meta Platfroms; 2024. Dostupno na: https://developer.oculus.com/get-started-platform/

[8]. HTC Corporation. HTC Vive [Internet]. HTC; 2024. Dostupno na: https://developer.vive.com/resources/viveport/sdk/documentation/english/viveport-scene-sdk/getting-started-viveport-scene-sdk-unity/

[9]. Microsoft Visual Studio [Internet]. Microsoft; 2024. Dostupno na: https://visualstudio.microsoft.com/vs/unity-tools/

[10]. Nvidia PhysX Engine [Internet]. Nvidia; 2024. Dostupno na: https://unity.com/solutions/programming-physics

[11]. Unity Asset Store [Internet]. Unity Technologies; 2024. Dostupno na: https://assetstore.unity.com/

[12]. Unity Network [Internet]. Unity Technologies; 2024. Dostupno na: https://unity.com/network

[13]. GitHub [Internet]. Github, Inc.; 2024. Dostupno na: https://github.com/

[14]. GitHub Desktop [Internet]. GitHub, Inc.; 2024. Dostupno na: https://desktop.github.com/download/

[15]. Microsoft Corporation. Visual Studio Code [Internet]. Microsoft; 2024. Dostupno na: https://code.visualstudio.com/docs/other/unity

[16]. Aseprite [Internet]. Aseprite; 2024. Dostupno na: https://www.aseprite.org/

ZVUKOVI

[1]. Renda D. 8 Bit Surf [Internet]. Fesliyan Studios; 2024. Dostupno na: https://www.fesliyanstudios.com/download-link.php?src=i&id=568

POPIS SLIKA

PRILOZI

Github repozitorij: https://github.com/EugenKostro/BouncerChapter1

Gameplay video: https://youtu.be/bUQpWG9f3no

SAŽETAK

Za razvoj jednostavne igre poput „Bouncer: Chapter 1“, Unity je savršeno okruženje sa fleksibilnosti i već nekim određenim implementiranim značajkama koje znatno olakšavaju kreiranje i implementiranje različitih drugih materijala. Unity kao besplatan alat također ima i veliku zajednicu koja nudi široki spektar resursa. U ovom radu opisan je razvoj jednostavne igre uz alate za kreiranje vlastitih materijala, poput Asepritea i alata za spremanje samog projekta poput GitHub Desktopa. Svrha igre je uspješno donesti odluke u dopuštanju ulaza u klub koji je pod vodstvom mafije. Igrač ima opcije odbiti i dopustiti ulaz drugima ovisno o točnosti dokumenata NPC-eva.

Ključne riječi: Unity, Videoigra, 2D, PixelArt, GitHub, Aseprite

ABSTRACT

For the development of a simple game like "Bouncer: Chapter 1," Unity is the perfect environment, offering flexibility and pre-implemented features that greatly simplify the creation and integration of various assets. Unity, being a free tool, also benefits from a large community providing a wide range of resources. This project outlines the development of a simple game using tools for asset creation like Aseprite and project management tools like GitHub Desktop. The game's goal is to make decisions on allowing or denying entry to a mafia-run club based on the accuracy of NPCs' documents.

Keywords: Unity, Videogame, 2D, PixelArt, Github, Aseprite