Vizualno skriptiranje u programskom alatu Unity

Sitarić, Matej

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matej Sitarić

Vizualno skriptiranje u programskom alatu Unity

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matej Sitarić

Studij: Informacijski sustavi

Vizualno skriptiranje u programskom alatu Unity

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Mladen Konecki

Varaždin, rujan 2022.

Matej Sitarić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Tema rada je izrada videoigre u alatu za izradu videoigara Unity. Primarni fokus je na koji način su napravljene pojedine mehanike igre no umjesto klasičnog kodiranja će se koristiti nova značajka alata, a to je vizualno skriptiranje (eng. *visual scripting*).

Ideja je prikazati da za izradu videoigara nije potrebno potpuno znanje sintakse jezika već, ako se poznaje logika, postoji i drugačiji pristup razvoju videoigara. Prikaz funkcioniranja i izgleda značajke će biti kroz primjer videoigre čije će mehanike biti razrađene u ovom radu.

Sadržaj

1. Uvod

Ovaj rad će se osvrnuti na razvoj videoigre pomoću nove značajke unutar programa Unity koja se zove vizualno skriptiranje. Ukratko, vizualno skriptiranje služi za programiranje koje sadrži vizualni prikaz naredbi i tok programa. Pomoću vizualnog skriptiranja je moguće napraviti skoro sve stvari što je moguće i klasičnim kodiranjem, iako pri kompliciranijim mehanikama videoigre zna izgledati zapetljano.

Ta nova značajka me potaknula na izradu ovog rada kako bi se predstavila kao alternativni način programiranja videoigara za sve one kojima je kod zastrašujući i zbog toga se udalje od programiranja. Vjerujem da kada ljudima primamljivije učiti logiku programiranja kroz slaganje i spajanje različitih blokova te gledanje koji rezultat time dobivaju nego učenje sintakse jezika kako bi mogli krenuti učenje logiku programiranja.

Kroz ovaj rad će se prikazati kroz praktičan primjer kako se pomoću ove značajke može razviti videoigra iako ona nije čisto kodiranje. Opisati će se razne mehanike, kako su implementirane i na kraju osvrt na sve prednosti i nedostatke ove značajke sve na temelju praktičnog rada.

2. Programski alati

1.1. Unity

Unity je program za razvoj videoigara i podržava više od 25 platformi među kojima su Windows, iOS, Android i mnoge druge. Izdan je 2005. godine od kompanije tada zvanom Unity Software Inc., sada zvana Unity Technologies. Može se koristiti za razvod trodimenzionalnih (3D) i dvodimenzionalnih (2D) videoigara, kao i za interaktivne simulacije i ostalo. [1] Programski jezik koji Unity koristi za razvoj videoigara je C#.

Neke od glavnih prednosti Unity-a u odnosu na druge programe su:

- Jako je rasprostranjen što omogućuje novim korisnicima lakši ulaz u razvoj videoigara
- Dobar je program za početnike
- Brz je i agilan
- Čini prenošenje videoigre na druge platforme lakše
- Ima veliku i raznoliku trgovinu asseta
- Omogućuje razvoj vlastitih alata
- Dobar je za VR developere [2]

Osim toga, Unity je 2021. godine dodao novu značajku zvanu vizualno skriptiranje koja će biti opisana u podpoglavlju.

1.1.1. Vizualno skriptiranje (eng. *Visual scripting*)

Vizualno skriptiranje omogućuje kreiranje logike za igre bez pisanja koda. Vizualno skriptiranje koristi vizualni, blokovski-baziran graf, koji programeri i ne-programeri koriste za dizajniranje finalne logike ili kreiranje prototipa. [3]

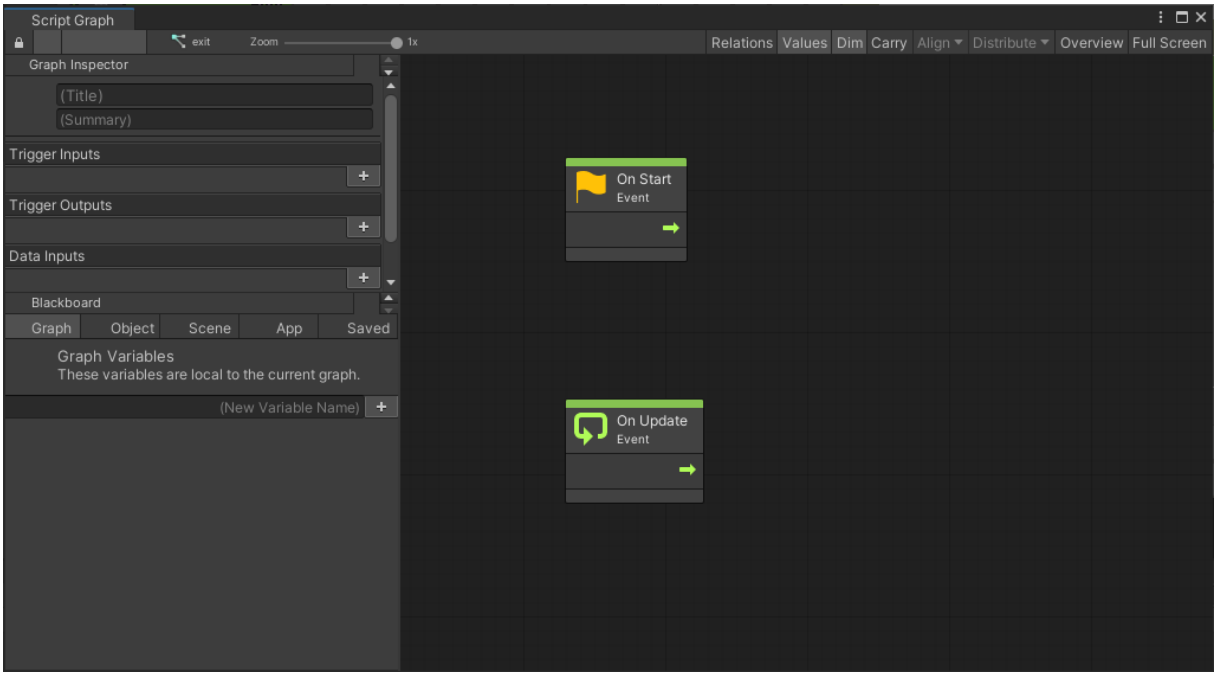
Kako bi se mogla koristiti značajka, potrebno je imati Unity verziju 2021 ili veću, jer je te godine Unity ugradio značajku u svoj program. Za korištenje nije potrebno ništa dodatno uključivati ili instalirati, jednostavno označimo objekt kojem želimo dodati skriptu stisnemo gumb „Add component“, tražimo „Script machine“ i odaberemo ga, time se dodaj nova komponenta objektu koja označava graf u kojem se može raditi skripta.

3. Videoigra

Praktičan dio rada je 2D videoigra žanra top-down shooter. Top-down se odnosi na pogled kamere odozgo, a shooter znači da se u igri puca iz puške. Inspiracija za temu rada je serija Community, a za žanr druge slične igre kao što je Enter the Gungeon.

1.2. Osnove vizualnog skriptiranja u Unity-u

Nakon dodavanja komponente objektu, unutar komponente se odabire opcija „Graph“ koja je unaprijed označena ili opcija „Embed“. Razlika je u tome što „Embed“ stvara graf samo za taj objekt, a „Graph“ stvara graf koji se može dodati i drugim objektima. U ovome radu se koristila samo opcija „Embed“. Nakon odabira opcije se može dodati naziv grafa i kratki opis. Za početak s radom treba kliknuti na gumb „Edit graph“ čime se otvara novi prozor „Script Graph“ (slika 1).



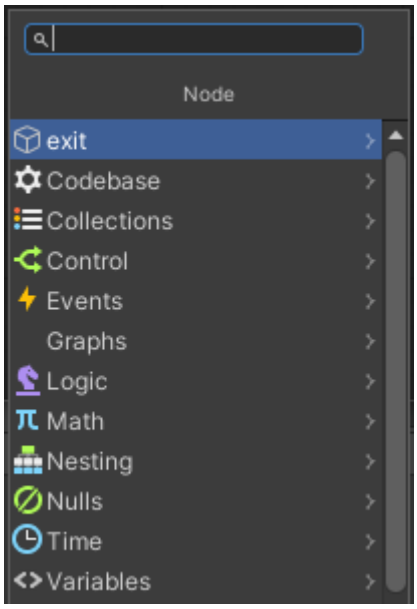
Slika 1. Početni prikaz prozora "Script Graph"

Na samom početku postoje 2 bloka, „Start“ i „Update“, oni označavaju početak akcije. „Start“ se izvodi jednom na pokretanju, a „Update“ se izvodi svaki frame (svake slike u sekundi) dok je videoigra pokrenuta. Bolje je koristiti „Fixed Update“ jer on ne ovisi o korisnikovom FPS-u nego o određenom intervalu koji je konstantan za sve korisnike.

S lijeve strane grafa se nalaze detalji o odabranom bloku, a ispod toga varijable. Postoji 5 vrsti varijabli:

- „Graph“ varijable – vrsta varijable koja vrijedi samo za taj graf i drugi grafovi joj ne mogu pristupiti (kao ključna riječ „private“ u programskom kodu“
- „Object“ varijable – vrsta varijable koja se dijeli kroz cijeli objekt i drugi objekti joj mogu pristupiti
- „Scene“ varijable – Vrsta varijable koje se dijeli kroz cijelu scenu
- „App“ varijable – vrsta varijable koja se dijeli između scena
- „Saved“ varijable – vrsta varijable koja se sprema i nakon što se ugasi videoigra

Pritiskom desnim klikom miša bilo gdje na grafu se otvara izbornik za dodavanje novog bloka. Može se upisati naziv bloka kojeg tražimo ili se može tražiti blok kroz sljedeće skupine: radnje s trenutnim objektom, „Codebase“, kolekcije, kontrole, događaji, grafovi, logika, matematika, ugnježđivanje, „Nulls“, vrijeme i varijable (slika 2).



Slika 2. Prozor za traženje blokova

S lijeve strane svakog bloka se nalaze ulazi koje blok treba za obavljanje radnje, a s desne strane se nalaze izlazi/rezultati izvođenja radnje pojedinog bloka. Većina blokova ima strelice što označava tok programa.

1.3. Igrač

Igrač je objekt kojeg se kontrolira. Kako bi se kreirao odabrao sam početni sprite iz „Project“ kartice i dati mu željeno ime, u ovom slučaju „Player“. Dodane su mu zatim sljedeće komponente: „Box Collider 2D“, „Rigidbody 2D“, „Animator“, „Script machine“. Sve funkcionalnosti će biti opisane u podpoglavljima koje slijede.

1.3.1. Kretanje

Kretanje je implementirano tako što je na samom početku uzet blok „On Fixed Update“ koji označava koja se početna radnja treba dogoditi da bi krenulo izvođenje tog djela grafa. Ovim blokom govorimo da se svaki interval pokrene radnja.

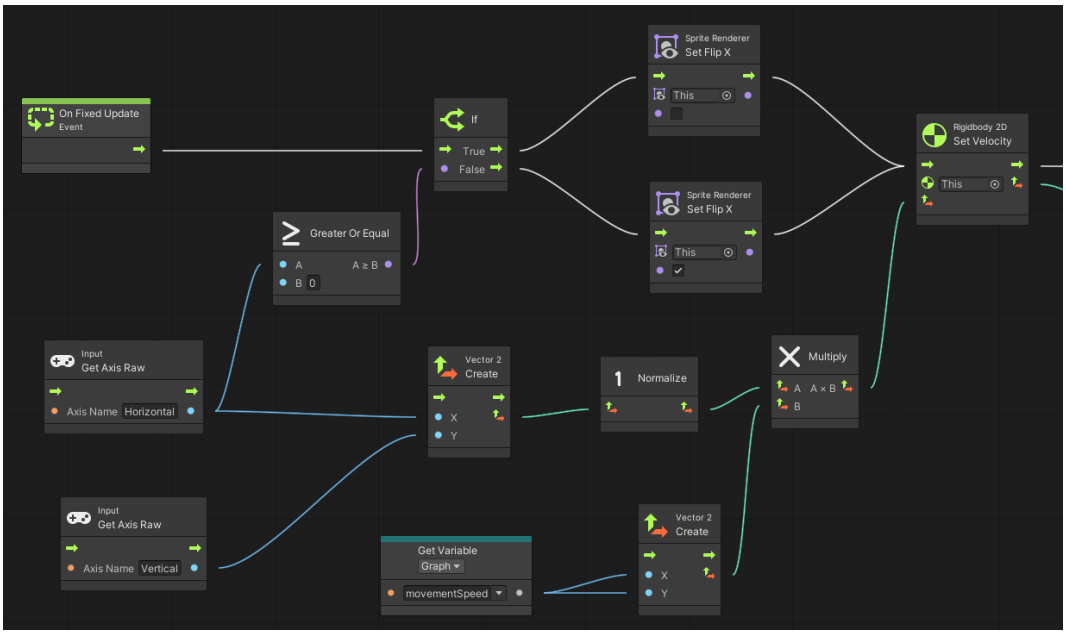
Zatim taj blok ulazi u „If“ blok. On ima ulazni parametar tipa bool, te izlazi ovisno da li je ulaz istinit ili lažan. Stavljen je „If“ blok jer ako je pritisnut ulaz za kretanje desno želimo da igrač gleda desno, inače neka gleda lijevo. Gledanje se može jednostavno podesiti tako da zrcalimo sliku po x osi pomoću bloka „Set Flip X“. U njega ulazi varijabla tipa „Object“ koaj je postavljena na „This“ jer želimo zrcaliti sliku igrača, te bool koji je zapravo kučica koju treba označiti prema potrebi.

U „If“ blok maloprije naveden ulazi upit bloka „Greater Or Equal“ koji provjerava da li je ulazna horizontalna varijabla veća ili jednaka 0. Neke ulazne varijable su unaprijed definirane u Unity-u. U ovome slučaju provjeravamo ulaz „Horizontal“ pomoću bloka „Get Axis Raw“ u kojemu se mora upisati ime varijable.

Koriste se dva bloka „Get Axis Raw“, jedan za „Horizontal“ i jedan za „Vertical“. Pomoću njih se stvara 2D vektor pomoću bloka „Vector 2 Create“, gdje „Horizontal“ ulazi u X, a u Y ulazi „Vertical“. Nakon toga se vektor mora normalizirati zato što bi inače igrač bio brži pri kretanju u koso nego li samo lijevo/desno/gore/dolje. Zatim se taj vektor mora pomnožiti pomoću bloka „Multiply“ sa 2D vektorom brzine igrača.

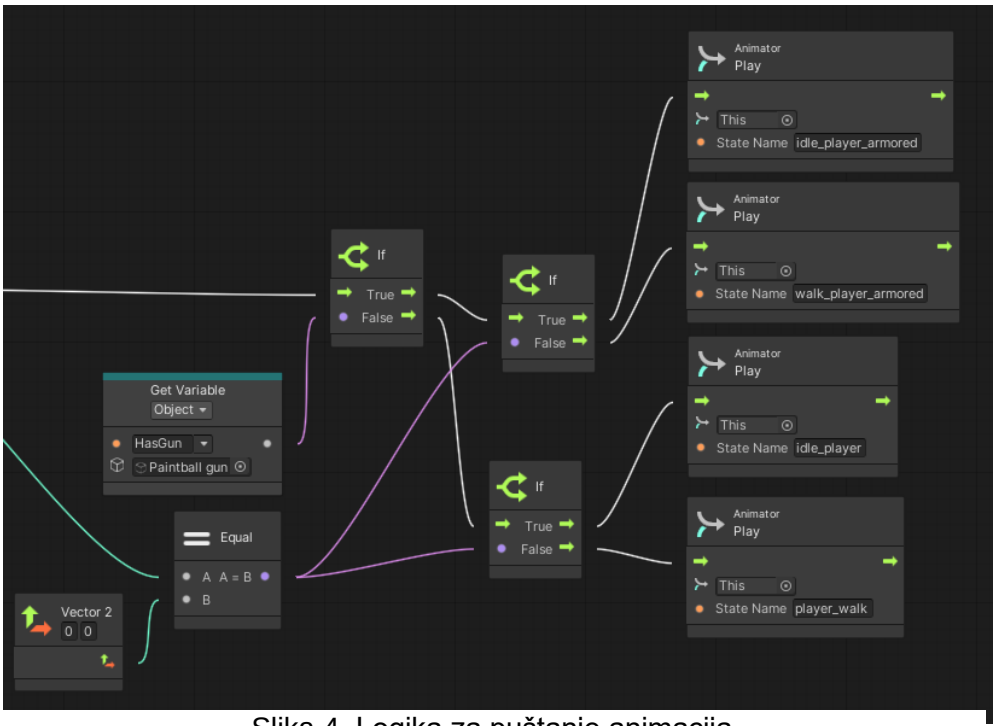
2D vektor brzine igrača se dobije tako što se stvori 2D vektor kojemu su X i Y jednaki brzini definiranoj pomoću „Graph“ varijable koja se dobiva blokom „Get Variable Graph“.

Nakon množenja vektora, treba se postaviti brzina igrača na dobiveni vektor blokom „Rigidbody 2D Set Velocity“. Opisani graf izgleda kao na slici 3.



Slika 3. Izgled grafa za kretanje igrača

Kretanje igrača još ima i jednostavnu logiku za animacije. Provjerava se da li igrač drži pušku. Ako da onda se provjerava da li je njegova brzina (uzeta iz bloka „Rigidbody 2D Set Velocity“) jednaka nuli. Ako je onda se pušta animacija (blok „Animator Play“) gdje igrač stoji sa puškom, a ako ne onda se pušta animacije gdje igrač hoda sa puškom. Ista je logika ako ne nosi pušku. Logika za animacije se može vidjeti na slici 4.



Slika 4. Logika za puštanje animacija

1.3.2.Ulaz/izlaz iz zgrade (prijelazi s lokacije na lokaciju)

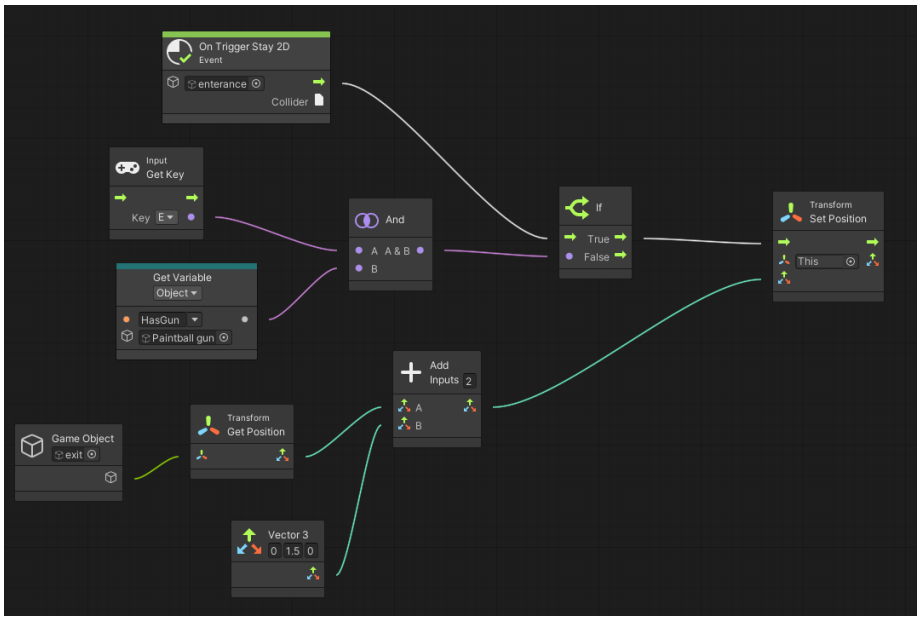
U igri postoje dva prijelaza: u/iz zgrade i na prvi kat/u prizemlje. Opisat će se logika za ulaz u zgradu jer je ista logika za sve ostale.

Kako bi igra prepoznala gdje je ulaz, a gdje izlaz, mora postojati objekt na tim mjestima. Ti objekti na sebi trebaju imati samo jednu komponentu: „Box Collider 2D“, te na njoj označenu kučicu „Is Trigger“. Ta kučica govori da se igrač ne zabija u taj objekt nego da provjerava da li ga igrač dodiruje.

Kako bi graf prepoznao da igrač dodiruje objekt za ulaz, potreban je blok „On Trigger Stay 2D“. Koristi se ostaje umjesto ulazi jer želimo da igrač uđe u zgradu tek kada stisne tipku „e“. Zato taj blok ide u „If“ blok i provjerava da li je pritisnuta. Ako je onda pomoću bloka „Set position“ postavljamo poziciju igrača na željenu poziciju.

Željena pozicija je u ovom slučaju objekt izlaz, ali malo udaljeno od njega kako igrač nebi odmah izašao iz zgrade kada bi stisnuli tipku „e“. Za dobivanje objekta izlaz koristi se blok „Game Object“ koji dohvaća odabrani objekt iz scene. Njegova se pozicija dobiva blokom „Get Position“ koja je tipa vektor 3. Njega zatim zbrojimo za onaj vektor koliko želimo da igrač bude udaljen od ulaza, u ovom slučaju to je 1,5 po Y osi. Kako bi se mogli zbrojiti vektori, drugi vektor se treba deklarirati pomoću bloka „Vector 3“.

Nakon zbrajanja vektora, rezultat ulazi u blok „Set Position“ kako je moguće vidjeti na slici 5.



Slika 5. Logika ulaska u zgradu

Logika izlaska iz zgrade, te logika za penjanje na 1. kat/silaženje u prizemlje je ista samo se gledaju drugi objekti.

1.3.3. Događaj „Hit“

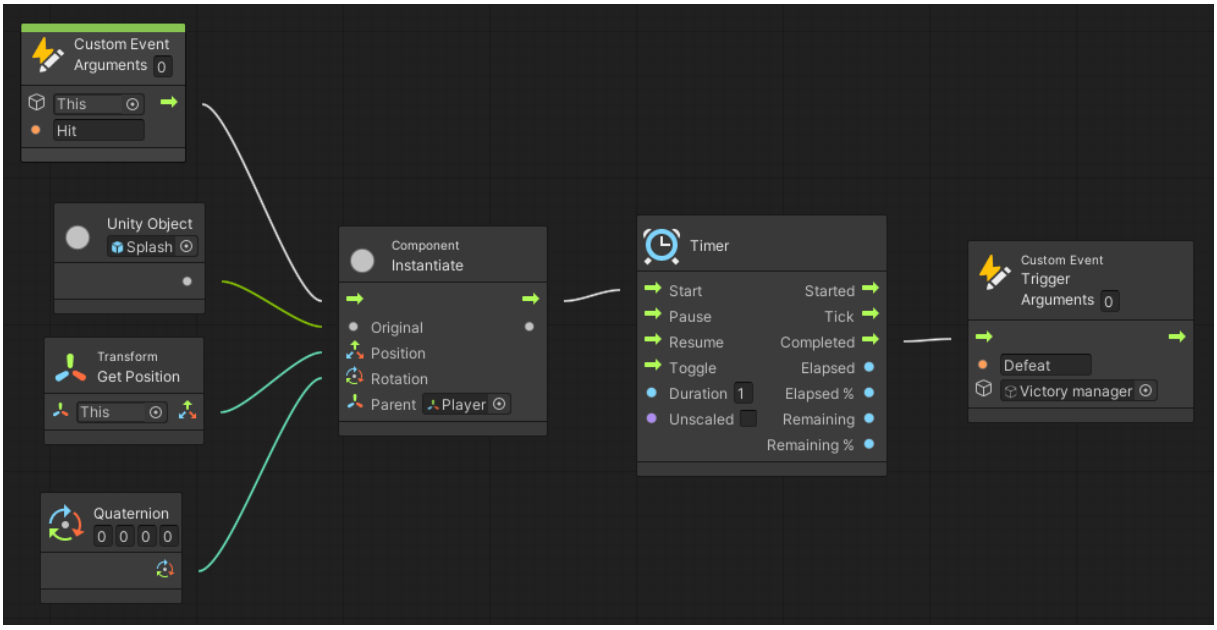
Ova funkcija, odnosno događaj, se izvodi samo ako se pozove, a definira se pomoću bloka „Custom Event“. Može se upisati njegov naziv te se može odrediti broj ulaznih argumenata. Ona se poziva ako neprijatelj pogodi igrača.

Pri pozivanju funkcije se instancira novi objekt iz predefinirani objekata zvanih „prefab“. To su objekti kojima je definirano njihovo ponašanje/izgled i pospremljeni su u datoteku ako ih treba instancirati više puta. Odabrani objekt je boja koja ostane nakon ispucanog metka iz puške za paintball.

Kako bi se mogao instancirati objekt, on treba poziciju gdje će se instancirati (u ovom slučaju na igrača), rotaciju i roditelja. Rotacija je tipa „Quanternion“ te se tako zove i blok kojim se on instanicira. A za roditelja instanciranog objekta je odabran igrač jer je on pogođen.

Nakon što je igrač pogođen, blok „Timer“ pričeka sekundu i onda poziva događaj drugog objekta kako bi on obradio sve što je dalje potrebno. Događaj se poziva blokom „Custom Event Trigger“, potrebno je upisati naziv događaja te odabrati kojem objektu on pripada.

Izgled funkcionalnosti se može vidjeti na slici 6.



Slika 6. Graf funkcije "Hit"

1.4. Spremnik za paintball pušku

Objekt igrača ima prazan objekt-dijete na sebi koji će postati roditelj paintball pušci kada se pokupi. On služi kako bi se promijenilo usidrenje puške zbog rotacije. Bez ovog objekta, puška bi se rotirala oko svoje sredine, a želimo da se okreće oko svojeg početka.

1.4.1. Rotacija prema pokazivaču

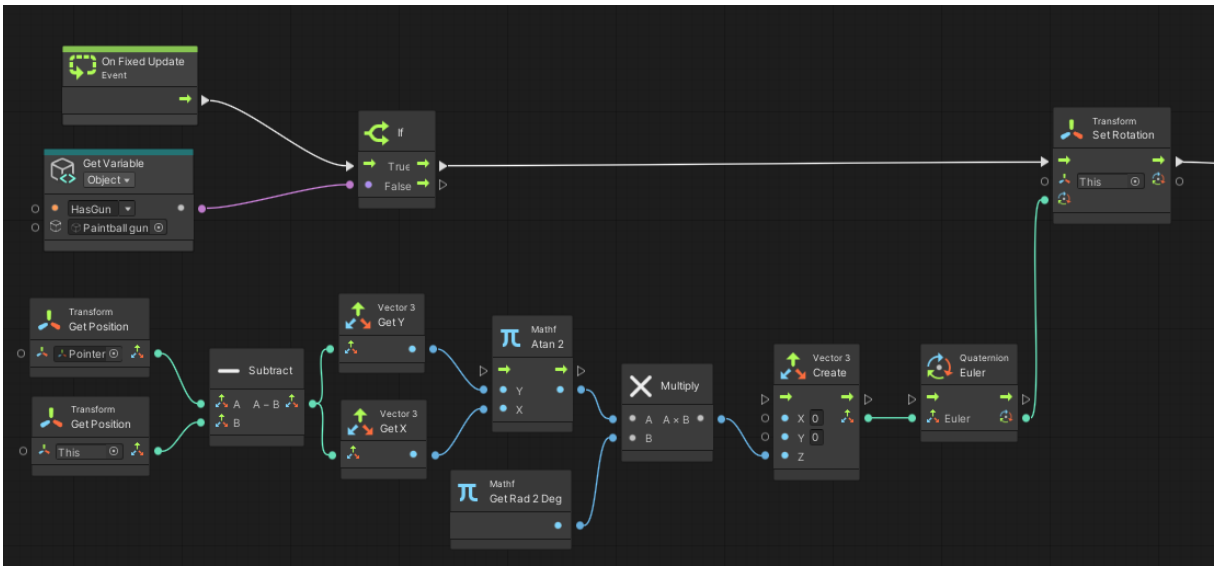
Ova logika služi za ciljanje puške prema cilju. Centar rotacije je prazan objekt maloprije naveden u kojemu se nalazi navedena logika.

Na početku je potrebno provjeriti da li je igrač pokupio pušku pomoću objektne varijable s puške preko „If“ bloka. Ako je pokupio pušku onda treba postaviti rotaciju ovog objekta pomoću bloka "Set Rotation".

Kako bi se dobila željena rotacija prema pokazivaču, na samom početku je potrebno oduzeti pozicije odredišta (pokazivača) i početka (ovog objekta). Rezultat oduzimanja je vektor 3,a za računanje stupnjeva potrebni su X i Y. Oni se dobivaju pomoću blokova „Get X“ i „Get Y“. Oni uzimaju samo jednu točku iz vektora. Te točke treba spojiti s blokom „Atan 2“ koji računa tangens Y/X i vraća rezultat kao kut u radijanima (Y je Y vektora, a X je X vektora). Kako bi se objekt pravilno rotirao, on treba kut u stupnjevima, a ne radijanima. Za pretvorbu je potreban blok „Multiply“ koji će pomnožiti rezultat u radijanima sa konstantom za pretvaranje radijana u stupnjeve. Tu konstantu nije potrebno ručno upisivati već se ona može dobiti blokom „Get Rad 2 Deg“.

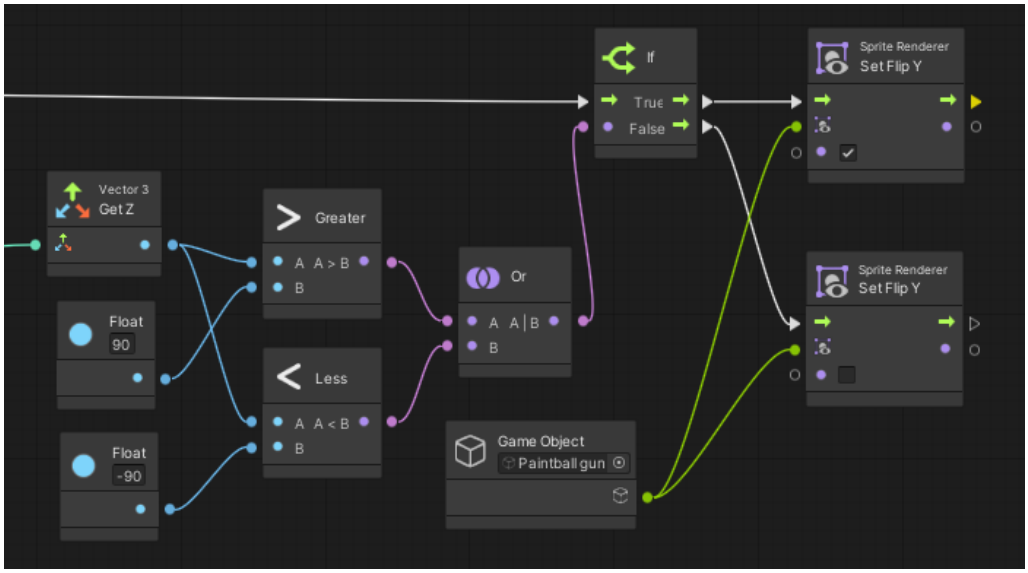
Rezultat množenja treba staviti u Z vrijednost novog vektora 3 pomoću bloka „Vectoor 3 Create“, a X i Y novog vektora trebaju biti 0. No blok „Set Rotation“ za ulaz ima podatak tipa „Quaternion“. Kako bi se dobio taj tip podatka iz vektora 3, treba u blok „Quaternion Euler“ staviti taj vektor, a rezultat bloka je tip podatka koji je potreban.

Vizualni prikaz grafa rotacije je moguće vidjeti na slici 7.



Slika 7. Graf rotacije prema pokazivaču

Za bolji izgled, da puška ne bi bila naopako kad se cilja prema desno, provjerava se da li je Z vrijednost vektora 3 (prije pretvaranja u quaternion) veća od 90 ili manja od -90. Ako je onda se postavi zrcaljenje puške po Y osi na istinu, inače na laž blokom „Set Flip Y“ koji radi na isti način ranije opisat blok „Set Flip X“.



Slika 8. Graf zrcaljenja puške

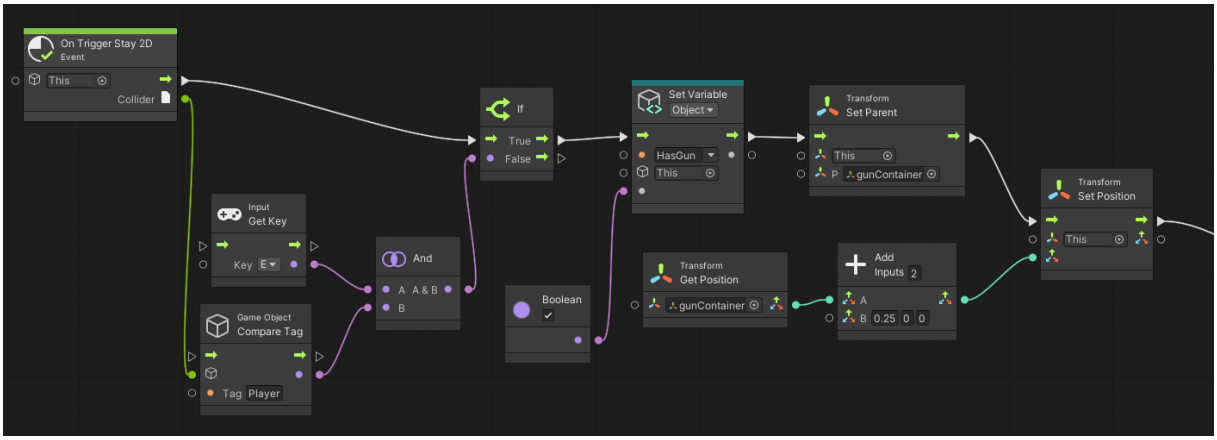
1.5. Paintball puška

1.5.1. Uzimanje puške u ruke

Na početku igre, puška se nalazi na stolu i igrač je treba pokupiti kako bi mogao pucati. Uz pušku igrač automatski pokupi i zaštitnu opremu koja se makne iz scene nakon što se puška pokupi. Puška na sebi ima istu komponentu kao i ulaz u zgradu „Box Collider 2D“ koji ima označeni okidač.

Kao i ulaz u zgradu, graf započinje blokom „On Trigger Stay 2D“, nakon kojeg se provjerava da li je pritisnuta tipka za interakciju i da li je objekt koji ga dodiruje igrač preko oznake (eng. *Tag*). Za usporedbu oznake koristi se blok „Compare Tag“ u čiji ulaz ide objekt koji daje početni blok te varijabla tipa „string“ koja označava tekst. Zato je igraču potrebno postaviti novu oznaku „Player“ te upisati „Player“ u ulazno polje „Tag“.

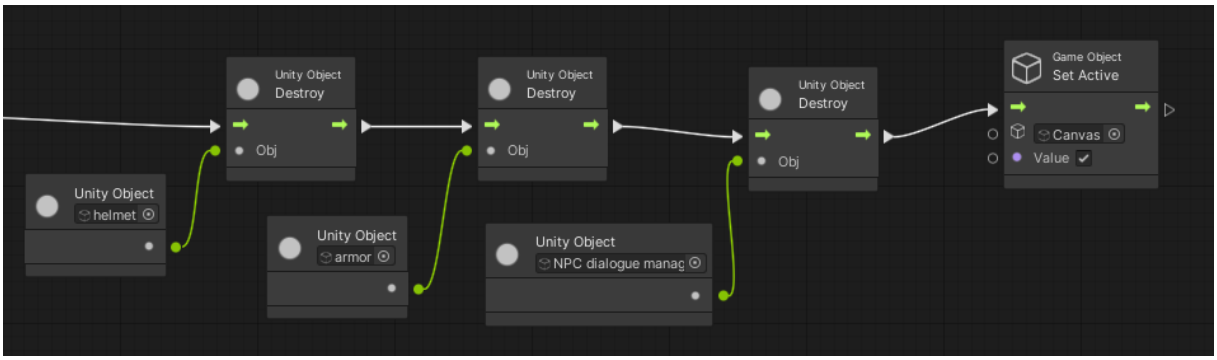
Ako to je igrač i pritisnuta je tipka „e“ onda se postavlja varijabla objekta u istinu blokom „Set Variable Object“. Zatim se postavlja roditelj pušci blokom „Set Parent“ na objekt opisan u poglavlju 3.3. Nakon postavljanja roditelja, postavlja se pozicija puške malo u desno od njegovog roditelja zbog pomaka centra rotacije pušci. To se radi tako što se uzme pozicija roditelja, zbroji se sa vektorom 3 kojemu je postavljena samo pozicija X na 0,25 i dobiveni vektor se stavlja kao pozicija pušci.



Slika 9. Graf uzimanja puške

Nakon uzimanja puške, potrebno je maknuti iz scene objekte koji vizualno prikazuju zaštitnu opremu, te objekt koji prikazuje dijalog NPC-a (kasnije opisan). Objekti se brišu iz

scene blokom „Unity Object Destroy“. Nakon micanja objekata iz scene, aktivira se objekt „Canvas“ koji u sebi sadrži sve elemente sučelja i njenu logiku (također kasnije opisano).



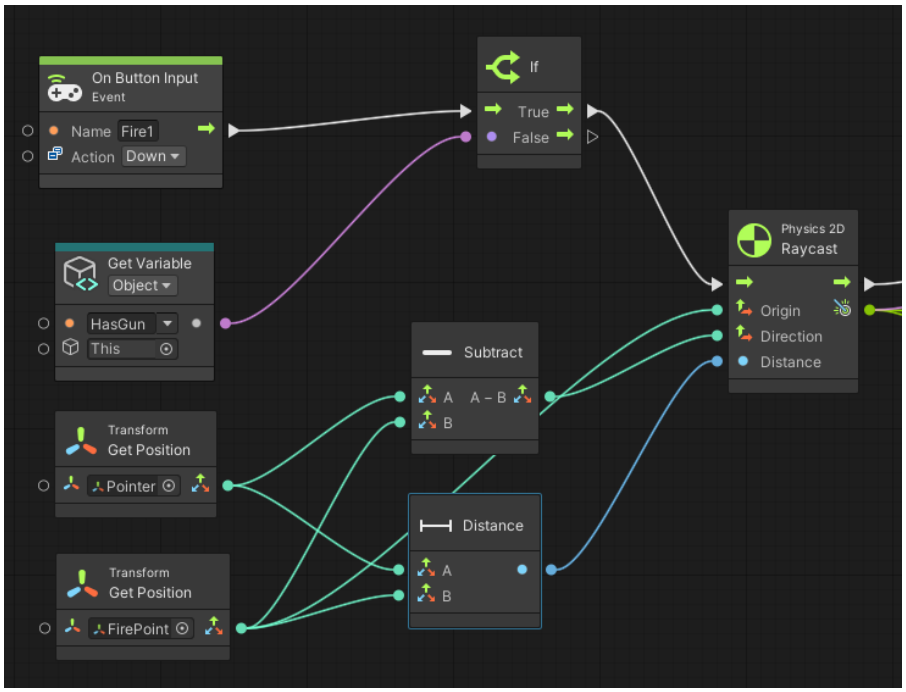
Slika 10. Micanje objekata iz scene

Puška ima prazan dijete-objekt koji treba biti pozicioniran na vrh puške. Služiti će kod pucanja da se zna od kuda se puca.

1.5.2. Pucanje

Pucanje nema kompliciranu logiku. Pokreće se pomoću bloka „On Button Input“ koji pokrene graf kada je napravljena akcija sa nekom tipkom, u ovom slučaju „Fire1“ je lijeva tipka miša, a „Down“ znači na njen pritisak. Ako je pritisnuta tipka i igrač drži pušku, stvara se linija pomoću bloka „Physics 2D Raycast“.

Da bi se linija mogla stvoriti, potrebni su joj izvor, smjer i duljina. Izvor linije je točka od kuda ona kreće, što je prazno dijete-objekt puške koje je pozicionirano na njen vrh. Smjer se dobiva oduzimanjem vektora odredišta od vektora izvorišta, u ovom slučaju to su pokazivač i točka pucanja. Udaljenost se dobiva blokom „Distance“ u koji ulaze dvije točke tipa vektor 3, a izlaz je njihova udaljenost tipa „float“.



Slika 11. Pucanje iz puške

1.5.3. Pogodak

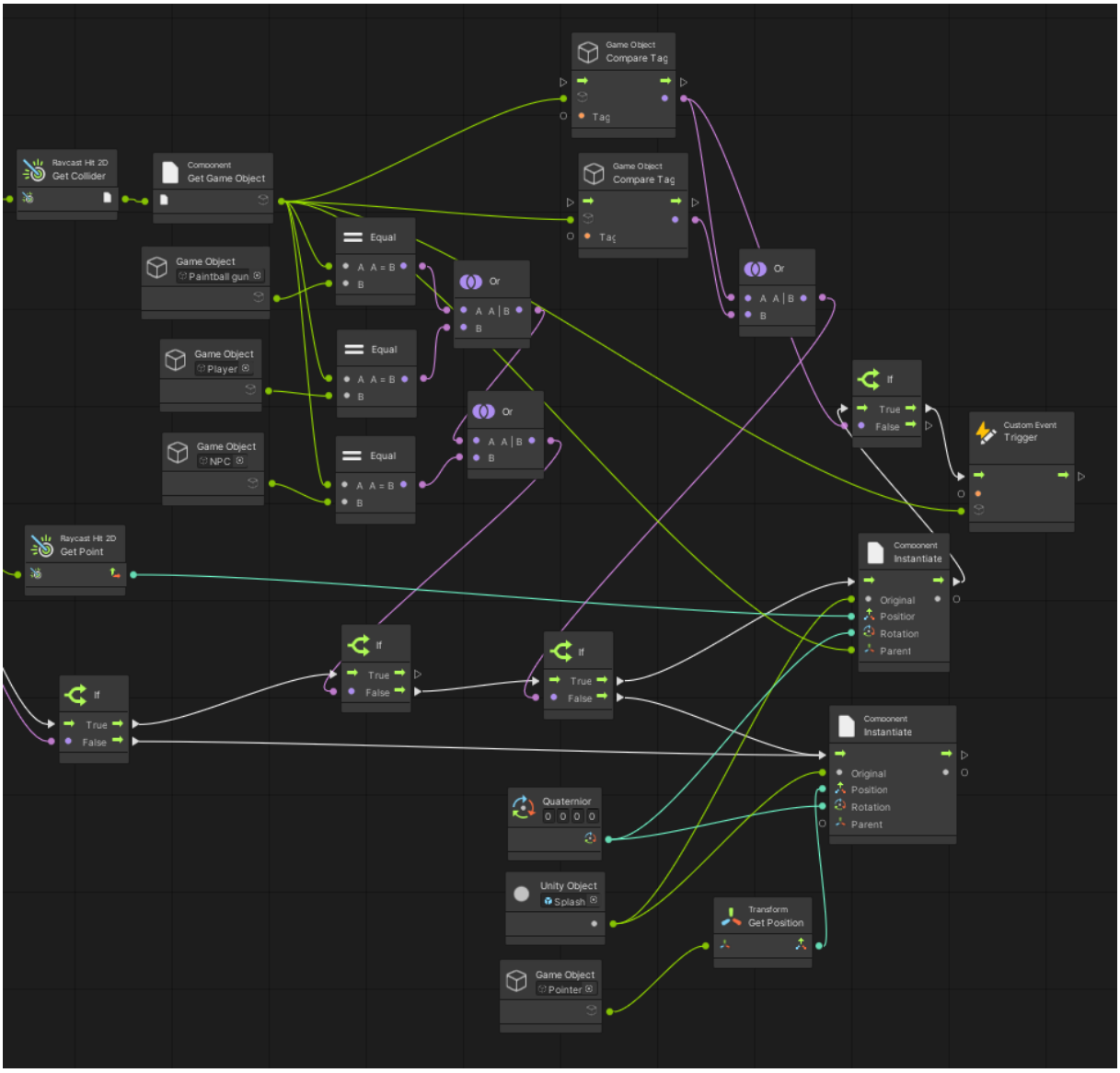
Nakon stvaranja dužine, provjerava se da li je ikoji objekt pogođen. Pogođen može biti samo komponenta „Collider“ nekog objekta. Ako nije onda se instancira objekt ranije definiranog „prefab“-a na krajnju točku linije s rotacijom 0 i roditelj je scena.

No ako je pogodilo objekt sa komponentom „Box Collider 2D“, uzima se pogođena komponenta blokom „Raycast Hit 2D Get Collider“, zatim njen pripadajući objekt blokom „Component Get Game Object“ i provjerava se da li je taj objekt puška, igrač ili NPC. Ako je onda se ne instancira „prefab“ jer nema smisla ako igrač može pogoditi svoju pušku, sam sebe ili neutralnog NPC-a.

Ako je neki drugi objekt, onda se provjerava oznaka tog objekta, da li je on neprijatelj ili prepreka. Ako nije onda se „prefab“ instancira na maloprije opisan način.

Ako je pogođen objekt s oznakom neprijatelj ili prepreka, onda se instancira „prefab“ na pogođenu poziciju, sa rotacijom 0 i postavi se kako dijete tog objekta.

Zatim se provjerava da li pogođeni objekt ima oznaku neprijatelj. Ako ima onda se poziva njegova funkcija da je pogođen.



Slika 12. Logika za pogodak

1.6. Pokazivač

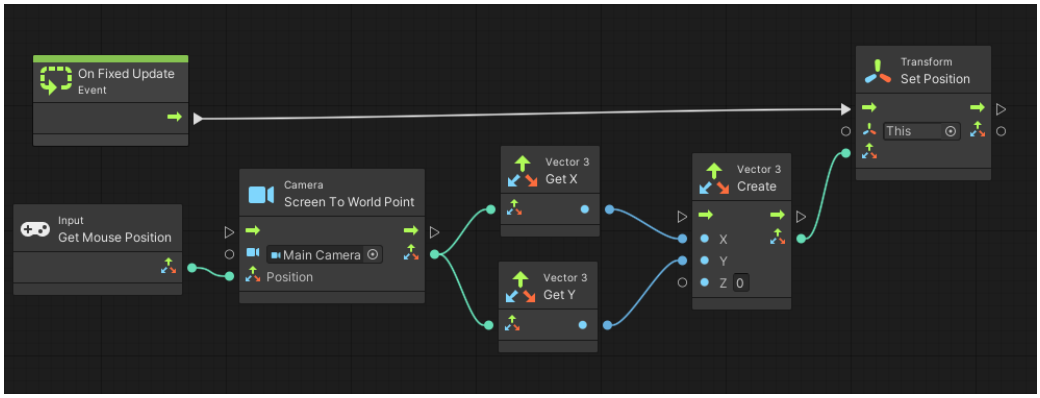
1.6.1. Postavljanje pokazivača na poziciju kursora

U smislu ovog rada, kursor je pokazivač miša, a pokazivač objekt u igri koji ima ulogu kursora.

Kako bi pokazivač u svakom trenutku bio na poziciju kursora, mora se objekt staviti svaki trenutak na tu poziciju. Zato se za početak grafa koristi blok „On Fixed Update“.

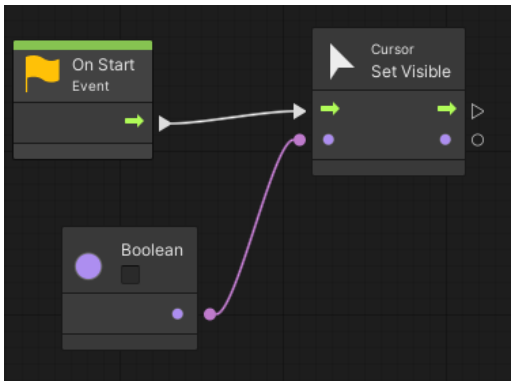
No pozicija kursora i objekta nije jednaka. Objekt se nalazi u svijetu igre, a kursor u svijetu ekrana. Pri uzimanju pozicije kursora blokom „Get Mouse Position“ pretvorba u poziciju svijeta igre poziva se funkcija blokom „Screen To World Point“. Ona radi pretvorbu pozicije svijeta ekrana u svijet igre.

Kako bi se mogla postaviti pozicija objekta, potrebni su X i Y dobivenog vektora. Pomoću njih stvorimo novi vektor 3 i na stvorenu poziciju se stavi objekt.



Slika 13. Postavljanje pokazivača na poziciju kursora

Nakon napravljenog grafa, pokazivač je na poziciji kursora, no tijekom igre se kursor i dalje vidi. Kako bi se sakrio, potrebno je samo na samom početku igre (blok „OnStart“) postaviti vidljivost kursora na laž blokom „Cursor Set Visible“.



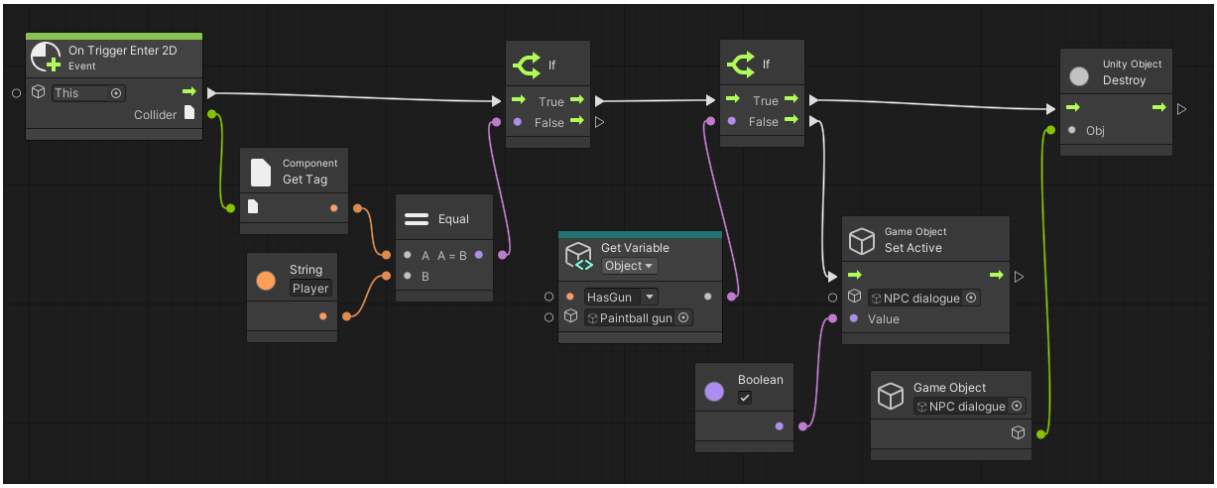
Slika 14. Postavljanje vidljivost kursora

1.7. Dijalog menadžeri

U igri su napravljena dva menadžera za dijalog. Jedan upravlja prikazivanjem teksta za NPC-a, a drugi za igrača.

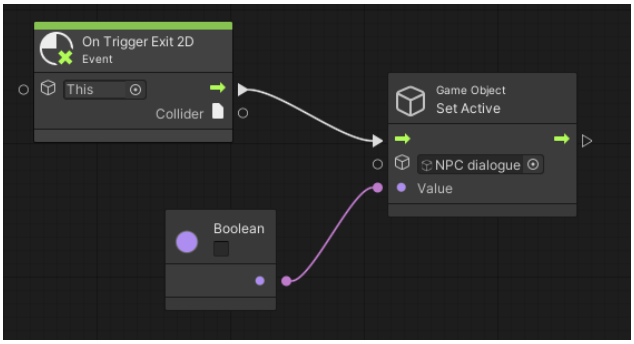
1.7.1. NPC dijalog menadžer

Ovaj dijalog menadžer ima postavljenu komponentu „collider“ kao i ulaz u zgradu. Kada igrač uđe unutar njega (blok „On Trigger Enter 2D“) provjerava se da li je oznaka objekta jednaka igraču. Ako je provjerava se da li igrač ima pušku, ako ne onda se prikaže tekst dijaloga. No ako je ima onda uništi samoga sebe tako da ne prikazuje dijalog ako igrač ponovno uđe u njega.



Slika 15. Logika prikaza dijaloga za NPC-a

Osim ulaska u komponentu, potrebna je logika za izlazak iz komponente. Blokom „On Trigger Exit 2D“ pokreće se tok tek kada igrač izađe iz komponente. Prilikom izlaska skriva se dijalog tako što se njegov objekt deaktivira blokom „Game Object Set Active“.

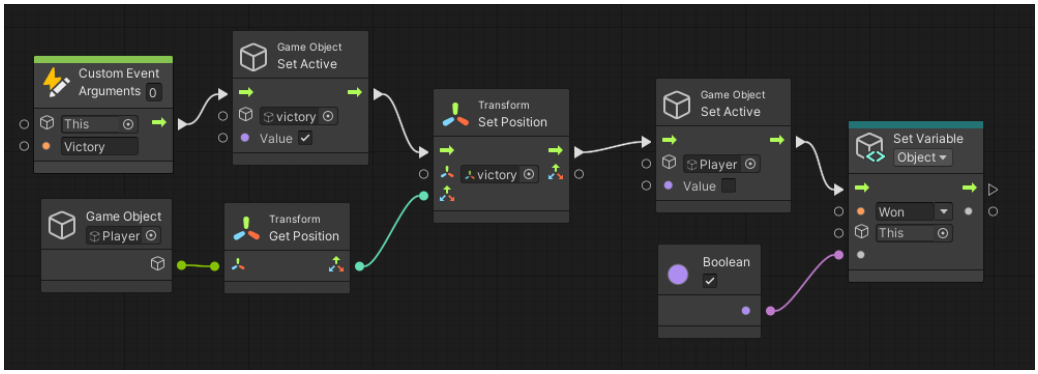


Slika 16. Logika skrivanja dijaloga

1.7.2.Menadžer pobjede/gubitka

Unutar ovog menadžera se nalaze dva događaja, jedan za pobjedu, a drugi za gubitak. Identični su, jedina je razlika koji dijalog se prikazuje igraču i nazivi događaja.

Prilikom poziva događaja, aktivira se objekt koji prikazuje poruku pobjede/gubitka, zatim se njegova pozicija stavlja na poziciju igrača, deaktivira se objekt igrača (kako se ne bi kretao dok je prikazana poruka) te se postavlja varijabla objekta pobijedio u istinu.

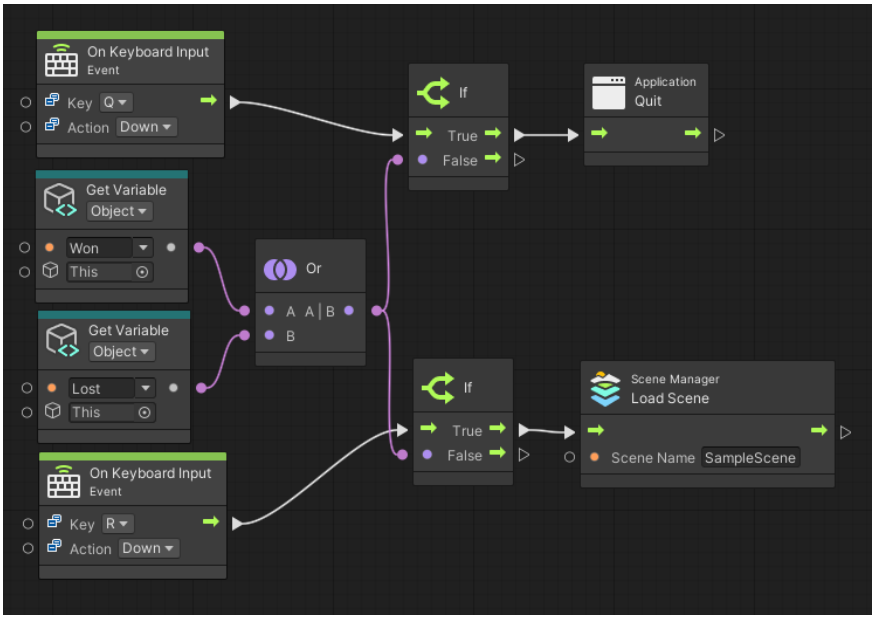


Slika 17. Logika prikaza poruke pobjede

Resetiranje igre i izlaz iz nje

Nakon prikaza poruke pobjede ili gubitka, igrač ima ponudu da ponovno pokrene igru ili da izađe iz nje.

Na pritisak tipke „q“ provjerava se da li je igrač pobijedio/izgubio pomoću varijable pobijedio/izgubio, igra se zatvori blokom „Application Quit“. A pritiskom tipke „r“ radi se ista provjera no onda se ponovno učita početna scena blokom „Load Scene“ koji kao ulaz ima varijablu „string“ u koji treba upisati naziv scene koju treba učitati.



Slika 18. Logika za izlaz iz igre i resetiranje

1.8. Neprijatelji

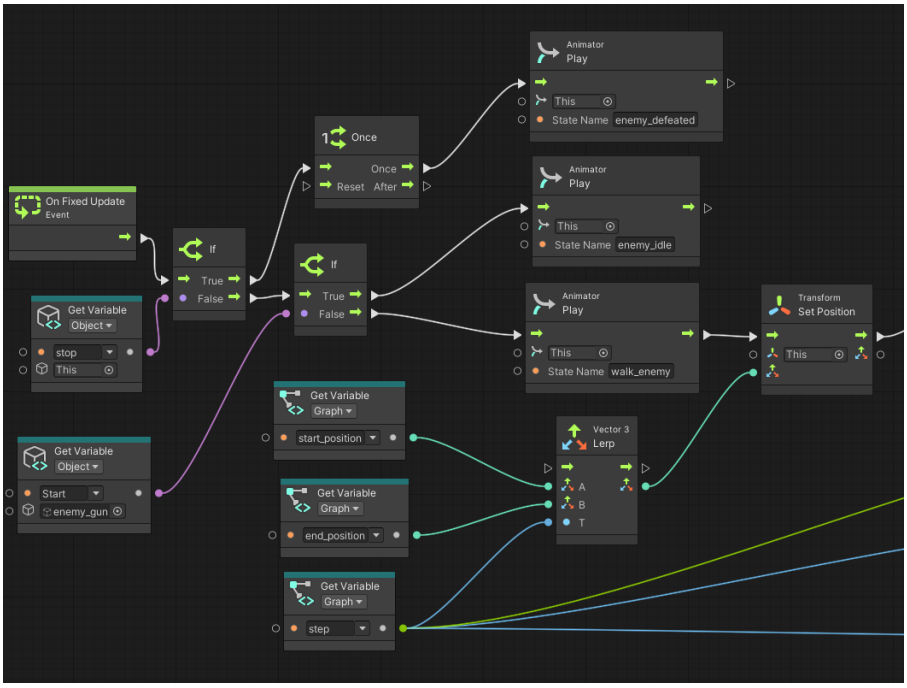
Svi neprijatelji su isti i imaju istu logiku. Jedino se razlikuju u točkama između kojih hodaju. Kao i igrač, imaju prazan dijete-objekt koji služi za okretanje puške, ali umjesto da cilja prema igraču, objekt na koji cilja je igrač. Također imaju objekt paintball pušku koja je dijete tog objekta i na kraju dijete puške je prazan objekt koji predstavlja točku iz koje puca neprijatelj.

1.8.1.Kretanje

Kretanje neprijatelja je malo drugačije od kretanja igrača. Ako igrač nije poražen i nema igrača u blizini, postavlja poziciju na točku koju dobiva iz bloka „Vector 3 Lerp“. To je funkcija koja stvara pravac između dvije točke i uzme postotak udaljenosti od početne točke kao trenutnu točku. Na samome početku se mora staviti pozicija neprijatelja na početnu poziciju spremljenu u varijabli.

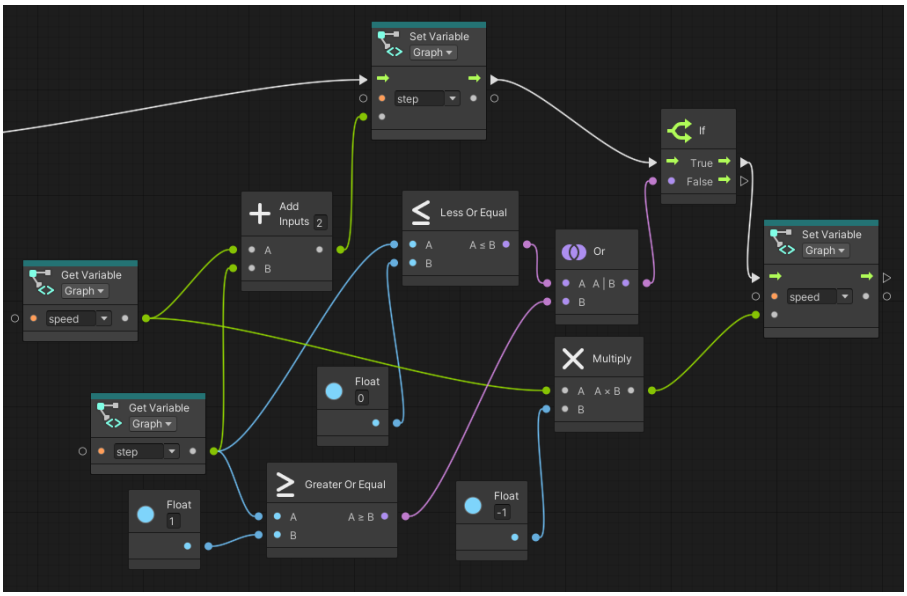
Početna i završna točka „Lerp“ funkcije su spremljene varijablu kako bi se lakše mijenjale prilikom dupliciranja neprijatelja. Osim njih, u funkciju ulazi „float“ vrijednost koja govori točku s kojeg postotka treba vratiti. Taj postotak je spremljen u varijablu tako da se nakon postavljanja pozicije može ona povećati za drugu varijablu koja određuje brzinu kretanja neprijatelja. Vrijednosti tog ulaza mogu biti samo između nula i jedan.

Ako je igrač poražen onda pomoću bloka „Once“ jednom pokrene animaciju za poraz, a ako vidi neprijatelja onda stane, inače hoda.



Slika 19. Logika kretanja neprijatelja

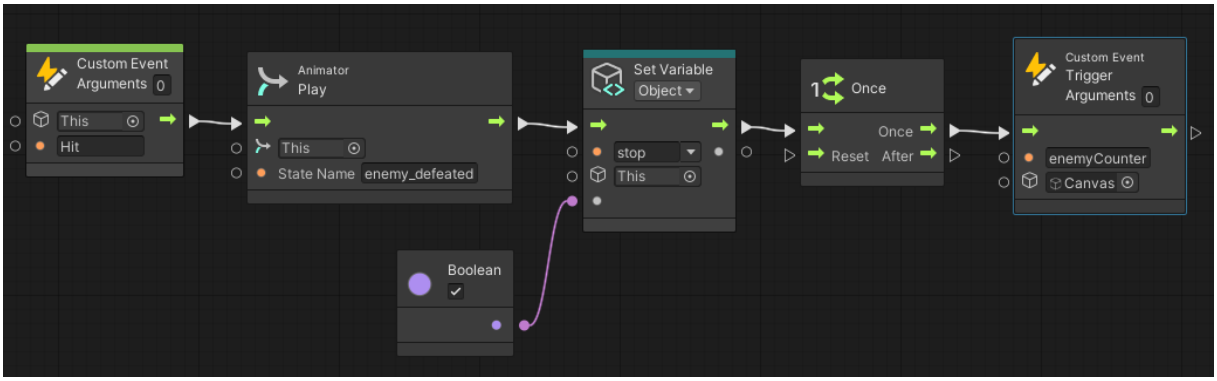
No da varijabla kojom se upravlja postotak dužine ne bi prešla jedan ili nulu, ako dođe do tih krajnjih vrijednosti onda se varijabla za brzinu množi sa -1 kako bi neprijatelj nastavio hodati u suprotnom smjeru.



Slika 20. Provjera graničnih vrijednosti

1.8.2.Poražen logika

Logika za poraz neprijatelja se nalazi u posebno događaju koji prvo pusti animaciju za poraz, onda postavlja varijablu da je poražen u istinu i onda jednom pozove funkciju korisničkog sučelja koje smanjuje broj preostalih neprijatelja.



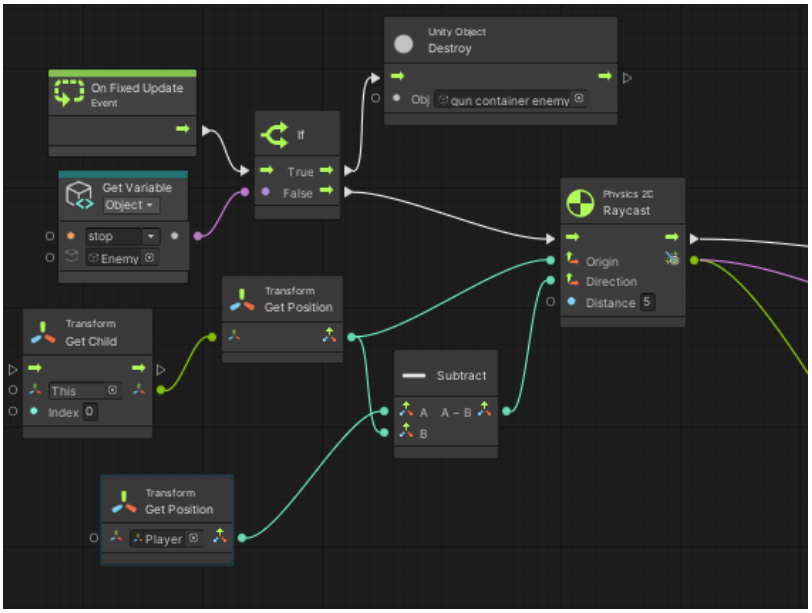
Slika 21. Logika za poraz neprijatelja

1.8.3.Logika pucanja

Logika pucanja se nalazi na objektu paintball puške neprijatelja, kao i kod igrača. Na samom početku treba napraviti liniju od vrha puške prema igraču.

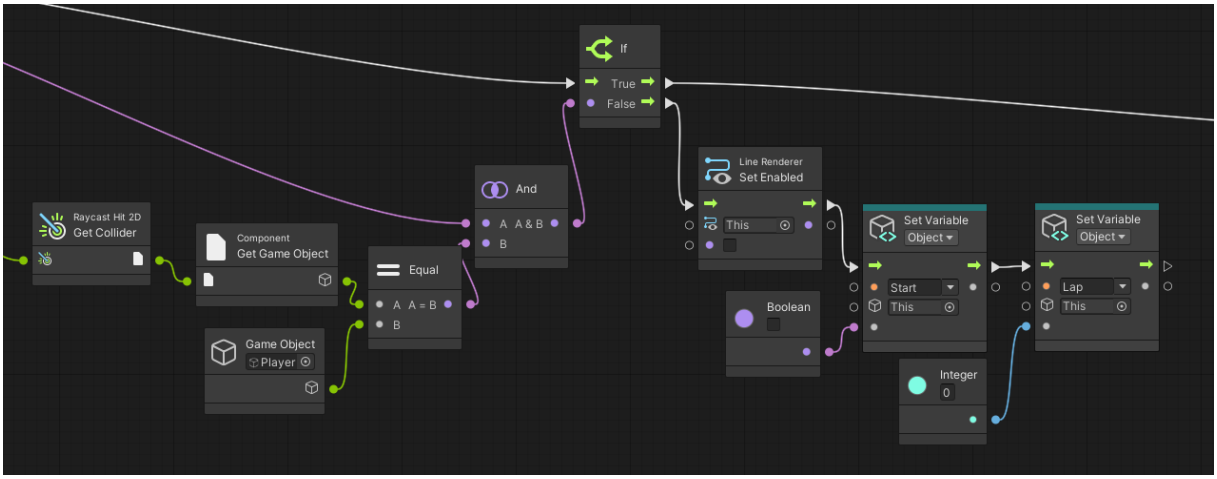
Logika je ista kao i kod igrača. Ako neprijatelj nije poražen onda radi liniju od svojeg djeteta (vrh puške) prema razlici pozicije igrača i pozicije djeteta. Razlika je u udaljenosti, kod neprijatelja udaljenost je postavljena na 5, što govori da neprijatelj može pucati na igrača jedino ako je unutar udaljenosti 5.

No ako je neprijatelj poražen, u ovom slučaju sam sebe obriše iz scene zbog izgleda animacije za poraz.



Slika 22. Pucanje iz puške neprijatelja

Ako je igrač unutar udaljenosti 5, onda se omogući komponenta „Line Renderer“ blokom „Line Renderer Set Enabled“, inače se onemogući. Ako se onemogući, tada se treba postaviti varijabla za određivanje početka prikazivanja indikatora na laž te varijabla koja broji koliko puta se izmijenio indikator na nulu.



Slika 23. Logika ako je neprijatelj van udaljenosti 5

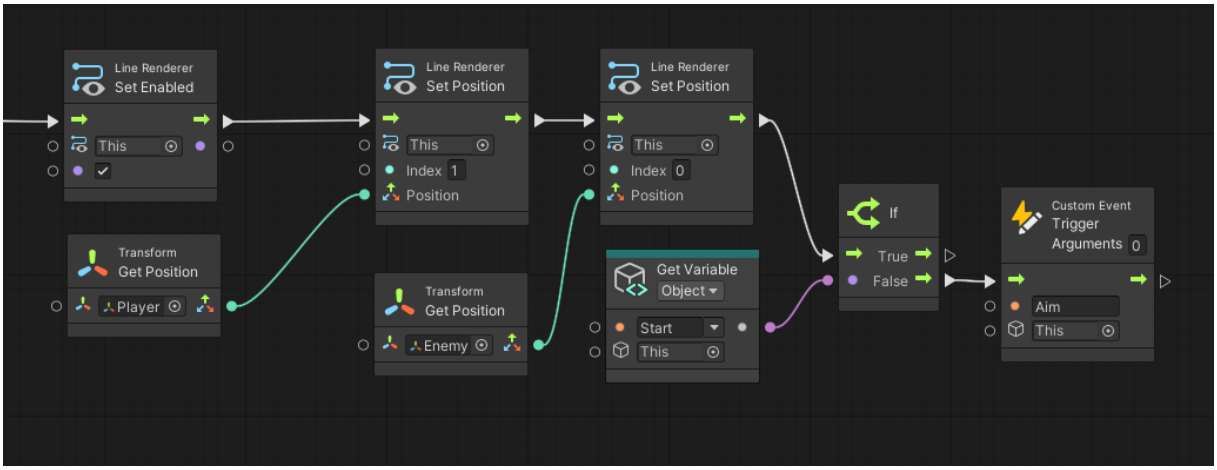
No ako je igrač u dometu, onda se omogući komponenta, postavi se vrijednost „Index 1“ komponente na poziciju igrača, a „Indeks 0“ na poziciju neprijatelja pomoću bloka „Line Renderer Set Position“. Zatim se poziva događaj koji pokreće proces indikacije. Ova komponenta radi tako što povlači liniju od indeksa 0 do indeksa 1. Nema ograničenja za količinu indeksa, ali u svrhu ovog rada su dovoljna dva. Kroz karticu „Inspector“ je postavljena debljina linije na vrlo nisko (slika 24).

Slika 24. Izgled grafa za debljinu linije komponente "Line Renderer"

1.8.4.Indikacija

Indikacija služi kao vizualni prikaz igraču da zna kada je unutar dometa neprijatelja. Prvo sporije izmjeni boje nekoliko puta, zatim brže, a brzina izmjenjivanja boja se uzima iz varijable.

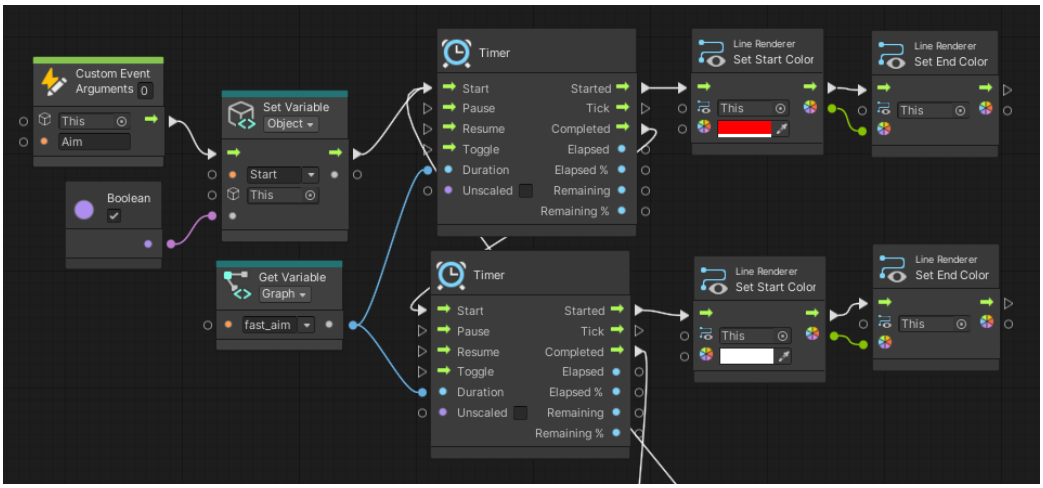
Događaj započinje tako što se prvo varijabla koja označava početak izvođenja postavi



Slika 25. Logika ako je igrač u dometu

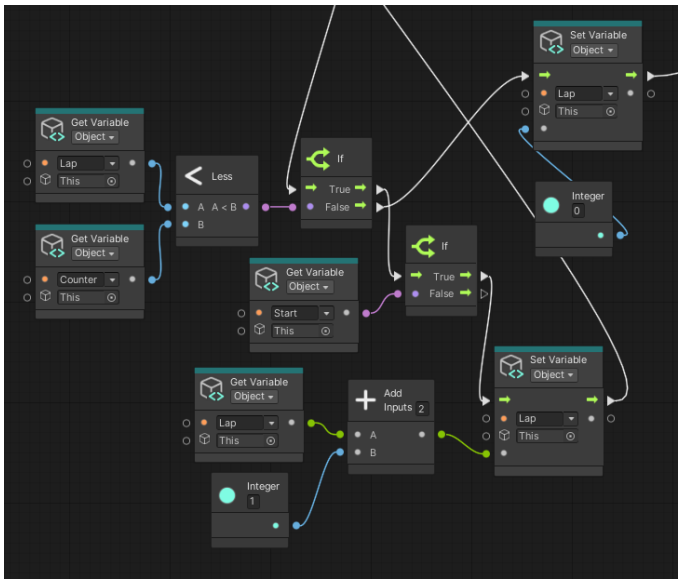
u istinu, onda se pokrene prvi brojač. Na početku brojanja, treba se postaviti početna i završna boja komponente „Line Renderer“ pomoću blokova „Set Start Color“ i „Set End Color“. Moraju se postaviti početna i završna boja jer inače prikazuje sve nijanse između dvije odabrane boje, a potrebno je prikazati samo jednu boju, u ovom slučaju crvenu. Duljina trajanja brojača je određena varijablom.

Završetkom prvog brojača se pokreće drugi brojač koji na početku postavi početnu i završnu boju u bijelu i trajanje je isto određeno istom varijablom.



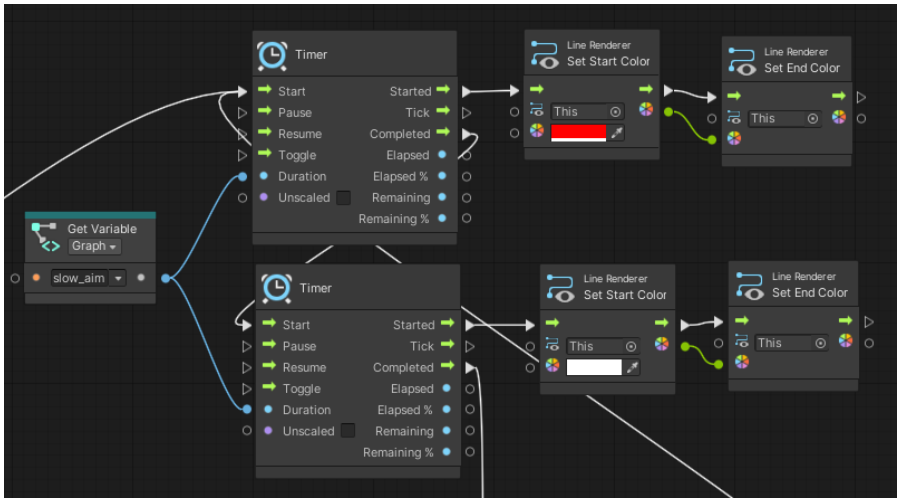
Slika 26. Logika izmjene boja između dva brojača

Razlika drugog brojača u odnosu na prvi je taj što on na završetku ispituje koliko puta su se boje izmijenile. Ako se nisu dovoljno puta izmijenile onda se provjerava da li se prekinuo proces indikacije (igrač je izašao iz dometa). Ako se nije prekinuo proces (varijabla je u stanju istina) onda se varijabla brojanja broja izmjena poveća za jedan i ponovno se pokrene prvi brojač, time proces kreće od početka.



Slika 27. Provjera broja proteklih krugova izmjene boja

Ako je završio posljednji krug, onda se brojač krugova postavlja natrag na nulu i pokreće se treći brojač koji radi na istom principu kao i prvi brojač, samo što njegova duljina trajanja se dobiva iz druge varijable. Isto vrijedi za četvrti brojač.



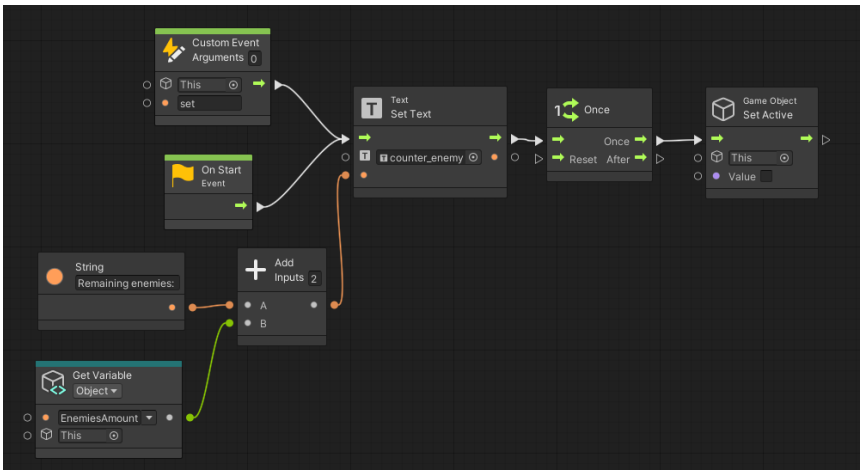
Slika 28. Izgled trećeg i četvrtog brojača

Nakon četvrtog brojača slijedi ista logika kao i nakon drugog brojača, samo što se na kraju poziva događaj „Hit“ koju ima igrač.

1.9. Korisničko sučelje

Korisničko sučelje služi za prikaz preostalih neprijatelja na mapi i nalazi se u svijetu ekrana. Kreira se desnim klikom na scenu, odabere opcija UI, zatim „Legacy“ i na kraju „Text“. Pomoću inspektora se postavlja u gornji lijevi kut ekrana, postavi se veličina teksta, pozicija na ekranu i font.

Na samome početku i pri pozivu svakog događaja „Set“ se mora ispisati tekst i broj koliko je neprijatelja ostalo. Broj se dobiva iz varijable, a prikazuje se tekst pomoću bloka „Text Set Text“ u koji ulazi objekt teksta iz korisničkog sučelja i podatak tipa „string“ teksta kojeg se prikazuje. Na samom početku se deaktivira objekt za prikaz teksta jer ga se želi prikazati tek kada igrač uzme pušku.

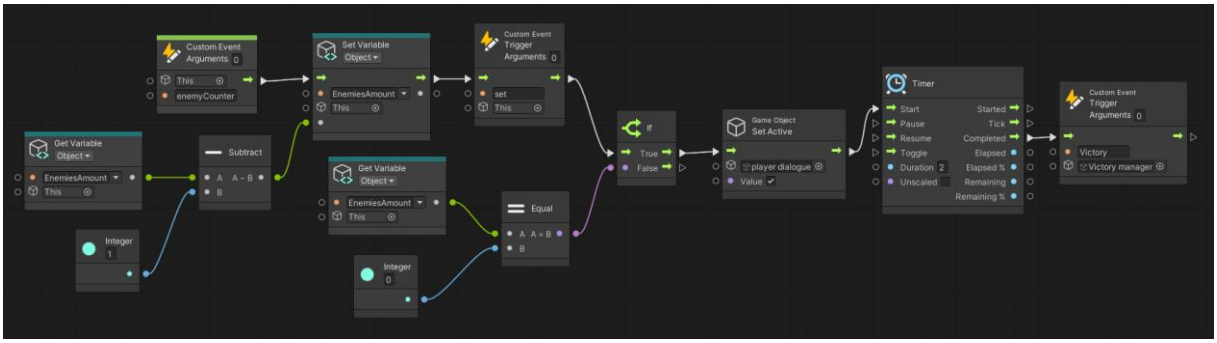


Slika 29. Početna logika korisničkog sučelja

1.9.1. Brojač preostalih neprijatelja

Događaj za brojanje neprijatelja na početku smanjuje broj preostalih neprijatelja za jedan jer se poziva kada igrač pogodi neprijatelja.

Zatim se poziva događaj „Set“ koji prikaže novi broj neprijatelja na korisničko sučelje. Ako je broj neprijatelja nula, onda se aktivira dijalog igrača, pokrene se brojač i kada završi pokrene se događaj pobjede od objekta menadžer pobjede/gubitka.



Slika 30. Logika brojača preostalih neprijatelja

4. Zaključak

Cilj izrade ovog rada je prikazati kako se može koristiti vizualno skriptiranje kao alternativni način izrade videoigara kroz praktičan rad top-down 2D shooter igrice. Također se mogu vidjeti i velike sličnosti sa klasičnim programiranjem pomoću C# koda, ali isto tako i razlike u radu.

Kod vizualnog skriptiranja je moguće predočiti kako izgledaju funkcije i kako se one mogu povezati sa drugim komponentama pomoću onoga što u njih ulazi, odnosno izlazi. Vrlo se brzo može uočiti što sve ulazi u pojedinu funkciju te što funkcija daje kao rezultat i odmah se zna što se očekuje kao rezultat. U prepoznavanju koji tip podataka gdje putuje pomažu boje. Svaka boja označava neki tip podatka ili tok funkcije, npr. bijela boja označava tok grafa, ljubičasta boja predstavlja varijablu tipa „bool“ itd. Također boje označavaju kojem tipu pripadaju koji blokovi kao što na primjer svjetlo-zelena boja označava događaje koji pokreću tok grafa. Odlična stvar je što za traženje grešaka, nije potrebno svaku varijablu ispisivati u log, već se tokom pokrenute igre vidi iznad poveznice dva bloka koja vrijednost se prenosi između njih.

Jedna od negativnih strana vizualnog skriptiranja je ta da kada se radi kompliciranija funkcionalnost, vrlo lako je tako rečeno „zapetljati“ blokove i ne razumjeti što se kako spaja ako se nije pazilo kako su oni posloženi. Odnosno, ako korisnik nema uredno posloženi graf, kasnije je teže pronaći i što je sa čime složeno. Također nema mogućnost komentiranja, već se blokovi mogu grupirati ako su stavljeni jedni blizu drugih i tako zajednički pomicati umjesto svaki put ih iznova označavati. Pronalaženje ponekih funkcija zna biti teže nego li u klasičnom kodiranju.

Ukratko, vizualno skriptiranje je zabavna nova značajka Unity-a i vrijedno ju je isprobati i s njom izrađivati prototipe mehanika i funkcionalnosti. Rekao bih da je odličan početak za početnike koji ne znaju kodirati kako bi mogli shvatiti logiku izvođenja funkcija na vizualno primamljiviji način nego li kroz klasično kodiranje koje je vrlo često odbojno apsolutnim početnicima. Ali na samome kraju je bolje i brže koristiti klasično programiranje jer uz to što je brže, ima jako veliku internetsku zajednicu koja također koristi klasično programiranje i uvijek je spremna pomoći.

5. Popis literature

[1] „Unity Technologies Files Registration Statement for Proposed Initial Public Offering“, 24. kolovoz 2020. https://www.businesswire.com/news/home/20200824005445/en/Unity-Technologies-Files-Registration-Statement-for-Proposed-Initial-Public-Offering (pristupljeno 10. rujan 2022.).

[2] „What is the best game engine: is Unity right for you?“, *GamesIndustry.biz*, 16. siječanj 2020. https://www.gamesindustry.biz/what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you (pristupljeno 10. rujan 2022.).

[3] „About Visual Scripting | Visual Scripting | 1.7.8“. https://docs.unity3d.com/Packages/com.unity.visualscripting@1.7/manual/index.html (pristupljeno 10. rujan 2022.).

6. Popis slika