Razvoj računalne 2D igre

Pufnik, Matija

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

Razvoj računalne 2D igre

Matija Pufnik

Zagreb, veljača 2019.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, datum.

Predgovor

Zahvaljujem se profesorima Visoke Škole za Primijenjeno Računarstvo koji su mi tokom studija prenijeli svoja znanja. Posebno se zahvaljujem profesoru Predragu Šuki na pomoći pri izradi završnog rada. Nakraju, zahvaljujem se i svojoj obitelji na podršci koju su mi pružili tokom cijelog školovanja.

Prilikom uvezivanja rada, umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Ovaj rad prikazuje proces razvoja računalne igre od početne ideje do gotovog proizvoda. Počinje objašnjenjem početne faze razvoja koja se odnosi na dizajniranje igre te se nastavlja prikazom procesa vezanih za izradu te igre. Objekti likova i ostali elementi koji će se rabiti u igri bit će izrađeni s pomoću alata *Adobe Photoshop*, a dizajnirana će igra biti izrađena u računalnom programu *Construct 2*. Prije prikaza izrade same igre objasnit će se važni pojmovi vezani za izradu koji se odnose na plan, događaj, objekt i sloj. Nakraju će se provesti i testiranje izrađene igre, a rezultati će biti prikazani i razjašnjeni u zadnjem poglavlju rada.

Ključne riječi: igra, dizajn, objekt, *Construct 2*, plan, događaj.

Abstract

This paper presents the process of developing a computer game from an initial idea to a finished product. It begins with an explanation of the initial phase of development which refers to a game design and it continues with showing the process of its creation. Objects of the characters and other elements that will be used in the game will be created using the *Adobe Photoshop* tool, and the designed game will be be created in *Construct 2* computer program. The important terms regarding the creation of the game itself will be explained before showing the process of its creation. Those terms refer to a plan, event, object and layer. The game testing will be performed after its creation and the results will be presented and clarified in the last chapter of this paper.

Key words: game, design, object, Construct 2, layout, event.

Sadržaj

1. Uvod

Ovaj će završni rad prikazati proces razvoja 2D računalne igre. Taj proces započinje dizajnom igre u kojemu se osmišljava koncept igre. Nastavak procesa odnosi se na izradu objekata koji će biti uporabljeni u igri, a za izradu tih objekata iskorišten je grafički računalni program *Adobe Photoshop* koji se primarno upotrebljava za obradu rasterske grafike. Nakon izrade objekata kreće izrada igre, a za izradu ove igre primijenjena je *Construct 2* arhitektura. *Construct 2* je računalni program baziran na HTML5 jeziku i služi za izradu 2D igara.

Cilj rada je od ideje za igru s pomoću računalnih programa *Adobe Photoshop* i *Construct 2* doći do finalnoga proizvoda.

Struktura rada pratit će proces izrade igre od ideje do krajnjeg rezultata. Najveći naglasak bit će stavljen na listu događaja koja je dio *Construct 2* arhitekture te na to kako s pomoću nje utjecati na ponašanje objekata.

Referencije za rad potječu sa službenih *Adobe Photoshop* i *Construct 2* internetskih stranica te besplatne literature dostupne na njima.

2. Dizajn igre

Dizajn igre je proces kojim započinje razvoj igara. U ovom će poglavlju biti opisana ideja igre, a osnovna je ideja napraviti dvodimenzionalnu (2D) igru pucanja s pogledom odozgo čija će se radnja odigravati u svemiru.

2.1. Tema igre

Ime igre koja je izrađena u praktičnom dijelu završnog rada naziva se *Soul Star Destroyer* i njezin je osnovni cilj uništiti sve protivnike koji vam se nađu na putu. Radnja igre odigrava se u budućnosti u *Soul Star* planetarnom sustavu koji je u procesu kolonizacije koju provode ljudi koji su napustili Zemlju. Pri dolasku u *Soul Star* planetarni sustav flota je upala u neprijateljsku zamku te je gotovo cijela uništena. Jedini borbeni lovac koji je ostao na raspolaganju jest najnoviji prototip kodnog imena *Blue Beast* koji se sada, koristeći se svojim naprednim naoružanjem, suprotstavlja protivničkoj floti kako bi obranio neborbene svemirske brodove od valova neprijateljskog napada, porazio glavne neprijatelje i osigurao jedan od planeta za kolonizaciju. Igra je osmišljena za jednog igrača (engl. *single-player*) koji će upravljati borbenim lovcem.

Žanr ove igre jest akcija s pucanjem (engl. *shoot 'em up*) i po tipu je svemirska pucnjava s vertikalnim pomakom (engl. *vertical scrolling space shooter*). U ovakvom tipu igre igrač gleda igru iz ptičje perspektive, a pozadina (engl. *background*) se pomiče vertikalno s vrha ekrana prema dnu, čime se stvara privid pomicanja lika kojim igrač upravlja u odnosu na pozadinu i druge objekte. Način igre tog žanra uglavnom se sastoji od pucanja na protivničke jedinice koje dolaze u velikom broju i izbjegavanje raznih projektila koje oni ispucavaju. Inspiracija za ovakav tip igre došla je od igara *DemonStar* i *Enigmata* koje imaju sličnu tematiku i sličan način igranja kao igra izrađena u praktičnom dijelu.

2.2. Likovi

Likovi u ovoj igri podijeljeni su u tipove:

- glavni lik
- neprijateljske postrojbe
- glavni neprijatelji
- završni (finalni) glavni neprijatelj.

Glavni je lik prototip borbenog lovca *Blue Beast,* i to je lik kojim upravlja igrač. Njegov je zadatak, koristeći se laserima, raketama i posebnim napadom, uništiti neprijateljske postrojbe te nakraju i glavnog neprijatelja. Pri tome se može služiti i raznim pojačanjima (engl. *power-ups*) koje može pokupiti tijekom igre.

Neprijateljske su postrojbe protivnici glavnog lika i cilj im je uništiti ga ili proći pokraj njega kako bi ugrozili svemirske brodove koje glavni lik štiti. Svaki tip neprijateljske postrojbe ima svoje posebno obilježje kojim koristi za ostvarenje svojeg cilja. Neprijateljske postrojbe dijele se na osnovne, koje se pojavljuju u svakoj razini, i napredne, koje su nešto jače od osnovnih i pojavljuju se nakon prve razine.

Glavni neprijatelj dolazi nakraju svake razine i cilj mu je isključivo uništenje glavnog lika. Specifičan je po tome što posjeduje naprednija oružja i, s obzirom na to da je najveća prijetnja nekoj razini, mnogo ga je teže uništiti nego obične neprijateljske postrojbe. Pojavljuje se nakon što igrač ispuni određeni uvjet.

Finalni glavni neprijatelj pojavljuje se nakraju posljednje razine. On je glavni neprijatelj kojeg je najteže uništiti i čije je napade najteže izbjeći.

2.3. Način igranja i rada igre

Ovaj dio poglavlja odnosi se na način igranja (engl. *gameplay*) i mehaniku igre i opisivat će ključni cilj igrača u igri, napredak kroz igru te interakciju igrača s glavnim likom i ostatkom igre.

2.3.1. Cilj igre

Cilj je igre uništiti neprijateljske postrojbe i onemogućiti im prolazak. Uništavanjem tih jedinica igrač se približava ispunjenju uvjeta da se pojavi glavni neprijatelj, a njegovom se

pojavom igraču otvara mogućnost da ga uništi te tako završi razinu. Glavni je cilj je doći do zadnje razine i uništiti završnoga glavnog neprijatelja koji nosi ime planetarnog sustava u kojem se igra događa. Time igrač dobiva titulu *Soul Star Destroyer* i završava igru.

2.3.2. Napredak kroz igru

Ova se igra sastoji od triju razina. Da bi se razina završila, potrebno je ispuniti uvjet da se glavni neprijatelj pojavi, a taj je uvjet moguće ispuniti ako je uništeno dovoljno neprijatelja ili ako je sakupljeno dovoljno bodova. Uništenjem postrojbe glavni neprijatelj igrač završava razinu i odlazi na sljedeću.

Svaka sljedeća razina bit će teža od prethodne, a tomu će pridonijeti:

- nove neprijateljske postrojbe
- više neprijatelja u istome vremenskom intervalu
- rjeđa pojava predmeta za regeneraciju zdravlja (engl. *health*)
- glavni neprijatelj kojeg će biti teže ubiti i čije će napade biti teže izbjeći.

Ako je glavni lik uništen ili je igrač dopustio da više od dopuštenog broja neprijateljskih postrojbi prođe pokraj njega, igrač gubi i, ako nije spremio igru, mora početi od početka neovisno o tome na kojoj je razini bio.

2.3.3. Mehanike igre

Mehanike igre odnose se na skup su pravila koja određuju način rada igre i dizajnirane su kako bi igrač s njima imao interakciju[1]. Glavni lik kao najvažnije mehanike posjeduje mogućnost kretanja i pucanja. To mu omogućuje da uništava neprijateljske postrojbe i pritom izbjegava njihove projektile. Mehanike pucanja moguće je podijeliti na:

- lasere
- rakete
- posebni napad.

Laserima je moguće pucati u neograničenom broju, a rakete su ograničene i nakon što se potroše potrebno je sakupiti predmet koji sadržava određen broj raketa. Rakete mogu biti obične ili navođene, koje su posebne po tome što će uvijek pratiti neprijatelja dok on ne bude uništen. Posebni se napad odnosi na bombu koja se nakon što je igrač ispuca detonira te

[1] https://www.gamedesigning.org/learn/basic-game-mechanics/

uništava sve neprijateljske postrojbe koje trenutačno vidi na ekranu ili u slučaju borbe s glavnim neprijateljem uzrokuje veliku količinu štete. Nakon što igrač iskoristi taj napad morat će pričekati određeno vrijeme prije negoli mu ponovno bude omogućeno da ga iskoristi. Sakupljanje pojačanja također je jedna od mehanika s pomoću koje je, osim toga što je moguće sakupljati rakete i regenerirati zdravlje, moguće i pojačati glavni napad, odnosno laser.

Predmeti s pomoću kojih je moguće pojačati laser:

- ubrzano pucanje (engl. *rapid fire*)
- dvostruko pucanje
- veća šteta (engl. *damage up*).

Ova su tri predmeta posebna po tome što njihov učinak traje samo određeno vrijeme, ali imaju mogućnost da djeluju zajedno ako je jedan od njih skupljen u vrijeme trajanja drugog. Svaki od predmeta za pojačanje može se nasumično pojaviti ili se može pojaviti nakon uništenja protivničke postrojbe.

Za pojavu neprijatelja odgovorna je mehanika nasumičnog kreiranja protivnika po x-osi te u raznim vremenskim intervalima i ima velik utjecaj na težinu igre. Neprijatelji koje igrač uništi, osim mogućim pojačanjima, nagrađuju igrača i bodovima s pomoću mehanike „rezultat“. Nakon svake razine, osim posljednje, igraču će s pomoću mehanike „nadogradi“ biti omogućeno da poboljša neku od specifikacija lovca kojim upravlja. Zadnja i neizostavna mehanika jest pauza. Njome igrač može zaustaviti igru te je, kad bude u mogućnosti, nastaviti tamo gdje je stao.

2.4. Sučelje igre

- statusna traka
- izbornici
- kamera

Statusna je traka element koji omogućuje igraču da prati trenutačno stanje svojeg lika i stanje u igri. Preko statusne trake igrač može pratiti:

- stanje zdravlja
- broj običnih i navođenih raketa
- vrijeme potrebno da mu se ponovno omogući posebni napad (engl. *cooldown*)
- koliko neprijatelja još smije proći pokraj glavnog lika

- trenutačni rezultat.

Koristeći se početnim izbornikom, igraču će biti omogućeno da započne igru. Ako odluči započeti igru, otvorit će mu se mogućnost biranja između lakše i teže verzije igre. Preko početnog izbornika igrač će također moći i pogledati kontrole, pogledati trenutačno najbolji rezultat te učitati igru ako ju je prije toga spremio.

Podizbornici će se pojavljivati nakon svake uspješno i neuspješno završene razine. Preko podizbornika koji se pojavljuje nakon uspješno završene razine igrač će moći nastaviti na iduću razinu, pogledati trenutačni rezultat i spremiti igru. Uz prije navedene mogućnosti, taj podizbornik također sadržava i sustav za prije spomenute nadogradnje. Preko tog sustava igraću će biti omogućeno da bira koju od navedenih specifikacija želi nadograditi prije nego nastavi s igrom. Ako igrač izgubi, pojavit će se podizbornik koji će mu omogućiti da ponovno započne igru ili da iz nje izađe. S obzirom na to da vertikalni pomak pozadine stvara privid pomaka glavnog lika s obzirom na pozadinu, kamera ne prati neki određeni objekt i ne pomiče se.

3. Istraživanje igara sličnog dizajna

Istraživanje sličnih igara proces je koji se može obavljati uz dizajn igre nakon razrade njezine teme. Korisno je jer se s pomoću njega mogu dobiti ideje o tome što bi sve jedna igra žanra koji je odabran za izradu trebala sadržavati i kako bi se one potencijalno mogle ukomponirati u vlastitu igru.

Istraživanje je provedeno igranjem računalnih igara *DemonStar* i *Enigmata* koje pripadaju svemirskim pucnjavama s vertikalnim pomakom. Cilj istraživanja je bio tijekom igranja spomenutih igara napraviti popis svih njihovih mehanika i obilježja koje bi se potencijalno moglo dodati i razraditi u dizajnu vlastite igre. Iako dvije navedene igre dijele mnogo zajedničkih mehanika i obilježja, one se ipak razlikuju u načinima izvedbe. Glavne su razlike u:

- načinu stvaranja neprijatelja
- načinu ponašanja neprijatelja
- posebnim napadima
- sustavu bodovanja
- sadržaju između razina (nadogradnje).

Tijekom igranja tih dviju igara praćeni su svi aspekti u kojima se te dvije igre razlikuju i na temelju toga je napravljena procjena koja bi se od izvedbi navedenih obilježja i mehanika bolje uklapala u dizajn igre.

Na temelju te procjene odlučeno je da će se način stvaranja neprijatelja i sadržaj između razina bazirati na onom koji se primjenjuje u igri *Enigmata*. To se odnosi na nasumično stvaranje neprijatelja umjesto valova neprijatelja i mogućnost nadogradnje glavnog lika između razina. Način ponašanja neprijatelja i sustav bodovanja bit će pak sličniji onima u igri *DemonStar*. Neprijatelji se neće nužno samo kretati vertikalno i nastojati što prije proći, kao što je to slučaj u igri *Enigmata*, nego će se neki nastojati malo duže zadržati na ekranu i boriti se s glavnim likom. Sustav bodovanja odnosit će se na trenutačni rezultat i na najbolji rezultat, tako da će se nadogradnje rješavati bez zlata, a igrač će i dalje imati priliku pokušati ostvariti što bolji rezultat. Posebni napad dijelit će način rada kao i u navedenim igrama, ali će imati nešto jednostavniji način primjene.

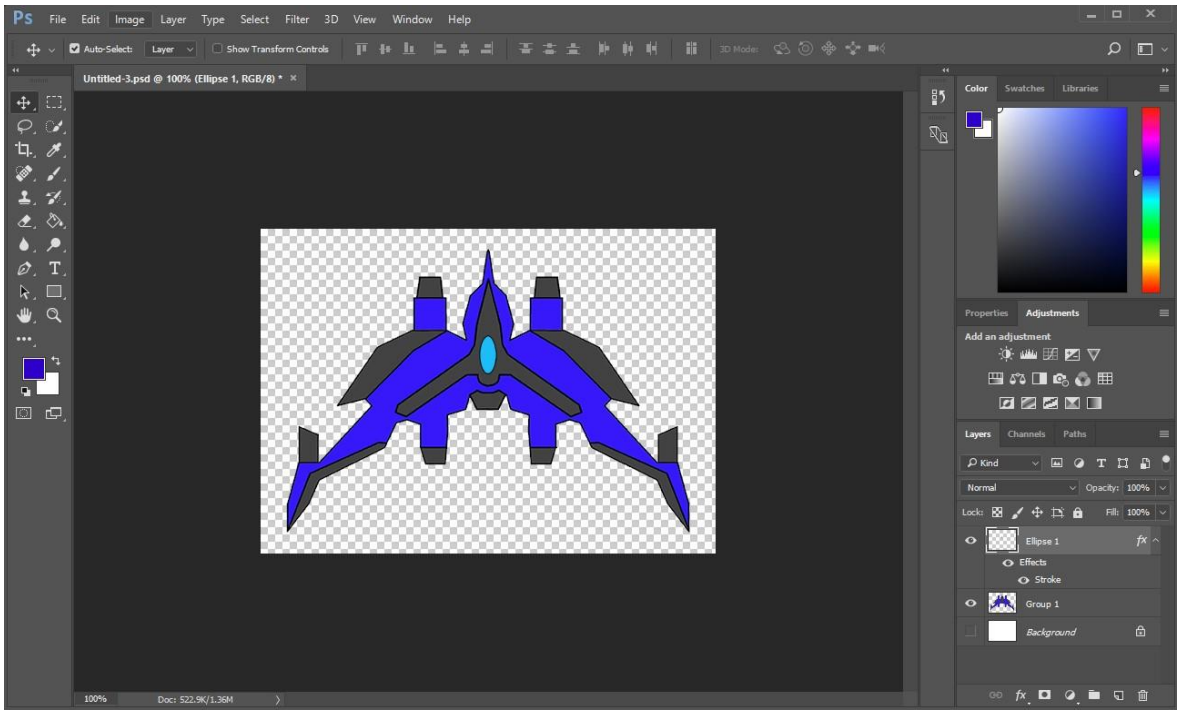
Glavno obilježje po kojemu se dizajnirana igra razlikuje od igara na kojima je provedeno istraživanje jest u tome da igrač može izgubiti a da i ne uništi glavnog lika ako propusti

previše neprijatelja. Kod većine igara tog žanra neuspješno se uništavanje neprijatelja baš i ne kažnjava pa je zbog toga to obilježje dodano u igru *Soul Star Destroyer* kako bi se ona ipak po nečemu razlikovala od ostalih sličnih igara. Ideja je s pomoću tog obilježja veću važnost staviti na svakoga pojedinog neprijatelja kako bi svatko od njih zaista bio prijetnja, a ne samo izvor dodatnih bodova. Pri izvedbama stvaranja neprijatelja te njihovih ponašanja u igrama *DemonStar* i *Enigmata* često se događa da se u jednome trenutku pojavi broj neprijatelja koji bi bilo nemoguće potpuno zaustaviti ili da neprijatelji djeluju tako da pokušavaju što prije proći pokraj igrača kako bi mu uskratili bodove. Takve načine izvedbe tijekom izrade igre bit će potrebno što je više moguće smanjiti kako bi igrač u većini slučajeva bio u mogućnosti zaustaviti neprijatelje s pomoću dostupnih sredstava.

4. Izrada objekata

Izrada objekata odnosi se na proces izrade likova, neprijatelja, pozadina i ostalih objekata koji se uporabljuju pri izradi kroz drugo poglavlje dizajnirane igre. Ti su objekti izrađeni u računalnoj aplikaciji *Adobe Photoshop* koja služi za digitalnu obradu slike.

Format u kojemu je izvedena svaka rasterska grafika za ovu igru jest PNG. Razlog korištenja PNG formatom jest mogućnost uporabe transparentne pozadine i podržani su djelomično transparentni pikseli.

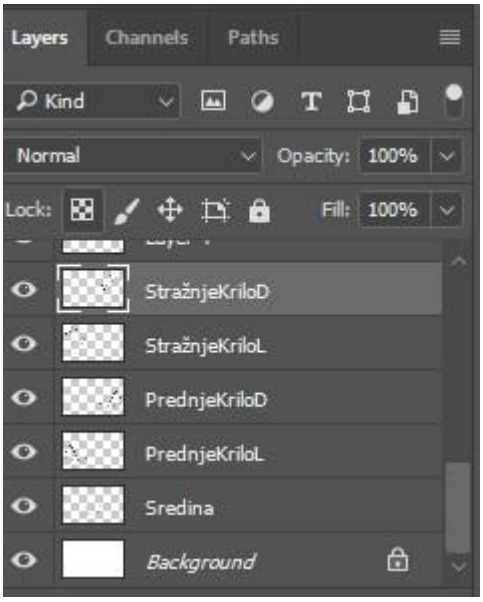


Slika 4.1 Korisničko sučelje alata *Photoshop*

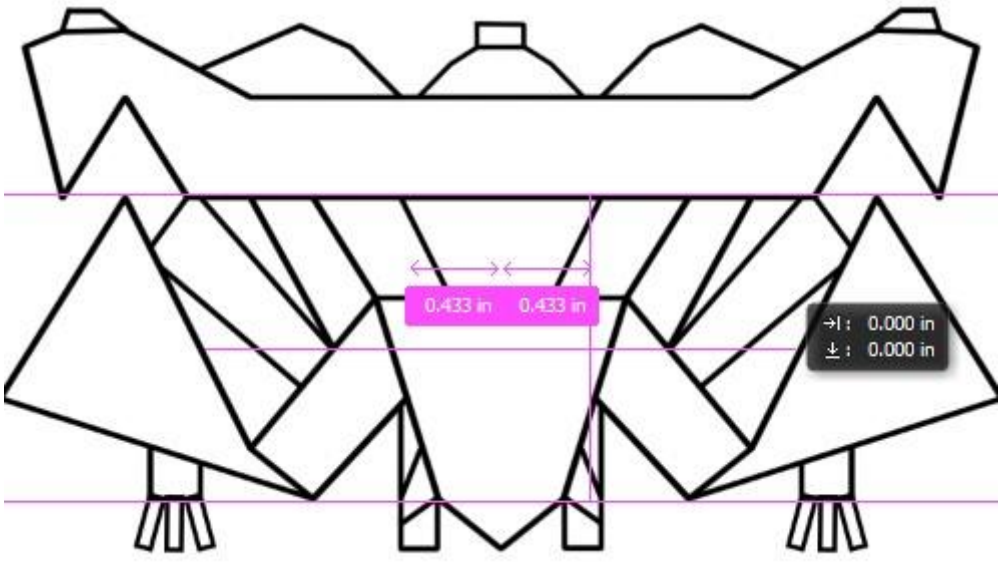
4.1. Izrada likova

Likovi se odnose na rastersku grafiku koja je dizajnirana tako da bude dio neke veće scene, što je u ovom slučaju objekt u 2D računalnoj igri. Svi objekti likova crtani su primjenom alata *brush*. Proces crtanja likova jest takav da se ne crta cijeli objekt odjednom na jednom sloju, nego se objekt podijeli na više dijelova koji se međusobno raspoređuju po slojevima. Kako bi se osigurala simetričnost objekta, dovoljno je samo nacrtati jednu polovicu svakog dijela objekta na zasebnom sloju, napraviti kopiju svakog od tih slojeva i te kopije potom okrenuti horizontalno. Sve dijelove objekta na kopiranim slojevima potrebno je poslagati

tako da budu simetrični s obzirom na originalnu polovicu. To *Photoshop* omogućuje asistencijom kao na slici (Slika 4.3) koju pruža pri preslagivanju objekata, koja je također korisna i pri početnom slaganju.

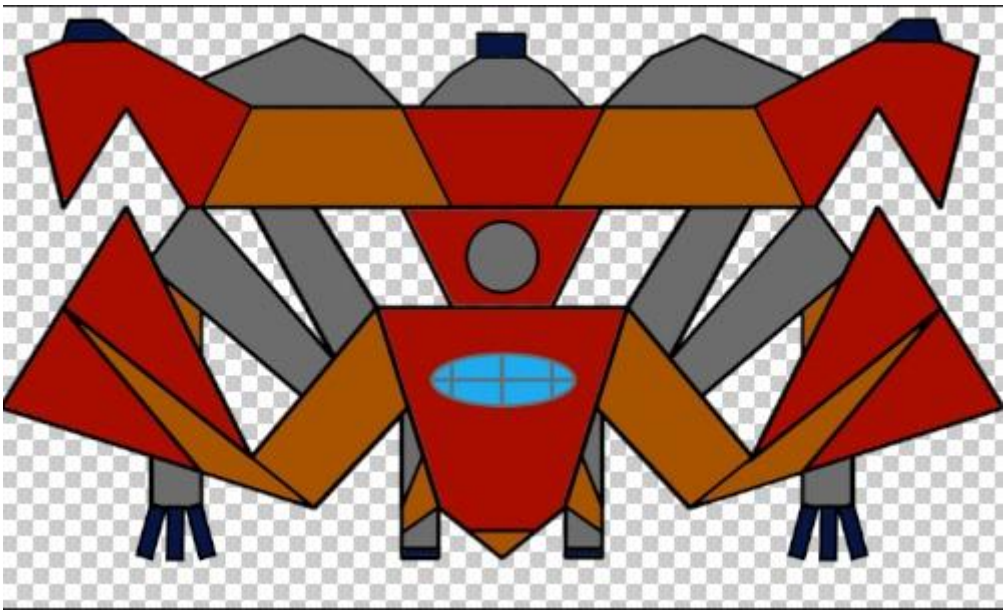


Slika 4.2 Podjela dijelova objekta po slojevima



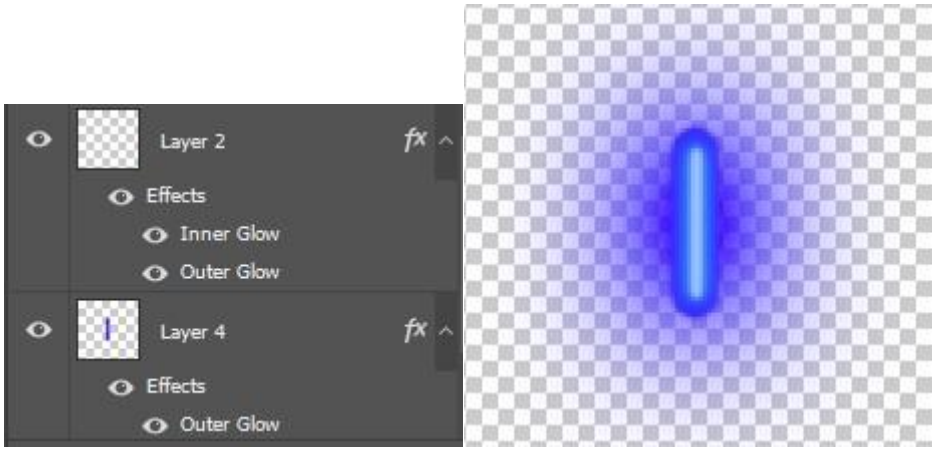
Slika 4.3 Objekt *završni (finalni) glavni neprijatelj* bez dodanih detalja

Kako bi se objekt spremio za bojenje, potrebno je sve slojeve koji su bili uporabljeni pri crtanju grupirati u jedan sloj. Na taj se način omogućuje da se svaki dio koji se nalazi između nacrtanih linija posebno ispuni željenom bojom s pomoću alata *paint bucket*.



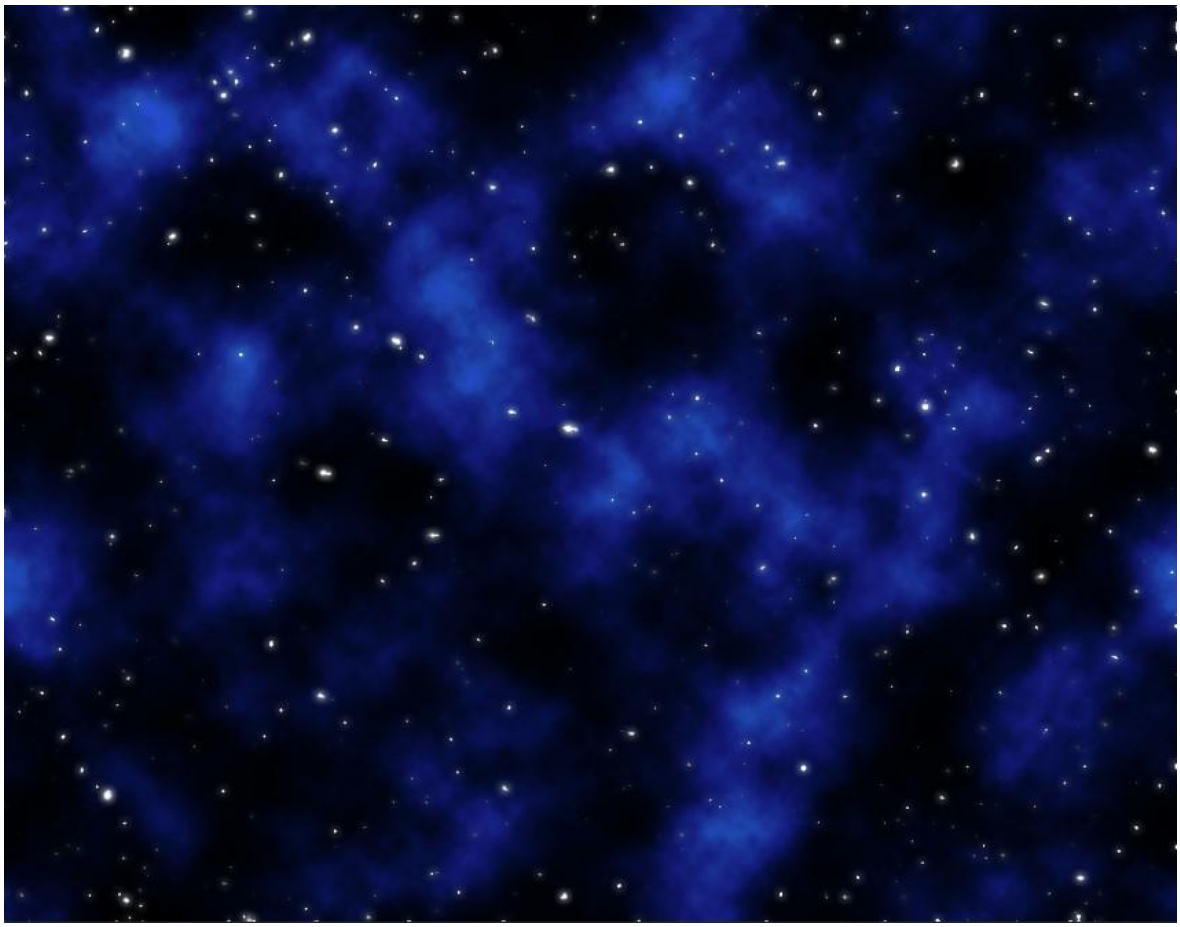
Slika 4.4 Završna verzija objekta *finalni glavni neprijatelj*

Na svaki je sloj prema potrebi moguće primijeniti razne efekte. Pri izradi objekata elipsama je s pomoću efekta *stroke* dodan obrub kako bi izgledale u s kladu s ostatkom objekta. Za izradu lasera i mlazova primijenjeni su efekti *inner glow* i *outer glow* zbog kojih objekt izgleda kao da svijetli.



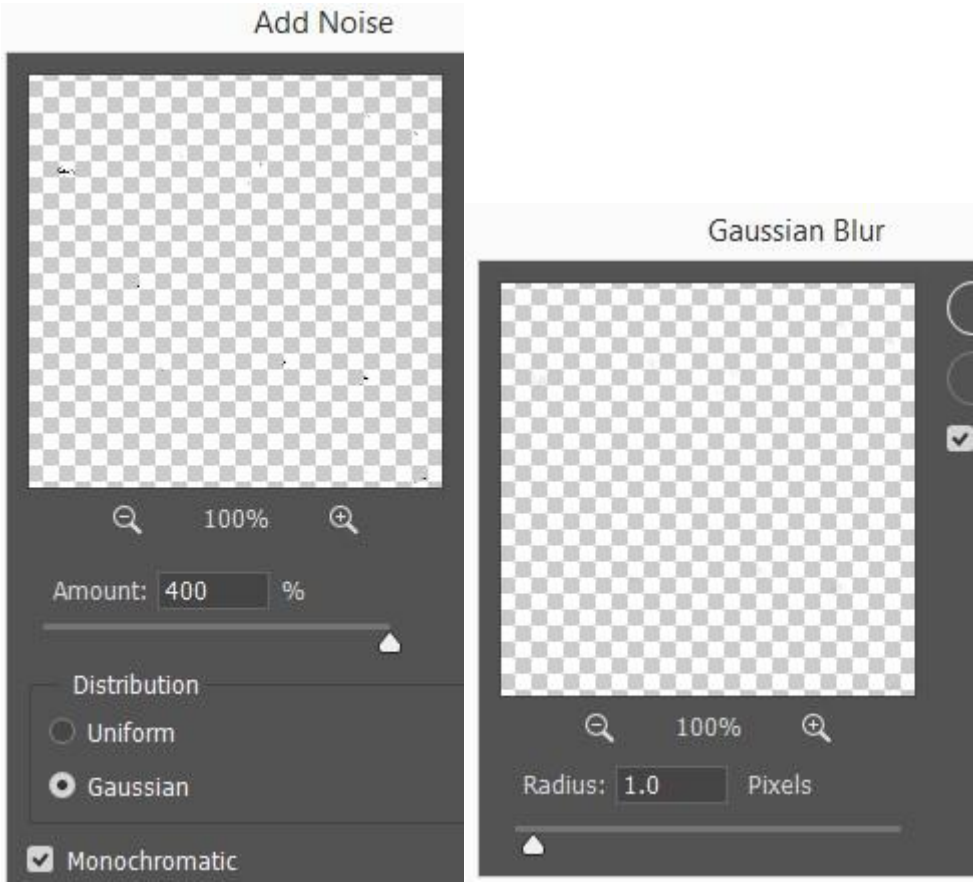
Slika 4.5 Izrada objekta *laser*

4.2. Izrada pozadine



Slika 4.6 Svemirska pozadina za igru

Kako bi se izradila svemirska pozadina kakva je prikazana na slici (Slika 4.6), potrebno je koristiti se filtrima. Izrada započinje dodavanjem zvijezda tako što se dodaje novi sloj kojem se dodaje filtar *noise* po postavkama sa slike (Slika 4.7) i dodavanjem kopije toga sloja. Prvom sloju koji će služiti za manje zvijezde zatim se dodaje filtar *gaussian blur* po postavkama sa slike (Slika 4.7). Nakraju je još potrebno u padajućem izborniku slika pod prilagodbe (engl. *adjustments*) odabrati alat *levels* koji omogućuje rastezanje i pomicanje razina svjetlosti histograma. U tom je alatu potrebno mijenjati *input* sve dok se ne dobije željeni izgled zvijezda. Isti se postupak ponavlja i za velike zvijezde kojima je potrebno povećati *gaussian blur* i po potrebi korigirati *input* u alatu *levels*.



Slika 4.7 Prozori za dodavanje efekata *noise* i *gaussian blur*

Gotove zvijezde potrebno je izrezati koristeći se alatom *magic wand* i dodati im efekt *outer glow*. Nakon zvijezda potrebno je dodati i zvjezdanu prašinu s pomoću efekta *render clouds* koji se dodaje na sloj između pozadine i zvijezda. Sloj na koji je dodan efekt *render clouds* kopira se i jednom se sloju se doda *blend mode multiply* kako bi efekt bio transparentan, a drugom *color dodge* kako bi se malo povećao sjaj zvjezdane prašine. Boju zvjezdane prašine određuje prvi sloj, odnosno pozadina, a tamni dijelovi dolaze od efekta *render clouds*.

5. Primjena alata *Construct 2* pri izradi igre

Ovo je poglavlje podijeljeno na dva dijela. Prvi dio poglavlja predstavit će alat uporabljen za izradu igre, a drugi će se dio odnositi na njegovu primjenu pri izradi igre. Igra, čija se izrada prati, izrađena je onako kako je dizajnirana, što je prikazano u drugom poglavlju ovog rada, koristeći se objektima opisanima u četvrtom poglavlju i objektima koji su izrađeni u alatu *Construct 2*.

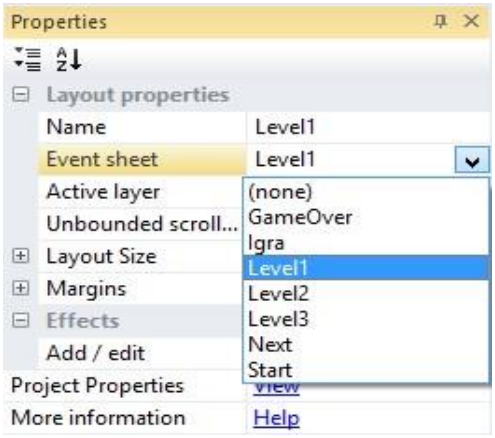
5.1. Construct 2

Construct 2 je alat baziran na HTML5[2] jeziku koji je razvila tvrtka Scirra. Alat je razvijen i dizajniran posebno za izradu 2D igara i sa svrhom da omogući njihovu izradu korisnicima koji nemaju znanje nekog programskog jezika.[3] To je omogućeno jednostavnim dodavanjem novih objekata u plan (engl. *layout*) te dodavanjem postojećih ponašanja (engl. *behaviour*) tim objektima i događajima (engl. *events*), a sve to služi kao zamjena za korištenje programskim jezikom.

5.1.1. Plan

Plan služi kao prikaz neke razine ili izbornika i sastoji se od slojeva i objekata. Svaki se plan koristi listom događaja (engl. *event sheet*) koja određuje ponašanje svih objekata koje taj plan sadržava. Pri kreiranju plana alat će ponuditi opcije da kreira listu događaja za taj plan, ali je moguće izabrati i opciju da se lista događaja ne kreira i naknadno kao listu događaja postaviti neku od postojećih listi.

[2] Prezentacijski jezik za strukturiranje na *web*-stranicama (engl. *HyperText Markup Language*)
[3] https://www.scirra.com/construct2



Slika 5.1 Obilježja plana

Plan ima i mogućnost primjene efekata koji djeluju na sve objekte koji se na njemu nalaze.[4] Svaki efekt ima neke parametre po kojima radi i kojima je iznos unaprijed definiran. Promjenom tih parametara moguće je povećati ili smanjiti efekt koji djeluje na originalan prikaz. Za prikaz efekata potrebna je WebGL[5] podrška.

Slika 5.2 Prikaz korištenja efektima

5.1.2. Slojevi

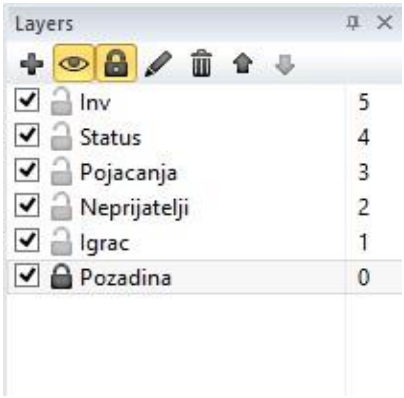
Slojevi su sastavni dio plana. Na slojeve se može gledati kao na prozirne staklene ploče na koje su smješteni različiti objekti[6]. Rabe se za prikaz objekata ispred ili iza drugog objekta (Slika 5.3). Najčešći redoslijed počinje od sučelja igre kao najvišega sloja, a nastavlja se na

[4] https://www.scirra.com/manual/67/layouts
[5] Biblioteka web grafika (engl. *Web Graphics Library*)
[6] https://www.scirra.com/manual/85/layers

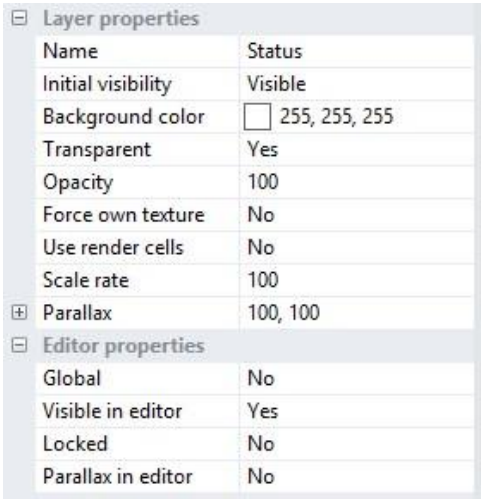
slojeve na kojima se nalazi igrač i ostali objekti s kojima ima interakciju, a na samome je dnu pozadina.

Slika 5.3 Prikaz odnosa između objekata ovisno o sloju na koji su smješteni

S pomoću trake sa slojevima moguće je dodavati nove slojeve, preimenovati ih, mijenjati im raspored prikazivanja, brisati ih i zaključati. Zaključavanje sloja onemogućuje daljnju promjenu objekata na tom sloju i korisno je služiti se ovom opcijom za sloj pozadine kako se ona ne bi slučajno pomaknula ili na neki drugi način izmijenila tijekom izrade igre. Osim opcija koje su ponuđene na traci sa slojevima, omogućena je i uporaba *drag and drop* slojeva u svrhu pozicioniranja i moguće je mijenjati parametre obilježja pojedinoga sloja koristeći se traku s obilježjima sloja.



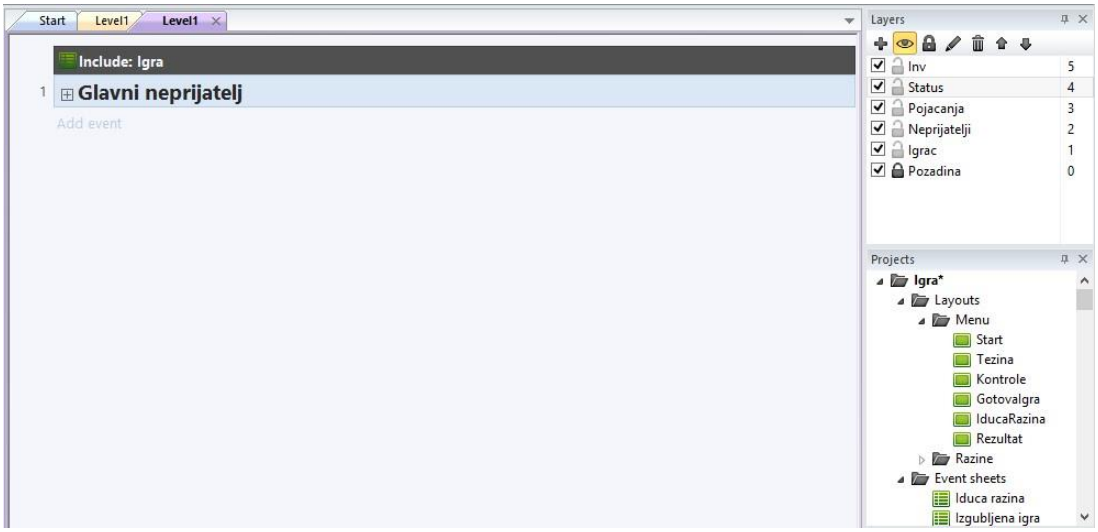
Slika 5.4 Traka sa slojevima



Slika 5.5 Obilježja sloja

5.1.3. Lista događaja

Lista događaja jest lista u koju se dodaju događaji. Svaki plan ima sebi pridruženu listu događaja koja definira kako će taj plan raditi. Pri dodjeli liste događaja pojedinom planu korisno je rabiti jednu listu događaja za više planova kako bi se izbjeglo ponavljanje događaja, a to je moguće postići dodavanjem liste događaja drugim listama događaja (Slika 5.6).[7] Na ovaj se način izbjegava potreba da se primjerice dvaput pišu događaji koji omogućuju kretanja.

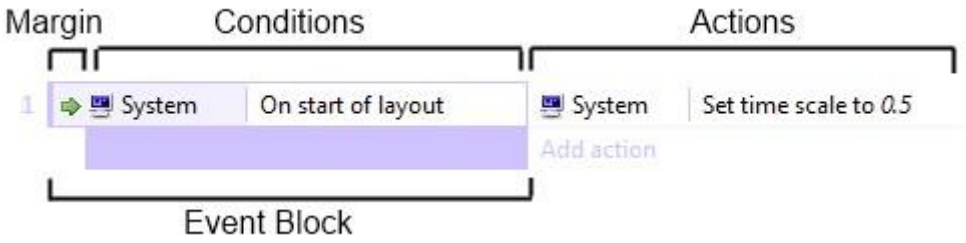


Slika 5.6 Prikaz liste događaja

[7] https://www.scirra.com/manual/121/event-sheets

5.1.4. Događaji

Događaji su glavno obilježje alata *Construct 2* i omogućuju da se rad igre definira s pomoću logičkih blokova i tako služe kao zamjena za uporabu nekog programskog jezika. Ti logički blokovi zajedno se nazivaju događajima, a više logičkih blokova ukupno čine listu događaja. Događaji su dizajnirani tako da korisnicima pruže osnovne alate koji su potrebni za izradu sofisticirane igre.[8] Osnovni koncept događaja jest da uvjeti (engl. *conditions*) filtriraju instancije koje odgovaraju tim uvjetima te se zatim akcije pokreću samo za te instancije.



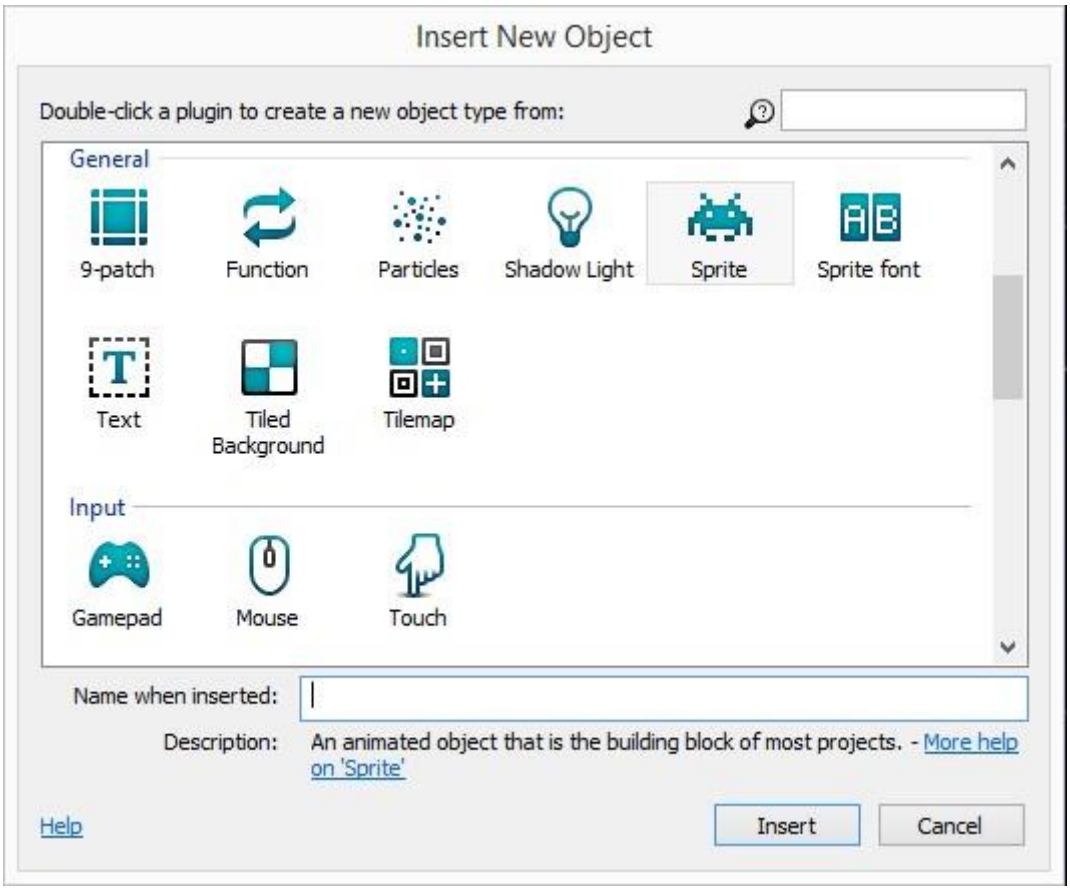
Slika 5.7 Primjer jednostavnog događaja

U pravilu, događaji se sastoje od uvjeta koji mora biti ispunjen i akcije koja se pokreće ako je uvjet zadovoljen. Moguće je i dodavanje poddogađaja (engl. *sub-event*) koji se izvršava nakon što se izvrši početni događaj. Redoslijed izvođenja događaja kreće od vrha liste događaja prema dnu.

5.1.5. Objekti

Objekti su gotovo sve stvari koje igrač vidi u *Construct 2* igri i izvode najviše korisnog posla u projektu. Pri dodavanju novog objekta ponuđeno je mnogo priključaka (engl. *plugin)*[9], od kojih se po potrebi izabire jedan i time se kreira novi tip objekta (engl. *object type*).

[8] https://www.scirra.com/manual/121/events
[9] *Plugin* je vrsta softvera koja se uporabljuje da bi se programu pridodalo neko specifično obilježje.



Slika 5.8 Prozor za dodavanje objekata

Priključci definiraju vrstu objekta i mogu se podijeliti na:

- vizualne (*sprite*)
- skrivene (polje)
- projektne (tipkovnica).

Vizualni se priključci pojavljuju unutar plana i prikažu nešto na ekranu, skriveni su stavljeni u određeni plan, ali se ne prikazuju na ekranu, a projektne (engl. *project-wide*) koji se odnose na cijeli projekt moguće je dodati samo jednom.[10]

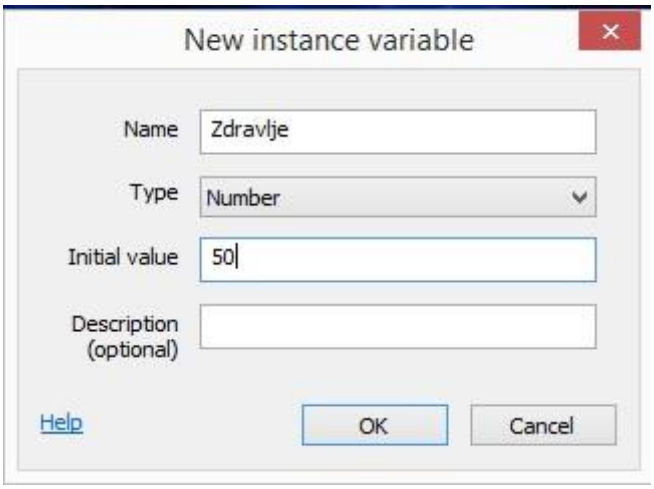
Svaki objekt posjeduje obilježje tip objekta koje definira klasu nekog objekta. Primjerice, objekti *Igrač* i *Neprijatelj1* dva su različita tipa objekta *sprite* priključaka. Većina tipova objekata može imati više instancija koje predočuju objekte koje vidimo u igri. To se odnosi na instancije koje imaju svoj kut, poziciju i veličinu unutar plana. Instancije je moguće kreirati s pomoću liste događaja ili ih poslagati u planu ručno. Svaki tip objekta može imati i neka određena obilježja. Obilježja nužna za izradu igara odnose se na:

[10] https://www.scirra.com/manual/69/plugins

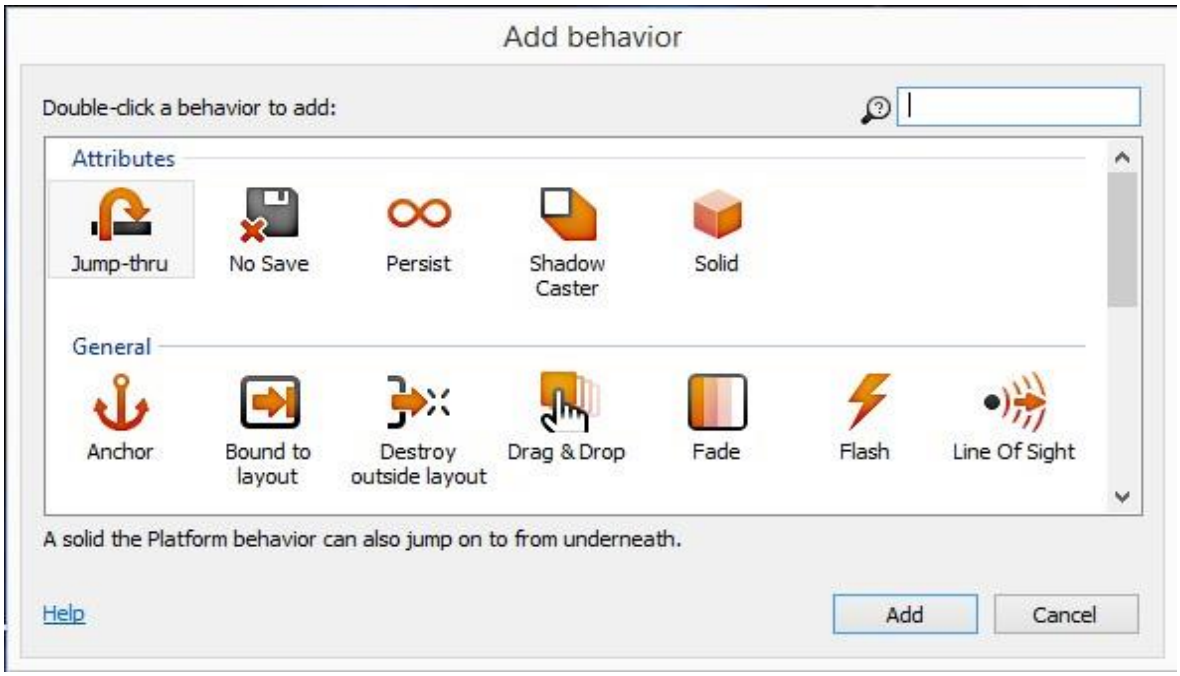
- varijablu instancije (engl. *instance variables*)
- ponašanja
- spremnici (engl. *containers*).

Varijable instancije dodaju se tipovima objekata i spremaju brojeve ili tekst za svaku instanciju posebno. Ponašanja se također dodaju tipovima objekata i sadržavaju već gotovu funkcionalnost. Njihova namjena nije zamijeniti događaje, nego samo olakšati i uštedjeti vrijeme pri dodavanju funkcionalnosti nekom objektu.

Spremnici su napredno obilježje koje služi kako bi se u događajima mogla prikupiti grupa instancija u isto vrijeme.



Slika 5.9 Prozor za postavljanje vrijednosti varijable instancije

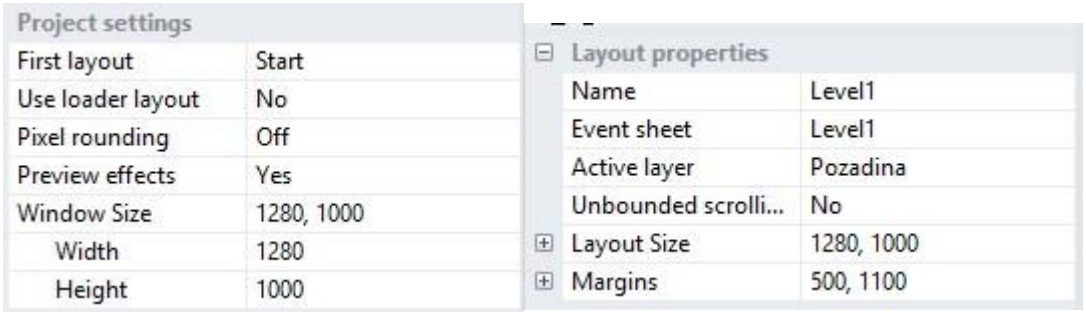


Slika 5.10 Prozor za dodavanje ponašanja

5.2. Izrada igre

U prethodnom dijelu poglavlja opisan je alat *Construct 2*. Ovaj će dio poglavlja pratiti primjenu tog alata u svrhu izrade igre i s pomoću primjera detaljnije objasniti i prikazati gore spomenute pojmove.

Izrada igre počinje od stvaranja novoga projekta. Pri stvaranju projekta moguće je odabrati prazan projekt, projekt s nekim već prije definiranim vrijednostima ili predložak. U slučaju ove igre odabran je prazan projekt. Tom je projektu potrebno postaviti parametre veličine ekrana. Veličina ekrana odabrana je kako bi odgovarala tipu igre, što je u ovom slučaju svemirska pucnjava i, s obzirom na to da je riječ o takvoj igri, veličina plana postavljena je na istu vrijednost (Slika 5.11). Razlog je u tome što će za pomak između pozadine i igrača biti odgovorna sama pozadina pa nema potrebe da plan bude veći od veličine ekrana.



Slika 5.11 Postavke projekta i obilježja plana

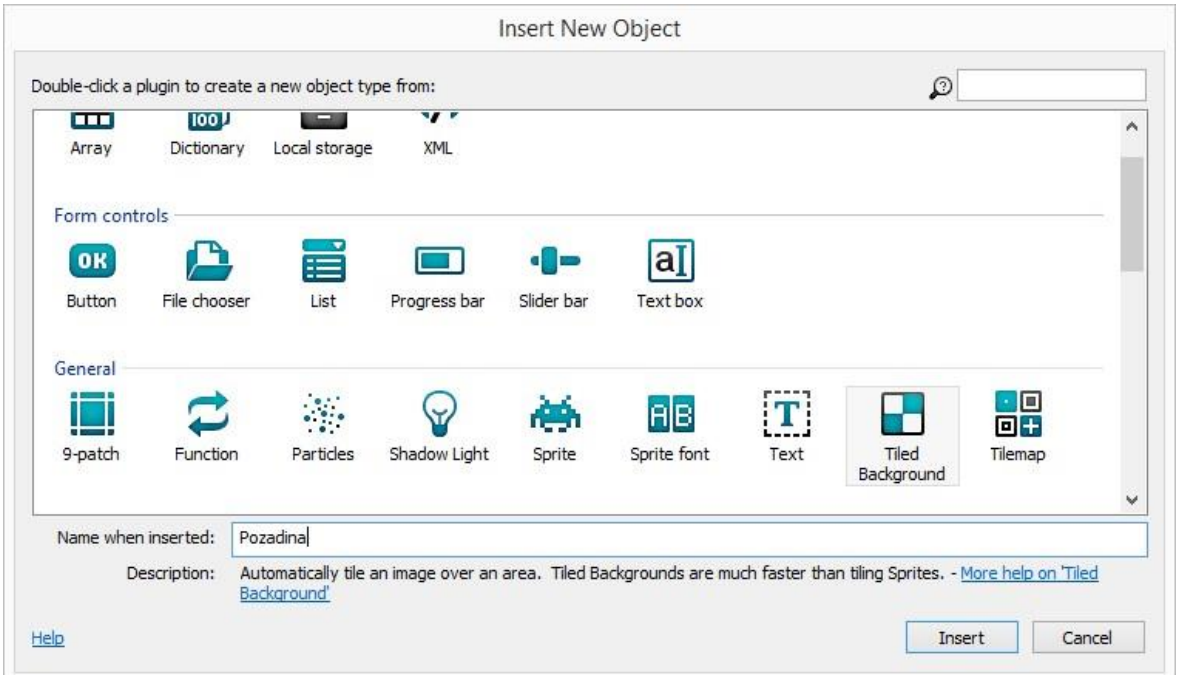
Prije nego što se krene s dodavanjem objekata poželjno je odmah definirati i dodati sve slojeve (Slika 5.5), a oni su u ovom slučaju:

- pozadina
- neprijatelji
- igrač
- pojačanja
- nevidljivi objekti
- sučelje (statusna traka).

Dodavanje slojeva prije dodjele objekata uvelike pomaže pri snalaženju, štedi vrijeme jer nije poslije potrebno za svaki tip objekta ili instanciju postavljati sloj u traci s obilježjima tog objekta i smanjuje mogućnost da se neki objekt stavi na pogrešan sloj. Sada kad su svi slojevi definirani i dodani, mogu se početi dodavati objekti.

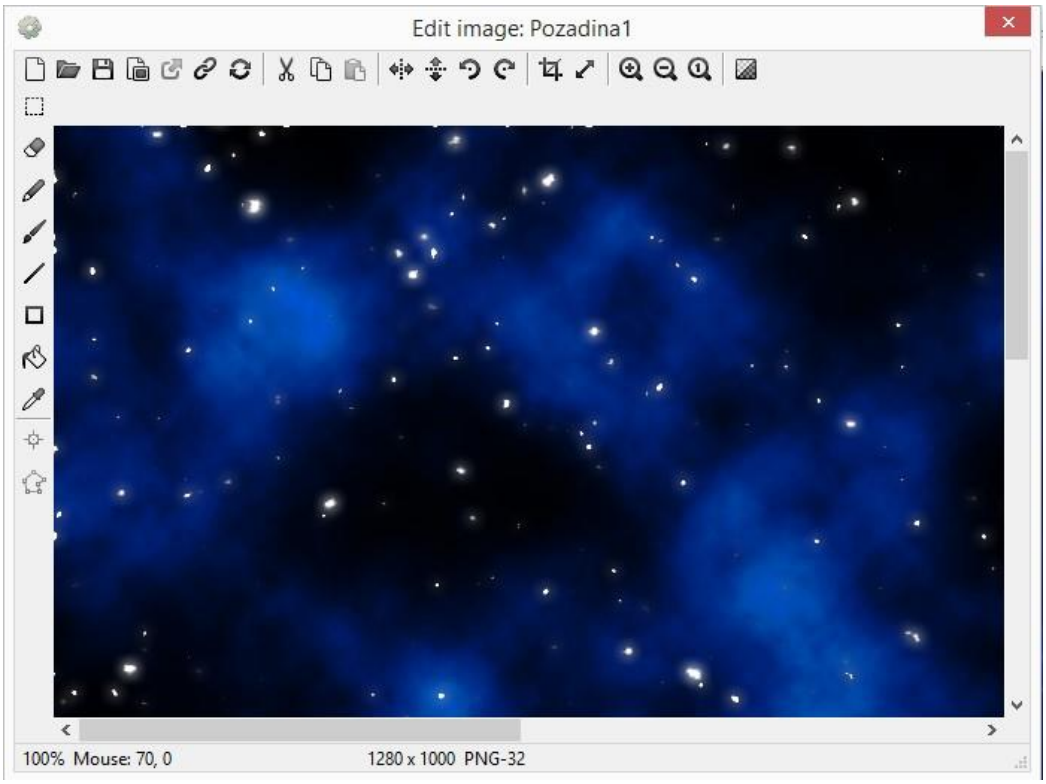
5.2.1. Pozadina

Pozadina je objekt koji se nalazi na najnižemu sloju igre. Osim toga što prikazuje okolinu u kojoj se nalaze likovi iz prvoga plana, u ovom slučaju ona ima i zadaću stvoriti privid pomaka između sebe i tih likova. Za dodavanje pozadine primijenjen je priključak *Tiled Background* (Slika 5.12). Razlog uporabe ovoga priključka jesu bolje performanse i način da pri promjeni veličine nastavlja ponavljati uzorak slike umjesto razvlačenja slike kao što to čini priključak *sprite*.



Slika 5.12 Odabir priključka za pozadinu

Nakon što se priključak umetne, otvara se prozor za obradu slike. U tom se prozoru nudi mogućnost da se umetne slika ili da se, koristeći se kistom, ispunom i ostalim sličnim alatima, izradi nekakva slika. U ovom slučaju umetnuta je već gotova slika prethodno izrađena u programu *Photoshop* i u editoru postavljena veličina kako bi odgovarala veličini plana. Zatvaranjem prozora proces dodavanja tipa objekta završava i omogućuje se rad s tim objektom. Pozicija i veličina čije su vrijednosti postavljene kao na slici (Slika 5.14) omogućuju vertikalni pomak. Visina je pozadine povećana dvostruko s obzirom na početnu veličinu i izvorišna joj se točka nalazi na visini dvostruko većoj od veličine plana. Na taj se način se stvara mogućnost da jedna polovica pozadine uvijek bude prisutna na planu, a kako objekt tipa *Tiled Background*, radi tako da povećanjem slike samo kopira već postojeći uzorak, pozadina koja će se vertikalno pomicati uvijek će biti onakva kakva je originalno i zamišljena.

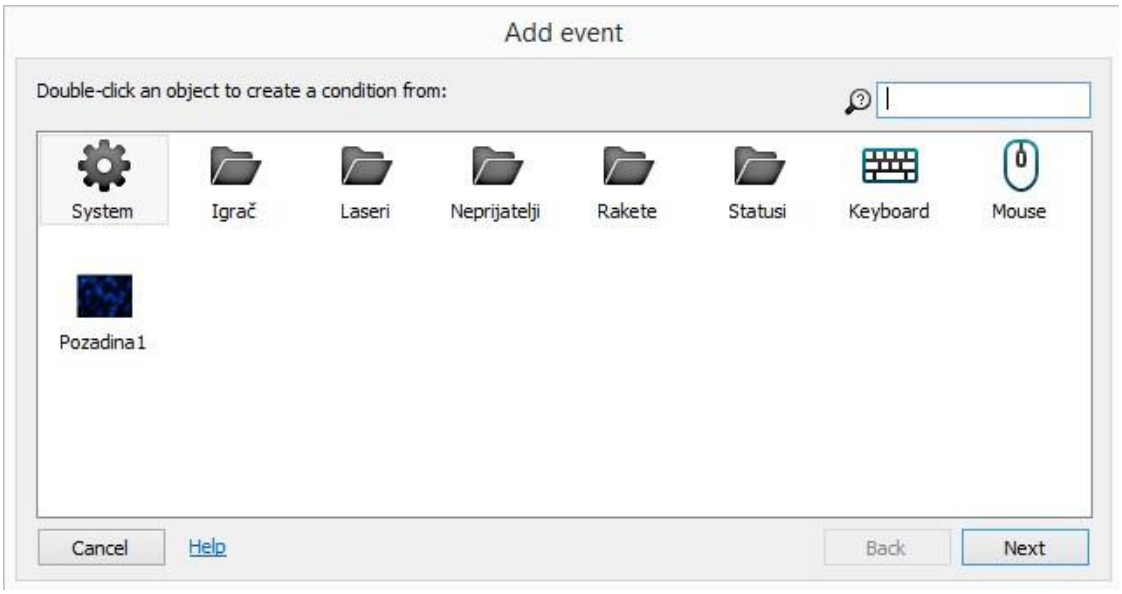


Slika 5.13 Editor slika

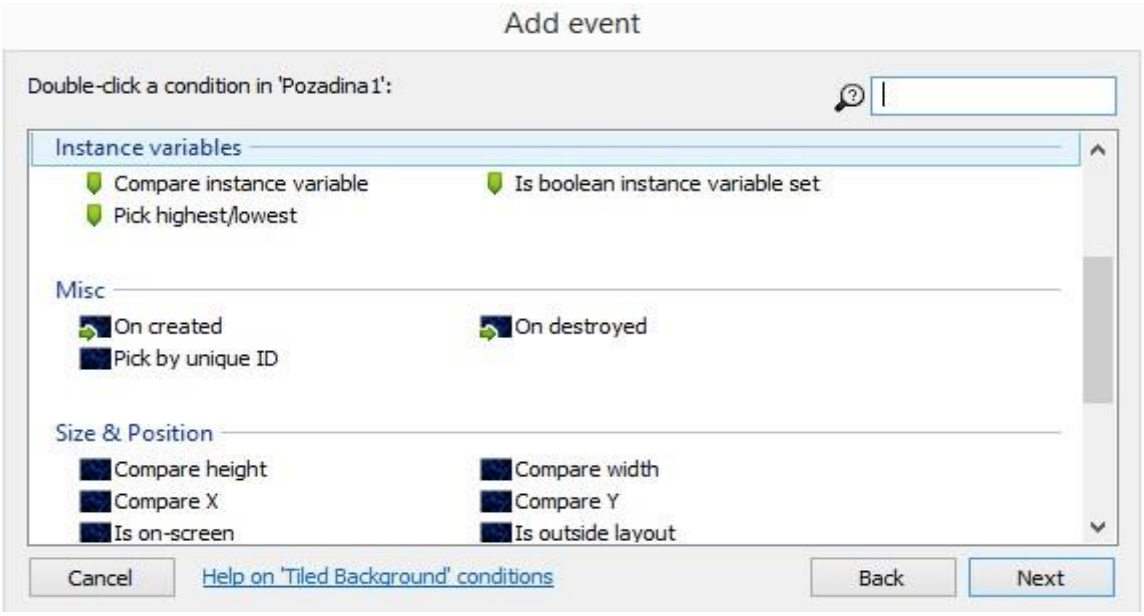
Object type properties	
Name	Pozadina1
Plugin	Tiled Background
UID	0
Global	No
Common	
Layer	Pozadina
Angle	0
Opacity	100
Position	0, -1000
Size	1280, 2000

Slika 5.14 Obilježja tipa objekta *pozadina*

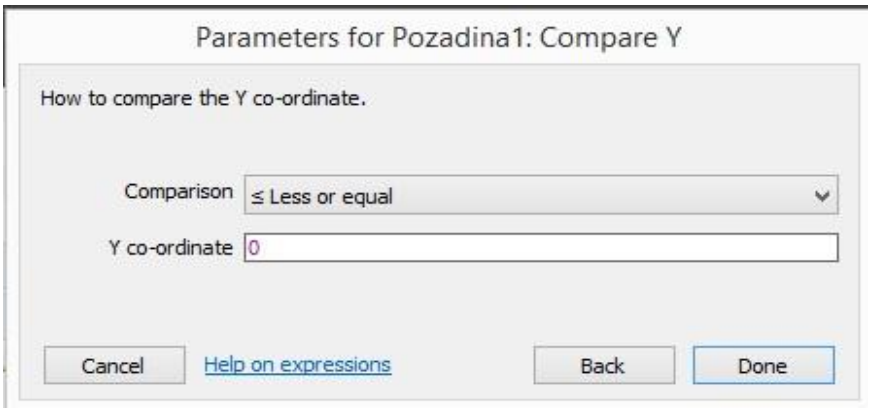
Da bi ovako postavljena pozadina obavljala vertikalni pomak, potrebno je definirati način njezina rada s pomoću događaja. Kako bi se događaj dodao u listu događaja, potrebno je prvo stisnuti desni klik miša i odabrati dodaj događaj. Nakon toga se otvara prozor u kojemu se odabire objekt koji će činiti uvjet (Slika 5.15). Nakon što se objekt odabere nudi se mogućnost odabira uvjeta (Slika 5.16) te nakraju parametri tog uvjeta (Slika 5.17).



Slika 5.15 Prozor za biranje objekta

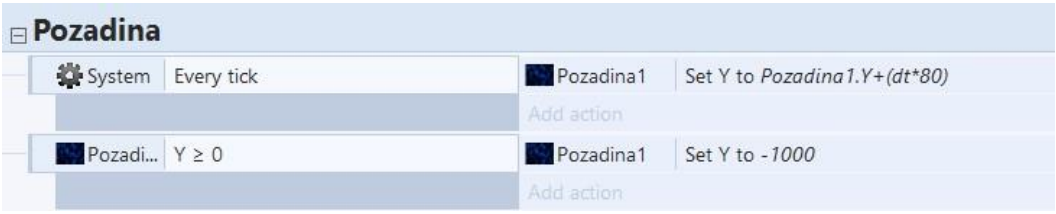


Slika 5.16 Prozor za biranje uvjeta

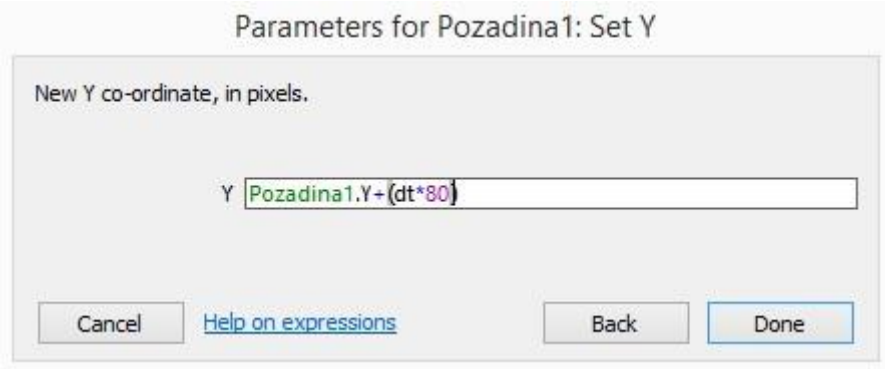


Slika 5.17 Prozor za postavljanje parametara

Vertikalni pomak riješen je s pomoću događaja (Slika 5.18) u kojem sistem svaki otkucaj (engl. *every tick*) obavlja akciju koja stavlja koordinatu pozadine Y na položaj zadan parametrima kao na slici (Slika 5.19). U ovom se slučaju *Pozadina1.Y* odnosi na trenutačni položaj objekta, 80 se odnosi na brzinu kojom će se izvoditi vertikalni pomak, a dt označuje promjenu u vremenu, odnosno vrijeme svakog otkucaja u sekundama i služi kako bi osiguralo glatko izvođenje ovoga neprekidnog pokreta. Nakon što se prvi događaj izvrši, izvršava se drugi koji je zadužen da, svaki put kad položaj pozadine bude nula (dođe do polovice) ili prijeđe nulu, vrati pozadinu na njezin početni položaj. Na taj način ova dva događaja čine petlju koja omogućuje neprekidan vertikalni pomak.



Slika 5.18 Događaji s pomoću kojih se odrađuje vertikalni pomak



Slika 5.19 Parametri za izvođenje vertikalnog pokreta

5.2.2. Igrač

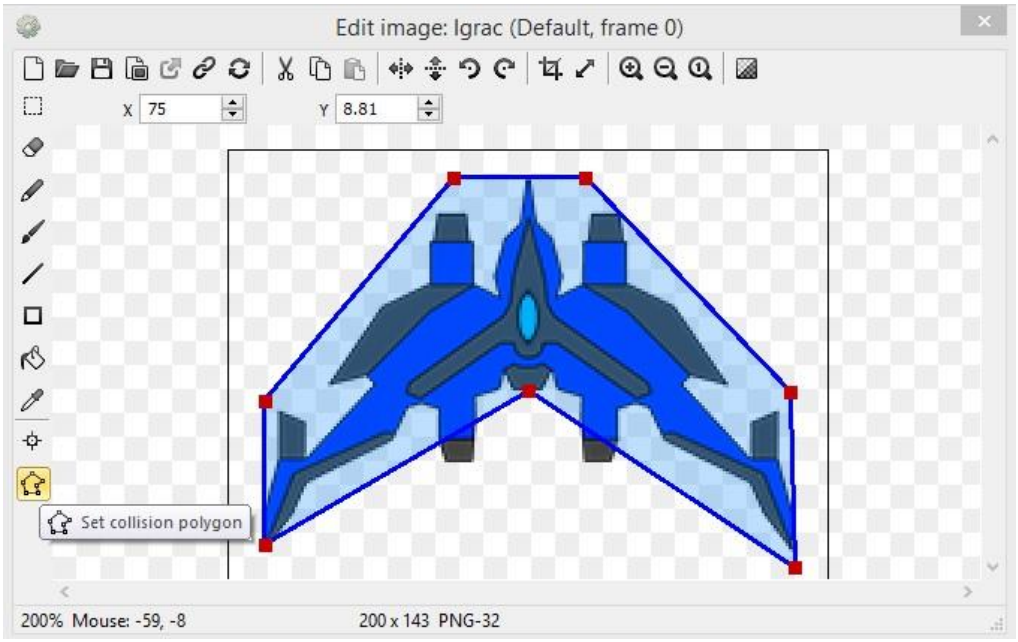
Igrač predočuje glavnog lika, odnosno objekt kojim igrač upravlja i ima interakciju s ostatkom igre. U ovom će se dijelu poglavlja prikazati načini kojima se ostvaruje temeljni način rada objekta *igrač*. Način rada objekta *igrač* odnosi se na:

- kretanje
- pucanje
- uporabu raketa
- animaciju mlaznog pogona.

Za dodavanje objekta *Igrač* uporabljen je priključak *sprite*. Taj je priključak iskorišten tijekom izrade igre i za objekte:

- neprijatelji
- glavni neprijatelji
- pojačanja
- napadi.

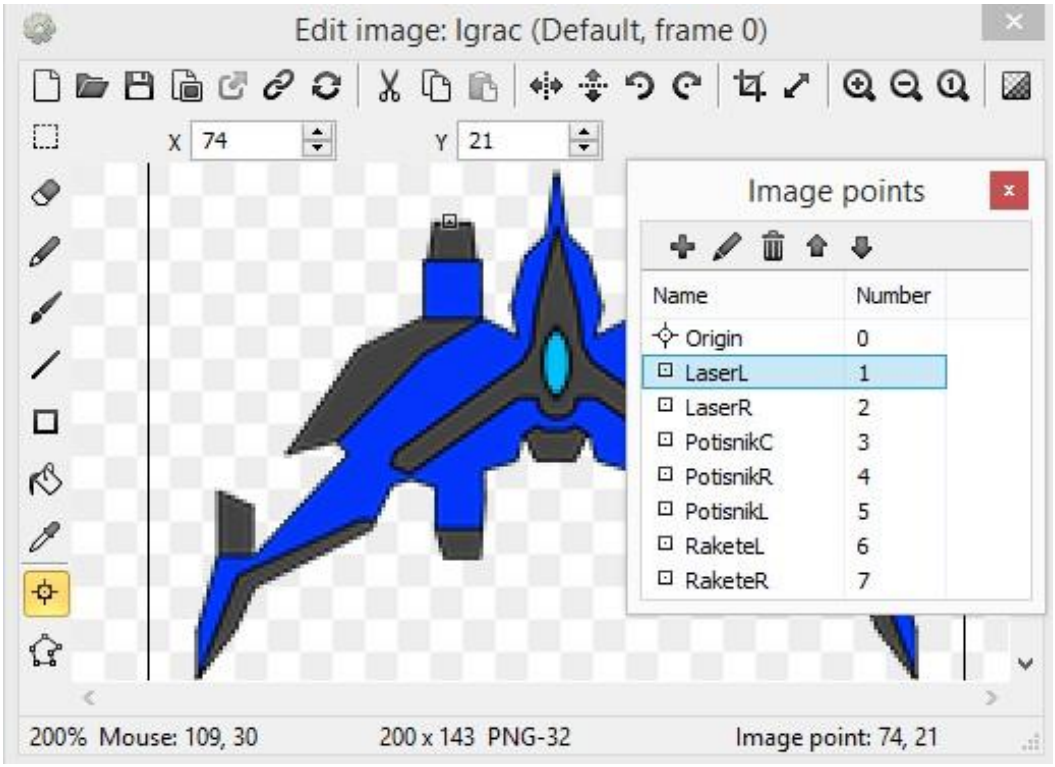
Kao i za pozadinu objekt *Igrač* potrebno je urediti u editoru slika. Osim veličine, tom je objektu potrebno postaviti *collision polygon*, točku izvora (engl. *origin point*) i točke slike (engl. *image points*). *Collision polygon* postavlja se s pomoću alata *set collision polygon* i služi za otkrivanje sudara (engl. *collision detection*) između objekata. Mnogokut (engl. *polygon*) koji je postavljen kao na slici (Slika 5.20) određuje granice područja objekta *Igrač* kojima se sudara s granicama drugih objekata. Točke koje su vidljive na svakom kutu mnogokuta moguće je ručno rasporediti i dodati prema potrebi kao što je to napravljeno u slučaju ovog objekta. Kod rasporeda točaka valja uzeti u obzir činjenicu da prevelik broj može uzrokovati lošije performanse. Kod manje kompliciranih objekata ili za uštedu performansi moguće je postaviti granični okvir (engl. *bounding box*) koji ima pravokutni oblik, odnosno samo četiri točke sudara (engl. *collision points*).



Slika 5.20 Postavljanje granica

Točka izvora točka je iz koje se izvodi rotacija objekta (engl. *the point of rotation*) i točka koja određuje poziciju objekta. Točke na slici (Slika 5.21) uporabljuju se za stvaranje novih

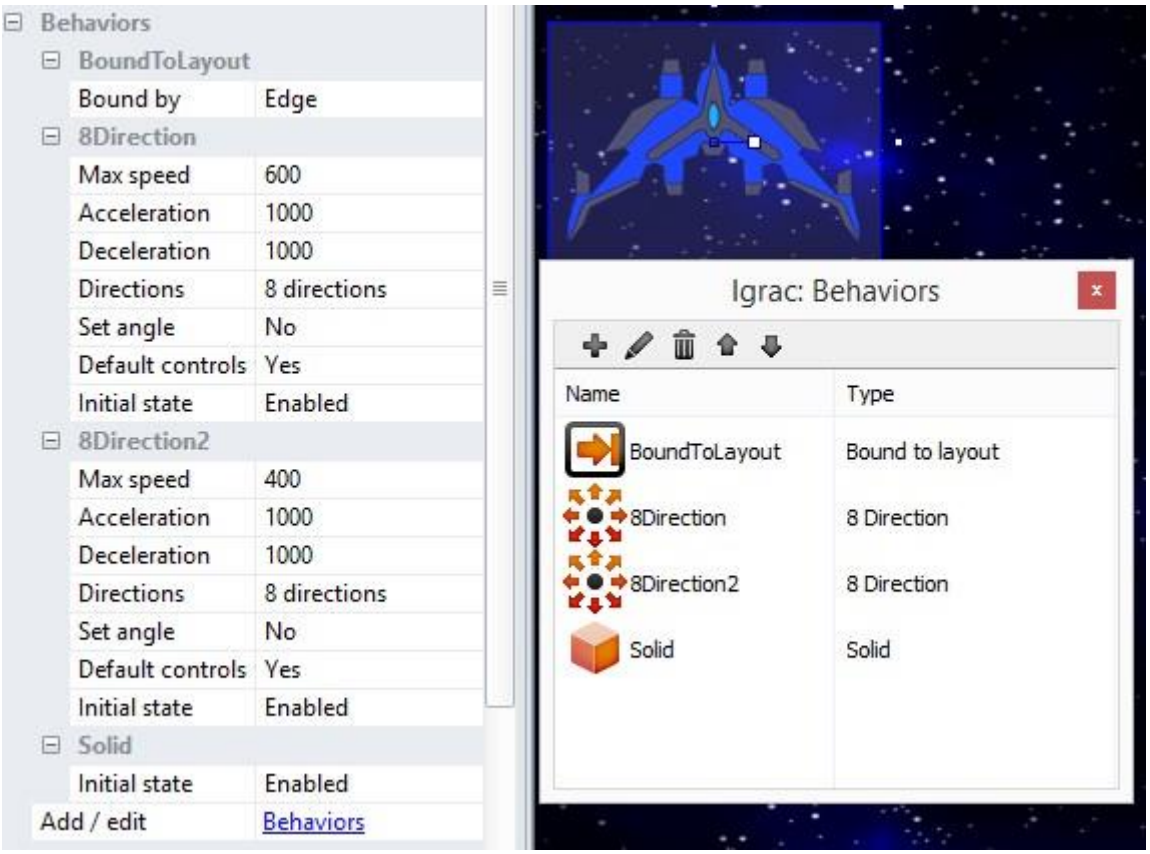
objekata i postavlja ih se tamo gdje planiramo s pomoću događaja stvoriti novi objekt. Na ovom objektu te su točke postavljene za ispucavanje lasera i raketa te za mlaznice.



Slika 5.21 Dodavanje točaka na sliku

Nakon što su sve točke postavljene objekt *igrač* dodaje se na plan. Pošto je objekt dodan na plan, moguće mu je pridodati neka određena ponašanja. Ponašanja pridodana objektu *igrač* prikazana su na slici (Slika 5.22) i služe kako igrač ne bi mogao napustiti plan, omogućuju mu micanje u više smjerova i sudaranja tog objekta s drugim objektima koji imaju ponašanje *čvrst* (engl. *solid*). Razlog za dodavanje ponašanja osam smjerova (engl. *eight direction*) 2 puta bio je kako bi pomak prema lijevo ili desno bio brži od pomaka prema gore, ali da se za dijagonalni pomak uporabljuje ponašanje koje ima veću vrijednost brzine kako taj pomak ne bi djelovao presporo.

Osim ponašanja, za rad ovog objekta potrebno mu je dodati i varijable instancije. Varijable instancije prikazane su na slici (Slika 5.23) i uporabljuju se za pucanje lasera i raketa.



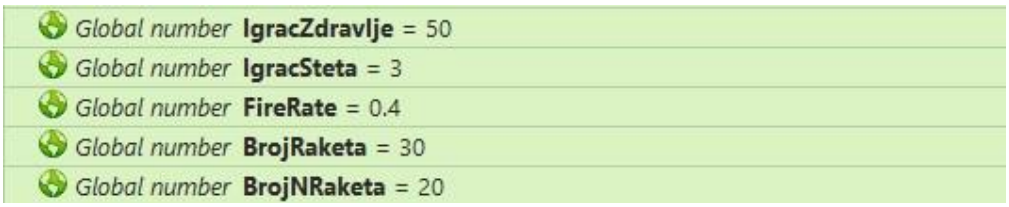
Slika 5.22 Ponašanja tipa objekta *igrač*

Slika 5.23 Varijable instancije objekta *igrač*

Osim varijabli instancija, postoje i drugi tipovi varijabli, a to su varijable događaja (engl. *event variables*). One se dijele na:

- globalne i
- lokalne.

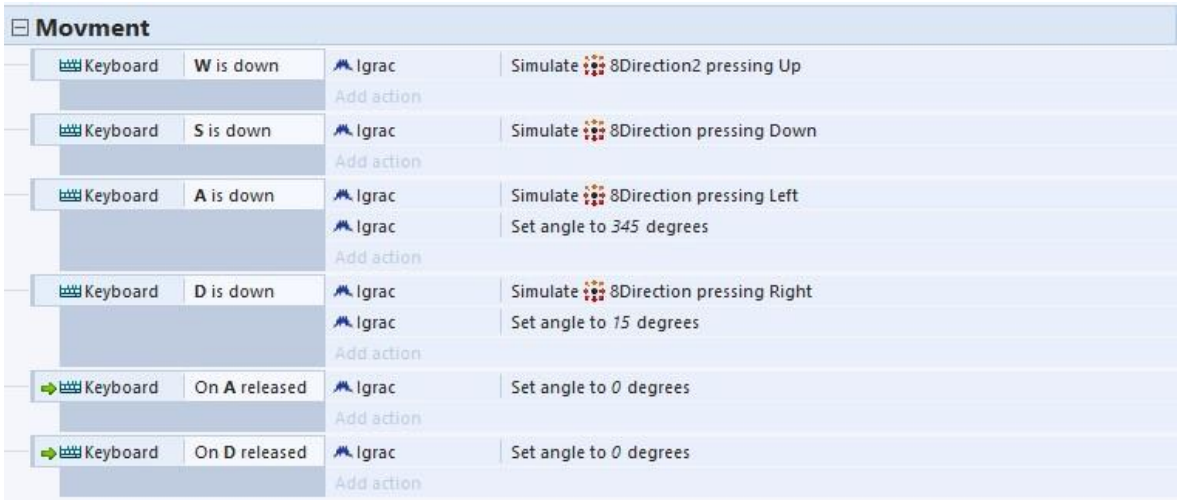
Globalne se varijable kreiraju u listi događaja i svoju vrijednost pohranjuju i između planova. Moguće im je pristupiti preko svih događaja, čak i ako su ti događaji u listi događaja u kojoj ta globalna varijabla nije kreirana. Lokalne se varijable kreiraju pak samo u određenom događaju i moguće im je pristupiti samo u tom događaju i u njegovim poddogađajima (engl. *sub events*).



Slika 5.24 Globalne varijable

Globalne varijable (Slika 5.24) kojima se objekt *igrač* koristi za pohranu podataka odnose se na njegove bodove zdravlja (engl. *health points*), na štetu koju će neprijatelj primiti ako ga pogodi laser, brzinu pucanja tih lasera i stanje raketa. Razlog uporabe globalnih varijabli umjesto varijabli instancije jest lakše ostvarivanje promjena vrijednosti s pomoću pojačanja i mogućnost da se prikupljena pojačanja prenesu na sljedeću razinu.

Kako bi objekt *igrač* izvodio sve ono što je zamišljeno da taj objekt radi, potrebno je u listu događaja dodati događaje koji će omogućiti sve te načine rada. Prvo i najvažnije jest omogućiti igraču kretanje po ekranu. To je moguće postići dodavanjem objekta tipkovnice i zadovoljavanjem uvjeta da, dokle god je određena tipka pritisnuta, ponašanje koje je prije dodano igraču radi pomak definiran ovisno o tome koja je tipka pritisnuta (Slika 5.25). Pri otpuštanju tipke igrač se prestaje micati dok ne pritisne novu tipku. Dodatna akcija koja se izvodi pri pomaku lijevo ili desno mali je pomak kuta igrača, a ona prestaje kada se neka od tih tipki otpusti.

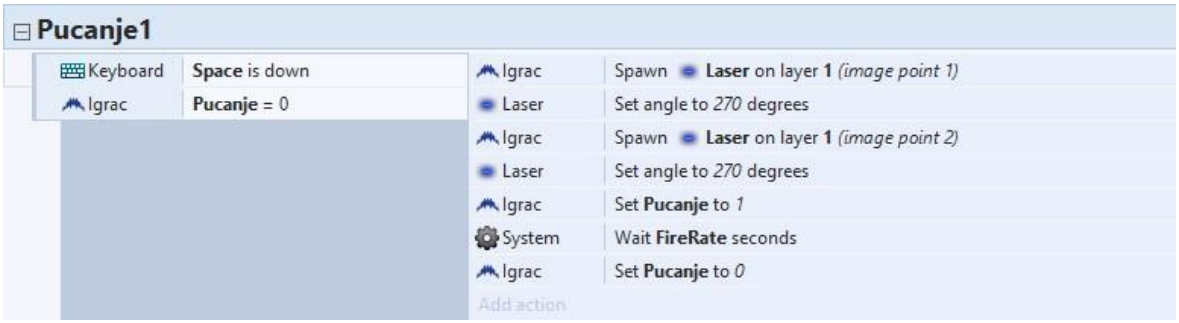


Slika 5.25 Događaji za kretanje igrača u igri

Nakon što je igraču omogućeno kretanje, potrebno mu je omogućiti i pucanje laserom. Da bi se to omogućilo, potrebno je prvo dodati objekt *laser* i pridodati mu ponašanje *metak* (engl. *bullet*). Za pucanje laserom potreban je događaj (Slika 5.26) koji se sastoji od dvaju uvjeta:

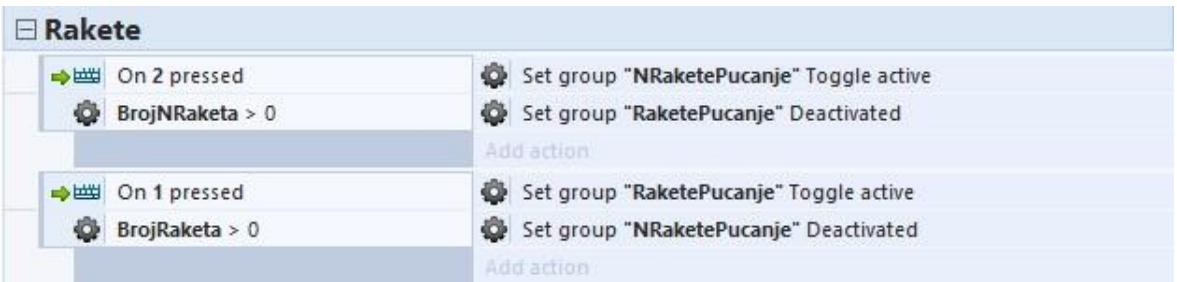
- ako je tipka pritisnuta
- je li varijabla instancije *Pucanje* jednaka 0.

Ako su ta dva uvjeta ispunjena, objekt *igrač* kreira instanciju objekta *laser* na sloju koji je odabran i na odabranoj točki slike, u ovom slučaju na vrhovima topova objekta *igrač*. Nakon što se laser kreira postavlja se njegov kut tako da se u trenutku stvaranja lasera krene kretati u željenom smjeru. Zadnje tri akcije koje se izvršavaju odnose se na brzinu pucanja. Akcije rade tako da nakon što se kreira laser stanje varijable *Pucanje* prelazi u jedan, a zatim se izvršava čekanje, ovisno o vrijednosti globalne varijable brzina pucanja koja određuje trajanje do izvršavanja posljednje akcije koja vraća varijablu pucanje natrag u nulu. Na ovaj je način omogućeno da, dokle god je pritisnuta tipka za pucanje, laseri će se kreirati u određenom ritmu, odnosno igrač će moći kontinuirano pucati.



Slika 5.26 Događaj koji omogućuje pucanje u igri

Osim laserom, igrač može pucati i dvjema vrstama raketa. Te se dvije vrste dijele na obične i navođene rakete čiji će princip rada biti objašnjen u kasnijem dijelu poglavlja. Pucanje raketa postiže se na malo drukčiji način od pucanja lasera. Cilj je bio omogućiti igraču da jednim klikom na tipku pokrene pucanje raketa i da, ako želi pucati drugu vrstu, ne mora prvo isključiti trenutačno, nego samo mora pritisnuti tipku za pucanje drugih raketa. To je postignuto logičkim blokovima koji su prikazani na slici (Slika 5.27).



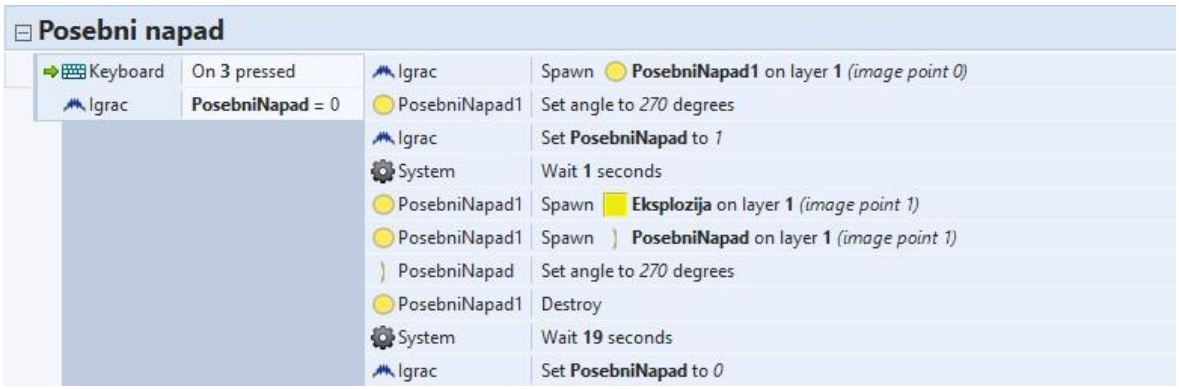
Slika 5.27 Događaji s pomoću kojih rade kontrole za pucanje raketa u igri

Ti logički blokovi, odnosno događaji rade na principu da se akcija izvrši ako je potrebna tipka bila pritisnuta i ako je iznos globalne varijable koja pohranjuje broj raketa veća od nule.

Kada je taj uvjet zadovoljen, pokreće se akcija koja aktivira zadanu grupu u kojoj se nalazi događaj (Slika 5.28) koji radi na istom principu kao i događaj za pucanje lasera. *While* petlja u ovom slučaju služi kao zamjena za kontinuirano držanje tipke, a provjera globalne varijable za broj navođenih raketa služi kako bi se, u slučaju da broj raketa padne na 0, zaustavilo pucanje. Akciju oduzimanja dviju raketa od njihova trenutačnog broja sistem provodi nakon svakog ispucavanja raketa.

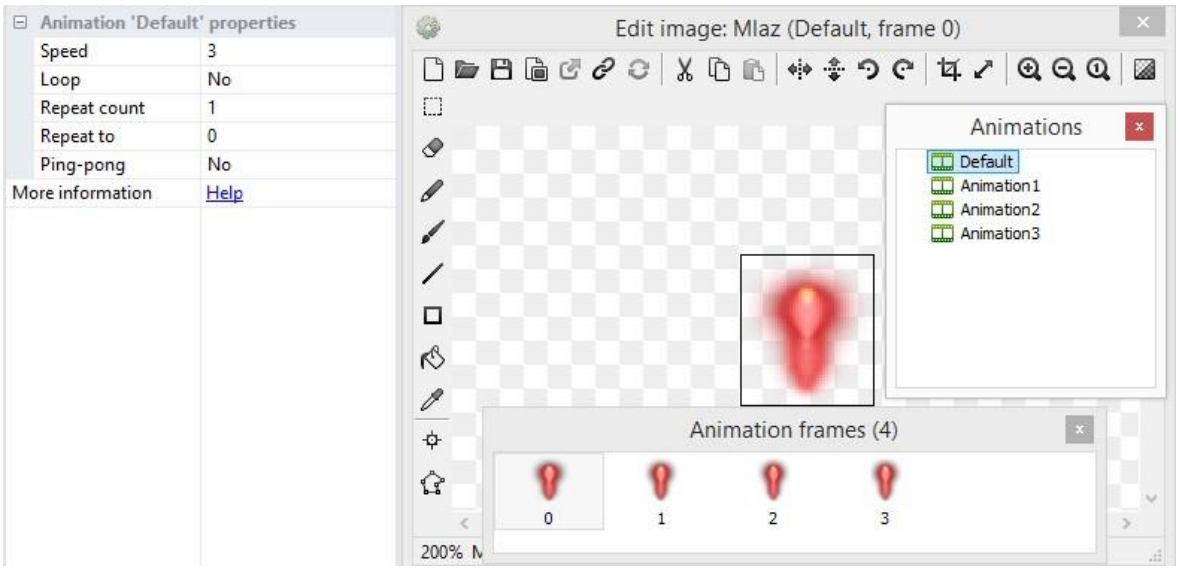
Slika 5.28 Događaj koji stvara rakete u igri

Zadnji od elemenata koji se odnosi na pucanje jest posebni napad. Taj napad radi tako da, svaki put kada igrač pritisne tipku za pucanje tog napada, glavni lik kreira bombu koja se nakon kratkoga vremena detonira te se nakon te detonacije stvara posebni napad koji može uništiti sve protivničke jedinice koje se trenutačno nalaze na ekranu, osim glavnih neprijatelja. S obzirom na to da je riječ o dosta jakom napadu, na njega je stavljeno malo dulje razdoblje čekanja kako bi se igraču ograničila njegova uporaba i kako bi pravodobna uporaba ovog napada bila što bolje nagrađena.



Slika 5.29 Događaj za posebni napad

Igrač kao objekt sam po sebi nije animiran, ali je animacija primijenjena na njegov mlazni pogon, odnosno objekt *mlaz*. Animacija se priprema tako da se u editoru slike (Slika 5.30) doda potrebni broj animacija i svakoj od tih animacija dodaju se animacijske sličice (engl. *frames*). U svojstvima animacija moguće je promijeniti parametre od kojih su najvažniji brzina i petlja (engl. *loop*).

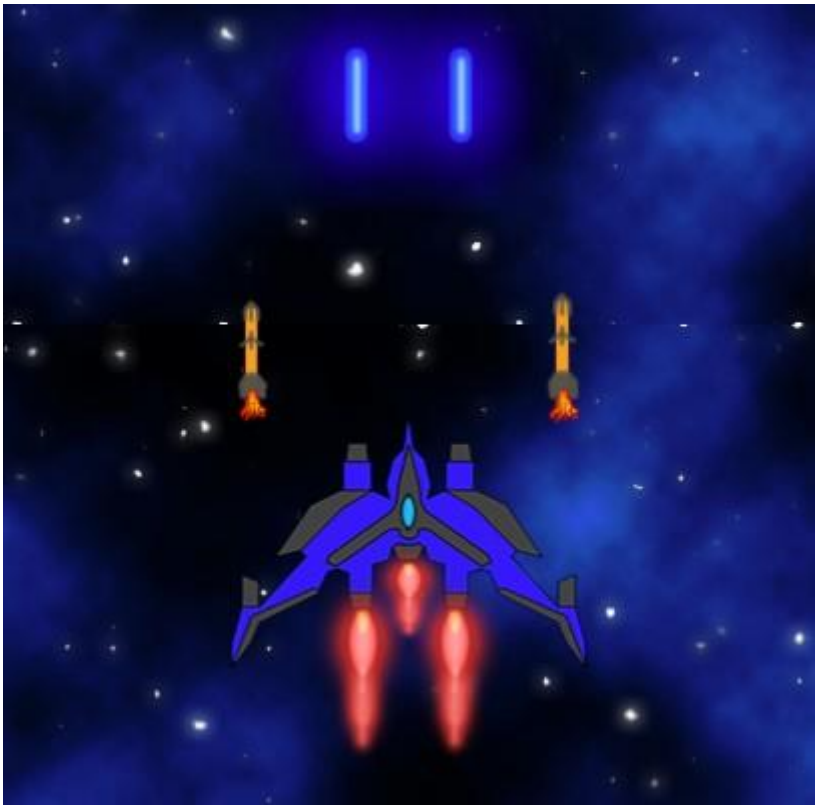


Slika 5.30 Postavljanje animacija u editoru slika

U ovoj igri postoje 2 tipa objekta za mlaz. Jedan se od njih odnosi na lijevi i desni mlazni pogon, dok se drugi odnosi na srednji. Lijevi i desni imaju samo jednu animaciju koja se s pomoću petlje cijelo vrijeme vrti, dok srednji pogon ima četiri različite animacije koje se pojavljuju ovisno o situaciji. Kako bi mlaz funkcionirao, potrebno ga je pri pokretanju igre stvoriti na točki slike, odnosno mlaznici za koju je predviđen, prikvačiti mlaz kojemu je potrebno dodati ponašanje prikvači (engl. *pin*) na poziciju na kojoj se nalazi ta točka slika i pokrenuti osnovnu animaciju koja se izvodi ako se objekt *igrač* ne pomiče (Slika 5.31). U trenutku kada se tipka za pomicanje prema naprijed pritisne, pokreće se animacija kojom se objekt postupno povećava. Kada se ta animacija dokraja provede, započne animacija kojom se koristi i drugi tip mlaza, a ona se odnosi na nekakvo pulsiranje toga mlaza. Kada se tipka za pomicanje prema gore otpusti, pokreće se animacija smanjivanja mlaza te se nakon završetka te animacije, animacija objekta vraća na početnu.

Slika 5.31 Događaji koji pokreću animacije za objekt *mlaz*

Animacija objekta *mlaz* posljednja je od značajki koje se odnose temeljan način rada objekta *igrač*. Interakcije koje igrač ima s neprijateljima i ostalim objektima bit će objašnjene nakon što se i ti, ostali objekti definiraju.



Slika 5.32 Prikaz objekta *igrač*

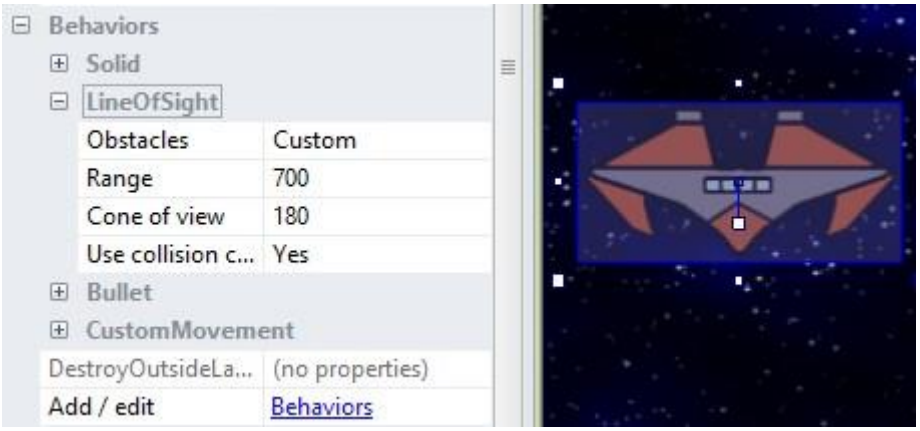
5.2.3. Neprijatelji

Neprijatelji se odnose na objekte koji stvaraju prijetnju igraču. Postupak dodavanja objekata neprijatelja na plan isti je kao postupak dodavanja objekta *igrač*. Objekti *neprijatelja* koriste se nešto drukčijim ponašanjima od objekta *igrač*. Ta se ponašanja, osim ponašanja *čvrst*, odnose na:

- liniju pogleda (engl. *line of sight*)
- posebno prilagođeno kretanje (engl. *custom movement*).

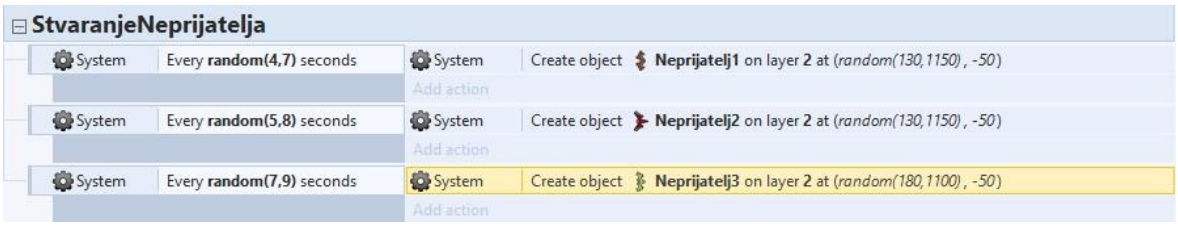
Ponašanje *linija pogleda* u ovoj igri služi kako bi se odredilo unutar koje udaljenosti od objekta neprijatelj igrač mora biti kako bi na njega neprijatelj počeo pucati, a posebno prilagođeno kretanje služi kako bi neprijatelj imao neku određenu brzinu kretanja u kojem god smjeru je potrebno, neovisno o kutu i tako pruža više mogućnosti s obzirom na ponašanje

metak kojim se objekt uvijek kreće u smjeru koji je određen kutom objekta. Nakon ponašanja objektu *neprijatelj* kreira se varijabla instancije bodovi zdravlja koja se uporabljuje pri interakciji s objektom *igrač* i objektima koje igrač ispucava.

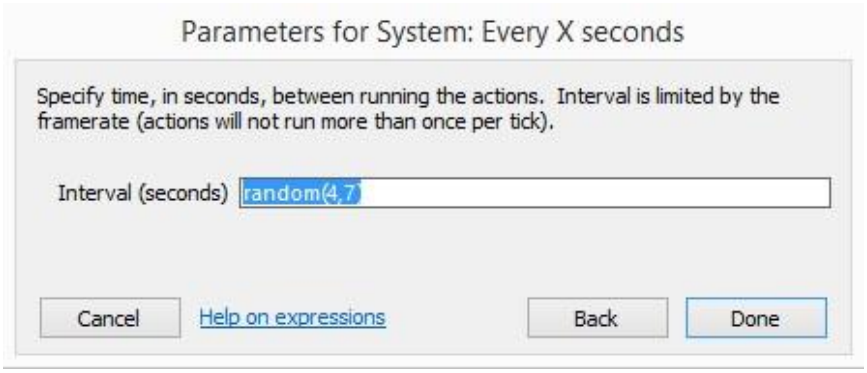


Slika 5.33 Ponašanje linije pogleda

Kako bi neprijatelji bili uopće prikazani na ekranu, potrebno ih je kreirati tijekom igre. Za potrebu kreiranja iskorištena je mogućnost da neprijatelja kreira sistem. Događaji (Slika 5.34) za radnju nasumičnog kreiranja neprijatelja rade na principu da se svakih nekoliko sekundi na nasumičnom dijelu plana pojavi novi neprijatelj. Parametri koji određuju vrijeme postavljeni su kao na slici (Slika 5.35). Uz pomoć izraza nasumično (engl. *random*) bira se broj iz vremenskog intervala koji će onda odrediti trajanje potrebno da se akcija krene izvršavati.

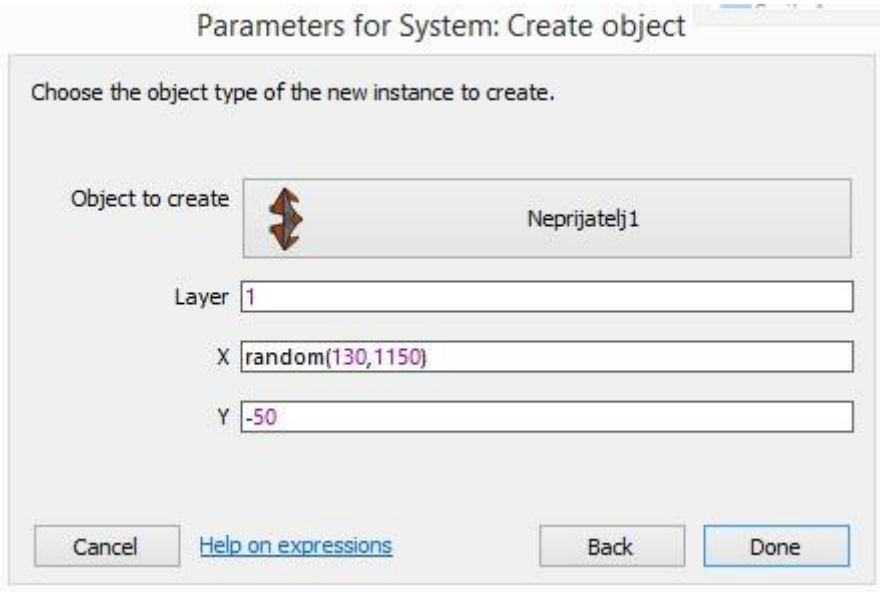


Slika 5.34 Događaji koji nasumično kreiraju nove neprijatelje



Slika 5.35 Parametar koji se uporabljuje kao uvjet za kreiranje jedinica

Akcija radi prema parametrima postavljenima kao na slici (Slika 5.36). Instancija neprijatelja uvijek se stvara na istome sloju i stvara se malo iznad nule, što omogućuje da igrač ne vidi stvaranje neprijatelja. Kako se neprijatelj ne bi uvijek stvarao na istoj poziciji umjesto fiksne vrijednosti parametru X dodan je izraz nasumično pomoću kojeg sistem onda bira gdje će na x-osi kreirati neprijatelja.



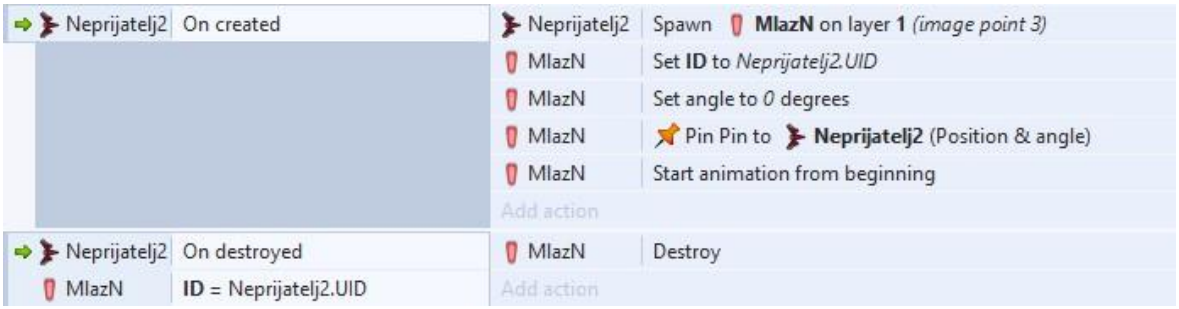
Slika 5.36 Parametri za određivanje gdje će se neprijatelj kreirati

Kako bi se neprijatelj počeo kretati nakon što ga se kreira, potrebno je odrediti mu brzinu i smjer kretanja. To se postiže tako da se svaki put kada se instancija *neprijatelj* kreira, toj instanciji dâ mogućnost vertikalnoga kretanja određenom brzinom (Slika 5.37). Ovakav način kretanja isti je za sve neprijatelje.

Slika 5.37 Događaj koji neprijatelju omogućuje kretanje u igri

Kao i objekt *igrač* i neprijatelji imaju mlazni pogon, ali, da bi se mlaz koji neprijatelj ima pravilno izveo, potrebno je rabiti UID. UID se dodjeljuje svakoj instanciji projekta i broj je koji se, počevši od 0, povećava za 1 pri kreiranju svake nove instancije. Razlog zašto je UID potreban za ovu radnju jest to što svaka nova instancija neprijatelja tijekom igre zahtjeva i novu instanciju objekta *mlaz*. U trenutku kada igrač uništi instanciju neprijatelja, potrebno je uništiti i njegov mlaz, a kako bi se osiguralo da se, ako postoji više neprijatelja, na planu uništi samo instancija *mlaz* kojom se koristi uništeni neprijatelj, potrebno je svakom mlazu kreirati varijablu instancije koja će služiti kako bi se u nju pohranila vrijednost UID instancije neprijatelja na kojemu je mlaz stvoren. Na taj se način, u trenutku kada je

neprijatelj uništen, može provesti drugi događaj sa slike (Slika 5.38), čiji uvjet služi tomu da pronađe mlaz koji je potrebno uništiti i uništi ga.



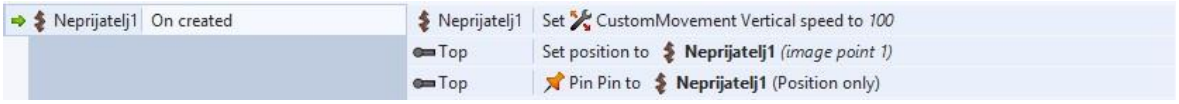
Slika 5.38 Događaji za mlaz neprijatelja

Svaki se neprijatelj, osim izgledom, razlikuje nekom posebnom značajkom. Te se značajke dijele na:

- rotirajući top
- okretanje neprijatelja prema liku
- micanje po X-osi (lijevo, desno).

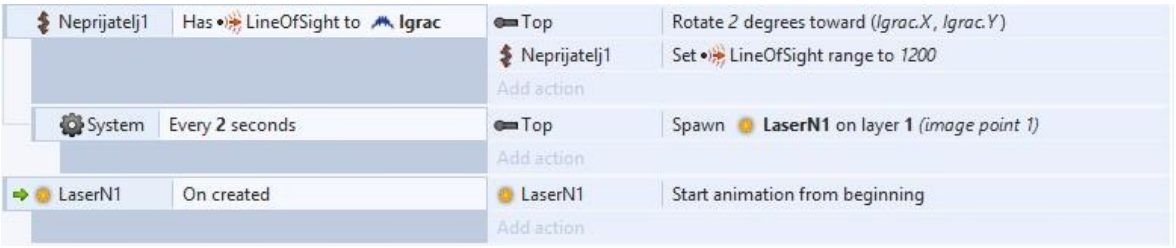
Rotirajući top zasebni je tip objekta kojim se tip objekta *neprijatelj1* koristi. Postavljen je u spremnik *neprijatelja1* i zbog toga objekt nije potrebno kreirati svaki put kada se kreira nova instancija *neprijatelja1*, već je top potrebno samo postaviti na željenu točku i prikvačiti ga (Slika 5.40).

Slika 5.39 Spremnik *neprijatelja1*

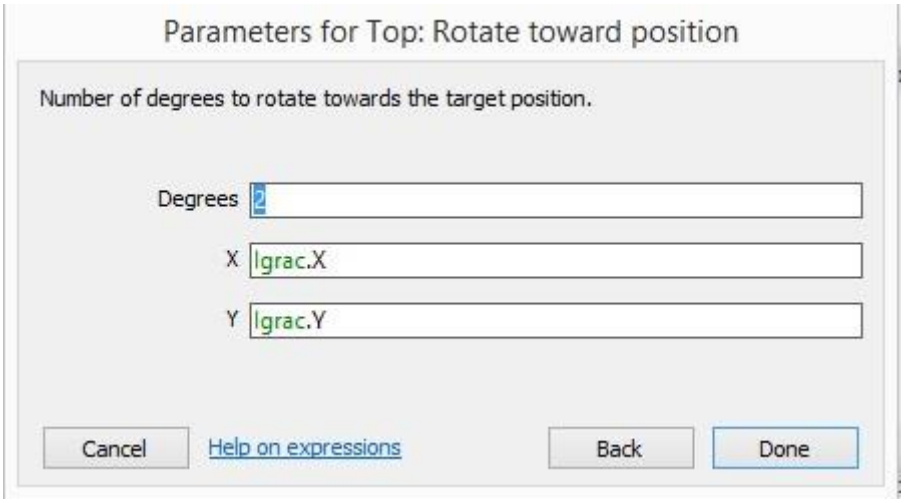


Slika 5.40 Događaj koji postavlja top na *neprijatelja1*

Događaj koji omogućuje način rada rotirajućeg topa prikazan je na slici (Slika 5.41). Top radi tako da se rotira prema igraču po parametrima na slici (Slika 5.42) dokle god neprijatelj ima pogled na igrača. Poddogađaj je zaslužan za pucanje i on, jednostavno, na vrhu topa kreira neprijateljski laser svake dvije sekunde.

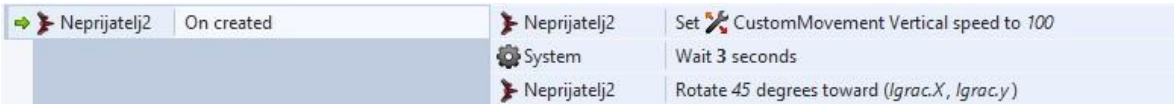


Slika 5.41 Događaji za rad topa i animaciju lasera

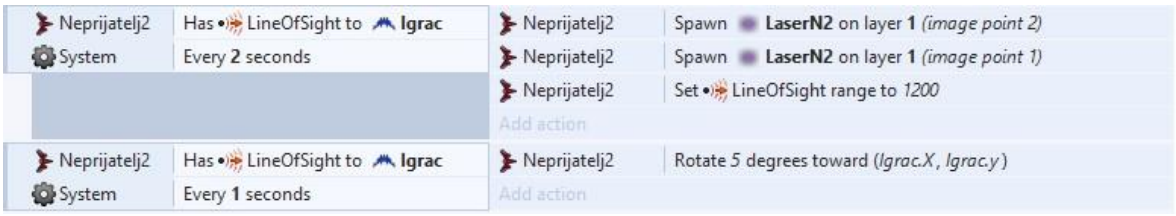


Slika 5.42 Parametri prema kojima se top rotira

Okretanje neprijatelja prema igraču značajka je koju ima *neprijatelj2*. Ona radi s pomoću događaja (Slika 5.43), koji radi tako da, svaki put kad se instancija neprijatelja kreira, čeka 3 sekunde i zatim rotira tu instanciju neprijatelja prema igraču, odnosno njegovoj poziciji. Nakon početne rotacije *neprijatelj 2* se svake sekunde rotira prema novom položaj igrača (Slika 5.44).



Slika 5.43 Događaj koji rotira neprijatelja prema igraču



Slika 5.44 Događaji koji omogućuju da neprijatelj puca i naknadno se rotira

Posljednje obilježje koje neprijatelj može posjedovati jest pomicanje po X-osi. Takvo obilježje posjeduje *neprijatelj 3* i za njega je potrebno prije navedeno posebno prilagođeno ponašanje. To obilježje radi pomoću događaja (Slika 5.45) koji svake 3 sekunde pomiče objekt desno ili lijevo, ovisno o njegovoj poziciji. Ako je njegova trenutačna pozicija na X-osi manja od iznosa polovice veličine plana, objekt će se kretati udesno (kut kretanja 45 stupnjeva), a, ako je veća, kretat će se ulijevo (kut kretanja 135 stupnjeva).

Slika 5.45 Događaji za kretanje neprijatelja

Pucanje je u slučaju *neprijatelja 3* riješeno na isti način kao kod *neprijatelja 2* jedina razlika je u šteti koju čine pojedini laseri koje neprijatelji pucaju.

5.2.4. Interakcija između igrača i neprijatelja

Sada kada su objašnjeni načini rada objekta *igrač* i objekata *neprijatelji* moguće je opisati i objasniti njihove interakcije. Interakcije i načini rada, kao i u prethodnom dijelu poglavlja, bit će objašnjene primjerima iz liste događaja.

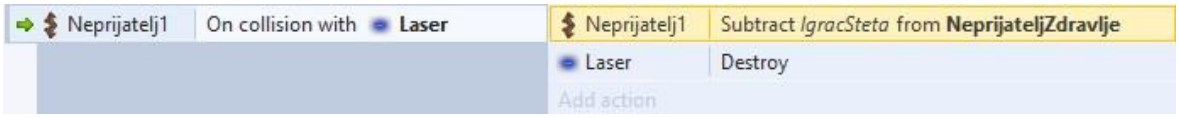
Interakcije se odnose na:

- sudar igrača i neprijatelja
- sudar neprijatelja s laserima ili raketama
- sudar neprijatelja s posebnim napadom
- sudar igrača s neprijateljskim laserima.

Prva interakcija koja se odnosi na sudar igrača i neprijatelja opisana je događajem sa slike (Slika 5.46). Ona radi tako da se pri sudaru neprijatelj pomakne unatrag te se nakon kratkog roka počne ponovno micati prema naprijed. I igrač i neprijatelj gube zdravlje, odnosno smanjuje se vrijednost koja je pohranjena u varijablama za vrijednost zdravlja.

Slika 5.46 Događaj za sudar igrača i neprijatelja

Sudar neprijatelja s laserima u ovoj je igri najvažnija i najčešća interakcija između igrača i neprijatelja. Događaj (Slika 5.47) omogućuje tu interakciju tako da svaki put kad se neprijatelj sudari s instancijom objekta *laser*, neprijatelj gubi zdravlje u iznosu globalne varijable *igrac steta*. Razlog za uporabu globalne varijable za štetu umjesto uporabe fiksne vrijednosti mogućnost je da se vrijednost promijeni ovisno o pojačanju ili nekom drugom čimbeniku te tako ostavlja prostor za naknadno uvođenje raznih nadogradnji.



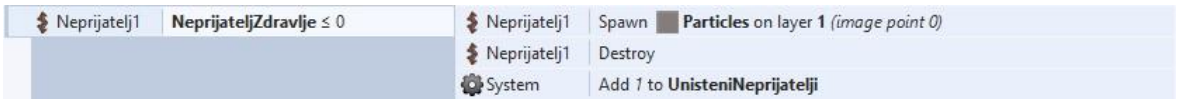
Slika 5.47 Događaj za sudar neprijatelja i lasera

Sudar neprijatelja s raketama radi na isti način kao i s laserima, a jedina je razlika što svaki od tipova raketa ima svoju fiksnu vrijednost za štetu. Iako je krajnji ishod pogotka rakete isti, one navođene specifične su po tome što jamče da će svaka od njih pogoditi protivnika ako se on nalazi na ekranu. U ovoj igri to je riješeno s pomoću UID-a i nevidljivog objekta pričvršćenog za svakog neprijatelja pod nazivom *vodič*. Događaji na slici (Slika 5.48) prikazuju proces navođenja rakete u kojemu se svakim otkucajem bira vodič najbliži položaju rakete i raketi kojoj je najbliži u varijablu se upisuje UID vodiča. Zatim *for each* petlja za svaku instanciju navođene rakete traži vodič kojemu je UID jednak vrijednosti varijable instancije objekta *navođena raketa* i poddogađaj onda svakim otkucajem rotira navođenu raketu prema tom vodiču. Na taj je način raketama omogućeno da odmah nakon kreiranja naciljaju najbližeg neprijatelja i prate ga sve dok on ne bude uništen. U slučaju uništenja njihova meta postaje sljedeći njima najbliži neprijateljski objekt.

Slika 5.48 Događaji za navođenje raketa

Ako igrač dovoljno puta laserom ili raketom pogodi neprijatelja, neprijatelj će se uništiti. Događaj na slici (Slika 5.49) omogućuje četiri stvari pod uvjetom da je vrijednost varijable instancije zdravlje manja ili jednaka nuli:

- eksploziju neprijatelja
- njegovo uništenje
- dodatak bodova na rezultat
- napredak kroz razinu.



Slika 5.49 Događaj koji se pokreće svaki put kad se uništi neprijatelj

Eksplozija se postiže tako da se prije uništenja neprijatelja kreira tip objekta *čestice* (engl. *particles*) s pomoću istoimenog priključka koje se raspršuju po zadanim parametrima koji su prikazani na slici (Slika 5.50).

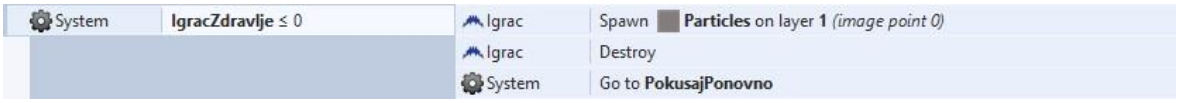
Properties			
Rate	30	Initial particle properties	
Spray cone	360	Speed	140
Type	One-shot	Size	4
Image	Edit	Opacity	100
Particle lifetime properties		Grow rate	2
Acceleration	-150	X randomiser	0
Gravity	0	Y randomiser	0
Angle rando...	0	Speed rando...	0
Speed rando...	800	Size random...	0
Opacity ran...	0	Grow rate ra...	0
Destroy mode	Fade to invis...		
Timeout	1		

Slika 5.50 Razni parametri kojima se utječe na način raspršivanja čestica

Nakon uništenja neprijatelja uvećava se vrijednost rezultata koja se razlikuje za svakog neprijatelja i dodaje se vrijednost 1 globalnoj varijabli koja broji uništene neprijatelje. Te se varijable uporabljuju za započinjanje borbe s glavnim neprijateljem (engl. *boss fight*) kao dva od triju mogućih uvjeta.

Posljednja interakcija igrača s neprijateljem odnosi se na sudar objekta *igrač* s neprijateljskim laserima. Važan dio ove igre jest da igrač bude u mogućnosti izbjegavati neprijateljske lasere jer, ako previše puta bude pogođen, igrač ostaje bez jedinica zdravlja. U slučaju da igrač ostane bez jedinica zdravlja, objekt *igrač* eksplodira i uništi se kao što je prikazano na slici (Slika 5.52). Igrač tada gubi igru i odlazi na izbornik *pokušaj opet*.

Slika 5.51 Događaj koji se aktivira kada je igrač pogođen



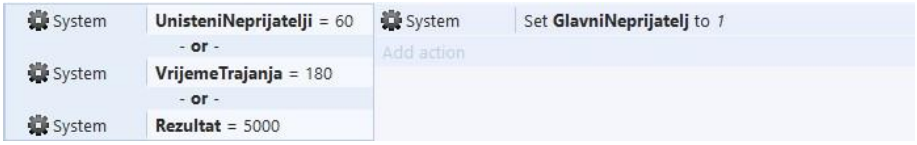
Slika 5.52 Događaj koji se pokreće kada igrač ostane bez zdravlja

Osim gubitka svih jedinica zdravlja, igrač gubi igru i ako previše neprijatelja prođe pokraj njega i on ih ne uspije uništiti. Za tu radnju odgovoran je nevidljivi objekt, odnosno *sprite* linija s pomoću koje se detektira neprijatelj. Događaj prikazan na slici (Slika 5.53) brine se da, svaki put kad neprijatelj dotakne liniju koja predočuje granicu, sistem smanji vrijednost globalne varijable koja čuva vrijednost koja određuje koliko još neprijatelja može proći za jedan. Kada vrijednost te varijable padne na nulu, ishod je isti kao i za gubitak svih bodova zdravlja.

Slika 5.53 Događaj koji se pokrene svaki put kad neprijatelj dođe do granice

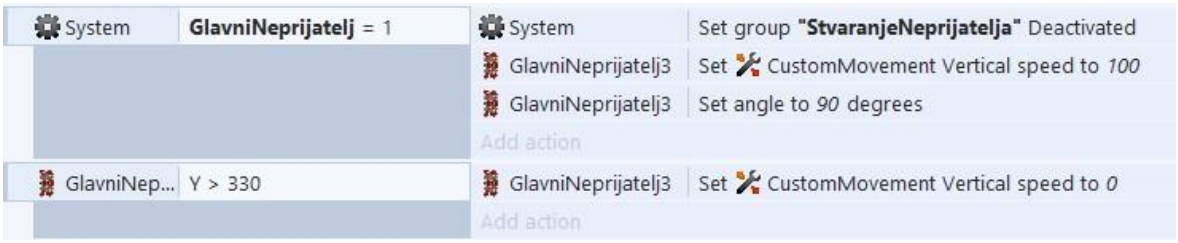
5.2.5. Glavni neprijatelji

Glavni neprijatelj zadnja je prepreku koju igrač mora svladati želi li prijeći razinu. I bitno se razlikuje od normalne neprijateljske postrojbe. Kada se jedan od uvjeta koji su prikazani na slici (Slika 5.54) ispuni, vrijednost globalne varijable *glavni neprijatelj* prelazi iz nule u jedan i time borba počinje.



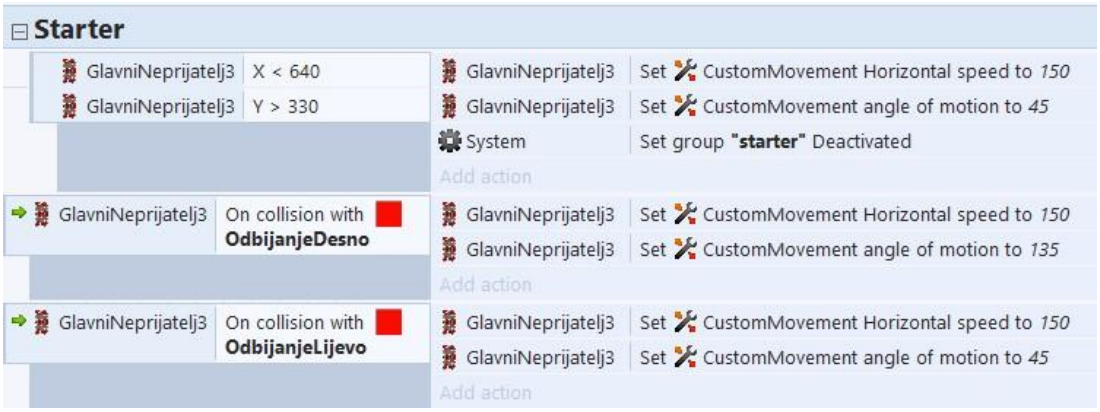
Slika 5.54 Logički blok „ILI“ koji služi za pokretanje bitke s glavnim neprijateljem

Tu borbu definira to što je glavni neprijatelj jedini neprijatelj s kojim se igrač suočava i u skladu s time potrebno je prekinuti kontinuirano stvaranje neprijatelja. Kako bi se to postiglo, potrebno je, kao što je prikazano na slici (Slika 5.55), deaktivirati grupu u kojoj se nalaze svi događaji koji kreiraju neprijatelje. Kada se ta akcija odradi, događaj tada vertikalno pokreće objekt *glavni neprijatelj* sve dok on potpuno ne uđe u ekran.



Slika 5.55 Događaji koji pripremaju borbu protiv glavnog neprijatelja

Na slici (Slika 5.56). prikazani su događaji koji su potrebni kako bi se glavni neprijatelj kontinuirano micao desno pa lijevo tijekom cijeli borbe. Logički blok koji se nalazi u grupi starter služi kako bi se objekt *glavni neprijatelj* počeo micati i nakon što odradi akcije za micanje deaktivira svoju grupu. Za nastavak pokreta rabe se dva nevidljiva objekta s pomoću kojih objekt *glavni neprijatelj*, ovisno o uvjetu koji je ispunjen, mijenja smjer kretanja.



Slika 5.56 Događaji koji omoguću kontinuirano kretanje lijevo-desno

Kako bi glavni neprijatelj opravdao da je vrijedan biti završna jedinica, njegovo naoružanje mora se nekako isticati u odnosu prema normalnim postrojbama. To je postignuto tako što je glavni neprijatelj u igri dobio pojačane verzije svih naoružanja kojima se obične postrojbe koriste. Rad tog naoružanja preko događaja prikazan je na slici (Slika 5.57). Pucanje radi na istom principu kao i kod normalnih postrojbi, ali u ovom slučaju potrebno je pokriti čak četiri topa od kojih dva imaju tri cijevi koje ispucavaju lasere u trima različitim smjerovima i jaču verziju rotirajućeg topa kakav ima objekt *neprijatelj1*.

Slika 5.57 Događaji koje glavni neprijatelj koristi za pucanje

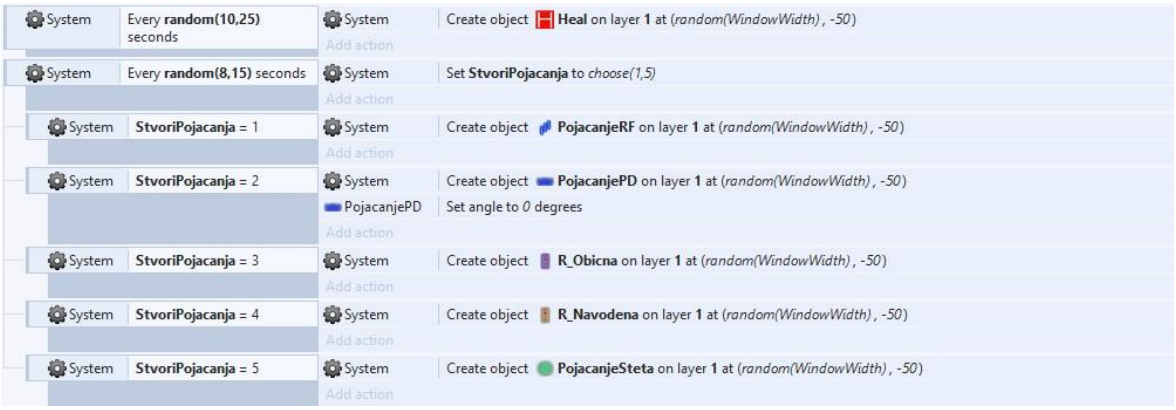
Ako igrač spusti bodove zdravlja glavnog neprijatelja na nulu, prelazi razinu i nakon što glavni neprijatelj eksplodira igrač dobiva mogućnost nastaviti dalje. U slučaju da je uništen završni (finalni) glavni neprijatelj, igrač pobjeđuje igru ostvarujući glavni cilj.

5.2.6. Pojačanja

Pojačanja se odnose na sve objekte u igri koji daju nekakav dodatan doprinos igraču. Njihova je uloga učiniti igru zanimljivijom i dinamičnijom. U ovoj igri takva se pojačanja dijele na:

- nadogradnje za lasere
- rakete
- regeneraciju zdravlja.

Pojačanja mogu biti kreirana nasumično ili ih se može kreirati uništenjem neprijatelja od kojih svaki ima neke svoje posebne parametre. Kreiranje se obavlja tako što sistem svaki nasumični broj sekundi u zadanom intervalu bira broj i pohranjuje ga u globalnu varijablu. Nakon što se broj pohrani u varijablu sistem će, ovisno o pohranjenom broju, kreirati objekt *pojačanja* čiji događaj zadovoljava uvjet (Slika 5.58). Svim objektima *pojačanja* potrebno je kao i kod neprijateljskih postrojbi dodati posebno prilagođeno ponašanje i prilagoditi im vertikalnu brzinu kako bi se omogućilo njihovo kretanje prema igraču.

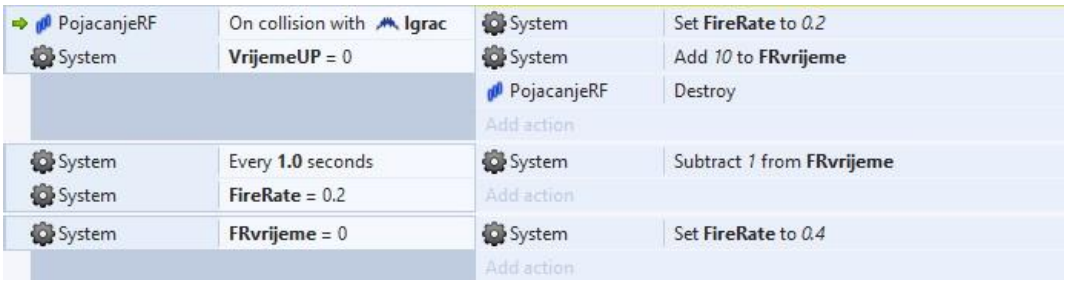


Slika 5.58 Događaji koji stvaraju pojačanja

Stvaranje pojačanja pri uništenju neprijatelja riješeno je slično kao i nasumično stvaranje. S obzirom na to da je, u slučaju uništenja neprijatelja, neprijatelj taj koji kreira pojačanja, potrebno je objektima *neprijatelja* dodati dvije varijable instancije. Prva varijabla instancije odnosi se na varijablu za koju će neprijatelj pri uništenju nasumično izabrati broj i, ako taj broj odgovara broju koji je naveden kao uvjet, onda će neprijatelj ponovno birati broj za drugu varijablu te će se kao i kod prethodnog primjera izvršiti onaj događaj kojemu je uvjet zadovoljen.

Svaki put kad igrač dođe u kontakt s objektom *pojačanja*, pojačanje se aktivira ili se samo pohranjuje, ovisno o tipu. Pojačanja koja se samo pohranjuju odnose se na rakete i zdravlje i, u slučaju sudara igrača s njihovim objektima, sistem samo dodaje određeni broj u globalne varijable koje su zadužene za pohranu vrijednosti raketa i zdravlja. Pojačanja koja se odnose na lasere nešto su kompliciranija zato što su osmišljena tako da imaju neko svoje vremensko trajanje i da mogu raditi u kombinaciji jedno s drugim.

Ubrzano je pucanje pojačanje koje utječe na to koliko često igrač može ispaliti laser. Tijekom ranije izrade umjesto fiksne vrijednosti za pucanje kreirana je varijabla koja se upotrebljava na njezinu mjestu i ta će varijabla sada doći do izražaja. Događaji za ubrzano pucanje prikazani na slici (Slika 5.59) omogućuju rad ubrzanog pucanja. Prvi događaj radi tako da, svaki put kad igrač sakupi pojačanje, sistem smanjuje vrijednost varijable za pucanje i time smanjuje vrijeme koje prođe između svakog pucnja. Sistem također globalnoj varijabli koja određuje vrijeme trajanja daje određenu vrijednost. Logički blok zatim svaku sekundu provjerava je li pojačanje još uvijek aktivno, odnosno je li vrijednost varijable ona koju pojačanje zadaje i oduzima jedan od varijable za vrijeme. Ako vrijeme istekne, posljednji će događaj pokrenuti akciju da se vrijednost varijable za pucanje vrati na početnu i time pojačanje prestaje djelovati.

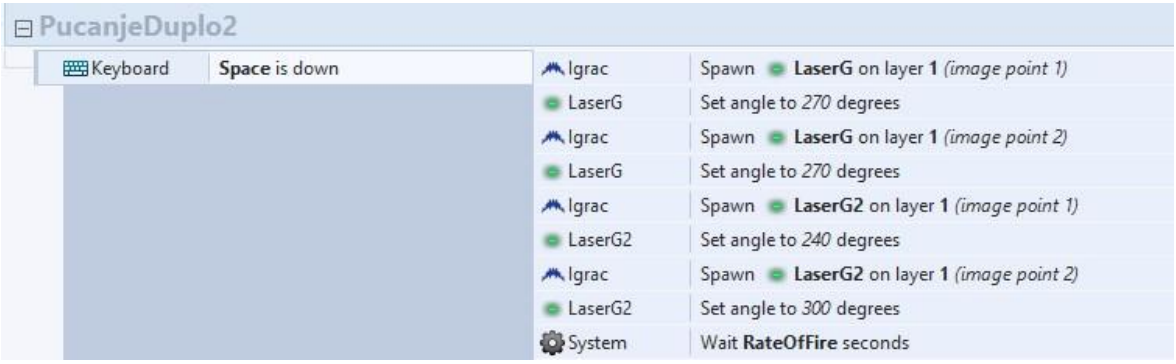


Slika 5.59 Događaji za ubrzano pucanje

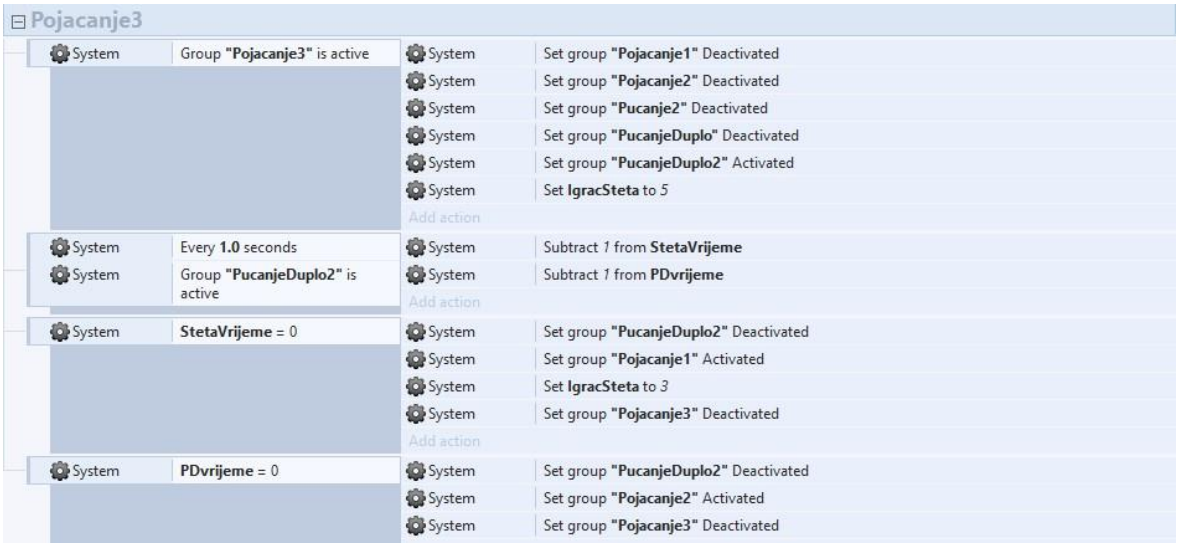
Osim ubrzanog pucanja, u poglavlju *Dizajn igre* navedeni su još *pucanje dvostruko* i *veća šteta*. Svako od tih pojačanja zasebno nije komplicirano izvesti, ali, kako bi im se omogućilo da u određenom trenutku djeluju zajedno, potrebno je napraviti određene pripreme. Treba odrediti sve moguće načine na koje igrač može pucati i podijeliti ih u grupe. Razlog tomu jest to što je svaku grupu moguće aktivirati i deaktivirati prema potrebi. Za ovu je igru pucanja bilo potrebno podijeliti u četiri grupe. Osim osnovne, potrebne su:

- grupa za dvostruko pucanje
- grupa za pucanje pojačanih lasera
- grupa za dvostruko pucanje pojačanih lasera

Te tri grupe koje se odnose na pojačanja započinju deaktiviranje kao na slici (Slika 5.60). Kako bi se grupe u potrebnom trenutku aktivirale i poslije deaktivirale, potrebne su još tri grupe po jedna za svaku prethodno navedenu koja će ih aktivirati. Na taj se način izbjegava mogućnost da više grupa za pucanje djeluje odjednom, a to je problem koji bi se bez njih pojavio u trenutcima pokretanja i deaktiviranja grupe koja puca dvostruko pojačane lasere. Primjer jedne od tih grupa prikazan je na slici (Slika 5.61) i ona prikazuje grupu koja se aktivira kad je jedno od pojačanja sakupljeno, dok drugo još uvijek djeluje. Ta grupa djeluje tako da, nakon što se aktivira, deaktivira sve grupe vezane za druga pojačanja, aktivira grupu pucanja kojoj je namijenjena i postavlja globalnu varijablu za štetu na veću vrijednost, što je potrebno ako je pojačanje za veću štetu pojačanje koje je pokrenulo grupu. Dok god je grupa za pucanje dvostruko aktivna, sistem će svake sekunde oduzeti jedan iz globalnih varijabli koje pohranjuju vrijednost jednaku preostalom vremenu rada te će se, ovisno o tome čija vrijednost prva padne na nulu, pokrenuti događaj koji će deaktivirati trenutačnu grupu i aktivirati grupu za pojačanje koje još ima preostalog vremena. Grupe za pojedinačna pojačanja djeluju slično kao i opisana, samo rade na jednostavniji način i nakon isteka vremena deaktiviraju grupe za pucanje pojačanja i vraćaju se na početnu grupu s kojom igrač započinje.

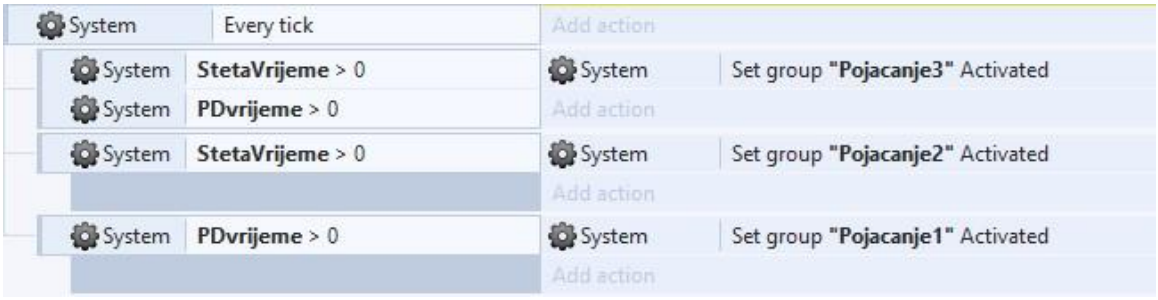


Slika 5.60 Grupa koja sadržava događaj za dvostruko pucanje i veću štetu



Slika 5.61 Grupa s događajima koji omogućuju djelovanje dvaju pojačanja istodobno

Pokretanje grupa koje aktiviraju grupe za pucanje prikazano je na slici (Slika 5.62). Ti događaji djeluju tako da sistem cijelo vrijeme provjerava vrijednosti globalnih varijabli za pojačanja i, kada igrač pokupi pojačanje i doda određenu vrijednost nekoj varijabli, sistem će to odmah prepoznati i, ovisno o njihovim vrijednostima, aktivirati događaj kojemu je uvjet ispunjen.



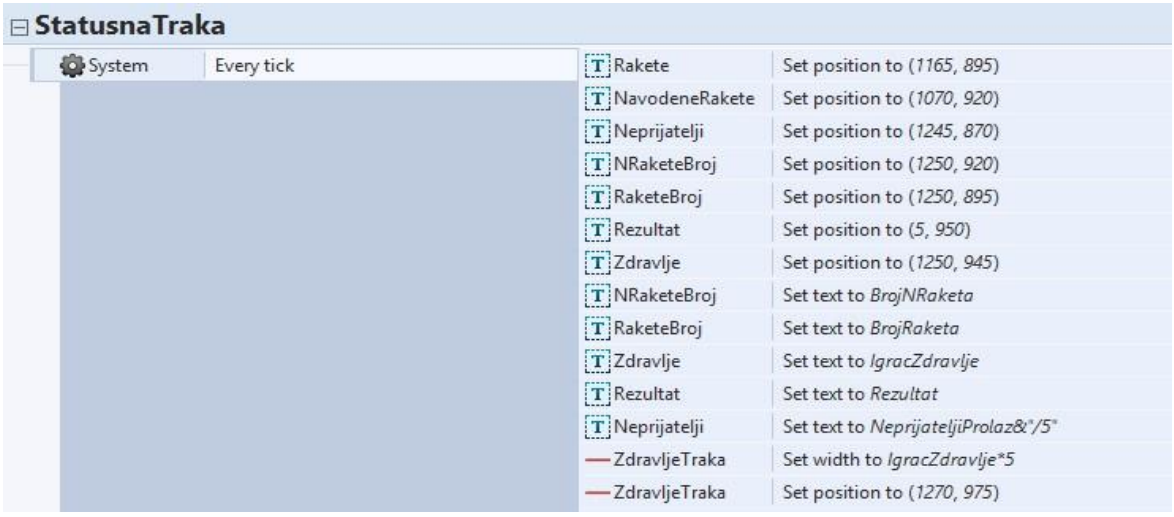
Slika 5.62 Događaji koji aktiviraju pojačanja

5.2.7. Statusna traka

Statusna traka predočuje sve podatke koje igrač može pratiti dok igra igru. Ti se podatci dijele na:

- rakete
- poseban napad
- zdravlje
- rezultat
- neprijatelje koji su prošli.

Događaj potreban da bi se statusna traka na ekranu prikazala onako kako je zamišljena prikazan je na slici (Slika 5.63). To se uglavnom sve svodi na jedan događaj koji sadržava akcije za svaki objekt koji pohranjuje neki podatak. Tekstove koji su statični moguće je odmah upisati u parametar koji se nalazi u obilježjima objekta, a za tekstove koji se mijenjaju potrebno je koristiti se globalnim varijablama koje je moguće ispisati kao tekst. Ako je potrebno, moguće je postaviti parametar tekst kao spoj statičnog teksta i globalne varijable tako da se statični tekst stavi u navodnike i s pomoću & prikaže uz vrijednost globalne varijable. Kao nekakva dodatna značajka za praćenje zdravlja dodana je i traka čija se duljina mijenja pri svakoj promjeni vrijednosti zdravlja igrača.



Slika 5.63 Događaji za ispis svih podataka za statusnu traku

Slika 5.64 Statusna traka prikazana u igri

5.2.8. Izbornici

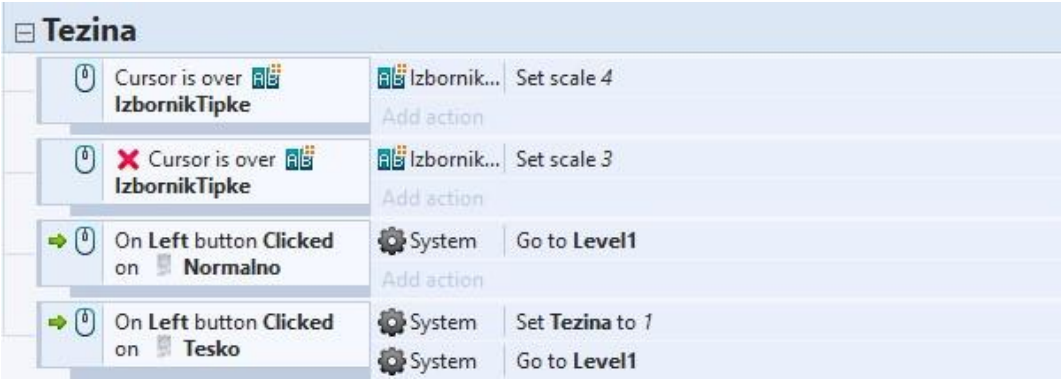
Glavni izbornik (Slika 5.65) prvo je što se pokreće kad igrač pokrene igru. Glavni izbornik pruža igraču mogućnosti da pogleda kontrole ili da započne novu igru. Kao zamjena za priključak *tekst* korišten je priključak *sprite font* koji ujedno služi i kao gumb na koji se može kliknuti mišem. Događaji s pomoću kojih izbornik radi prikazani su na slici (Slika 5.66) i oni rade tako da sistem ode na plan ovisno o tome na koji gumb je igrač kliknuo. Dodatna interaktivnost gumbima ostvaruje se tako da se veličina tipke malo poveća svaki put kada igrač dođe mišem iznad tipke. Kako bi se malo uštedjelo na događajima za povećavanje tipki, svi *sprite font* objekti postavljeni su u jednu familiju koja čini grupu tipova objekta. Način dodavanja objekata u familiju prikazan je na slici (Slika 5.67).

Slika 5.65 Prikaz glavnog izbornika

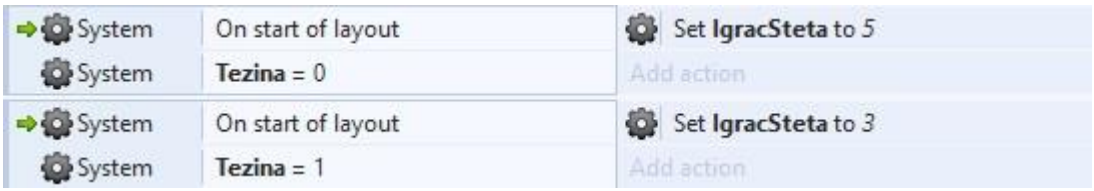
Slika 5.66 Događaji za glavni izbornik

Slika 5.67 Dodavanje tipova objekata u familiju

Ako igrač odluči započeti novu igru, ponudit će mu se novi izbornik u kojemu će mu se pružiti mogućnost da odabere između dviju težina igre. Težina igre odabire se tako da sistem, ovisno o tome koji gumb igrač klikne, postavlja globalnu varijablu za težinu igre na određenu vrijednost i ta se vrijednost onda poslije upotrebljava kao dodatni uvjet u određenim događajima. Primjer utjecaja na težinu prikazan je na slici (Slika 5.69). Ako igrač odabere lakšu verziju, pri pokretanju razine pokrenut će se događaj u čijem je logičkom bloku uvjet da je vrijednost globalne varijable za težinu jednaka onoj koju postavlja klik na lakšu verziju i laseri kojima igrač puca nanosit će više štete neprijateljskim postrojbama nego što bi je uzrokovali laseri u težoj verziji.



Slika 5.68 Događaji za odabir težine

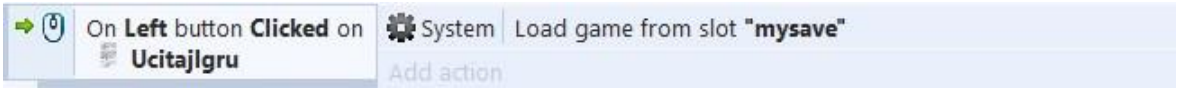


Slika 5.69 Događaji za određivanje štete

Osim toga da započne igru, igraču su ponuđene i ove mogućnosti:

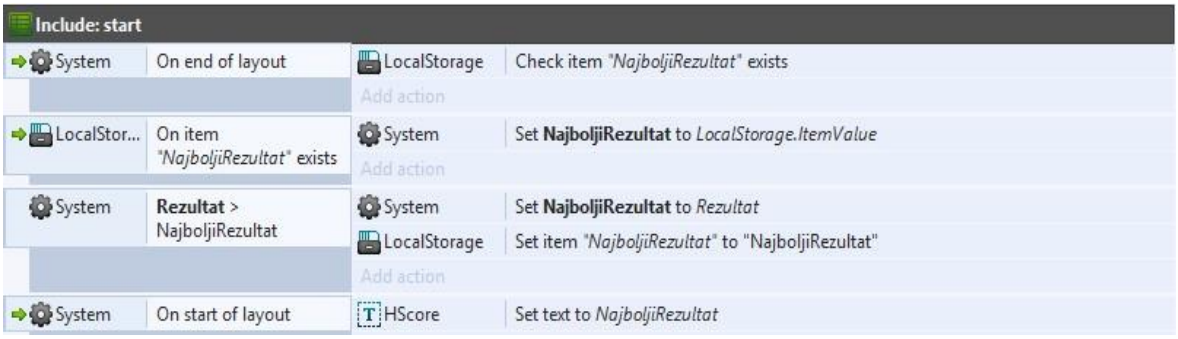
- pogled na kontrole
- učitavanje spremljene igre
- pogled na najbolji rezultat.

Učitavanje igre izvodi se tako da, kada igrač klikne na *učitaj igru*, sistem učitava sve spremljene podatke vezane za igru s mjesta *mysave*. S pomoću ove mogućnosti igrač može spremiti igru nakon prelaska razine, isključiti igru i poslije nastaviti s mjesta na kojemu je stao.



Slika 5.70 Događaj za učitavanje igre

Za točan prikaz i pohranu najboljeg rezultata potrebni su događaji sa slike (Slika 5.71). Ti događaji prikazuju način pohrane vrijednosti u lokalni spremnik i prikaz te vrijednosti. Pohrana radi tako da provjerava postoji li element *najbolji rezultat* i, ako postoji, sistem pohranjuje vrijednost globalne varijable *najbolji rezultat* u lokalni spremnik. Ako igrač postigne rezultat bolji od trenutačno najboljeg, sustav u globalnu varijablu *najbolji rezultat* upisuje vrijednost novog rezultata i pohranjuje tu vrijednost. Kada igrač u glavnom izborniku klikne na najbolji rezultat i uđe u plan, prikazat će mu se vrijednost koja je pohranjena u lokalni spremnik.



Slika 5.71 Događaji za prikaz i pohranu najboljeg rezultata

Ostali se izbornici odnose na moguće ishode igre:

- prelazak razine
- neuspješni prelazak razine
- prelazak zadnje razine, odnosno igre.

Kako bi se pokrenuo izbornik za prelazak razine, igrač mora uspješno završiti borbu s glavnim neprijateljem. Izbornik se pokreće s pomoću događaja (Slika 5.72) koji radi tako da, pošto je glavni neprijatelj uništen, sistem čeka nekoliko sekundi dok se ne odrade eksplozije i onda prije nego što ode na plan izbornika postavi sve potrebne vrijednosti globalnih varijabli na njihove početne i aktivira grupu koja stvara neprijatelje na novoj razini. Iako program pruža opciju ponovnog postavljanja svih globalnih varijabli na njihove početne vrijednosti, to u ovoj igri ne bi funkcioniralo. Razlog je u tome što bi se onda, primjerice, ako igrač odabere težu verziju nakon prelaska razine, težina postavila na početnu vrijednost, odnosno na vrijednost lakše razine.

Slika 5.72 Događaj za postavljanje globalnih varijabli prije prelaska razine

Izbornik koji se pokreće nakon prelaska razine nudi igraču ove mogućnosti:

- nastavak igre
- promjena težine
- nadogradnja lika
- spremanje igre.

Od tih četiriju mogućnosti sve, osim nadogradnje lika, rade kao i na početnom izborniku. Nadogradnja lika sastoji se od traki koje predočuju status specifikacija i gumba koji služi kako bi se neka od njih nadogradila. Nadogradnja lika djeluje tako da se, kada igrač klikne na gumb za nadogradnju, traka koja prikazuje status poveća kao što je prikazano na slici (Slika 5.73). Za nadogradnje su također potrebne i globalne varijable čija se vrijednost uporabljuje pod uvjetima da igra može pravilno određivati vrijednosti ne samo pri pokretanju plana nego i nakon isteka pojačanja koja su skupljena tijekom igre.

Izbornik, uz mogućnosti, daje igraču i pogled na trenutačni rezultat i pojačanja koja zadržava nakon prelaska razine. Ako igrač prijeđe posljednju razinu, pokreće se izbornik koji igraču

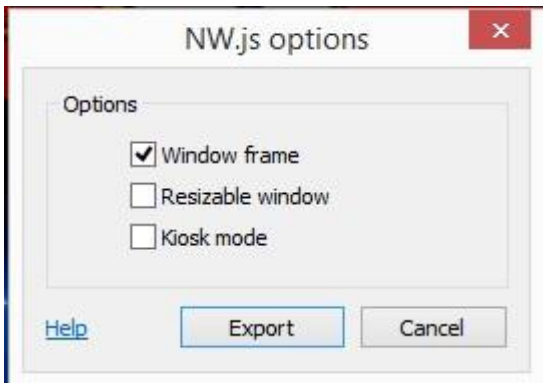
daje na znanje da je pobijedio, daje mu pogled na završni rezultat i nudi mu mogućnost da se vrati na početni izbornik.

Slika 5.73 Događaj za nadogradnju pucanja

Izbornik za neuspješni prelazak razine pokreće se ako je objekt *igrač* uništen, odnosno ako je igrač izgubio igru. Sistem prije njegova pokretanja također ponovno postavlja vrijednosti određenih globalnih varijabli kao i u prethodnom primjeru, ali je razlika u tome što u ovom slučaju mora vratiti još i sve varijable pojačanja na njihove početne vrijednosti. Ovaj izbornik igraču nudi mogućnost da ponovno započne igru i promijeni težinu.

5.3. Finalizacija igre

Kako bi se izrađena igra spremila za upotrebu, potrebno ju je izvesti. *Construct 2* nudi više raznih mogućnosti, ali za potrebe ove igre ona je izvezena u NW.js[11]. NW.js u ovom je slučaju samostalna verzija *Google Chrome browsera* koji prikazuje samo projekt. Pri izvozu ponuđene su tri opcije (Slika 5.74) s pomoću kojih se bira način prikaza.



Slika 5.74 Opcije izvoza igre

Odabirom NW.js opcije za izvoz, *Construct 2* igru izvozi za *Linux*, *Mac* i *Windows* platformu. Datoteke za svaku platformu spremljene su u zasebne foldere i u svakom folderu, osim raznih datoteka, nalazi se i NW aplikacija preko koje se pokreće igra. Pokrenuta se igra

[11] Okruženje za izradu *web*-aplikacija

igraču prikazuje u prozoru kao na slici (Slika 5.75) kojemu je dana opcija za promjenjivu veličinu (engl. *resizable*) kako bi se riješio problem crnih rubova koji se pojavljuje pri izvozu s NW.js opcijom. Još jedan problem kod NW.js opcije jest nemogućnost postavljanja ikone za aplikaciju, nego se ona uvijek izvozi s već definiranom ikonom. Pri ovoj igri taj je problem riješen s pomoću aplikacije za uređivanje resursa[12] koja daje mogućnost promjene zadane ikone. Za promjenu ikone potrebna je slika u ICO[13] formatu u koji se s pomoću *on-line* pretvarača može pretvoriti slika koja je u formatu PNG.

Slika 5.75 Ikona prečaca za pokretanje igre

[12] http://www.angusj.com/resourcehacker/
[13] *Microsoft Windows* format za kompjutorske ikone

Slika 5.76 Prikaz gotove igre pokrenute kao aplikacija

6. Testiranje igre

Testiranje igre proces je koji se obavlja za vrijeme izrade igre i nakon njezine izrade, a svrha mu je poboljšanje cjelokupne kvalitete igre i provjera koliko je ona spremna za lansiranje.

S pomoću testiranja igri se:

- otklanjaju greške
- poboljšavaju elementi i funkcionalnosti
- dodaju dodatne nadogradnje

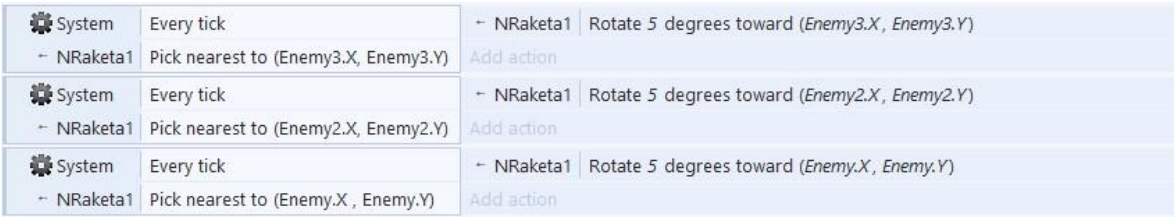
6.1. Testiranje kroz izradu

Testiranje kroz izradu počinje dodavanjem i programiranjem prvog objekta i nastavlja sa svakim novim objektom. Kako se dodaju novi objekti i time nove funkcionalnosti, tako i testiranje postaje sve opsežnije zbog sve većega broja objekata čiji se način rada mora pratiti. Takvo je testiranje prije svega odgovorno za otklanjanje grešaka i kako bi se osiguralo da se svaki element ponaša onako kako je zamišljen.

Construct 2, uz normalno pokretanje igre, u pregledniku ima i mogućnost pokretanja u načinu za otklanjanje grešaka (engl. *debug*). Primjenom tog načina moguće je u vrijeme igre pratiti trenutačno stanje svih globalnih varijabli i na taj se način najlakše mogu detektirati pogreške kod svih elemenata igre koji se koriste globalnim varijablama. Uz praćenje globalnih varijabli, također je moguće pratiti i performanse igre, parametre svakoga sloja koji se trenutačno nalazi na planu i parametre svakog objekta zasebno. Radi lakše preglednosti moguće je označiti i samo one parametre koje se trenutačno žele pratiti odnosno one koji se odnose na objekte i njihove funkcionalnosti koje se trenutačno testiraju.

Osim izravnog otklanjanja grešaka, tijekom testiranja kroz izradu također se mogu unositi izmjene na elementima i funkcionalnostima koji ne rade nužno pogrešno, ali ne rade točno onako kako su zamišljeni. Primjer toga nalazi se na slici 6.1. koja prikazuje način navođenja navođenih raketa prije dodavanja objekta *vodiča* kao pomoć pri navođenju. Takav bi način djelovao radio odlično ako se samo jedan tip neprijatelja nalazi na ekranu, ali, s obzirom na to da se u ovoj igri mogu pojaviti tri ili više različitih neprijatelja, to nije slučaj. Problem kod ovog načina bio je u tome što su navođene rakete, ako je više neprijatelja bilo na ekranu, uvijek imale prioritet na neprijatelja koji je bio naveden u događaju koji prvi dolazi na

izvršavanje. Iako bi igrač tijekom igre osjetio prednost i ovako navođenih raketa u usporedbi s nenavođenima, one ipak ne bi djelovale onako kako su zamišljene te su stoga događaji za njih promijenjeni kao što je prije objašnjeno na slici 5.49.



Slika 6.1 Događaji za navođenje rakete prije promjena

Osim popravljanja raznih elemenata igre, tijekom ovakvog testiranja isto je tako važno pratiti težinu igre i njezin cjelokupni balans. Težina i balans igre primjerice mogu ovisiti o broju neprijatelja koji se stvaraju i o tome koliko je teško ubiti svakog od njih.

6.2. Testiranje na korisnicima

Nakon što je igra izrađena korisno ju je prije lansiranja dati određenoj grupi ljudi na testiranje. Tijekom testiranja korisnici mogu uočiti greške i nedostatke koji nisu uočeni u tijeku izrade, ali i dati korisne povratne informacije vezane za sam dizajn igre, koje mogu pridonijeti boljem igraćem iskustvu.

Testiranje izrađene igre provelo je pet ispitanika, sve redom iskusni igrači igara. Primarni se cilj ovog ispitivanja bili su provjera funkcionalnosti i otkrivanje nedostataka u dizajnu igre, odnosno utvrđivanje koje bi promjene bilo dobro napraviti u svrhu poboljšanja kvalitete igraćeg iskustva. Od svakog se ispitanika zahtijevalo igru odigra otpočetka pa do kraja te da nakon toga odgovori na postavljena pitanja ocjenama od jedan do sedam, uz kratko obrazloženje ocjene.

Ispitivanje se sastojalo od sljedećih pitanja.

1. Jesu li razine dovoljno zahtjevne?
2. Sviđa li vam se nasumičan način stvaranja neprijatelja?
3. Sviđa li vam se način interakcije s neprijateljem?
4. Jesu li sve borbe protiv glavnih neprijatelja dobro dizajnirane (način kretanja, način pucanja, koliko ih je teško uništiti)?
5. Da li sva pojačanja i dodatni napadi pozitivno utječu na način igranja?

6. Koje je vaše općenito mišljenje nakon odigranih triju razina i što biste voljeli vidjeti ako se kreira još razina i proširi priča?

Tablica 6.1 Rezultati testiranja

	Razine	Stvaranje neprijatelja	Interakcija s neprijateljem	Glavni neprijatelji	Napadi i pojačanja
Ispitanik 1	4	4	4	5	6
Ispitanik 2	5	4	5	5	6
Ispitanik 3	4	4	4	4	5
Ispitanik 4	5	5	5	6	6
Ispitanik 5	4	4	5	5	5
Prosjek	4,4	4,2	4,6	5	5,6

Na temelju rezultata testiranja dolazi se do zaključka da je od elemenata igre koji su testirani najveći nedostatak pomalo osrednja zahtjevnost i angažman koji su rezultat ne baš najboljeg načina kreiranja neprijatelja. Najpozitivniju ocjenu postigli su napadi i pojačanja, iz čega se može zaključiti da se dodavanjem takvih dodatnih obilježja može pozitivno utjecati na kvalitetu igre. Drugi dio testiranja zahtijevao je od ispitanika da još jednom odigraju igru u svrhu praćenja grešaka koje nisu bile primijećene tijekom njezine izrade. Svaki bi od ispitanika nakon što primijeti grešku s pomoću *pauze* zaustavio igru i dodao grešku na popis grešaka. Primjer greške koju su uočili i otklonili ispitanika jest pojačani laser koji se nije uništavao u kontaktu s jednim od glavnih neprijatelja.

6.2.1. Budući razvoj igre

Šesto pitanje postavljeno ispitanicima usredotočeno je isključivo na komentare ispitanika i ne zahtijeva ocjenu. Cilj tog pitanja jest dobiti pozitivne i negativne komentare na temelju

kojih bi se planirao daljnji razvoj igre. Osim problema oko dizajna razina i stvaranja neprijatelja, većini je ispitanika nedostajao izbor između likova kojima bi mogli igrati. Na osnovi toga daljnji razvoj mogao bi se usmjeriti prema tome da se nastavak priče prilagodi izboru više likova.

Tema igre jest da igrač tijekom triju razine osvaja jedan planet i pobjeđuje glavnog neprijatelja. Pri budućem razvoju priča bi se mogla proširiti tako da, pošto igrač osvoji prvi planet, dobije mogućnost da osvoji i druge te da osvajanjem tih planeta dobije mogućnost korištenja drugim tipovima lovaca za glavnog lika koji se pojavljuju jer je igrač osvajanjem planeta stekao i tehnologiju koja se tamo razvijala. Ti, novi tipovi lovaca, osim razlike u izgledu, posjedovali bi drugačije napade i specifikacije kako bi se međusobno što više razlikovali. Uz nove tipove lovaca također bi se mogla dodati i nova pojačanja, jer su ona vrlo pozitivno ocijenjena, te prošireni sustav nadogradnje glavnog lika. Da bi se ova proširenja omogućila, bilo bi potrebno redizajnirati razine i njihove glavne neprijatelje kako bi se tri već napravljene razine prilagodile postojanju razine iza njih.

7. Zaključak

Ovaj je završni rad izrađen sa svrhom da se objasni cjelokupni proces razvoja videoigre na praktičnom primjeru. Razvoj igre *Soul Star Destroyer* započeo je osmišljavanjem njezine priče i na temelju te priče osmišljeni su likovi te su opisani napredak kroz igru i cilj igre i napravljen je opis mehanika koje će se nalaziti u igri. Igra je dizajnirana tako da dijeli mnoge sličnosti s igrama istog žanra, ali se po tome što je moguće igru izgubiti a i da se glavni lik ne uništi, ona ipak razlikuje od ostalih igara. U nastavku je objašnjen način primjene računalne aplikacije *Photoshop* pri izradi raznih tipova 2D objekata koji će se poslije iskoristiti pri izradi same igre.

Uz spreman dizajn igre i objekte koji će se upotrijebiti počinje najzahtjevniji dio razvoja, a to je izrada igre. Pri izradi igre uporabljen je alat *Construct 2* koji je posebno dizajniran za izradu 2D igara. Iako ne zahtijeva znanje programskoj jezika kako bi se upotrebljavao, potrebno je dobro proučiti priručnik i svladati dijelove programa koji su u ovom radu opisani prije opisa izrade igre. Pri opisu izrade igre objašnjeno je kako je svaki pojedini njezin element realiziran s pomoću sustava događaja koji su u programu *Construct 2* zamjena za klasično programiranje. Iako je *Construct 2* odličan za izradu jednostavnih 2D igara i svladavanje logika po kojima igre djeluju, pri radu s velikim brojem događaja, objekata i varijabli prilikom izrade igre lako je naići na probleme zbog loše organizacije događaja i lošeg planiranja. Pri izradi potrebno je gledati ne samo ponaša li se pojedini objekt kako treba nego valja razmišljati o tome kako bi na njegovo ponašanje mogao utjecati neki od idućih objekata koji će se dodati u igru. Prilikom izrade također je bilo iznimno važno napraviti dobru logiku za ponašanje svakoga pojedinog objekta kako bi se realiziralo točno ono što je zamišljeno. Gotova igra na kraju izrade izvozi se kao desktop aplikacija (NW.js) za *Linux*, *Mac* i *Windows* platformu.

Nakon izrade igru je testiralo pet ispitanika i, prema rezultatima tog testiranja, isplanirana su poboljšanja izvedbe trenutačnih obilježja i moguće nadogradnje igre. Tim bi se nadogradnjama proširio sadržaj igre te bi se igraču omogućilo da tijekom igranja iskusi veću raznolikost.

Popis kratica

2D	Two-dimensional	Dvodimenzionalno
HTML	HyperText Markup Language	Hipertekst označni jezik
PNG	Portable Network Graphics	Prijenosna mrežna grafika
WebGL	Web Graphics Library	Knjižnica web grafika
UID	Unique identifier	Jedinstveni identifikator
DT	Delta-Time	Delta-vrijeme

Popis slika

Popis tablica

Literatura

[1] https://gamicus.gamepedia.com/Shoot_%27em_up_video_games

[2] https://www.gamedesigning.org/learn/basic-game-mechanics/

[3] https://helpx.adobe.com/uk/photoshop/user-guide.html

[4] https://www.scirra.com/construct2

[5] https://www.scirra.com/manual/1/construct-2

Prilog

CD sa završnim radom.

ALGEBRA
VISOKO UČILIŠTE

Razvoj računalne 2D igre

Pristupnik: Matija Pufnik, 0321004559

Mentor: Prof. dipl. ing. Predrag Šuka