Izrada strateške igre u programskom alatu Game Maker Studio 2

Štulić, Jakov

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:033697

Rights / Prava: Attribution 3.0 Unported/Imenovanje 3.0

Download date / Datum preuzimanja: **2025-03-14**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Jakov Štulić

Izrada strateške igre u Game Maker Studiju 2

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Jakov Štulić

Matični broj: 44867/16–R

Studij: Informacijski sustavi

Izrada strateške igre u Game Maker Studiju 2

ZAVRŠNI RAD

Mentor/Mentorica:

Dr. sc. Mladen Konecki

Varaždin, lipanj 2019.

Jakov Štulić

Izjava o izvornosti

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

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

_____

Sažetak

Tema rada je izrada igre u GameMaker Studiju 2, programu specijaliziranom za izradu igri. Obrađivati će se teme gotovo svih aspekata izrade desktop igre koja nema posebnih online funkcionalnosti. Posebna pažnja će biti posvećena: programu za izradu, inspiraciji, kodu, izgledu, zvukovima i funkcionalnosti. Rad bi trebao poslužiti bilo kojoj osobi koju zanima izrada igri u Game Maker Studiju 2, posebice 2D igri za koje se alat specijalizira iako pruža i 3D mogućnosti. Treba imati na umu da ovo nije potpuni vodič za izradu slične igre stoga će se obraditi alati i glavne teme, ali ne i svi koraci izrade.

Ključne riječi: igre; izrada; gms2; strategija; kod; skripte; zvukovi; rešetka; sprite;

Sadržaj

1. Uvod

Stvaranjem igre u alatu GameMaker Studio 2, ovaj završni rad prikazuje proces izrade igre na nižoj razini složenosti te većinu relevantnih uspona i padova pri izradi ovakvih igri. U procesu je u cijelosti izrađen velik udio elemenata igre, s izuzetkom zvukova. Igra je strateškog tipa po uzoru na igre koje sam igrao u djetinjstvu, a od kojih se ističe Heroes of Might and Magic serijal. Poneki moderniji elementi nataknuti su na već postojeće temelje i principe igre.

Alat sam odabrao jer sam se s istim prethodno susreo na manjim zadatcima kojima sam se bavio u slobodno vrijeme. Korisničko sučelje je jednostavno i nalikuje ostalim alatima iste svrhe dok je dokumentacija dovoljna kako bi razjasnila skoro sve probleme i nedoumice. Alat je već uspješno korišten od strane manjih grupa ili samostalnih developera, a primjeri uključuju popularni Undertale, Hyper Light Drifter i mnoge druge.

Rad se u manjoj mjeri bavi estetskim dijelovima igre zbog osobnog manjka iskustva u tom području. Do te odluke sam došao iz dva razloga: željom za istraživanjem novih područja izrade igri te specifičnih artističkih potreba igre. Zvukove u igri nisam osobno snimao već su preuzeti s interneta, dok je glazba napravljena uz pomoć besplatnih alata.

2. Metode i tehnike rada

Zbog prirode rada prvo je u potpunosti napravljena igra te je tek po završetku izrade i testiranja krenulo pisanje. Nije korištena nikakva posebna metoda ili tehnika, već samo iterativno poboljšavanje do granice s kojom sam zadovoljan. Većina stvari koje se u procesu izrade igri smatraju važnim, osobama u sličnim prilikama i autoru osobno, su dokumentirane.

Glavni alat za izradu je GameMaker Studio 2 što je vidljivo iz imena rada („GameMaker Studio 2“, bez dat.). Alat sadrži mogućnosti crtanja uz već postojeće mogućnosti prisutne u svim sličnim programima. Ipak, kao pomoćni alat za crtanje korišten je Pyxel od strane Per Nybloma, radi funkcionalnosti koje nisam pronašao u već integriranom dijelu Game Maker Studia 2 (Kvartford, bez dat.). Za generiranje glazbe upotrijebljen je online alat Abundant Music razvijen od strane Daniela Kvarforda i koji je besplatan za korištenje (Nyblom, bez dat.). Rezultati ovih programa nisu fokus rada već njihova implementacija u igru. Programski jezik GML (eng. *Game Maker Language*) specifičan je za korišteni alat. Za sve informacije u vezi jezika poslužila je službena dokumentacija dostupna na internetu i unutar alata („GameMaker Studio 2 Docs“, bez dat.).

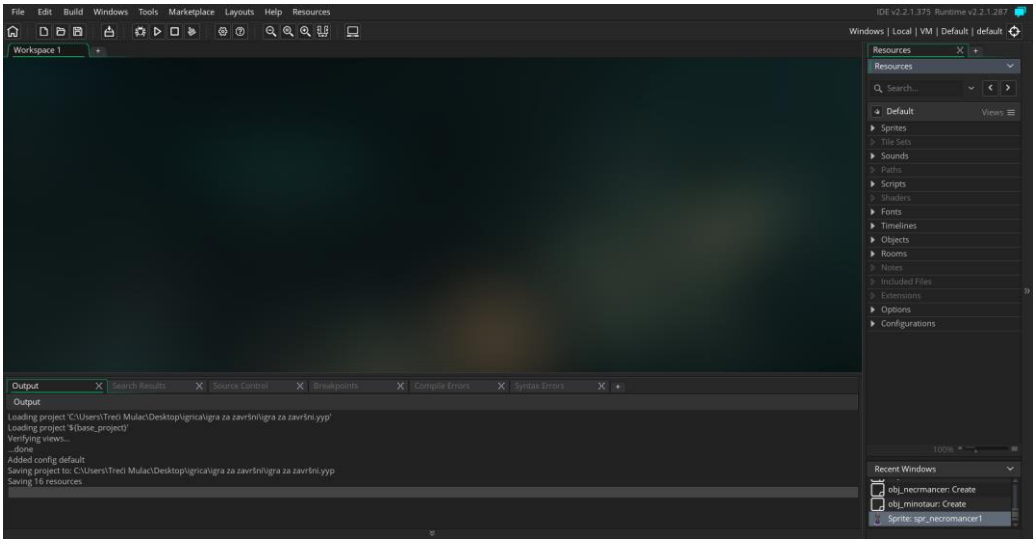
3. GameMaker Studio 2

GameMaker Studio 2 je razvojni alat za igre koje se mogu igrati na više platformi. Trenutno je u posjedu Yoyo Studioa, a ime je mijenjao više puta. Inicijalno je nazvan „Animo“ pri izlasku na tržište 1999. godine po izvoru (GameMaker Studio, 28.03.2019). Trenutno je moguće koristiti dvije glavne verzije programa, verziju 1 i verziju 2. Prva starija verzija nije korištena već je sve napravljeno u drugoj verziji. U drugoj verziji postoje dva važna stila izrade igri: Drag and Drop i GML, od kojih je odabran GML. Do tog je izbora došlo zbog jednostavnije implementacije naprednih mehanika. Drag and drop ima svoje prednosti posebno pri izradi manjih projekata niže razine složenosti. Iako postoji mogućnost prilagodbe igre za više platformi, u našem slučaju igra je napravljena isključivo za korištenje na osobnom računalu. Program je inicijalno napravljen kako bi izradu igri napravio što lakšom za početnike, štoviše postoji licenca za studente kao što je vidljivo na stranici alata (GameMaker Studio 2, bez dat.). GameMaker Studio 2 je fokusiran na izradu 2D igri po kojima je i poznat. Navedeno se da zaključiti iz trenda igri koje dolaze na tržište, a izrađene su u alatu.

Trenutno postoji više licenci, a od njih je korištena „Creator“ licenca. Sadrži sve nužno za potrebe ovog rada uz nedostatak manjka naprednih alata za testiranje i slično. Skuplje licence nisu namijenjene za obične korisnike već napredni rad.

3.1. Izgled i korisničko sučelje

Korisničko sučelje nalik je sučeljima u alatima slične ili identične svrhe, recimo Unitiy. Na slici broj 1 nalazi se izgled sučelja prilikom stvaranja ovog projekta gdje su svi prozori zatvoreni. Ukoliko ste upoznati sa sličnim programima, snalaženje je poprilično jednostavno. S desne strane vidljiva je lista svih resursa od važnosti za projekt: objekti, sprite, skripte, sobe, zvukovi. Osobno bih preporučio korištenje sufiksa ili prefiksa imena resursa umjesto prisutne pretrage. Na dnu slike se vidi output prozor kod kompajliranja i rada igre, korišten pretežno za debugiranje, preformanse i tome slično. Postoje i druge kartice koje se otvaraju kod rada sa specifičnim resursima, primjerice sobama. Dodatno, prisutni sprite editor pokazao se poprilično korisnim, posebice za male izmjene gdje se korištenjem editora štedi vrijeme. Dalje u specifične elemente se neće ulaziti detaljno jer je alat naime samo popratni dio rada.

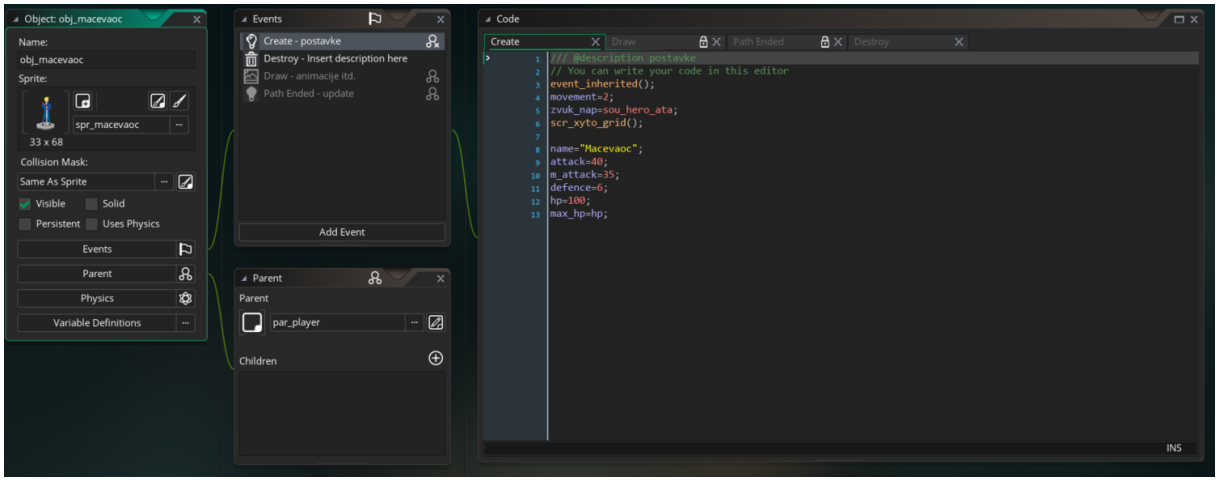


Slika 1: Sučelje (Izvor: Vlastita produkcija)

3.2. Objekti i GameMaker Language

GameMaker Language je skriptni jezik koji se koristi u potpunosti kod već spomenutih GML pristupa, a kod Drag and Drop pristupa izrade u pozadini. Jezik ima sličnosti s interpreterskim jezicima poput JavaScripta upravo zbog načina izvođenja i nedostatka tipova varijabli. U pogledu izgleda sintakse, primjetne su sličnosti s jezicima na bazi C-a kao što su sveprisutni C# i C++ u sličnim alatima. Osobno nisam imao nikakvih problema u prilagodbi na ovaj jezik s već poznatih programskih jezika. Smatram da je to rezultat dobre dostupne dokumentacije, a zatim i količine izvora, intuitivnosti i suradnje sa sučeljem.

Objekti su glavni dio funkcionalnosti igre, sve mehanike se postižu događajima (eng. *event*) u koje se piše kod za isti. Na slici broj dva koja je niže, vidimo objekt mačevaoca s relevantnim prozorima već otvorenima. Krenuvši od samog objekta, on ima određena svojstva i potencijalno pridružen sprite koji ga predstavlja. Sljedeće su veze roditelja i djece koje prvenstveno služe grupiranju objekata ili kako bi objekt koristili kao sučelje (eng. *interface*). U danom primjeru par_player je objekt koji se nasljeđuje i sadrži sva svojstva zajednička svim jedinicama u posjedu igrača. Svojstva se kasnije proširuju i potencijalno dalje nasljeđuju. Nadalje, vidljiv je otvoren događaj Stvori (eng. *Create*) koji se događa pri kreiranju objekta. U slučaju ovog objekta koristi se da bi se postavile osnovne varijable na pretpostavljene vrijednosti. Postoje brojne vrste događaja koje su već uključene u alatu, no ukoliko nisu dostatne ili pogodne mogu se simulirati i nove uz pomoć skripti, tajmera i tako dalje. Kao najvažnije događaje za rad bih istaknuo: Stvori (eng. *Create*), Uništi (eng. *Destroy*), Nacrtaj (eng. *Draw*) te one vezane uz putove (eng. *Path*).



Slika 2 : Rad s Objektima (Izvor: Vlastita produkcija)

3.3. Editor soba

Sobe (eng. *rooms*) su mjesta gdje se odvija cjelokupna radnja igre stoga je nužno imati barem jednu. Na slici broj 3 je prikazan prozor unutar alata za rad sa sobama, u ovom slučaju sobe prve razine. Sprite prvog nivoa je pridružen sobi kao pozadina te se za isti postavlja odgovarajuća veličina sobe. Objekti se mogu postavljati direktno u sobe, a ne isključivo generirati tokom rada igre, u primjeru su tri objekta. Prvi objekt obj_control predstavlja simbol upitnika što označava da je dani objekt nevidljiv tokom igre. Za pravilno smještanje objekata potrebno je odabrati odgovarajuću veličinu rešetke. Sljedeća dva imaju pridruženi sprite i to su gumb koji služi za završavanje poteza i svitak koji otvara izbornik magije. Editor soba sadrži dodatne postavke sobe i korisnu mogućnost definiranja preodređenih puteva.

Slika 3 : Rad sa sobama (Izvor: Vlastita produkcija)

4. Strateška igra

4.1. Žanr

Strateške igre su igre s naglaskom na pažljivo planiranje i predviđanje budućih odluka samog igrača te protivnika. Cilj im je nagraditi logičko i racionalno razmišljanje od strane igrača, dok snaga i tehnička sposobnost predstavljaju manji čimbenik. Trenutno razlikujemo dva glavan podtipa ovog žanra: Turn-Based Strategy (TBS) i Real-Time Strategy (RTS). Od navedenih, prvi je baziran na potezima gdje igrači naizmjence odigravaju svoje poteze. U RTS igrama igrači istodobno upravljaju svojom igrom. Iz toga slijedi da RTS igre ne nagrađuju isključivo zdravo razmišljanje, već spoj razmišljanja, brzine i tehničke sposobnosti. Tematski je većina strateških igri smještena u povijesno važnom dijelu prošlosti ili intrigantnom izmišljenom scenariju. Perspektiva je pretežno od gore prema dolje s rijetkim izuzecima. Sučelje igre od iznimne je važnosti za ovaj tip igre; potrebno je izraditi sučelje koje odgovara igri u tematskom i funkcionalnom pogledu. Neprijateljske jedinice najčešće posjeduju jasne snage i mane koje specifične za određenu vrstu ili tip jedinica pa s obzirom na njih korisnik planira svoje korake. Nužno je spomenuti da AI (Artificial Intelligence) ipak treba biti na određenoj razini sposobnosti s tim da se izbjegava osjećaj nepoštenosti. Trebaju biti postavljena jasna pravila koja se kasnije potencijalno nadograđuju. Uvid u pravila igre je ključan za donošenje ispravnih odluka koje se nagrađuju. Po Andrew Rollingsu i Ernest Adamsu iza svih uspješnih strateških igri stoje tri važna koncepta: sukob, istraživanje i razmjena (Rolling i Adams, 2003). Na slici broj 4 vidimo primjer jedne TBS igre, Into the Breach. Ista je smještena u izmišljenoj budućnosti gdje su čudovišta napala svijet pa se naši heroji vraćaju iz budućnosti kako bi ga spasili. Na danom primjeru ćemo objasniti koncepte strateških igri.

1. Sukob – Zanimljiva borba koja je glavni fokus igre i ima za cilj zadržavanje vaše pažnje. Konstruirana je na način da imate dovoljno vremena za prave i proračunate odluke.
2. Istraživanje – ulazeći u određene sukobe niste upoznati s više faktora poput broja i vrste neprijatelja. Nepoznata i nasumična priroda igre predstavljaju kontinuirani izazov korisniku.
3. Razmjena – na kraju svake cjeline u igri omogućeno je trošenje resursa koji su stečeni tokom igre za poboljšanja vaših jedinica. Kako biste skupili resurse morat ćete donositi ispravne odluke ili žrtvovati druge resurse.

Slika 4: Into the Breach (Izvor: Vlastita produkcija)

4.2. Osnovna ideja rada igre

Cilj je stvoriti stratešku igru zasnovanu na heksagonalnoj rešetki s dva igrača od kojih svaki kontrolira vlastite jedinice. Igra se bazira na potezima po principu da prvo jedan igrač može kretati sve jedinice, zatim kreće drugi igrač i tako naizmjence. Igrač koji ostane bez jedinica je izgubio borbu. Jednog igrača igra korisnik igrice, a drugog računalo s tim da postoje tri borbe, odnosno razine s rastućom težinom. Igračeve jedinice se iz razine u razinu vraćaju na potpuni život dok kroz razine jedinice računala dobivaju dodatne mehanike. Tematski je igra zamišljena u izmišljenom svijetu gdje još vladaju srednjovjekovna oružja i magija. Igrač upravlja grupom avanturista u potrazi za blagom koje se nalazi u špilji. U slučaju da ne uspiju dolazi druga grupa, koja sve radi iz početka. Za igranje je nužan miš, a tipkovnica dodatna jer se koristi samo kao alternativa za završavanje poteza i startanje igre.

4.3. Inspiracija za projekt

Glavna inspiracija za priču i način rada su igre koje sam igrao u djetinjstvu, od kojih se kao glavna ističe Heroes of Might and Magic (HoMM). Franšiza trenutno ima 7 nastavaka od kojih je za ovaj projekt kao najvažnija istaknuta druga igra. Franšiza je tu već neko vrijeme još od 1995. godine, a posljednji nastavak izašao je 2015. godine („Heroes of Might and Magic“, 21.04.2019).Kroz godine se mijenjalo podosta stvari uz iznimku borbe koja je okosnica igre.

Na slici broj 5 s prikazom borbe u igri, teško je ne uočiti heksagonalnu rešetku koja je umetnuta i u stvorenu igru. U HoMM 2 su prisutni i dijelovi koji se neće pojaviti u našoj igri: kretanja na mapi, upravljanja resursima i gradovima jer je ona bazirana na razinama, a ne na mapi. Nadalje, na slici 3 vidimo jednog lika u gornjem lijevom kutu koji izgleda kao heroj (eng. *hero*), ali ima samo mogućnost čaranja magije i nudi pasivne bonuse svojoj vojsci. U stvorenoj igri neće postojati, ali će se njegove funkcionalnosti zadržati na malo drugačiji način. HoMM 2 također omogućava da se uključi računalo koje igra umjesto vas što neće biti implementirano. Smatram da će uvođenje igrača računala kao protivnika biti dovoljna za demostraciju potencijala i načina rada. U igri inspiraciji nemoguće je odabrati redoslijed akcija jedinica jer se za određivanje poteza to koristi inicijativa likova.

Dodatni dijelovi koji se ne tiču izravno mehanika igre su glazba, tema i izgled igre. Pokušao sam ih što više približiti uzoru u skladu s umjetničkim vještinama i dostupnim alatima. U svemu je potrebno zadržati i originalnost te aspekata drugih igri, kako ne bih izradio kopiju. Razmišljajući danas o HoMM serijalu vratio sam se u djetinjstvo i prisjetio dobrih ideja za igre. Svakako najpoznatija igra u serijalu je HoMM 3 bez obzira na godine igrača, što po meni dokazuje solidnost mehanika u serijalu.

Slika 5: HoMM 2 (Izvor: Vlastita produkcija)

Izvan Homm serijala najveća inspiracija i temelj na kojem sam radio je nedovršeni vodič (eng. *tutorijal*) za izradu 2D igre na heksagonalnoj podlozi. Vodič je izrađen od strane youtube korisnika Talent Lost koji je napravio manji serijal od osam videozapisa (Talent Lost, 2017). Nažalost, postoji više problema sa spomenutim među kojima su: bugovi, nema razina, nema izbornika, nema magije niti posebnih mehanika za jedinice. U usporedbi s predmetom ovog rada tema je također drugačija, tim superheroja naspram timu avanturista. Iako je vodič upotrijebljen kao osnova, cjelokupni projekt igrice je zaokružen s poboljšanjima i novom pričom.

4.4. Jedinice

U ovom dijelu će biti objašnjene jedinice, od svojstava do mogućih akcija s njima. Jedinice su razvrstane u dvije vrste; igračeve jedinice i one kojima upravlja računalo. Sve jedinice dijele više svojstva, a to su:

- Životni bodovi (hp) - odnosi se na broj štete koju jedinica može primiti prije nego što se uništi.
- Minimalni napad – Minimalna količina štete koju jedinica može napraviti pri napadu (bez izračuna s obranom).
- Maksimalni napad - Maksimalna količina štete koju jedinica može napraviti pri napadu (bez izračuna s obranom).
- Obrana –Količina štete koju jedinica ignorira u slučaju da je napadnuta.
- Posebne mehanike – ovisne o vrsti jedinice.

4.4.1. Jedinice igrača

Tri su različite jedinice kojima upravlja igrač kroz sva 3 nivoa:

1. Mačevalac - jedinica s velikim napadom i osrednjom obranom čija je zadaća eliminiranje važnih protivničkih jedinica kad se približe.
2. Kopljanik - jedinica fokusirana na obranu s osrednjim napadom, trebaju služiti kao prva linija obrane.
3. Strijelac – jedinica koja ima mali napad i obranu, ali posjeduje posebnu mogućnost napadanja preko cijelog bojišta. Ova jedinica služi kako bi ''omekšali'' protivnike tijekom borbe i treba je štiti s ostalim jedinicama.

Na svakoj ćete razini na raspolaganju imati dva kopljanika, mačevaoca i dva strijelca. U slučaju da izgubite neku od jedinica na jednom nivou one su vraćaju na sljedećem. Na slici

broj 6 su vidljive sve jedinice kojima korisnik upravlja. Imajte na umu da je slika skalirana zbog bolje vidljivosti što je primjetno na slikama s malim brojem piksela.

Slika 6: Figure korisnika(Izvor: Vlastita produkcija)

4.4.2. Jedinice protivnika

Neprijatelj, odnosno računalo, također ima tri tipa jedinica, no one se mijenjaju kroz razine igre u broju i vrsti. To su:

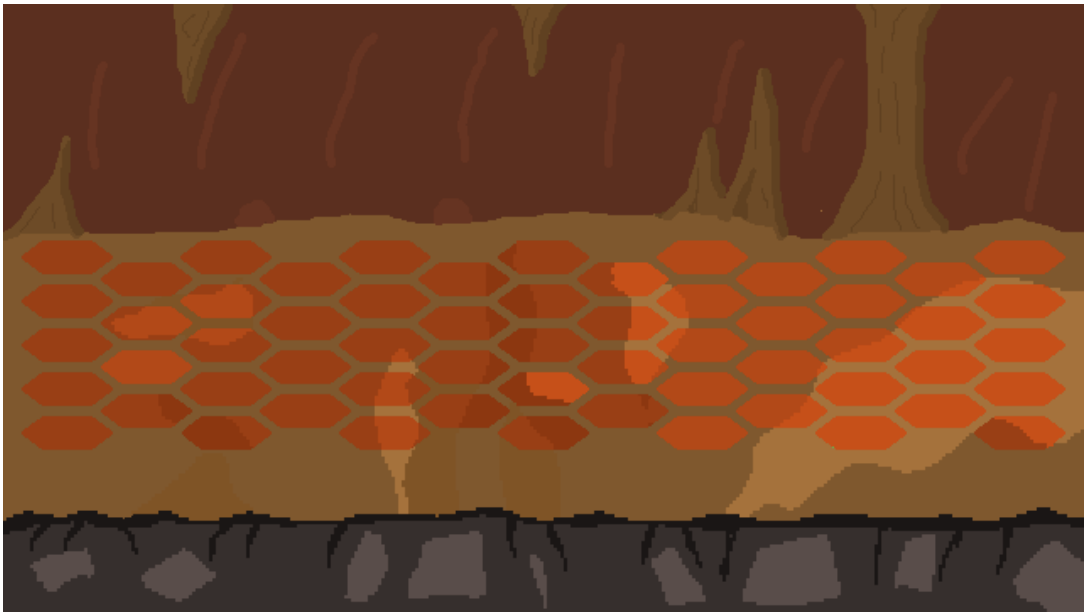
1. Kostur - jedinica bez ikakvih posebnih obilježja i osrednja u svim svojstvima. Ipak, predstavlja najčešćeg i najbrojnijeg neprijatelja u igri te ostvaruje interakciju s jednom drugom jedinicom.
2. Slime - jedinica sa slabim napadom, ali jakom obranom što je ne čini visokim prioritetom za eliminaciju. Jake defenzivne mogućnosti naglašava posebna mehanika regeneracije na početku svog poteza to jest kraju igračeva.
3. Necromancer – jedinica koja je glavni neprijatelj u igri, osjetno je jača od ostalih i nema određenih slabosti. Posjeduje moć oživljavanja kostura u slučaju njihove smrti što je popraćeno animacijom kako bismo lakše razaznali da je ova jedinica uzrok oživljavanja.

Na slici broj 7 su prikazane spomenute jedinice, a slike su također skalirane. Kosturi se pojavljuju na svakom nivou dok se ostale jedinice pojavljuju samo na jednom.

Slika 7: Jedinice protivnika(Izvor: Vlastita produkcija)

4.5. Pozadina i rešetka

Pozadina i rešetka su povezani u našem slučaju; dio pozadine se mijenja kroz razine, ali rešetka ostaje nepromijenjena. HoMM igre imaju različite rešetke, no kroz iteracije igri odlučio sam se za heksagone kao ćelije. Osobno smatram da su superiorne drugim vrstama ćelija, iako je dosta uspješih strateških igri koristilo kvadratne čelije. Na slici broj 8 je vidljiv primjer pozadine i rešetke s nivoa broj dva. Razmak između rešetki nije važan za kretanje jedinica i tu je zbog lakšeg diferenciranja individualnih pozicija.



Slika 8: Pozadina špilje s rešetkom (Izvor: Vlastita produkcija)

Iz priloženog su vidljive dimenzije rešetke; 5 redaka s 13 stupaca, dakle s naglaskom na širinu. Na lijevoj strani kreću jedinice igrača, a s desne protivničke jedinice. Sve jedinice

uvijek započinju na određenim pozicijama koje se ne mogu mijenjati. Ostatak pozadine je slobodan za ostale dijelove igre koji se naknadno ubacuju pri pokretanju igre poput kamenja.

4.6. Moguće akcije igraća

Na raspolaganju korisnika je više stvari: kretanje, napadanje i magija. Svaku od navedenih akcija ćemo detaljnije obraditi. Akcije se mogu izvoditi samo tokom poteza igrača i u slučaju da su uvjeti ispunjeni. O očitoj funkciji kraja poteza u igrama ovoga tipa nećemo podrobnije raspravljati.

4.6.1. Kretanje

Jedinice se mogu kretati u radijusu od 2 heksagona u svim smjerovima oko sebe, također mogu preskakati druge jedinice i prepreke. Nije moguće kretanje na pozicije na kojima su druge jedinice ili zapreke kao kamenja. Pozicije na koje se možete kretati će biti osvjetljene u slučaju da označite jedinicu koju želite pomaknuti. Desnim klikom na te označene čelije se vaša jedinica kreće na tu poziciju, a nakon kretanja gubite pravo kretanja do novog poteza. Kao što vidite na slici broj 9 već je predodređena udaljenost u pikselima od jedne ćelije do druge. Dakle, možemo se kretati na ćelije označene tamno žutom bojom. Primijetite da se ne možemo kretati na poziciju gdje je kamen ili na poziciju gdje su protivničke jedinice i ćelija je crvena. Kretnja na ćelije koje su udaljene više od 2 ćelije od trenutne pozicije također je onemogućena.

Slika 9: Primjer kretanja (Izvor: Vlastita produkcija)

Označene su udaljenosti u pikselima od središta jedinice. U slučaju da želimo pomaknuti jedinicu na poziciju ispod trenutne treba je pomaknuti za 26 piksela dolje. Ukoliko želimo desno mičemo se 47 piksela desno i u ovisnosti o željenoj ćeliji 13 piksela više ili niže. Sukladno s time se možemo pomaknuti na bilo koju drugu osvjetljenu ćeliju primjenom iste logike. U slučaju igre je taj radijus zajedno s pikselima hardkodiran jer trenutno nisam našao dobru zamjenu za ovaj sistem.

4.6.2. Napadanje

Svaka jedinica ima mogućnost napadanja jedinica koje su direktno susjedne njoj, odnosno unutar radijusa veličine jedne ćelije. Iznimka ovom pravilu je strijelac koji ima mogućnost napadaja preko cijele mape. U slučaju da ste napali u nekom potezu ta jedinica gubi pravo na daljnje kretanje i napadanje unutar trenutnog poteza. Za kalkulaciju štete se koristi više svojstava jedinica: od napadača maksimalni (MaxN) i minimalni napad (MinN) te od jedinice koja se napada obrana (Obr). Računa se uz pomoć pseudoslučajnog broja između MaxN i MinN od kojih se oduzima Obr. Izračunato je šteta koju prima napadnuta jedinica, a kada ta jedinica nakon napada nema više životnih bodova ona se uništava. Na slici broj 10 vidimo kopljanika koji napada kostura te mu u procesu radi 15 štete prema životnim bodovima. Iznad jedinica prikazana je razina životnih bodova na relativnoj skali od 0 do 100 posto. Iako nije vidljivo na slici, jedinice imaju animaciju i zvuk za napad.

Slika 10: Primjer napadanja (Izvor: Vlastita produkcija)

4.6.3. Magija

Na raspolaganju igrača je ujedno i magija koju je moguće koristiti jedan put po potezu. Za razliku od ostalih aktivnosti igrača, ograničeni ste resursom koji se zove mana te je ima 15.

Svaka vrsta magije košta različiti broj mane kako bi se koristila, a treba je štedjeti jer se ne regenerira ili vraća prelaskom razine. U slučaju da želite koristiti magiju kliknete na svitak kako bi ga otvorili. On sadrži popis magija koje možete koristiti. Te magije su redosljedom s lijeva na desno:

1. Grom – 6 mane - radi 30 bodova štete protivničkoj jedinici ignorirajući njenu obranu.
2. Izlječi – 4 mane - vraća 30 životnih bodova svojoj jedinici, ali samo u slučaju da joj nedostaje životnih bodova; ne može se jedinici dati više životnih bodova od maksimuma.
3. Štit – 2 mane - štit daje vlastitoj jedinici 5 bodova obrane stoga reducira štetu koju jedinica uzima od neprijateljskih napada za istu vrijednost. Imajte na umu da se štit može gomilati na istoj jedinici, ali se na kraju nivoa bonusi oduzimaju.

Slika 11: Magija u igri(Izvor: Vlastita produkcija)

Savjet za korištenje magije svakako je čuvanje mane za posljednje razine i njezina primjena na ključnim jedinicama za vašu strategiju. S pogleda funkcionalnosti, napravljene su animacije te klikom na magiju u svitku će prozirna verzija ikone magije pratiti vaš kursor. Kad ste odabrali metu jednostavno desnim klikom aktivirate magiju.

4.7. Računalo

Računalo ima na raspolaganju sve što korisnik ima na raspolaganju, izuzev magije. Računalo uvijek traži najbližeg neprijatelja po udaljenosti u pikselima te se kreće prema njemu. U slučaju da je susjedna jedinica igračeva tada napada ili postoji više susjednih igračevih jedinica, bira se najbliža po udaljenosti u pikselima. Sjetite li se udaljenosti između heksagona možete zaključiti da će se uvijek napadati jedinicu koja je vertikalno susjedna, a ne horizontalno. Nakon kretanja jedne jedinice računalo ide na sljedeću dok ne prođe sve jedinice. Nakon što su jedinice odradile zadane akcije, računalo završava svoj potez te tada kreće naš potez. Na početku poteza računalo također radi provjere za razne aktivnosti, primjerice

regeneraciju i slično. Skripte za rad računala će biti posebno obrađene kasnije u zasebnom dijelu.

4.8. Editori

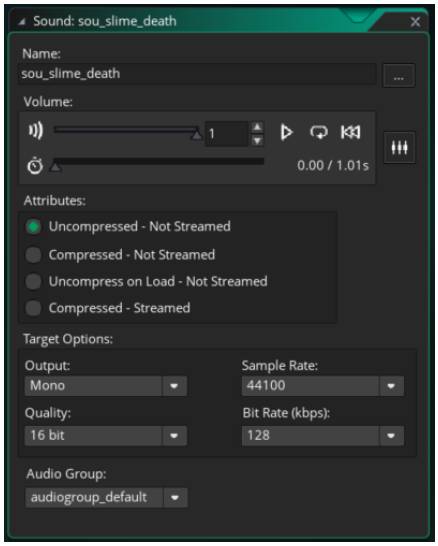
Za svrhe izrade izgleda ovog završnog rada su korišteni Pyxel (Kvarford, bez dat.) i editor unutar GameMaker Studija 2. Za jednostavne stvari je već integrirani editor odličan dok sam koristio Pyxel za crtanje iz nule i veće pothvate. Pretpostavljam da je to posljedica broja mogućnosti u Pyxelu koji je napravljen od strane jednog pixel artista. Dok je integrirani editor relativno nova adicija u GameMakeru, ipak služi svrsi i olakšava izmjene na već postojećim slikama.

4.9. Zvukovi

Igra sadrži glazbu koja je stalno uključena i zvukove koji odgovaraju specifičnim događajima. Glazba je napravljena u online programu Abundant Music koji omogućava generiranje nasumične glazbe (Nyblom, bez dat.). To se čini postavljanjem sjemenke (eng. *seed*) na temelju koje se generira glazba, a moguće je postaviti i sjemenke za različite dijelove pjesme. Postoji i mogućnost postavljanja velikog broja parametara vjerojatnosti različitih pojava u pjesmi. Dostupan je velik broj drugih opcija uz navedene, primjerice odabir broja i vrste instrumenata koji sviraju pjesmu.

Svi zvukovi u igri su skinuti sa stranice Freesound, koja omogućava besplatno skidanje već snimljenih i prenesenih zvukova od strane drugih korisnika (Music Technology Group, bez dat.). Zvukovi se aktiviraju samo kod određenih događaja poput napada, magije i slično.

Zvukovima i glazbom se upravlja iz prozora nakon unošenja u GameMaker Studio 2, što je vidljivo na slici broj 12. Moguće je upravljati samo osnovnim postavkama, od kojih je najrelevantnija glasnoća. Stoga, u slučaju da želite napredne mogućnosti morati ćete ih tražiti u drugim programima koji su za to specijalizirani. Uneseni se zvukovi jednostavno pozivaju s naredbom audio_play_sound gdje proslijedimo parametre: ime, prioritet i ponavljanje.



Slika 12: Prozor s zvukom(Izvor: Vlastita produkcija)

5. Programski kod

U ovom dijelu ću se dotaknuti programskog koda koji se nalazi u skriptama i objektima. Imajte na umu da neće biti pokazano sve te će se u većini slučajeva pokazati samo dio koda zbog svrhe prezentacije izvedbe.

5.1. Objekt igra

Objekt obj_game je zaslužan za upravljanje igricom u cjelokupnosti, ne samo kroz jedan nivo za razliku od ostalih objekata. Ima više događaja pa krenimo s događajem stvori (eng. *Create*).

```
window_set_size(1280,720);
alarm[0]=1;
mana=15;
audio_play_sound(background_music,1000,true);
level=1;
boja=make_color_rgb(51,51,0);
```

Inicijalno pri pokretanju igre, a time i stvaranja ovog objekta, trebamo povećati igru koja je inače rađena na 640x360 piksela na željenu dimenziju. Napravljena je u manjoj dimenziji zbog jednostavnosti crtanja i zadržavanja pikseliranog stila. Zatim postavljamo alarm koji će centrirati prozor kod prvog učitavanja. Postavljamo manu na 15, a razinu na 1 te puštamo glazbu u petlji. U zadnjoj liniji je stvorena boja koja će se kasnije koristiti kod pisanja uputa i priče naše igre. Sljedeći dio koda se nalazi u događaju step i bavit će se prijelazom iz jedne sobe u drugu.

```
if (keyboard_check_pressed(vk_enter)){
      switch(room){
            case rm_start:
                  room_goto(rm_combat1);
                  break;
            case rm_prijelaz1na2:
                  room_goto(rm_combat2);
                  level+=1;
                  break;
            case rm_prijelaz2na3:
                  room_goto(rm_combat3);
                  level+=1;
                  break;
```

```
            case rm_gameover:
                  game_restart();
                  level=1;
                  break;
            case rm_win:
                  game_restart();
                  level=1;
                  break;
      }
}
```

Točnije, ova se soba ne bavi prijelazima iz borbe već svega ostalog; prijelaznih soba, početka, kraja i pobjede. Jednostavno ako je uvjet ispunjen to jest kliknut gumb ili pritisnuta tipka, pogledamo u kojoj smo sobi trenutno te izvršavamo kod. Primjerice, kod prijelaza između razina povećamo razinu za jedan te ju otvorimo, dok kod gubitka restartamo igru i vraćamo razinu na 1. Sljedeći događaj je crtaj (eng. *draw*) i zadužen je za ispisivanje teksta na ekranu kod određenih soba. Za potrebe demostracije koda neće se prikazati sav događaj već jedan slučaj.

```
switch(room){
      case rm_start:
            draw_set_halign(fa_left);
            draw_set_color(boja);
            draw_set_font(fnt_pixel);
            var c= c_black;
            draw_text_transformed_color((room_width/2)-250,160, "Kako
igrati:",2,2,0,c,c,c,c,1);
            draw_text(room_width/2-250,200,"Pobjedi sve protivnike za
pobjedu");
            draw_text(room_width/2-250,220,"Lijevi klik za selekciju.");
            draw_text(room_width/2-250,240,"Desni klik za akciju.");
            draw_text(room_width/2-250,260,"E/GUMB za kraj poteza.");
            var c= c_red;
            draw_text_transformed_color( (room_width/2)-250,280,
"ENTER/GUMB za pocetak...",1,1,0,c,c,c,c,1);
            break;
```

Za početak pogledamo o kojoj se sobi radi i ukoliko smo došli do sobe rm_start kao u primjeru trebamo postaviti lijevo poravnanje teksta, odabrati već stvorenu boju kao željenu za crtanje i font s kojim želimo pisati. Tada krećemo s pisanjem teksta uz pomoć dvije funkcije; draw_text i draw_text_transformed. Druga funkcija za pisanje se koristi za određen izgled teksta.

5.2. Objekt kontrole igre

Objekt imena obj_control je zaslužan za većinu upravljanja borbom, još od početnog postavljanja jedinica. Krenimo prvo s događajem stvori (eng. *create*), imajte na umu da je dio koda izostavljen jer je sličan već objašnjen.

```
game_state = state.player;
player_selected = noone;
player_selected_r = noone;
player_hoover=noone;
p_enemy_selected=noone;
p_enemy_hoover=noone;

enemy_pos=0;

instance_create_layer(38,202,"Instances",obj_macevaoc);
instance_create_layer(38+47,202-13,"Instances",obj_kopljanik);
instance_create_layer(38+47,202+13,"Instances",obj_kopljanik);
instance_create_layer(38,202+13*2,"Instances",obj_strijelac);
instance_create_layer(38,202-13*2,"Instances",obj_strijelac);

if(obj_game.level==1)
{
	instance_create_layer(38+47*5,202-13,"Instances",obj_debris);
	instance_create_layer(room_width-38,202-(26*2),"Instances",
obj_skeleton);
	instance_create_layer(room_width-38,202,"Instances",obj_skeleton);
	instance_create_layer(room_width-38,202+(26*2),"Instances",
obj_skeleton);
}
```

Prvo postavljamo game_state varijablu na state.player jer svaka borba kreće s potezom igrača, a ne računala. Zatim se inicijaliziraju varijable nužne za daljnji rad skripata i objekata, kao one koje će držati informacije o označenoj jedinici i slično. Na preodređene lokacije u rešetki stvaraju se jedinice igrača. Nakon toga slijedi kreiranje protičničkih jedinica koje su za razliku od igračevih različite po nivou. Na primjeru je pokazano koje se jedinice i gdje stvaraju na nivou broj jedan. Sljedeći događaj koji ćemo pokazati je za crtanje (eng. *draw*), dakle ispisivanje informacija o jedinicama i magiji.

```
draw_set_halign(fa_left);
draw_set_color(c_black);
draw_set_font(fnt_pixel);
```

```
var xx=10;
var yy = 15
draw_text(xx+1, yy+1,"MANA:"+ string(obj_game.mana));
draw_text(xx-1, yy-1,"MANA:"+ string(obj_game.mana));
draw_text(xx,   yy+1,"MANA:"+ string(obj_game.mana));
draw_text(xx+1, yy,"MANA:"+  string(obj_game.mana));
draw_text(xx, yy-1 ,"MANA:"+ string(obj_game.mana));
draw_text(xx-1, yy ,"MANA:"+  string(obj_game.mana));
draw_text(xx-1, yy+1 ,"MANA:"+ string(obj_game.mana));
draw_text(xx+1, yy-1 ,"MANA:"+ string(obj_game.mana));
draw_set_color(c_blue);
draw_text(xx,yy,"MANA:"+string(obj_game.mana));

draw_set_font(fnt_pixel);
draw_set_halign(fa_left);
draw_set_color(c_white);
if (instance_exists(player_hoover))
{
      var name= player_hoover.name;
      var attack= player_hoover.attack;
      var m_attack=player_hoover.m_attack;
      var defence= player_hoover.defence;
      var hp= player_hoover.hp;
      draw_text(10,room_height-80,string(name));
      draw_text_ext_transformed(10,room_height-50,"NAPAD: " +string(attack)
+ "-" + string(m_attack)+ " OBRANA: "+string(defence)+" ZDRAVLJE: "+
string(hp),20,210,1,1,0);
}
else if (instance_exists(player_selected_r))
{
      var name= player_selected_r.name;
      var attack= player_selected_r.attack;
      var m_attack=player_selected_r.m_attack;
      var defence= player_selected_r.defence;
      var hp= player_selected_r.hp;
      draw_text(10,room_height-80,string(name));
      draw_text_ext_transformed(10,room_height-50,"NAPAD: " +
string(attack) + "-" + string(m_attack)+ " OBRANA: "+ string(defence)+"
ZDRAVLJE: "+string(hp),20,210,1,1,0);
}
```

Prvi dio događaja bavi se ispisivanjem mane to jest resursa za korištenje magije. Prvo postavimo postavke teksta, a zatim ispišemo mana s obrubom. To napravimo tako da u svim smjerovima u crnoj boji ispišemo isti tekst, a zatim željeni tekst u sredini. Kasnije mijenjamo boju u bijelo te provjeravamo koje su jedinice selektirane ili preko kojih je miš. U slučaju da je preko neke jedinice miš ispisuju se njezina svojstva, napad i druga. Ako miš nije preko bilo koje jedinice igrača tada se prikazuju svojstva za selektiranu jedinicu. Isto vrijedi i za protivničke jedinice samo na desnoj strani ekrana, to nije uključeno u prikazu koda jer je gotovo identično navedenom. Sljedeća malo kraća skripta se odnosi na događaj pri kliku gumb za kraj poteza ili tipci E.

```
if (game_state==state.player)
{
	with(player_selected)
	{
			timeline_index=tml_float;
			timeline_position=0;
			timeline_running=false;
	}
	scr_hex_destroy();
	game_state=state.ai;
	spell_cast=false;
	with(par_player)
	{
		attacked=false;
		moved=false;
	}
}
```

U slučaju da je pokrenut događaj isključimo vremensku crtu (eng. *timeline*) i vratimo u početno stanje. Uništimo stvorene heksagone za kretanje i napadanje, ali o tome još naknadno. Postavimo varijablu koja prati jesmo li koristili magiju ovaj potez na nekorišteno. Prebacimo potez na računalo te svim jedinicama varijable za napad i kretanje stavimo na false.

5.3. Objekt roditelj jedinicama igrača

Ovaj objekt sadrži više događaja od kojih se ističe jedan važni događaj: crtaj (eng. *draw*). On se odvija pri svakom crtanju u radu igre, a to se najčešće odnosi na okvir (eng. *frame*).

```
var is_selected=obj_control.player_selected;
var is_hovered=obj_control.player_hoover;
```

```
var sx = 5;
var sy = -50;

gpu_set_fog(true,c_black,0,1);
draw_sprite_pos(sprite_index,image_index,
      x-(sprite_width/2)-sx*2-10,y+float-sy,
      x+(sprite_width/2)-sx*2-10,y+float-sy,
      x+(sprite_width/2)+sx*2-10,y+float,
      x-(sprite_width/2)+sx*2-10,y+float,0.5);
gpu_set_fog(false,c_white,0,0);
draw_set_alpha(1);

if (is_selected==self)
{
      draw_sprite_ext(sprite_index,image_index,x-1,y-
float,image_xscale,image_yscale,0,c_black,1);
      draw_sprite_ext(sprite_index,image_index,x+1,y-
float,image_xscale,image_yscale,0,c_black,1);
      draw_sprite_ext(sprite_index,image_index,x,y-1-
float,image_xscale,image_yscale,0,c_black,1);
      draw_sprite_ext(sprite_index,image_index,x,y+1-
float,image_xscale,image_yscale,0,c_black,1);
      draw_sprite_ext(sprite_index,image_index,x,y-
float,image_xscale,image_yscale,0,c_white,1);
}

else if (is_hovered==self)
{
      draw_sprite_ext(sprite_index,image_index,x-
1,y,image_xscale,image_yscale,0,c_black,1);
      draw_sprite_ext(sprite_index,image_index,x+1,y,image_xscale,image_yscale,0,c_black,1);
      draw_sprite_ext(sprite_index,image_index,x,y-1,image_xscale,
image_yscale,0,c_black,1);
      draw_sprite_ext(sprite_index,image_index,x,y+1,image_xscale,image_yscale,0,c_black,1);
      draw_self();
}
else
{
      draw_self();
}
```

```
draw_healthbar(x-5*image_xscale,y-sprite_height-1,x+7*image_xscale,y-
sprite_height,(hp/max_hp)*100,c_black,c_green,c_green,0,true,true);
```

Prvi dio skripte iz drugog objekta obj_control uzima podatke o tome koji je objekt selektiran i iznad kojeg je miš. Na njega se nastavlja dio stvaranja sjene za jedinice, sy i sx su varijable koje su korištene za manipuliranje sjenom. Po potrebi su se mijenjale varijable kako bi se sjena pomicala, u konačnici su postavljene na 5 i -50. Postavimo gpu_set_fog na crno kako bismo crtali sjene, a ne dobili kopiju jedinice. Nacrtamo jedinicu uz pomoć draw_sprite_pos , i zatim vratimo gpu_set_fog natrag na bijelo. Nadalje, ako je jedinica koja je selektirana upravo ova, treba je podebljati i micati gore-dolje. Koristimo vremensku crtu (eng. *timeline*) koji upravlja varijablom float kako bi znali za koliko piksela trebamo sprite pomaknuti. Uz sve to crtamo sprite pomoću draw_spr_ext funkcije. Kada je miš iznad jedinice, tada ju je samo potrebno podebljati što je lako s već poznatim funkcijama. U slučaju da ništa od navedenog nije ispunjeno sprite se jednostavno nacrta s funkcijom draw_self. Zadnja linija se odnosi na funkciju draw_healthbar koja crta liniju zelene i crvene boje. Ona predstavlja helthbar kojim znamo relativnu količinu života jedinice. Za izračun uzimamo varijable iz specifičnih objekata hp trenutnu količinu života i max_hp maksimalnu količinu života. Podijelimo ih pa sve pomnožimo s 100 jer funkcija očekuje postotak života.

5.4. Skripta poteza igrača

Ova skripta se većinom bavi stvarima koje se događaju u potezu igrača, a poziva i podosta ostalih skripti. Kod je rastavljen je na dva dijela; jedan koji se tiče lijevog i drugi desnog klika miša.

```
if (mouse_check_button_pressed(mb_left))
{
      if (instance_position(mouse_x,mouse_y,par_player))
      {
            with(player_selected)
            {
                  timeline_running=false;
                  timeline_position=0;
                  float=0;
            }
            scr_get_top_selected();
            with(player_selected)
            {
```

```
                timeline_index=tml_float;
                timeline_position=0;
                timeline_running=true;
                scr_hex_destroy();
                scr_hex_move(movement);
            }
        }
        else if (instance_position(mouse_x,mouse_y,par_enemy))
        {
            scr_get_top_selected_enemy();
        }
        else
        {
            with(player_selected)
            {
                timeline_running=false;
                timeline_position=0;
                float=0;
                scr_hex_destroy();
            }
            player_selected=noone;
            player_selected_r=noone;
            p_enemy_selected=noone;
        }
}
```

Krenuvši s vrha, prvo provjeravamo selekciju to jest lijevi klik miša u našem slučaju. Ako je kliknuto, provjeravamo nalazi li se miš iznad neke jedinice igrača. Prvo gasimo vremensku crtu (eng. *timeline*) od prošle selektirane jedinice, a zatim pokrećemo skriptu za pronalazak selektirane jedinice i pokrećemo vremensku crtu za tu jedinicu. U sljedećem koraku uništavamo moguće preostale heksagone i stvaramo nove uz pomoć skripti. Kada je miš preko protivničke jedinice pokreće se skripta za pronalazak protivničke selektirane jedinice. Ako ipak nismo kliknuli na jedinicu, selektirana jedinica se deselektira i otkazuje se njena vremenska crta.

```
else if (mouse_check_button_pressed(mb_right))
{
    if (instance_position(mouse_x,mouse_y,par_enemy))
    {
        scr_get_top_hovered_enemy();
        var being_attacked=obj_control.p_enemy_hoover;
```

```
            if (player_selected!=noone and player_selected.attacked==false)
            {
                  scr_legal_attack(player_selected,being_attacked);
            }
            scr_hex_destroy();
      }
      else if (instance_position(mouse_x,mouse_y,obj_hex_move))
      {
            var hex_move=instance_position(mouse_x,mouse_y,obj_hex_move);
            with(player_selected)
            {
                  var move_path;
                  move_path = path_add();
                  path_add_point(move_path,x,y,20);
                  path_add_point(move_path,hex_move.x,hex_move.y,20);
                  path_set_closed(move_path,false);
                  path_start(move_path,20,0,0);
                  if (hex_move.x>=x) {image_xscale=1;}
                  else {image_xscale=-1;}
                  obj_control.zadnja_x=hex_move.x;
                  obj_control.zadnja_y=hex_move.y;
                  depth=-hex_move.y;
                  scr_hex_destroy();
                  moved=true;
            }
      }
}
```

Desni klik uglavnom se odnosi na akciju, a ne selekciju, u primjeru je demostrirano za kretanje i napadanje. Ako je desni klik pritisnut, gledamo nalazi li se miš iznad protivničke jedinice. U slučaju da je to ispunjeno pokrećemo skriptu koja pronalazi preko koje se jedinice nalazi miš te isto stavlja u varijablu. Zatim gledamo postoji li selektirana igračeva jedinica i je li napadala ovaj potez. Ako je selektirana jedinica koja nije napala prosljeđuju se podaci skripti koja će odrediti legalnost napada i obračunati štetu. Za kraj uništavamo sve preostale heksagone. Ukoliko je miš preko objekta koji označava mogućnost kretanja njega stavljamo u varijablu. Zatim selektiranu jedinicu treba pomaknuti na poziciju heksagona. To radimo kreiranjem puta (eng. *path*) počevši od jedinice do pozicije odabranog heksagona. Pokrenemo put i podesimo dubinu jedinice kako ne bih došlo do preklapanja s ostalim jedinicama. Tu su također dvije varijable koje služe za izbjegavanje bugova pri velikoj brzini rada računala. Naime, heksagoni za kretanje se uništavaju ako su na njima jedinice kojima treba vrijeme za

dolazak. Vrijeme je minimalno, ali zbog brzine rada računala je korištenje ovih varijabli nužno kako bi znali koje još heksagone treba uništiti. Uništimo sve heksagone i postavimo varijablu moved na true što označava da se jedinica u ovom potezu kretala.

5.5. Skripta poteza računala

Ovaj dio se odnosi na skripte koje upravljaju radom računala, to jest načinom na koji dolazi do svojih odluka.

```
var enemy_count=instance_number(par_enemy);
player_selected=instance_find(par_enemy,enemy_pos);
p_enemy_selected=player_selected;

with(player_selected)
{
	if(!instance_exists(obj_hex_move)) {scr_hex_move(movement);}
	else
	{
		timeline_index=tml_float;
		timeline_position=0;
		timeline_running=true;

		var nearest_player=instance_nearest(x,y,par_player);
		with(nearest_player) {
		var hex_move=instance_nearest(x,y,obj_hex_move);
		}
		if(attacked==false){
			if (name=="Slime" and hp<max_hp and regened==false)
			{
				var regen=0;
				regened=true;
				if ((max_hp - hp)<=regeneration)
				{
					regen=max_hp - hp;
					hp+=regen;
				}
				else
				{
					regen=regeneration;hp+=regen;
				}
```

```
                        var heal_display = instance_create_layer(x,y-
sprite_height,"Instances",obj_heal_display);
                        heal_display.heal=regen;
                  }
                  scr_legal_attack(id,nearest_player);}


            if(attacked==false and moved==false)
            {
                  var move_path;
                  move_path=path_add();
                  path_add_point(move_path,x,y,20);
                  path_add_point(move_path,hex_move.x,hex_move.y,20);
                  path_set_closed(move_path,false);
                  path_start(move_path,20,0,0);
                  obj_control.zadnja_x=hex_move.x;
                  obj_control.zadnja_y=hex_move.y;
                  if(hex_move.x>=x) {image_xscale=1;}
                  moved=true;
            }
      }
      if(next==true)
      {
            if (name=="Slime") {regened=false;}
            var nearest_player=instance_nearest(x,y,par_player);
            if(attacked==false){ scr_legal_attack(id,nearest_player);}
            scr_hex_destroy();
            timeline_running=false;
            timeline_position=0;
            float=0;
            if (obj_control.enemy_pos==enemy_count-1)
            {
                  with(par_enemy)
                  {
                        moved=false;
                        attacked=false;
                        next=false;
                  }
                  obj_control.enemy_pos=0;
                  obj_control.player_selected=noone;
                  obj_control.game_state=state.player;
                  exit;
```

```
            }
            else {obj_control.enemy_pos+=1;}
      }
}
```

Nakon što postavimo inicijalne varijable poput broja jedinica kojima upravlja računalo to jest broj jedinica koje treba pomicati, onda idemo na prvi slučaj. U slučaju da nisu generirani heksagoni za kretanje i napadanje, to sad učinimo. Ako jesu, pokrenemo vremensku crtu i tražimo najbližu jedinicu igrača. Kada smo je pronašli, tražimo heksagon za kretanje najbliži njoj jer se na taj heksagon namjeravamo kretati. Ako selektirana protivnička jedinica još nije napala to prvo pokušamo, a provjerimo ima li i jedinica mogućnost regeneracije. Posjeduje li tu sposobnost tad se izračuna količina životnih bodova koja će se nadodati te se stvara objekt koji će navedeno prikazati. Zatim provjeravamo varijable attacked i moved, ako su obje neistinite, radimo sljedeće: krećemo se do već predodređenog heksagona koji je najbliži najbližoj igračevoj jedinici. Nakon toga se provjerava varijabla next koja je istinita ako se jedinica kretala ili napadala. Pripremimo određene varijable za sljedeći potez kao regened za regeneraciju. Još jednom provjeravamo da li je moguće izvršiti napad i isključimo vremensku crtu. Provjerimo jesmo li prošli kroz sve jedinice kroz koje treba proći. Ako jesmo vraćamo sve varijable poput je li jedinica napala i slično na false. Broj neprijatelja se vraća na 0 i selektirana jedinica se deselektira, nakon čega računalo završava potez i završava ovu skriptu. U slučaju da nismo prošli sve jedinice jednostavno ćemo varijablu enemy_pos povećati za jedan jer ona drži podatke o broju prođenih jedinica.

5.6. Skripta legalnog napada

Dio koda je izostavljen jer se dobrim dijelom ponavlja, stoga će za svrhu demostracije biti pokazan samo napad obične jedinice. Imajte na umu da se defender odnosi na napadnutu jedinicu, a attacker na jedinicu koja napada.

```
else if (attacker.gx == defender.gx-1 || attacker.gx ==defender. gx+1)
{
      if (attacker.gy == defender.gy-1 || attacker.gy == defender.gy+1)
      {
            audio_play_sound(attacker.zvuk_nap,0,false);
            with (attacker)
            {
                  var move_path;
                  move_path = path_add();
                  path_add_point(move_path,x,y,30);
                  path_add_point(move_path,defender.x,defender.y,30);
```

```
                path_add_point(move_path,x,y,30);
                path_set_closed(move_path,false);
                path_start(move_path,30,0,0);
                if (defender.x>=x) {image_xscale=1;}
                else {image_xscale=1;}
                attacked=true;
                moved=true;
            }
            var damage = irandom_range(attacker.attack,attacker.m_attack)-
defender.defence;
            if(damage<0){damage=0;}
            defender.hp-=ceil(damage);
            var dmg_display = instance_create_layer(defender.x,defender.y-
defender.sprite_height,"Instances",obj_damage_display);
            dmg_display.damage=ceil(damage);
            if (defender.hp<1)
            {
                with(defender) {instance_destroy();}
                scr_prijelaz();
            }
      }
}
```

Krećemo s elseif dijelom koda jer smatram da najbolje opisuje što je važno u ovom dijelu skripte. Drugi slučajevi su za ostale vrste provjera legalnosti napada i neke jedinice koje ne provjeravaju susjednost drugih jedinica za napad. Prvo provjeravamo jesu li jedinice susjedne uz pomoć gx i gy varijabli, u slučaju da jesu napad može početi. Pustimo zvuk napada, a zatim s napadačem pripremamo i izvršavamo napad. Napravimo put slično kao u dijelu skripte za kretanje samo dodamo još jednu točku, onu početnu. Zatim treba obračunati štetu koja se dobiva generiranjem nasumičnog broja između maksimalnog napada i minimalnog napada od kojeg se oduzme obrana. Zaokružimo broj štete te taj broj oduzmemo od životnih bodova jedinice. Generiramo objekt koji će nam pokazati količinu štete i proslijedimo mu nanesenu štetu kako bi je mogao ispisati. U konačnici provjeravamo je li napadnuta jedinica uništena. Ako jest, uništavamo objekt jedinice i pozivamo skriptu za prijelaz. Ta skripta će provjeriti jesu li sve jedinice s jedne strane uništene i ako jesu tada će prebaciti sobu.

6. Zaključak

Utjecaj računalnih igara na modernu kulturu i život mlađih generacija je neporeciv što dovodi do sve većeg interesa i potražnje za istima. Otvaraju se brojni novi poslovi u industriji izrade igri stoga se nadam da vam je ovaj rad dao barem mali uvid u razvijanje sličnog projekta. Dakako, s pogleda solo developera koji radi kraće vrijeme na igri te se uči u procesu.

GameMaker Studio 2 je alat za izradu igri koji se pokazao dostojnim i poznatijih alata poput Unity. Trenutno je program je jako pristupačan manjim i većim grupama developera te jednostavan za korištenje. Dokumentacija je izvrsna i krucijalna za rad s ovim alatom po mom mišljenju. Smatram da se alat može istinski iskoristiti tek nakon dobrog upoznavanja s dokumentacijom. Također, dopala mi se količina aspekata igre koje alat obuhvaća, što ga čini poprilično samodostatnim. Izuzetak ovom pravilu su po mom mišljenju samo auditorni aspekti igre za koje je bolje imati dodatne alate.

Mišljenja sam da je rezultat projekta sasvim dobar, igra je ostala vjerna glavnoj inspiraciji uz nadogradnju s nekoliko originalnih ideja. Ipak, najbolja stvar koja je izašla iz cijelog procesa je znanje i iskustvo koje mogu primjenjivati i dalje. Ovakva igra, s potencijalno manjim opsegom odlična je za učenje studenata na zanimljiv način. Na kraju svega mogu reći da sam više nego zadovoljan izborom alata i teme te bih i drugima preporučio slične radove.

Popis literature

GameMaker Studio (zadnje editiranje 28.03.2019) u Wikipedija. Preuzeto 11.06.2019. s https://en.wikipedia.org/wiki/GameMaker_Studio

Heroes of Might and Magic (zadnje editiranje 21.04.2019) u Wikipedija. Preuzeto 11.06.2019. s https://en.wikipedia.org/wiki/Heroes_of_Might_and_Magic/

Kvarford D. *Pyxel Edit* (verzija 0.4.8) [alat]. dostupno: https://pyxeledit.com/

Music Technology Group, *Freesound* [web stranica s zvukovima] Preuzeto 11.06.2019. dostupno : https://freesound.org/

Nyblom P. *Abundant music* (verzija nepoznata) [web alat]. dostupno : https://pernyblom.github.io/abundant-music/index.html

Rollings , A. i Adams , Ernest (2003). *Andrew Rollings and Ernest Adams on Game Design*. SAD , San Francisco, New Riders.

Talent Lost, (08.06.2017) *TBS Super Hero Edition* [video]. Preuzeto 11.06.2019. s https://www.youtube.com/watch?v=9qJYUkibloI

YoYo Games. *GameMaker Studio 2 .* YoYo Games. (verzija 2.2.1.375) [alat]. dostupno: https://www.yoyogames.com/get.

Yoyo Games. *GameMaker Studio 2 Docs .* (verzija nepoznata) [službena dokumentacija]. dostupno: http://docs2.yoyogames.com/

Popis slika