Razvoj strategije u realnom vremenu u okruženju Game Maker

Poljak, Vanja

Sveučilište Jurja Dobrile u Puli
Fakultet ekonomije i turizma
«Dr. Mijo Mirković»

VANJA POLJAK

RAZVOJ STRATEGIJE U REALNOM VREMENU U OKRUŽENJU GAME MAKER

Završni rad

Pula, kolovoz, 2020. godine

Sveučilište Jurja Dobrile u Puli
Fakultet ekonomije i turizma
«Dr. Mijo Mirković»

VANJA POLJAK

RAZVOJ STRATEGIJE U REALNOM VREMENU U OKRUŽENJU GAME MAKER

Završni rad

JMBAG: 0303070919, redoviti student
Studijski smjer: Sveučilišni preddiplomski studij informatički menadžment

Predmet: Osnove programiranja

Znanstveno područje: Društvene znanosti
Znanstveno polje: Ekonomija
Znanstvena grana: Poslovna ekonomija

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

Pula, kolovoz, 2020. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisan, Vanja Poljak, kandidat za prvostupnika poslovne ekonomije ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Vanja Poljak

U Puli, kolovoz, 2020. godine

IZJAVA

o korištenju autorskog djela

Ja, Vanja Poljak, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Razvoj strategije u realnom vremenu u okruženju Game Maker “ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, kolovoz, 2020. godine

Potpis

Vanja Poljak

Sažetak

Rad opisuje koncept razvoja video igre tipa strategije u realnom vremenu pomoću Game Maker okruženja. Igra se sastoji od 3 razine gdje težina razine ovisi o lokaciji resursa unutar igre. Igrač upravlja sa rudarskom kompanijom koja kopa rijetke minerale. Igra je napravljena u 2D obliku sa pogledom odozgora prema dolje gdje je misija igre sakupiti određeni broj rijetkog minerala s pažljivim širenjem svojih postrojenja. U okruženju Game Maker opisuju se korištene metode i postupak izrade određenih dijelova koda video igre te resursa s kojima igra raspolaže. Cilj video igre je da igrač ima višestruke opcije kako bi prišao problemu na više načina, te kako bi samostalno razvio najoptimalniju strategiju da pobijedi.

Ključne riječi: *Game Maker, Game Maker Language, Drag and Drop, 2D, Video igra, UI, Strategija u realnom vremenu*

Abstract

This bachelor thesis describes the concept of developing a real time strategy game by using the Game Maker engine. The game contains 3 levels where the difficulty of each level depends on the location of resources within the level. Player takes control of a mining company that excavates rare minerals. The 2D game is set with a top down view of the level. The mission of the game is to collect a specific amount of resources by carefully managing and expanding the mining instalations. This bachelor thesis describes the development of the game with Game Maker engine, the methods used and explanations of certain code segments and resources created within the engine. The goal of the game is to give the player multiple choises on how to aproach the mission so he can develop an optimal strategy to win the game.

Keywords: *Game Maker, Game Maker Language, Drag and Drop, 2D, Video game, AI, Real time strategy*

Sadržaj

1. Uvod

Tema ovog završnog rada je izrada strategije u realnom vremenu uz pomoć Game Maker okruženja. Inspiracija za temu video igre dolazi od flash igre nazvane „Super Energy Apocalypse: RECYCLED!“ izdane 2008. godine, a napravio ju je Lars A. Doucet [12]. U njoj igrač mora preživjeti određeni broj dana od invazije zombija koji napadaju po noći. Postrojenja koje igrač gradi, obrana koju koristi i resurse koje troši za proizvodnju električne energije onečišćuje okoliš i ostavljaju smeće koje uvelike utječe na brojke i snagu dolazećih zombija. Zato treba pažljivo koristiti resurse i postrojenja da bi se razvila strategija kako bi imali dovoljnu razvijenu obranu sa što manjim onečišćenjem. U igri rađenoj u ovom radu igraču nije cilj preživjeti noći od invazije zombija koji su nastali apokalipsom uzrokovane globalnim zatopljenjem. Cilj je eksploatacija i rafiniranje određenog broja rijetkog minerala uz postepeno i strateško onečišćenje planeta i širenje postrojenja, kako lokalna fauna ne bi prebrzo počela napadati da spriječi igrača. Igra sadrži 3 razine gdje težinu definiraju lokacije resursa unutar razine. Video igra napravljena je sa pogledom odozgo prema dolje u 2D obliku.

Opće je poznato da su video igre postale svakodnevica velikog dijela populacije mladih ali i odraslih ljudi. Količina video igara povećava se kao i platforme na kojima se igraju. Nagli porast korisnika dolazi do razvitka mobilnih igra, gdje svatko može instalirati besplatnu ili plaćenu igru na svoj mobitel u par minuta. Svaki igrač ima preferirani žanr koji odgovara njegovom načinu igranja. Najpopularniji su oni žanrovi koji su jednostavni, grafički privlačni i koji ne zahtijevaju puno razmišljanja poput akcijskih igara ili pucačina, a manje popularne igre su one suprotne tome. Strateške igre ili igre gdje korisnik upravlja kolonijama zahtijevaju potpunu pozornost i spremnost igrača da razvije optimalnu strategiju i prilagodi se svakom problemu te su zbog toga su manje popularne među igračima.

U prvom poglavlju ovog rada govori se o povijesti i žanrovima video igara. Drugo poglavlje opisuje razvojno okruženje Game Maker i njegove komponente koje se koriste za izradu video igara, te se opisuju metode s kojima se korisnik može služiti. U zadnjem poglavlju prolazi se kroz samu video igru koja je izgrađena u ovom okruženju i metode

koje su korištene za njeni razvoj. Na kraju zadnjeg poglavlja objašnjavaju se pojedini dijelovi koda i elementa igre te razvitak strateškog dijela.

2. Video igre

U poglavlju o video igrama govori se o njihovim početcima od 1950-ih pa sve do danas. Objašnjeni su neki od najpoznatijih žanrova te najpoznatija okruženja za izradu video igara u današnje vrijeme.

2.1. Povijest

Prva pojava video igre dolazi 1948. godine. pod nazivom Zabavna sprava od katodnih cijevi (eng. *Cathode-Ray Tube Amusement Device*) [13]. Patent su dali izraditi Thomas T. Goldsmith i Estle Ray Mannu. Ta sprava igrala se tako da je igrač s pomoću ručice kontrolirao brzinu i kut ispucavanja projektila na određeni cilj. U tim godinama nisu postojali zasloni koji su mogli išta iscrtavati po njima, već su se koristile prozirne folije na kojima su bile crtane mete. U šezdesetim godinama razvija se prva igra koja se mogla igrati na televizoru. Autor Ralph Baer napravio je igru koju je nazvao Potjera (eng. Chase). Sedamdesete godine dovode nagli razvitak arkadnih igra poput igre PONG [1] koja je prikazana na slici 1.

Slika 1: Igra PONG [1]

Igrače konzole se prvi puta pojavljuju osamdesetih godina. Njihovom pojavom dolazi do rasta broja igara jer se na njima moglo pokretati više od jedne igre. Devedesetih godina pojavljuje se prve 3D igre poput Quake. To je prva igra koja je koristila tu tehnologiju i primjer scene iz igre može se vidjeti na slici 2. U novije vrijeme 3d tehnologija sve se brže razvija do razine foto realizma, a u najnovije vrijeme dolazi do razvijanja virtualne realnosti.

Slika 2: 3D igra Quake [2]

2.2. Žanrovi

Postoji mnogo žanrova video igara gdje svaki igrač ima svoje preferencije. U ovom poglavlju opisuje se nekoliko najpoznatijih koju svaki igrač zna i barem jednom je probao. Igrač preferira žanrove ovisno o njihovim načinom igranja. Neki igrači vole brze i akcijske igre, dok neki vole smirene i opuštajuće video igre.

2.2.1. Akcijske igre

Igre ovog žanra stavljaju igračeve reflekse, reakcijsko vrijeme, preciznost i donošenje brzih odluka kao glavne utjecaje na uspješnost [3]. Akcijske igre često su brzog tempa i najpopularnije su od svih žanrova. Zbog toga što je žanr jedan i on prvih kod nastanka video igra, sa vremenom se počeo dijeliti na pod-žanrove tipa Beat-em-Ups, pucačine,

platformere i mnoge druge. Na Slici 3 prikazan je primjer horor akcijske pucačine – „Doom III“

Slika 3: Doom III - horor akcijska pucačina [4]

2.2.2. Sportske igre

Sportske igre nastale su isto rano u povijesti video igra. Cilj igre orijentirana je na nekakav sport gdje se jedan igrač natječe protiv jednoga ili više igrača, ili pak kompjutersko kontroliranih umjetnih inteligencija [14]. Sportska igra ne mora biti isključivo o samom sportu, već može biti kombinacija sa strategijom gdje igrač upravlja cijeli svoj tim. Jedna on najpoznatijih sportskih igra je FIFA 18 prikazana na slici 4.

Slika 4: FIFA 18 - sportska igra [5]

2.2.3. Battle Royale igre

Ovo je žanr koji je postao aktualan prije 3 godine [6]. Cilj u ovom tipu igre je eliminirati ostale igrače i ostati zadnji preživjeli. Popularnost ovog žanra naglo je porasla s igrom Player Unknown's Battle Grounds (PUBG) prikazana na slici 5. koja je prodala preko 50 milijuna kopija na PC i Xbox platformi od lipnja 2018, dok sa mobilnom verzijom ta brojka prelazi 400 milijuna korisnika. Pojava nove igre Fortnite dodatno je pojačala potražnju za ovim žanrom, a sve više industrija video igara pokušavaju uhvatiti ovaj trend te zgrabiti udio u tržištu.

Slika 5: Player Unknown's Battle Grounds - battle royale igra [7]

2.2.4. Avanturističke igre

Ovaj žanr uvelike se fokusira se na rješavanje zagonetki, otkrivanje priče koja se pruža kroz cijelu igru te istraživanju okoline u kojoj se igra nalazi [9]. Avanturističke igre tipično su opuštajućeg i sporijeg tempa bez pretjeranih borbi, akcija ili korištenja strategija. Početne verzije ovog žanra bile su bazirane samo na tekstu, gdje se nisu koristili nikakvi grafički elementi. Na slici 6 može se vidjeti scena od jedne moderne avanturističke igre - Detroid become human.

Slika 6: Detroit become human - avanturistička igra [8]

2.2.5. Strategije u Realnom Vremenu

Ovaj žanr počeo se razvijati nakon pojave poteznih strategija. Razlika između njih je što u strategiji u realnom vremenu nema pauza, odnosno igrač može izdavati komande samo dok igra teče, a u poteznim strategijama igrač je ograničen na određenu količinu odluka koje može napraviti u jednom potezu [10]. U strategijama u realnom vremenu cilj je najčešće uništiti sve neprijateljske baze i vojnike kao što je vidljivo na slici 6 gdje se prikazuje jedna takva borba. Igrač gradi postrojenja i trenira vojnike, skuplja resurse i sudjeluje u manjim ili većim bitkama. Igrač koji najbolje iskoristi diplomaciju, resurse, vojsku i informacije koje je sakupio može s lakoćom pobijediti ostale igrače. Igra Starcraft izuzetan je primjer strategije u realnom vremenu.

Slika 7: Starcraft - strategija u realnom vremenu [11]

2.3. AAA i Indie video igre

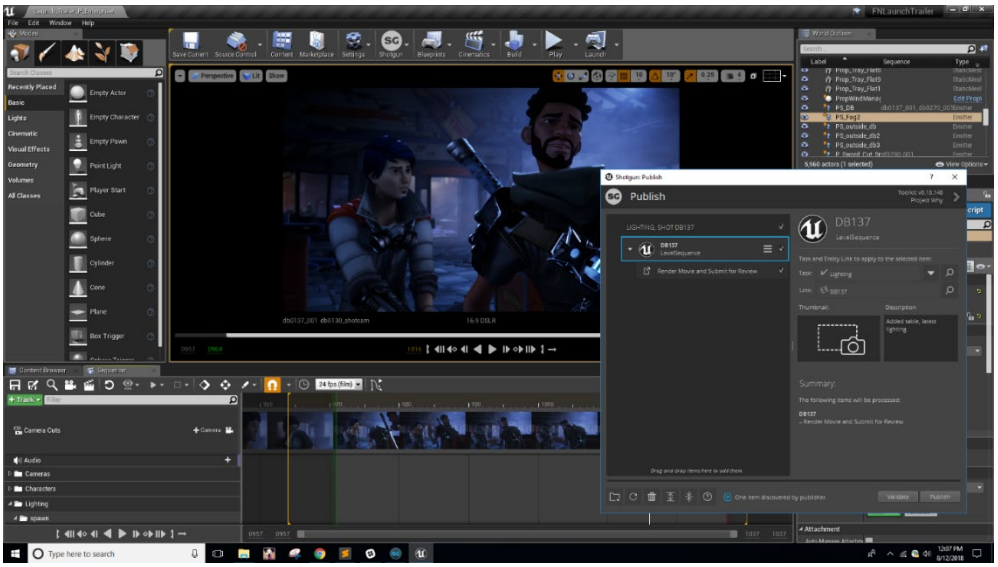
U današnje vrijeme igre se najviše klasificiraju kao indie ili AAA igre. Indie studiji, odnosno neovisni studiji karakterizirani su po malom broju zaposlenika, najviše do 30 zaposlenika te niskom budžetu i malenim igrama. Kako bi počeli razdvajati visoko kvalitetne igre od manje kvalitetnijih, pojavio se pojam AAA igra [30]. U američkom načinu ocjenjivanja A daje najveću ocjenu, te ovdje svako A stoji za kritični uspjeh, inovativnu igru i marketinški uspjeh te igre. AAA igre karakterizirane su po vrlo visokom budžetu, jakim marketingom i velikom timu, gdje je svaka osobo specijalizirana za jedan dio izrade video igre. EA, Bethesda i Activision jedni su od mnogih AAA studija.

2.4. Sučelja za izradu video igara

Kako ima sve više video igra na tržištu, tako se pojavljuju sve naprednija i jednostavnija okruženja za izradu video igra. Okruženja svojim korisnicima pruža već izrađene komponente fizike, inputa, skriptiranja, otkrivanja kolizije, umjetne inteligencije i ostalih korisnih funkcija bez da ih moraju isprogramirati od nule. U ovom poglavlju su navedena i opisana nekolicina najpoznatijih okruženja i neke poznate video igre koje su napravljene pomoću njih.

2.4.1. Unreal Engine

Jedan on najpopularnijih i najkorištenijih okruženja napravljen od strane Epic Games [15]. Prva verzija lansirana je 1998. godine i još uvijek se unaprjeđuje, a dovoljno je fleksibilan za sve veličine timova koje rade na nekom projektu. Pruža sve od foto realističnih pregleda kao što je prikazano na slici 8, do visoko kvalitetnih umjetnih i virtualnih realnosti. Unreal okruženje potpuno je besplatno za korištenje, no ako je igra lansirana, korisnik mora plaćati 5% od ukupne prodaje video igre koja je napravljena na tom okruženju [16]. Jedni od najpoznatijih igra napravljena u ovom okruženju uključuje Gears of War, Mass Effect, Bioshock i Batman: Arkham.



Slika 8: Sučelje okruženja Unreal [17]

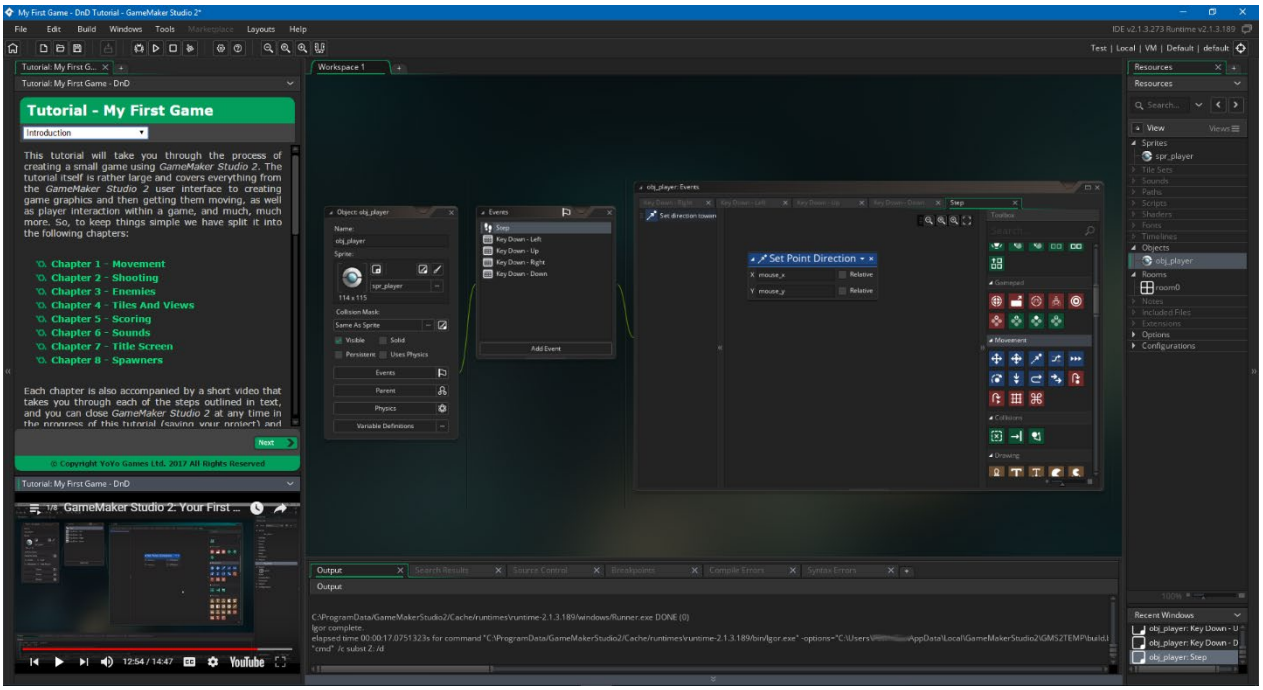
2.4.2. Unity Engine

Unity podržava mnogo platforma i s njim se vrlo jednostavno izrađuju 2D i 3D video igre, a primjer jednog projekta rađena u tom okruženju može se vidjeti na slici 9. Visoka funkcionalnost ovog okruženja pruža korisniku da napravi bilo koji žanr igre koji poželi. Unity ima dvije verzije, personalnu i profesionalnu. Personalna je besplatna sve dok korisnikova firma koja je registrirana na taj račun ne prelazi zaradu veću od $100'000 godišnje, dok se profesionalna verzija s više alata plaća mjesečno [18]. Neki od poznatih video igra napravljeni s ovim sučeljem su: Lara Croft Go, Her Story, Pillars of Eternity i Kerbal Space Program.

Slika 9: Sučelje okruženja Unity [19]

2.4.3. Game Maker Studio 2

GameMaker Studio 2 popularan je zbog svoje iznimne jednostavnosti i prilagodbe neiskusnim programerima [20]. Ovo sučelje nije nimalo napredno kao i prethodno navedena sučelja, no omogućuje izradu manje složenih igara i aplikacija. Postoji besplatna verzija no vrlo je ograničavajuća zbog manjka važnih funkcija koja su potrebna za izradu video igra. Poznati titlovi napravljeni putem ovog sučelja su: Spelunky, Hotline Miami i Super Crate Box. Više o sučelju Game Maker nalazi se u sljedećem poglavlju, a prikaz tog okruženja vidljiv je na slici 10.



Slika 10: Sučelje okruženja GameMaker Studio [21]

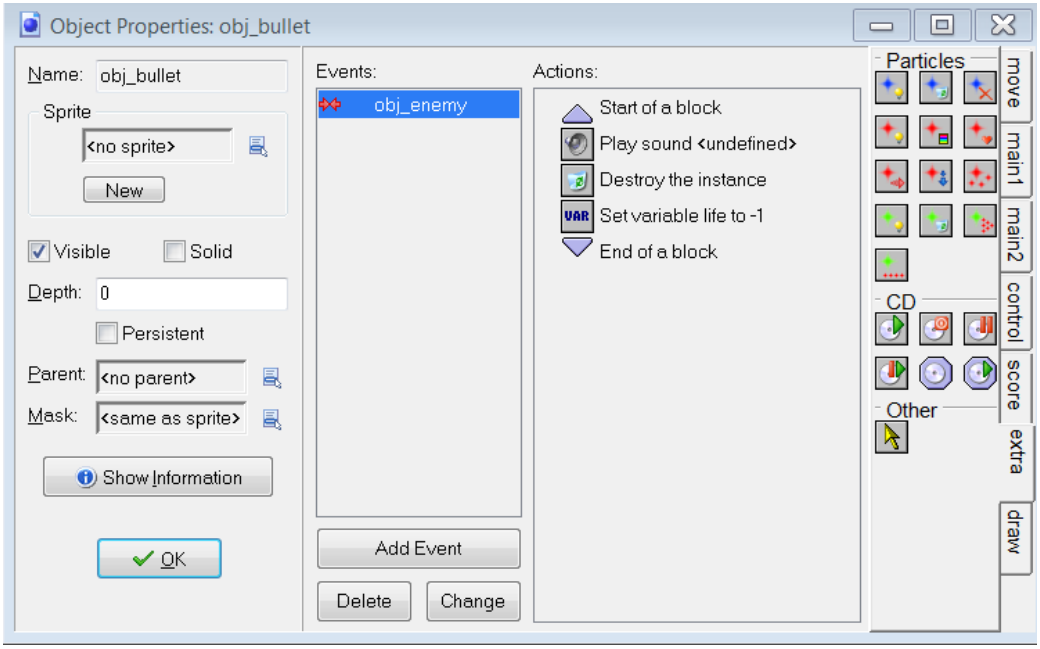
3. Game Maker pro 8.0

Game Maker programsko je okruženje koje omogućuje početnicima i naprednim korisnicima izradu vlastitih video igara. Korisnici koji nemaju nikakvo ili vrlo ograničeno znanje u programiranju mogu izraditi jednostavnu video igru s *Drag and Drop (DnD)* metodom dok napredni korisnici koriste Game Maker Language (GML) metodu, integrirani programski jezik, za izradu naprednih video igra. Obje metode biti će detaljnije objašnjene u sljedećem poglavlju. U vrijeme izrade ovog rada najnovija i naprednija verzija okruženja Game Maker je GameMaker Studio 2.

3.1. Metode

Game Maker podržava dvije metode za izradu projekta koje se mogu međusobno kombinirati, a to su Drag and Drop (DnD) metoda, primjer korištenja te metode vidljivo je na slici 11, i Game Maker Language (GML) metoda koju možemo vidjeti na slici 12.

DnD metoda namijenjena je za početnike koji nemaju nikakvo iskustvo sa kodiranjem. Korisnik može povući unaprijed definirane akcije u elemente objekta i tako odrediti ponašanje objekata [22]. Ova metoda vrlo je ograničena i pomoću nje nije moguće koristiti napredne izračune ili pozvati napredne funkcije koje se mogu koristiti samo pomoću GML-a. Zbog ove metode brojni korisnici odlučili su okušati svoju sreću u izradi vlastite video igre.



Slika 11: Primjer Drag and Drop metode kod kolizije dva objekta - Izradio autor

Game Maker Language (GML) metoda koristi vlastiti programski jezik [23]. GML jednostavni je programski jezik adaptiran vlastitom sučelju te je sličan Python-u i JavaScript-u [24][25]. Ova metoda također je korisna za učenje programiranja jer je vrlo jednostavan programerski jezik i specifično je namijenjen da se brzo prilagodi korisniku. Iako korisnik neće odmah znati kako programirati u drugim programerskim jezicima, učenje ovoga uvelike će mu pomoći da ih brže savlada.

```
//Request resources
if(build.materials[1,0] < build.materials[1,1]/1.5 && build.active_job != 3 &&
build.active_job != 4){
    ds_list_add(global.GLB_module.lst_jobs,build)
    scr_update_resources(1,2,build.materials[1,1] - build.materials[1,2])
    build.active_job = 3 // Supply
    }

//Initiate Repair
if(build.hp < build.hp_max && build.active_job != 3 && build.active_job != 4 &&
ds_list_size(build.lst_assigned_drones) == 0){
    ds_list_add(global.GLB_module.lst_jobs,build)
    build.active_job = 4 // Repair
        }
```

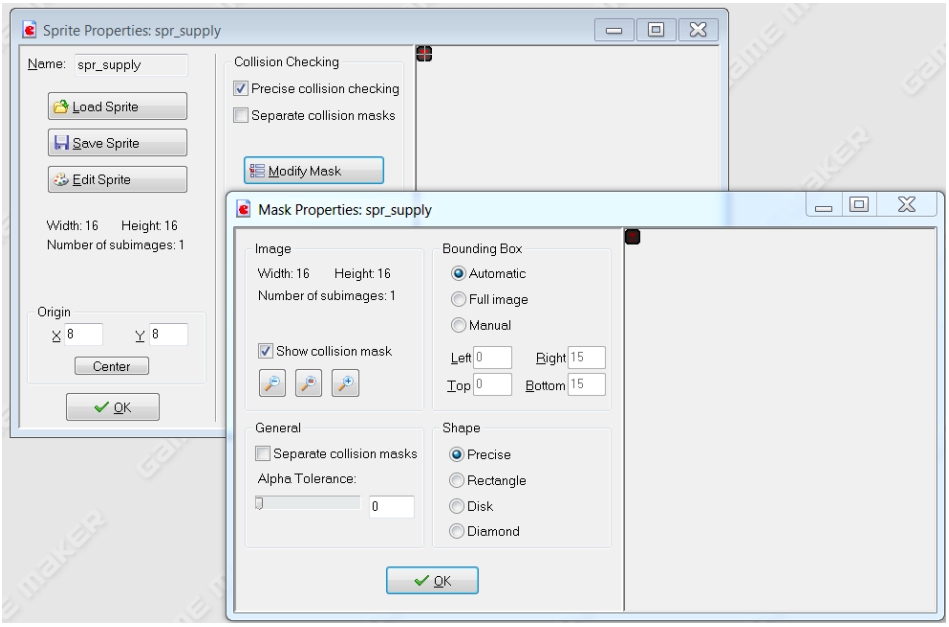
Slika 12: Primjer primjene Game Maker Language metode – Izradio autor

3.2. Sučelje

U ovom poglavlju objašnjavaju se neka od važnijih sučelja koje okruženje sadrži. Pomoću ovih sučelja korisnik definira i izrađuje ikone, pozadinu, skripte, zvukove, prostorije i objekte.

3.2.1. Ikone

Game maker ima vrlo jednostavan integrirani program za izradu ikona i njegovo sučelje je prikazano na slici 13. U tom programu definiraju se grafički elementi te se u njima izrađuju ikone i njihovi indeksi, definira se kolizijska maska te x i y koordinata kod ucrtavanja [26]. U postavkama se može navesti neki drugi program za crtanje koji se otvara prilikom uređivanja slika.



Slika 13: Prikaz uređivanja kolizijske maske ikone - izradio autor

3.2.2. Skripte

Game maker podržava pozivanje funkcija izvan onih definiranih kod elemenata u objektu s proslijeđenim argumentima koja će skripta koristiti [27]. Skripta može primiti do 10 argumenata i može se pozvati GML i DnD metodom. Cilj skripte je eliminiranje redundantnog koda koji mogu koristiti više različitih objekata, izračun formula ili zadataka, provjera istinitosti i neistinitost, bolje preglednosti koda i još mnogo toga. Jedan primjer prikazan je na slici 14 gdje se radi provjera da li zgrada ima dovoljno električne energije.

```
//scr_has_energy(build, demand)

var build, demand;
build = argument0   //building checking for energy,
demand = argument1  //Amount to check for
argument0 = 0       //Default value. Just check if there is enough energy when
already draining it


//If Supply is bigger than demand OR if Reserves are bigger than demand
if(global.GLB_module.materials[3,2] >= demand*(-1) ||
global.GLB_module.materials[3,0] >= build.materials[3,2]*(-1)){
    //We have energy. Return TRUE
    return 1;
    }
    //We don't have energy. Return FALSE
return 0;
```

Slika 14: Primjer skripte sa dva argumenta za provjeru dostupnosti električne energije - Izradio autor

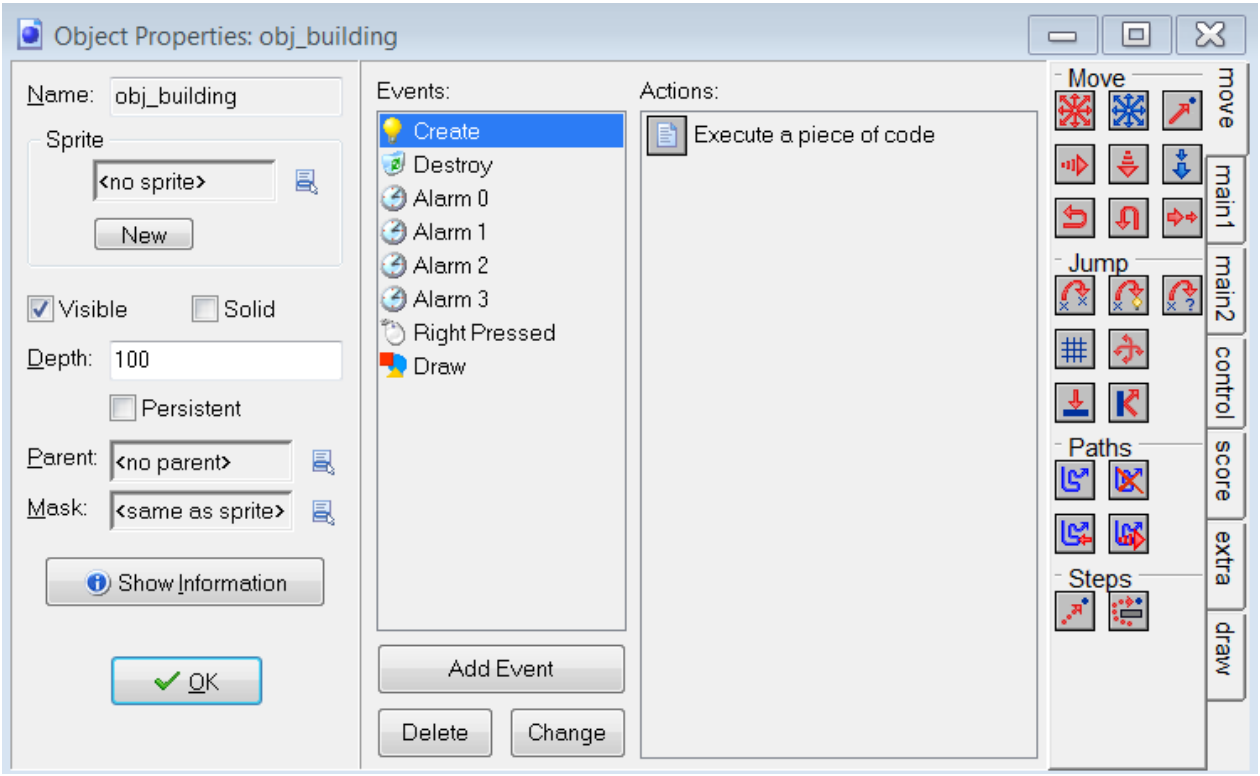
3.2.3 Objekti

Objekti su još jedni resursi koji se koriste za kontroliranje projekta [28]. Objekt sadrži evente i akcije koje definiramo kako bi ih natjerali da rade ono što mi želimo. Sami objekti nikad nisu postavljeni u prostorijama, već su postavljene njihove instance, odnosno zasebne kopije tih objekata. Objekti također mogu imati relaciju roditelj – dijete. Objekti koji imaju roditelja povlače sve evente i akcije koje nismo uredili. U slučaju da smo ih uredili, moramo pozvati posebnu funkciju prije nego naših promjena da kopira kod iz roditelja. Ovi resursi imaju zasebno sučelje koje je prikazano na slici 15.

Eventi definiraju pod kojim se uvjetima akcije događaju. Postoji ukupno 12 eventa sa više podeventa koji ih detaljnije definiraju. Za svaki objekt nužni su eventi Create, Step, Draw, Alarm, Collision, Mouse i Keyboard.

Create event definira početne varijable nakon što je njegova instanca stvorena. Step event pokreče akcije svaki korak. Koliko koraka je pokrenuto po sekundi ovisi o brzini koja je definirana u prostoriji. Draw event ucrtava grafičke elemente na ekran svaki korak. Tu definiramo sve vezano za prikazivanje teksta, ikona objekta, HUD elementa, linija i ostalih tekstura. Alarm je event koji pozivamo manualno s navedenim vremenskim

razdobljem. Collision se aktivira ako se kolizijske maske od definiranih objekata sudare. Nakraju su ostali Mouse i Keyboard koje pokreću akcije pritiskom, držanjem, ili otpustom klika ili tipke na navedeni objekt ili bilo gdje u prostoriji. Ostali eventi su od manje koristi i koriste se u manjim slučajevima, pretežito jer se mogu pozvati preko GML-a.

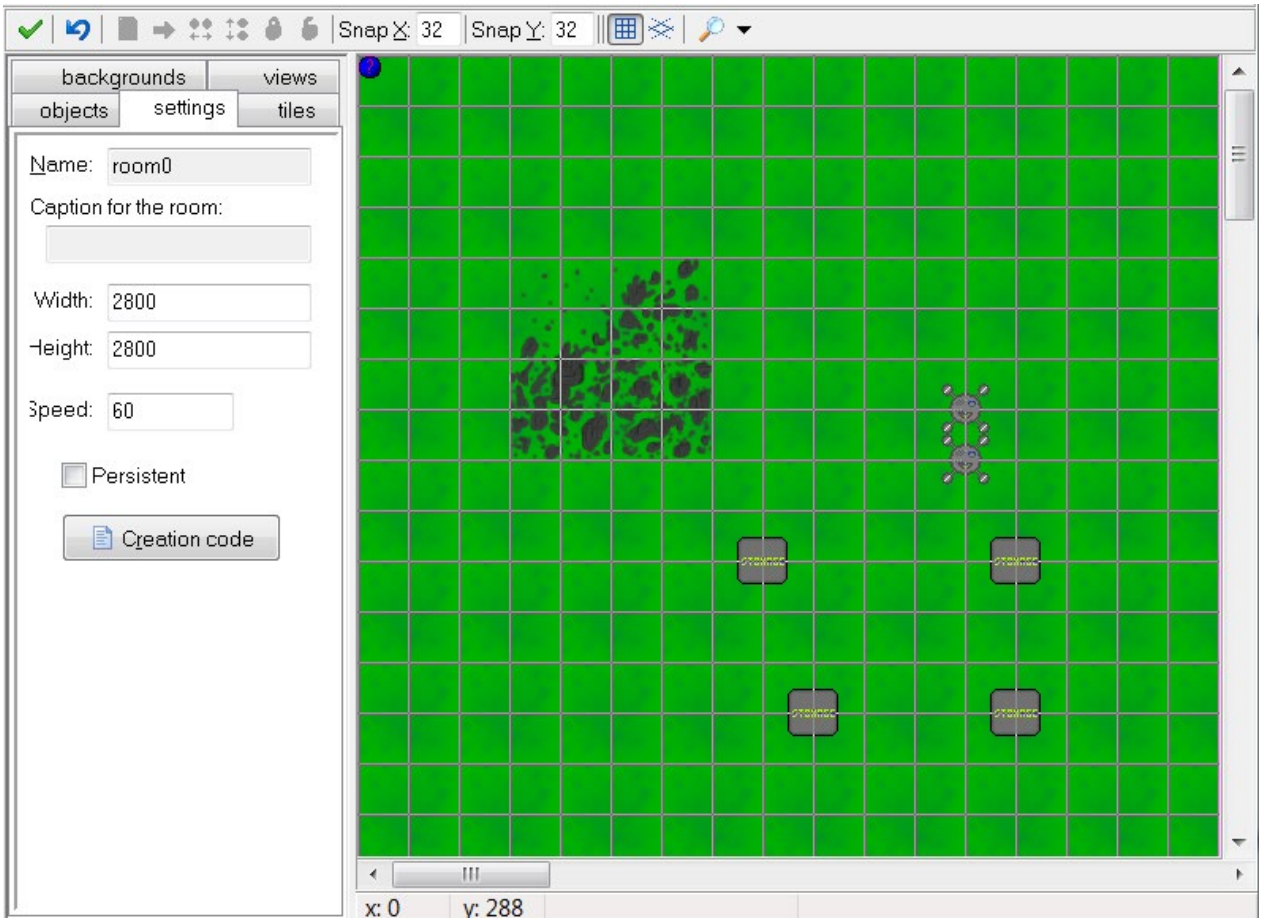


Slika 15: Prikaz objekta obj_building - Izradio autor

2.2.4. Prostorije

Da bismo mogli pokrenuti naš projekt, moramo definirati prostoriju ili prostorije u kojima će se odvijati sva radnja. Svaka prostorija ima svoje zasebno sučelje i postavke kao što se vidi na slici 16. U svakoj prostoriji može se definirati početne pozicije objekta, visina i širina prostorije, količina koraka po sekundi, pozadina ili pločice koje definiraju izgled prostorije, jedan ili više pogleda koji se prikazuju na ekranu te postavke za veličinu svakog pogleda i brzinu pomicanja pogleda ako je određeni objekt blizu rubu tog

pogleda [29] . Projekt može lako mijenjati prostorije kroz D&D ili GML metodu. Prostorije se koriste za definiranje nivoa u video igrama i/ili za glavne izbornike.



Slika 16: Prikaz jedne prostorije - Izradio autor

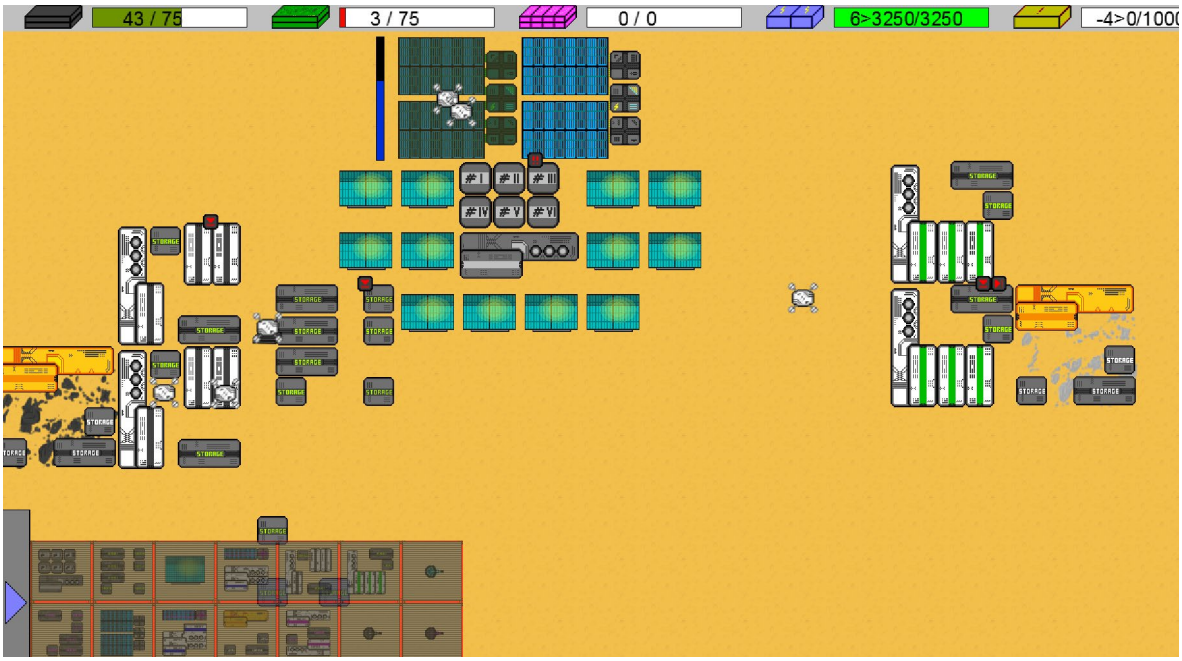
4. Izrada Strategije u realnom vremenu

U ovom poglavlju prolazi se kroz koncept video igre i proces izrade pojedinih elemenata projekta: procesor, dodjela i obavljanje poslova, neprijatelja i automatske obrane. Video igra izgrađena je pomoću okruženja Game Maker pro 8.0.

4.1. Uvod u video igru

„Interdyne inc.“ je strategija u realnom vremenu. Igrač upravlja sa rudarskom kompanijom „Interdyne inc.“ koja se bavi eksploatacijom rijetkih minerala i ruda. Na novootkrivenom planetu „Pharetra“ pronađen je Unobtanium, ruda iznimno visoke vrijednosti i gustoće koja se može koristiti kao izvor energije i u vojne svrhe. Cilj igre je skupiti i rafinirati određeni broj tog resursa. Na lici 17 prikazana je igračeva baza.

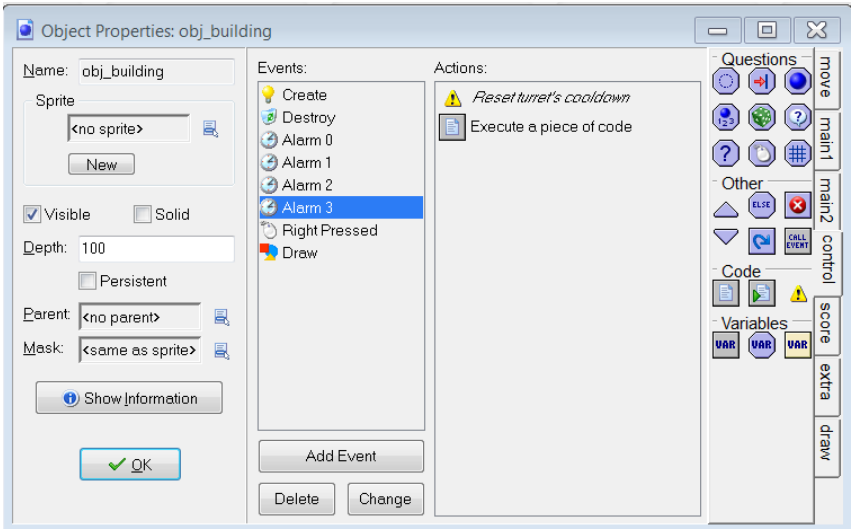
Igrač uz pomoć dronova gradi postrojenja za generiranje energije, kopanje ruda, te procesiranje i rafiniranje istih da bi ih koristio za danje širenje svog utjecaja. Što se igrač više i agresivnije širi, to više zagađuje okoliš koja utječe na lokalnu faunu tog planeta. Ozbiljnost zagađenje uvelike tjera faunu na agresivnost te se zbog toga igrač mora osigurati od napada na njegova postrojenja sa automatiziranom obranom. Igrač mora paziti na raspoložive resurse, potrošnju energije, količinu zagađenja te lokaciju obrambenih sustava kako bi osigurao pobjedu. S obzirom da samo jedna osoba stoji u izradi ove igre i budžet za izradu i marketing je nepostojeći, možemo ju klasificirati kao indie video igru.



Slika 17: Interdyne inc.. - strategija u realnom vremenu - Izradio autor

4.2. Korištene metode

Za izradu ove video igre korišteno je pretežito GML metoda i vrlo malo DnD metode. Ova igra je prekomplicirana za izradu s DnD metodom pa je na slici 18 prikazano korištenje te metode samo za postavljanje komentara na eventima, posebice alarmima kako bi se znalo o čemu se radi u kodu bez da se otvara akcija. Iako se mogao koristiti za veći broj jednostavnijih akcija, izbjegnuto je prekomjerno miješanje oboje metode radi veće preglednosti i održavanje koda.



Slika 18: Prikaz korištenja DnD metode - Izradio autor

GML metoda pretežito je korištena u cijelom projektu zbog potpune slobode programiranja koje ona nudi. Mnogo akcija i funkcija nije moguće pokrenuti pomoću DnD metode.

4.3. Izrada pojedinih elemenata igre

U ovom poglavlju predstavlja se proces izrade pojedinih elemenata igre. Neki dijelovi elemenata su skraćeni pa će se prikazati samo njihov najvažniji dio, a ostalo će se kratko opisati.

4.3.1. Procesor

Umjesto da svaka zgrada ima vlastiti alarm koji obavlja zadatke, na slici 19. koristimo glavni kontroler za pozivanje skripte za svaku instancu u definiranoj listi. Ovim načinom ne preopterećujemo kalkulacije i pojednostavnjujemo kod tako da se skripte za određene zgrade nalaze i definiraju na jednom mjestu.

U objektu koji nam služi kao globalni kontroler video igre definiramo listu ***lst_process_objects = ds_list_create();*** u create eventu. Nakon što dron završi izradu konstrukcije, ID novonastalog objekta dodaje se u listu zajedno sa varijablom koja sadrži ime skripte koje je namijenjena za nju.

Ako postoje objekti koji zahtijevaju procesiranje, globalni kontroler ide redom kroz listu uz pomoć ***for*** petlje, sprema ID vrijednosti na trenutnoj poziciji liste koju je funkcija ***ds_list_find_value(list,n)*** pronašla, a funkcija ***script_execute(script,arg0…)*** pokreće skriptu koja je spremljena u objektu u varijabli ***process_script.*** Nakon što je petlja došla do kraja, ***alarm[0]*** ponovno se pokreće svakih 10 koraka, odnosno svakih 1/6 sekunde i ovaj proces se nastavlja.

```
var i, process_this;
    for(i = 0; i < ds_list_size(lst_process_objects); i += 1){
        process_this = ds_list_find_value(lst_process_objects,i)
        if(instance_exists(process_this)){
            script_execute(process_this.process_script,process_this)
        }

}

alarm[0] = 10
```

Slika 19: Jednostavna metoda za procesiranje aktivnih objekata - Izradio autor

4.3.2. Neprijateljske instance

Sami objekt služi nam kao modul za više tipova neprijatelja. U njemu su definirane sve potrebne varijable što svaki neprijatelj mora imati, no varijable svakog tipa neprijatelja postavljamo prilikom poziva skripte ***scr_spawn_enemy(x,y,type)*** po potrebi.

Pozivom skripte stvara se nova instanca objekta ***obj_enemy*** sa koordinatama specificirani u argumentima tokom poziva skripte, a taj novi objekt referirati ćemo u varijabli ***new_enemy*** prikazanoj na slici 20. ***Switch*** funkciju, koja je kraća verzija ***if else*** funkcije, koristimo kako bi definirali naš tip neprijatelja iz argumenta ***type.***

```
var xx, yy, type;


xx = argument0
yy = argument1
type = argument2


new_enemy = instance_create(xx,yy,obj_enemy)

switch(type){
  case("Swarmer"):
      new_enemy.hp = 25
      new_enemy.hp_max = 25
      new_enemy.max_speed = 1.5
      new_enemy.sprite_index = spr_swarmer
      new_enemy.mask_index = spr_swarmer
      new_enemy.damage = 2
      new_enemy.attack_speed = 30
      new_enemy.armor = 0
      new_enemy.mass = 100
      break
  case("Berserker"):
      new_enemy.hp = 225
      new_enemy.hp_max = 225
      new_enemy.max_speed = 1.5
      new_enemy.sprite_index = spr_berserker
      new_enemy.mask_index = spr_berserker
      new_enemy.damage = 25
      new_enemy.attack_speed = 60
      new_enemy.armor = 0
      new_enemy.mass = 250
      break
  case("Brute"):
      new_enemy.hp = 325
      new_enemy.hp_max = 325
```

Slika 20: Skripta za stvaranje nove instance neprijatelja određenog tipa - Izradio autor

Umjetna inteligencija neprijateljskih objekata je vrlo jednostavna: Pronađi najbližu igračevu zgradu, kreni na nju, i napadni kad dođe u kontakt s njom Isječak koda odgovoran za to može se vidjeti na slici 21. Ako neprijatelj nema metu, on prolazi kroz listu svih instanca objekta ***obj_building*** i izračuna udaljenost s funkcijom ***distance_to_object(object)*** od neprijatelja do te instance. Nakon toga uspoređuje maksimalnu udaljenost pregleda sa udaljenosti do te instance. Ako je instanca unutar te udaljenosti, to nam je privremena meta. Mogli smo završiti s pronalaskom mete ovdje, no to će navesti neprijatelja da ide na prvu instancu koju je pronašao, a ne i najbližu. Da bismo ga naveli da ide na najbližu instancu, nastavljamo dalje sa traženjem instanci no ovog puta uspoređujemo udaljenost te instance sa zadnjom privremenom metom. Ako je distanca manja, to nam je nova trenutna meta. To ponavljamo sve do zadnje instance.

```
var distance, tmp_target, current_distance;

distance = 9991200 // distance
current_distance = 0
tmp_target = noone

//Find target
if(!instance_exists(target)){
  with (obj_building){
      if(constructed){
          current_distance = distance_to_object(other.id);
          if (current_distance < distance){
              distance = current_distance;
              tmp_target = id;
          }
      }
  }
  if (instance_exists(tmp_target)){
      target = tmp_target;
  }
}
```

Slika 21: Jednostavna metoda za pronalazak najbliže mete neprijatelju - Izradio autor

Sada kad imamo konačnu metu, naređujemo mu da krene prema meti. Na slici 22 provjeravamo da li će se neprijatelj u sljedećem koraku sudariti sa kolizijskom maskom svoje mete pomoću funkcije ***place_meeting(x,y,object),*** u koordinatama izračunatim funkcijom ***lenghtdir_x(dist,direction)*** i ***lenghtdir_y(dist,direction)*** u smjeru u kojem trenutno gledamo. Ako neće biti kolizije postavljamo neprijatelja da se pomiče brzinom od dvije koordinate po koraku u smjeru mete sa funkcijom ***move_towards_point(x,y,speed)***. Ako je prijašnja provjera neistinita, onda zaustavljamo neprijatelja i započinjemo napad koji se ponovno aktivira za određeno vrijeme u eventu ***alarm[0]*** ovisno o tipu neprijatelja.

```
//Go to target and attack
if(instance_exists(target)){
  if (!place_meeting(x+lengthdir_x(5,direction), y+lengthdir_y(5,direction),
target)){
      move_towards_point(target.x, target.y, max_speed);
 }
  else{
      speed = 0;
      if(can_attack){
          target.hp -= damage
          scr_check_building_health(target)
          can_attack = false
          alarm[1] = attack_speed
          speed = 0
          }
      }
}
alarm[0] = 30
```

Slika 22: Pomicanje neprijatelja prema meti i izvršavanje napada - Izradio autor

4.3.3. Automatska obrana

Automatska obrana također koristi jedan objekt kao modul za više tipova instanca, što znači da 3 različita tipa automatske obrane koristi iste skripte, samo sa drugačijim argumentima. Jedino što razlikuje ove zgrade je tip materijala koji se koristi kao projektil. Strojnica koristi metal, energetska obrana koristi energiju a obrana koja ispaljuje guste

grumene unobtaniuma koristi rafinirani unobtanium. Svaka zgrada konzumira te resurse, ima određeni broj projektila koji ispali u rafalnoj vatri, te ima određeno vrijeme potrebno da ponovno ispali projektile.

Umjetna inteligencija automatske obrane slična je kao kod neprijatelja s nekoliko izmjena, a one su vidljive na slici 23. Nakon što je automatske obrana pronašla metu, provjerava da li je moguće doći do nje, odnosno da li se meta skriva iza druge zgrade. To dobivamo pomoću funkcije ***collision_line(x1,y1, x2, y2, object, precise, notme)***. Ona nam govori da li linija od zgrade do mete dodiruje kolizijsku masku od ***obj_building***, i istovremeno isključuje samog sebe jer je automatska obrana također ***obj_building*** pa bi dolazilo do uvijek točne tvrdnje.

```
var distance, tmp_target, current_distance;

distance = 1200 // distance
target = noone
tmp_target = noone
current_distance = 0


if(!instance_exists(target) && instance_exists(obj_enemy)){
  with (obj_enemy){
      current_distance = distance_to_object(other.id);
      if (current_distance < distance){
          with(other){
              if(!instance_exists(collision_line(x, y, other.x, other.y,
    obj_building,0 ,1 ))){
                  distance = current_distance;
                  tmp_target = other.id;
              }
          }
      }
  }
  if (instance_exists(tmp_target)){
      target = tmp_target;
  }
}

if(instance_exists(target)){
  if(can_fire){
      scr_turret_fire(ammo_type,target)
  }

}
```

Slika 23: Modificirana metoda za pronalazak najbliže mete koja nije iza prepreke - Izradio autor

Ako postoji meta, pokreće skriptu koja se bavi sa ispaljivanjem projektila prema meti, te odabiru projektila, brzine, snage i ikone baziranu na materijalu. Automatska obrana također neće ciljati direktno u metu jer ako je meta pokretna, svi projektili će proletjeti iza mete. Zato automatska obrana uzima u obzir udaljenost, brzinu mete i brzinu projektila kako bi došla do točne lokacije pogotka.

4.3.4. Poslovi i Dronovi

Najkompliciraniji i najzahtjevniji dio cijele igre leži u umjetnoj inteligenciji dronova koji obavljaju različite poslove umjesto igrača. Dronovi donose resurse potrebne za izgradnju, konstruiraju postrojenja, odnose resurse po potrebi te popravljaju oštećenja nastala tokom napada lokalne faune.

Svaka zgrada može imati samo jedan aktivni posao koji traži i maksimalno dva drona koji rade na njemu. Mogući poslovi su konstrukcija, dostava, opskrba i popravak.

Ako dron nema posla, on prolazi kroz listu **lst_jobs** svih poslova i dodjeljuje je se prvom poslu koji je pronađen a da nije suspendiran ili stavljen pod pauzom, i ima slobodnog mjesta. Isječak koda s tom logikom nalazi se na slici 24. Do suspenzija dolazi kada dron nije u mogućnosti izvršiti trenutni zadatak. Suspenzija je bitna da dron ne zapne u vječnom pokušaju da završi nemogući posao. Nakon što dohvati posao, on spremi ID od zgrade koja traži posao i vrstu posla koju zahtjeva, a dronov ID spremljen je u listu **lst_assigned_drones** koja sadrži podatke o dronovima koji trenutno rade na njoj.

```
switch(active_job){
  //NO JOB
  case(0):{
      //If we have no job, go find one!
      if(ds_list_size(global.GLB_module.lst_jobs) > 0){
          var i, potential_job;

          //Search trough all global jobs
          for(i = 0; i < ds_list_size(global.GLB_module.lst_jobs); i += 1){
              potential_job = ds_list_find_value(global.GLB_module.lst_jobs,i)

              //If we found a job that has space for a drone and is not paused
or suspended
              if(ds_list_size(potential_job.lst_assigned_drones) <
potential_job.assigned_drones_max && potential_job.suspended == false &&
potential_job.paused == false){
                  //Add that job to the drone
                  assigned_job = potential_job
                  active_job = potential_job.active_job
                  ds_list_add(potential_job.lst_assigned_drones,id)
                  break
              }
          }
      }
      //Else search again
      alarm[0] = 30
      break
```

Slika 24: Kod za dodjelu poslova dronovima - Izradio autor

Dostava zahtjeva da se podmire svi potrebni materijali koji su potrebni za početak konstrukcije. Postoje 7 resursa, no 3 su potrebna za dostavu: metal, elektronika i unobtanium. Svaka zgrada ima dvodimenzionalnu Array listu za resurse i prikazana je na slici 25. Prva dimenzija nam govori o tipu resursa koji su označeni brojevima od 0 do 7 dok nam druga dimenzija govori o specifičnim vrijednostima tog resursa.

```
case "Drone Center":
//MATERIALS
//Metal
hud.materials[0,0] = -25   //Requires
hud.materials[0,1] = 0    //Max Storage
hud.materials[0,2] = -25   //Free
hud.materials[0,3] = 0     //Demand
//Circuits
hud.materials[1,0] = -10 //Requires
hud.materials[1,1] = 5 //Max
hud.materials[1,2] = -10 //Free
hud.materials[1,3] = 0     //Demand
//Unobtanium
hud.materials[2,0] = -5 //Requires
hud.materials[2,1] = 0 //Max
hud.materials[2,2] = -5 //Free
hud.materials[2,3] = 0     //Demand

//Energy
hud.materials[3,0] = 0 //Stored
hud.materials[3,1] = 500 //Max
hud.materials[3,2] = -7 //Supply
hud.materials[3,3] = 0   //Demand

//Pollution
hud.materials[7,0] = 0 //Stored
hud.materials[7,1] = 0 //Max
hud.materials[7,2] = 1 //Give Pollution
break
```

Slika 25: Primjer 2D array liste za spisak potrebnih materijala za konstrukciju solarne panele - Izradio autor

Tablica 1: Prikaz vrijednosti unutar 2D array liste – Izradio autor

Stupac / Redak	Currently Stored / Required for construction	Max storage capacity	Unreserved amount / Energy usage / Pollution	Demand, Supply or Demand & Supply
Metal	-20	0	-20	0
Circuits	-10	10	-10	1
Unobtanium	0	0	0	0
Energy	0	500	-7	0
Metal Ore	0	0	0	0
Silicon Ore	0	0	0	0
Unobtanium Ore	0	0	0	0
Pollution	0	0	1	0

U Tablici 1 se vidi da zgrada zahtjeva 20 **metala [0,0]** i ima slobodnih 20 mjesta za taj **metal [0,2]**. Zahtjeva 10 **elektronike [1,0]** i oslobađa 10 mjesta za **elektroniku [1,2]**. Također postavlja da je **maksimalni kapacitet** 10 **elektronike [1,1]** i ta zgrada, nakon što je konstruirana, označena je kao izvoznik **elektronike [1,3]** što znači da će dronovi uzimati elektroniku iz ove zgrade i proslijediti je drugdje po potrebi. Pruža 500 **maksimalnog kapaciteta energije [3,1]** i **konzumira** 7 jedinica **energije [3,2].** Na kraju povećava **zagađenje** za 1 **[7,2].** Tablica 2 prikazuje koordinate prijašnjih vrijednosti.

Tablica 2: Prikaz koordinata 2D array liste za prva 4 materijala – Izradio autor

2D 1D	Currently Stored / Required for construction	Max storage capacity	Unreserved amount	Demand, Supply or Demand & Supply
Metal	[0,0]	[0,1]	[0,2]	[0,3]
Circuits	[1,0]	[1,1]	[1,2]	[1,3]
Unobtanium	[2,0]	[2,1]	[2,2]	[2,3]
Energy	[3,0]	[3,1]	[3,2]	[3,3]

Posao dostave započinje tako da dron sa ***for*** petljom provjerava koji je prvi slobodan resurs koji zgrada zahtjeva i koji nije već rezerviran. Zatim spremi indeks tog materijala u varijablu **material_type**. Ako nije pronađen materijal ili ako je igrač pauzirao posao, dron taj posao suspendira. Kod zadužen za tu logiku prikazan je na slici 26.

```
 //DELIVERY
 case(2):{
   if(instance_exists(assigned_job)){
       //If we don't know what to grab
           if(material_type < 0){
               var i;
               //Find and Reserve the materials that the building requires
               for(i = 0; i < 3; i += 1){
                   //If the storage has all the materials of that group, reserve
all
                   if(scr_do_we_have_it(i,assigned_job) &&
 assigned_job.materials[i,0] < 0 && assigned_job.materials[i,2] < 0){
                       material_type = i
                       break;
                   }
               }
                   //If it requires no delivery, end the drone task now!
               if(material_type < 0 || assigned_job.paused == true){
                   alarm[0] = 10
                   assigned_job.suspended = true
                   assigned_job.alarm[0] = 250
                   scr_clear_drone_delivery(id,assigned_job)
                   return 0;
                   }
           }
```

Slika 26: Dron pronalazi indeks materijala koji je potreban zgradi - Izradio autor

Nastavak koda nalazi se na slici 27. Ako nema već pronađeno skladište, dalje traži zgradu koje sadrži taj resurs koji mora odnesti. Zgrada mora imati postavljen taj resurs da ga izvozi **[x,3] = 1** ili uvozi-izvozi **[x,3] = 2**. Ponavlja se sve dok ne dođe do kraja svih zgrada kako bi pronašao najbližu lokaciju.

```
//Find the closest storage
if(!instance_exists(grab_loc) && !moving && !material_held){
  if(ds_list_size(global.GLB_module.lst_buildings)){
      var i, temp_grab_loc, ii;
      for(i = 0; i < ds_list_size(global.GLB_module.lst_buildings); i += 1){
          temp_grab_loc = ds_list_find_value(global.GLB_module.lst_buildings,i)

          //Search for all material types inside building
              //If that building has materials that are free to take
              if(temp_grab_loc.materials[material_type,0] > 0 &&
temp_grab_loc.materials[material_type,2] > 0 &&
temp_grab_loc.materials[material_type,3] > 0){
                  if(!instance_exists(grab_loc)){
                      grab_loc = temp_grab_loc
                      //show_message("FOUND GRAB LOC")
                  }
                  else if(distance_to_object(grab_loc) >
distance_to_object(temp_grab_loc)){
                      grab_loc = temp_grab_loc
                      //show_message("FOUND CLOSER GRAB LOC")
                  }
              }
      }
  }
}
```

Slika 27: Kod za pronalazak najbližeg skladišta koji sadrži željeni materijal - Izradio autor

Ako skladište još uvijek nije pronađeno, onda se posao suspendira. Ako se dron ne kreće, rezervira materijal u skladištu i u zgradi te kreče prema skladištu skriptom **scr_move_to_work()**. Nakon što dođe do skladišta, uzme resurs u sebe, te kreče prema poslu. Ta logika prikazana je na slici 28.

```
//If still no location to grab from, suspend the delivery
if(!instance_exists(grab_loc) && !material_held){
  //show_message("Delivery suspended")
  assigned_job.suspended = true
  assigned_job.alarm[0] = 500
  scr_clear_drone_delivery(id,assigned_job)

}

//If we have everything, reserve the materials in Storage and job location
//Move to the storage
else if(!moving){
  grab_loc.materials[material_type,2] -= 1
  assigned_job.materials[material_type,2] += 1
  //Initiate moving
  scr_move_to_work(id,grab_loc)
}

//Stop when we get to the grab_loc
if(moving && instance_exists(grab_loc) && !material_held){
  scr_move_to_work(id,grab_loc)
}
//If we have the material, go to delivery location
if(working && !material_held){
  //Transfer materials from storage to drone
  grab_loc.materials[material_type,0] -= 1
  material_held = true
  working = false
  scr_move_to_work(id,assigned_job)
}
//Stop when we get to the delivery location and deliver
if(moving && material_held){
  scr_move_to_work(id,assigned_job)
}
```

Slika 28: Kod za pomicanje prema skladištu, uzimanju resursa i povratak prema zgradi - Izradio autor

Zadnji dio koda vezan za dostavu prikazan je na slici 29. Na kraju dostavi materijal kad se približi poslu, te provjerava da li je to zadnji materijal koji je trebao biti dostavljen. Ako je, posao se pretvara u konstrukciju.

```
//If we have the material, deliver
if(working && material_held){
  //Transfer materials from drone to building
  assigned_job.materials[material_type,0] += 1
  scr_update_resources(material_type,0, -1)
  scr_update_resources(material_type,2, -1)
  material_held = false  //Drone is no longer carrying anything
  material_type = -1
  working = false
  grab_loc = noone

  //Is delivery done in total?
  var i, isdone;
  isdone = true
  for(i = 0; i < 3; i += 1){
      if(assigned_job.materials[i,0] < 0){
          isdone = false
          break //It is not finished
      }
  }
  if(isdone){
      //It is, finish it
      //show_message("FINISH!!")
      scr_finish_delivery(id, assigned_job)
  }

}
}
alarm[0] = 10;
break
```

Slika 29: Kod za dostavu materijala i provjera - Izradio autor

Konstrukcija je vrlo jednostavna. Ako su svi materijali dostavljeni, dronovi započinju konstrukciju i nakon određenog vremena zgrada se aktivira, a posao se briše i oslobađa se dron kao što je prikazano na slici 30. Posao za popravke koristi sličan kod, samo što povećavaju njezine životne bodove umjesto vrijeme konstrukcije.

```
//CONSTRUCTION
case(1):{
  if(instance_exists(assigned_job)){
          if(!moving && working){
              if(assigned_job.con_time < assigned_job.con_time_max){
                  assigned_job.con_time += 1
                  assigned_job.suspended = false
              }
              else{
                  scr_finish_construction(id,assigned_job)
              }
          }
          else if(!working){
              scr_move_to_work(id,assigned_job)
          }

  }
  alarm[0] = 10;
  break
}
```

Slika 30: Kod za konstrukciju – Izradio autor

Zgrade koje zahtijevaju resurse zbog rafiniranja ruda, proizvodnju energije, održavanje dronova i za obranu dodaju novi posao tipa opskrbe. Opskrba je kodom slična dostavi, no postoje dodatne provjere kako ne bi došlo do repeticija. Na primjer skladište izvozi i uvozi metal i elektroniku. Problem ovdje nastaje što bi jedno skladište moglo drugome slati svoje resurse, pa ih to skladište može vratiti nazad i tako u nedogled. Zato moramo postaviti određene provjere da objekt koji pruža posao ne može biti isti objekt kao i onaj što pruža resurse. Također moraju biti izvoznici ili uvoznici / izvoznici

5. Zaključak

Za početnika u izradi video igara, Game Make okruženje odličan je odabir za izradu prve jednostavne video igre. Programski jezik u kojem je pisan i drag & drop metoda vrlo je jednostavan za razumjeti i koristiti, a dokumentacija vezana za njega je detaljno opisana. Online zajednica vrlo je aktivna koja redovito i ljubazno odgovaraju na pitanja novih korisnika koji su u poteškoćama sa svojim projektom. Forumi posjeduju veliku kolekciju već odgovorenih pitanja kroz sve ove godine koji se mogu lako pretražiti putem tražilice. To omogućuje korisniku laki pronalazak rješenja problema i siguran nastavak rada na svom projektu. Grafičko sučelje, iako je zastarjelo prema današnjim trendovima, jednostavno je za pregled i ne opterećuje korisnika koji nema iskustva u njegovom korištenju. Do problema dolazi ako korisnik s ovim okruženjem pokušava napraviti neku složenu video igru. Prve i najveće prepreke su što ovo okruženje ne podržava 3D igre, gdje su ostala okruženja poput Unity i Unreal Engine-a primarno rađene za 3D igre. Game Maker s druge strane nema nikakvih integriranih naprednih dodataka za animacije, optimizacije, integriranu trgovine sa besplatnim i plaćenim paketima tekstura, gotovih 3D i 2D objekata i skripta, alata za svjetlost, fiziku i ostalih kompliciranih i korisnih alata koje uvelike pomažu pri izgradnji video igre. S ovim okruženjem korisnik je primoran pretraživat forume njihove stranice i ručno tažiti dijelove koje mogu iskoristiti u svom projektu. Iako nije namijenjen za izradu video igara novih generacija, količina jednostavnih igara koje u lansirani putem ovog okruženja govori nam da još uvijek ima potražnje za jednostavnim okruženjima. Verzija Game Maker Pro podržava samo Windows kompatibilnost, dok nove verzije okruženja Game Maker Studio podržavaju kompatibilnost prema više operacijskih sustava. U njemu također dolazi do znatnog poboljšanja izgleda grafičkog sučelja i mogućnost još lakše izgradnje kompliciranijih video igara.

U projektu Interdyne Inc postoji mnogo dijelova koji se mogu prepraviti i nadograditi. Ovaj projekt nije uzimao u obzir dobre vizuale i sistem čestica, jer velika količina neprijateljskih instanca i općenito objekata koji se prikazuju na ekranu nisu dozvoljavale da igra teče bez zastajanja. Tu dolazimo do jednog većeg problema koji je snašao kako ovaj tako i svaki projekt neiskusnog programera, a to su optimizacije. Game Maker nije

idealan za nove programere ako gledamo na optimizacije, jer ako se želi imati nekoliko stotina elementa na istom zaslonu bez usporavanja, treba znati par trikova. Pametnim korištenjem lista i što manje alarma ne naprežemo CPU i daje se ostalim funkcijama neometan rad, što uvelike pomaže ovom projektu da se ne zaustavlja. Ovaj se projekt definitivno može još optimizirati kako bi se dobili dobri vizuali, a to je uz korištenje pozadinske slike, gdje se na njemu jednom ucrtavaju statični objekti poput zgrada. Nakon toga se ta slika ucrtava na ekran kao jedan objekt što pomaže kod optimizacije. Druga stvar koju se u projektu može iskoristiti je recikliranje objekata koje se ponavljaju, poput metaka. Metak koji je pogodio metu se sakrije i deaktivira te nakon što je ponovno potreban, umetnu se nove varijable u njega te ga pomaknemo i aktiviramo na određenim pozicijama. Ovim načinom se troši manje resursa za stvaranje i brisanje objekta. Prilikom izrade ovog projekta najviše problema dolazilo je do muzike. Funkcije koje su navedene u dokumentaciji nisu radile točno kako je navedeno u primjerima i učitavanje glazbe je bilo izrazito sporo, i to do tolike mjere da zbog jedne muzičke datoteke cijela igra bila zamrznuta do 10 sekundi prije nego je počela svirati.

Game Maker Pro i Game Maker Studio okruženja idealni su za učenje programiranja i izrade jednostavnih video igra. Napravljeni su s jednim ciljem, a to je da svaka osoba bez ikakvog iskustva može napraviti jednostavnu vlastitu 2d video igru uz malo vremena i čitanja dokumentacije.

6. Literatura

[1] Wikimedia Foundation, Inc. , (05.09.2019.), Pong, https://en.wikipedia.org/wiki/Pong, 15.09.2019.

[2] Reddit, (15.08.2017.), Quake 1 from 1996 in HD at 2560x1440 at 144hz., https://www.reddit.com/r/pcmasterrace/comments/6trq2e/quake_1_from_1996_in_hd_at_2560x1440_at_144hz_you/, 13.09.2019.

[3] Blue Flame Labs (n.d.), Genre Definitions, https://www.mobygames.com/glossary/genres, 15.09.2019.

[4] Jim01, (21.05.2018.), Doom 3 Underrated, https://www.mgtow.com/forums/topic/doom-3-underrated/, 13.09.2019.

[5] Kieran Francis, (26.09.2017.), FIFA 18 gameplay: Five key features, https://www.goal.com/en-om/news/fifa-18-gameplay-five-key-features/1wm72h2x8rq0y1az6w1plwh40g, 13.09.2019.

[6] Killian Bell, (21.03.2018.), You're winning PUBG Mobile because you're playing against bots, https://www.cultofmac.com/536311/youre-winning-pubg-mobile-because-youre-playing-against-bots/, 13.09.2019.

[7] Phil Hornshaw, (10.04.2019.), The history of Battle Royale: From mod to worldwide phenomenon, https://www.digitaltrends.com/gaming/history-of-battle-royale-games/, 13.09.2019.

[8] Will Fulton, (24.05.2018.), 'Detroit: Become Human' Review, https://www.digitaltrends.com/game-reviews/detroit-become-human-review/, 13.09.2019.

[9] Marek Bronstring, (12.02.2012.), What are adventure games?, https://adventuregamers.com/articles/view/17547, 13.09.2019.

[10] Mark H. Walker, (n.d.)Strategy Gaming: Part V -- Real-Time vs. Turn-Based, https://web.archive.org/web/20081221074049/http://archive.gamespy.com/articles/february02/strategygames05/, 13.09.2019.

[11] Lutris, (n.d.), StarCraft Anthology, https://lutris.net/games/starcraft-anthology/, 13.09.2019.

[12] JayisGames, (14.04.2018.), Super Energy Apocalypse, https://jayisgames.com/review/super-energy-apocalypse-review.php, 16.09.2019.

[13] Wikispaces, (n.d.), Povijest računalnih igara, https://dmkbj10.wikispaces.com/Povijest+racunalnih+igara, 13.09.2019.

[14] Technavio (10.09.2018.), https://blog.technavio.com/blog/top-10-most-popular-game-genres , 14.09.2019.

[15] Epic Games, (29.08.1998.), Unreal Engine, https://www.unrealengine.com/en-US/, 15.09.2019.

[16] Samit Sarkar, (02.03.2015.), Epic makes Unreal Engine 4 free, https://www.polygon.com/2015/3/2/8134425/unreal-engine-4-free-epic-games, 15.09.2019.

[17] Epic Games, (29.08.2018.), Shotgun integration now available in Unreal Engine 4.20, https://www.unrealengine.com/en-US/blog/shotgun-integration-now-available-in-unreal-engine-4-20, 15.09.2019.

[18] Unity Technologies, (08.06.2005.), Unity, https://unity.com/, 15.09.2019.

[19] Digital Inovative, (01.02.2018.), Unity Engine: A Unicorn Powering the Video Game and VR/AR Economy, https://digital.hbs.edu/platform-digit/submission/unity-engine-a-unicorn-powering-the-video-game-and-vr-ar-economy/, 14.09.2019.

[20] GameDesigning, (24.07.2019.), The Top 10 Video Game Engines, https://www.gamedesigning.org/career/video-game-engines/, 14.09.2019.

[21] Common Sense Education, (02.2018.), GameMaker Studio 2, https://www.commonsense.org/education/game/gamemaker-studio-2, 13.09.2019.

[22] YoYo Games Ltd., (n.d.), Drag and Drop, https://docs2.yoyogames.com/source/_build/3_scripting/1_drag_and_drop_overview/index.html, 14.09.2019.

[23] YoYo Games Ltd., (n.d.), GML overview, https://docs.yoyogames.com/source/dadiospice/002_reference/001_gml%20language%20overview/, 14.09.2019.

[24] queenguin, (07.09.2016.), https://www.reddit.com/r/gamemaker/comments/51jztx/is_gamemaker_language_gml_based_on_or_is_similar/, 15.09.2019.

[25] griffin1973, (02.09.2017.), Game Maker Language similar to JavaScript?, https://www.reddit.com/r/gamemaker/comments/6xob4n/game_maker_language_similar_to_javascript/, 15.09.2019.

[26] YoYo Games Ltd., (n.d.), Sprites, https://docs.yoyogames.com/source/dadiospice/002_reference/game%20assets/sprites/index.html, 12.09.2019.

[27] YoYo Games Ltd., (n.d.), Scripts, https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/006_scripts.html, 12.09.2019.

[28] YoYo Games Ltd., (n.d.), Rooms, https://docs.yoyogames.com/source/dadiospice/002_reference/rooms/index.html, 12.09.2019.

[29] YoYo Games Ltd., (n.d.), Objects and Instances, https://docs.yoyogames.com/source/dadiospice/002_reference/objects%20and%20instances/index.html, 12.09.2019.

[30] Raka Mahesa, (10.02.2017.) PacktPub, https://hub.packtpub.com/difference-between-working-indie-and-aaa-game-development, 15.09.2020.

7. Slike

8. Tablice