šolski center ptuj
elektro in računalniška šola

RAČUNALNIŠKA IGRA PO MERI

RAZISKOVALNA NALOGA

Mentor: David Drofenik

Somentor: Franc Vrbančič

Avtor: Matej Sardinšek, 4.a

Program: SSI, tehnik računalništva

Ptuj, Marec 2017

POVZETEK

Namen te naloge je, da sem moral narediti svojo računalniško igro. Izbral sem si to nalogo zato, ker se mi je ta tema zdela zelo poučna in me je zelo navdušila. Vedno me je zanimalo, kako so igre narejene in skozi kakšen proces gredo preden so prodane po vsem svetu. Ta veda se še danes širi in se vedno najdejo novi načini za izdelavo iger, predvsem nastanek novih idej, zgodb in okolij.

Preden sem začel izdelovati svojo igro, sem potreboval načrt, v katerem sem oblikoval svojo idejo in kaj bo moja igra vsebovala. Odločil sem se za arkadno igro, ki bo izdelana v programu GameMaker Studio Pro in je lahko igrana na namiznem računalniku, ki ima nameščen operacijski sistem Windows 7 ali Windows 10. Cilj igre je, da premagaš vse sobe in prideš do konca.

Med izdelavo sem naletel na nekaj težav glede programiranja igre, predvsem so bile težave z interakcijo igralca z njegovim okoljem. Imel sem še tudi par težav z dodajanjem novih sob in kaj naj bi vsebovale. S pomočjo interneta in svojega mentorja sem lahko razrešil vse težave, ki sem jih imel.

Igra je bila ustvarjena z namenom, da se ob njej zabavaš ter preganjaš dolgčas. Ob tem še je namen izdelave bil, da sem se lahko naučil, kako so igre načrtovane, izdelane in da pri tem dobim izkušnje le-tega.

Ključne besede: igra, izdelava, programiranje

SUMMARY

The purpose of this task was for me to create my own computer game. I picked this task due to it being an interesting and educational topic, as well as it being in my interests. I always wanted to know how people created games and through what process they had to go through before being released throughout the whole world. This knowledge is still being researched to this day and new ways to develop games are arising, mostly consisting of new ideas, stories and development enviroments.

Before I started making my game I needed a plan where I shaped my idea and figure out what things my game will have. I decided to make an arcade game that was created with GameMaker Studio Pro and it was able to be played on desktop computers that ran the operating system Windows 7 or Windows 10. The goal of the game is to beat all rooms and reach the end.

While I was developing my game I encountered a few problems, which were mostly related to interaction with the player and his enviroment. I also had a few issues making the levels for the game and what they would contain.

The game's purpose is to enjoy yourself with it and to kill time. Its purpose has also let me learn about how games are made, planned and I was able to gain experience in those fields.

Keywords: game, production, programming

KAZALO

KAZALO SLIK

UVOD

Igra je po Slovarju slovenskega knjižnega jezika definirana kot dejavnost, ki je namenjena razvedrilu in zabavi. Računalniška igra je podobna katerikoli drugi igri, le da je igrana na osebnem računalniku za razliko od drugih medij, kot sta konzola ali igralna plošča.

Za samo izdelavo računalniške igre je potrebno imeti ogromno znanja, ki ga sestavlja znanje iz večih področij, zato se lahko en projekt razdeli na več delov : Programiranje, animacija, načrtovanje, zvok, glasba in mnogo drugih zadev. Če je več oseb na enem projektu, se lahko to delo razporedi med njimi, če pa je izdelovalec le en sam, mora prevzeti vso delo. Sama veda o računalniških igrah se je zelo razširila skozi leta in še se danes napreduje ali se najde nekaj novega. Ta veda ni veliko cenjena med preostalimi, ampak z njo lahko dosežeš velike cilje.

Sprva bom začel z načrtom same igre, nato pa sledil po navodilih načrta. Vedno me je zanimalo, ali lahko nekdo z majhnimi izkušnjami v programiranju izdela preprosto računalniško igro. S tem me tudi zanimalo, kako preprosto je uporabljati program, ki ti omogoča izdelavo računalniških iger, ter koliko časa moraš vložit, da se naučiš uporabljati program.

Cilj te naloge je, da pridobim izkušnje izdelave računalniških iger in da spoznam to obširno vedo, da bom v prihodnosti začel izdelovati igre boljše kvalitete in mogoče postal del teh podjetij. Namen mojega izdelka je, da se ob njem zabavaš v svojem prostem času.

Empirični podatki

1.1 Znanje iz področja računalniških iger

V področju video iger je dandanes težko izumiti novo izkušnjo, saj je že skorajda vse bilo narejeno. Včasih se lahko najde igra, ki je lahko popolnoma drugačna v primerjavi z drugimi igrami in je lahko samo od tega uspešna. Število iger danes je ogromno in z vsakim dnem narašča. Te igre so narejene s programi, ki so si jih ljudi sami ustvarili ali so uporabili program, ki ga lahko kdorkoli uporablja. Za začetnike je najboljše, da si najdejo zastonj program za izdelovanje iger, saj je potrebno veliko znanja da takšen program narediš.

Danes lahko vsakdo naredi majhno igro s pomočjo interneta, kjer lahko najdeš raznorazna navodila za izdelavo preproste igre katerekoli zvrsti. V programu GameMaker Studio so ljudje naredili igre, ki so postale velike uspešnice. To okolje ima zelo malo omejitev, ampak lahko narediš karkoli v njem. Namenjen je ljudem, ki so komaj začeli izdelovati svojo prvo igro, ampak je tudi namenjen ljudem, ki so že izkušeni v tem področju, kjer lahko naredijo igro, ki ima veliko vsebine. Primer tega je igra z imenom »Cook, Serve, Delicious!«, ki je zaslužila $130,000 v prvih treh mesecih, ter je bilo prodanih 50,000 kopij. Celotna igra je bila napisana v GameMaker Studio in lahko celo najdeš na spletu originalno kodo te igre.

V tem področju je bilo ogromno narejeno, ampak se vedno lahko najde nekaj novega, čeprav je dandanes to težko izvesti.

Slika 1: Cook, Serve, Delicous! – Igra, napisana v GML jeziku

1.2 Namen izdelka

Glavni namen mojega izdelka je, da uporabnika zabava in da mu preganja dolgčas, tako kot vsaka druga igra. Ta izdelek lahko zabava osebo približno deset minut. Igra ni namenjena, da bi jo oseba večkrat igrala, saj bi se igre naveličala in ne bi bila zabavna. Drugi namen tega izdelka je to, da pridobim osnovne izkušnje za izdelovanje preprostih video iger. Po pridobitvi osnov lahko začnem izdelovati bolj zapletene in daljše igre.

Izdelek simulira digitalni svet, kjer uporabnik dobi izziv, da mora priti do konca tega sveta. To naredi tako, da premika svoj lik na zaslonu s pomočjo tipkovnice. Izdelek postavi raznorazne ovire v svetu, ki morajo preprečiti uporabnikov uspeh. Te ovire so lahko sovražniki ali sama okolica. Igra deluje na preprost način:

-Igra se začne, kjer se lahko uporabnik takoj začne premikati,

-Če se dotakne ovire ali sovražnika, uporabnika prestavi na začetek sobe,

-Če se uporabnik dotakne shranjevalne točke, se po smrti prestavi na lokacijo »Checkpoint«-a namesto na začetek sobe,

-Ko pride do konca sobe, se igralec prestavi v novo sobo (Igra ima 4 sob).

Tako kot vsaka video igra izdelek je namenjen zabavi. Igre lahko primerjaš po dolžini igre, po zgodbi, po glasbi, po videzu itd. Moj izdelek nima nobene glasbe ter ima preprost videz. Uporabnika lahko zabava le za nekaj časa, preden se igre naveliča. Igro je preprosto razširiti, kjer s tem lahko povečamo čas zabave uporabnika. Izdelek je majhen in preprost in ni namenjen preveliki uporabi.

1.3 Metoda dela

Igro bom izdeloval po posebnih navodilih:

-Sprva bom naredil igralca, osnovni objekt za zid in blok, ki igralca rani,

-naredil bom tri sovražnike različnih vrst,

-raznorazne pripomočke, ki jih igralec lahko pobere in s tem postane močnejši na določen način,

-shranjevalne točke, kjer ko igralec umre, bo postavljen na to točko namesto na začetek sobe,

-objekt, ki bo prestavil igralca v naslednjo sobo in

-police, ki se lahko premikajo levo in desno ali police, skozi katere lahko padeš skozi, če klikneš tipko za premik dol.

Predvsem bom meril, kako se težnost doda v igro, kako deluje trčenje z zidovi in na kakšne načine se lahko sovražniki premikajo. Glede trčenja (ang. »Collision«) obstajata dva znana načina, ki jih lahko uporabiš. Prvi način je, da ko se poskusiš premakniti v zid, to igra zazna in te premakne nazaj. Drugi način je, da igra zaznava, če je zid pred tabo in če je to res, se ti naj hitrost ponastavi. Poskusil in meril bom drugi način, saj ga je lažje izvesti. Uspešen bo takrat, ko bo delovalo tako, kot je napisano in načrtovano. Meril bom tudi obnašanja sovražnikov in kako se odzovejo svojemu okolju. Testiral bom, če sovražniki lahko padejo iz višine ali ne. Na koncu še bom meril, kaj so najbolj potrebne stvari v igri in kakšen vpliv imajo nanj.

1.4 Raziskovalna vprašanja in hipoteze

Moje glavno raziskovalno vprašanje bi bilo: »Kako lahko je narediti preprosto igro, ki bi jo lahko znal kdorkoli igrati?«. Večina ljudi pravi, da je za izdelavo igre potrebno imeti veliko znanja programiranja, imeti veliko logike in s tem tudi nadpovprečno znanje matematike. Drugi pravijo, da za izdelavo igre potrebuješ skorajda nobenega znanja programiranja. Moj predpostavek bi bil, da če hočeš narediti igro, potrebuješ vsaj osnovno znanje programiranja in da je najbolj odvisno od velikosti in kakovosti igre. Drugo vprašanje bi bilo: »Ali je uporaba programa GameMaker Studio res zelo preprosta?«. Ko vidiš katerikoli program za izdelavo iger, je razvidno, da vsak program ima veliko nastavitev in opcij. Začetnik si bi mislil, da je težko uporabljati te programe, saj potrebujejo precej znanja, da sploh lahko začneš izdelovati svojo igro. GameMaker Studio naj bi bil preprost za uporabo in se ga lahko zelo hitro naučiš uporabljati. Moj predpostavek je, da je program precej preprost za uporabo in da se ga ni potrebno temeljito naučiti preden začneš izdelovati igro.

Računalniška igra

2.1 Kratka zgodovina

Prve video igre so bile narejene v petdesetih letih prejšnjega stoletja, kjer so računalničarji ustvarili preproste igre. Ena od teh iger je zelo znana, ki se imenuje »Pong«. Igre so postale priljubljene v osemdesetih in devetdesetih letih na kabinetih za video igre in igralnih konzolah. Te igre so imele le dve dimenziji, dokler se niso konzole razvile in so ljudje lahko igrali igre s tremi dimenzijami. Ob istem času so ljudje začeli opuščati arkadne kabinete za igralne konzole in igre na računalnikih. Igre so se razvijale in izboljševale ob novejši in boljši tehnologiji računalnikov in igralnih konzol. Čez nekaj let, ko so prišli prvi pametni telefoni v javnost, so se igre razvile tudi tam.

2.2 Zvrsti računalniških iger

Video igre lahko razvrščamo tako kot katerikoli drugi tip medij kot so filmi, knjige, televizijske oddaje itd. Video igre je s tem tudi težko razdelit in razvrstit, saj lahko imajo veliko različnih elementov. Razvrstimo jih v žanre, kjer imajo igre podoben ali enak izziv. Ena igra lahko ima različne elemente iz drugih žanr, zato še jih dodano razvrščamo v »podžanre«. Primer tega bi lahko bila igra, ki ima elemente strateške igre in športne igre.

2.3 Delitev računalniških iger

Večino iger spada med petimi najbolj znanimi kategorijami. Obstaja tudi veliko iger, ki niso v nobeni kategoriji, saj jih je težko razvrstiti.

2.3.1 Akcijska igra (Action Game)

Za igranje akcijske igre je potrebno imeti dobro usklajenost med rokami in očmi ter osredotočenost. V teh igrah ima igralec največji vpliv na dogajanje v njej. Akcijske igre lahko razvrstimo kot: »Platformer« igre, »Shooter« igre, »Fighting« igre, »Stealth« igre in »Survival« igre. Ključno znanje teh iger je imeti hiter reakcijski čas in narediti pravilne odločitve v kratkem času.

Slika 2: Primer "Platformer" igre, Fez

VIR: http://www.gematsu.com (2017)

2.3.2 Igra z vlogo (»Role Playing Game«)

Igra z vlogo ali RPG je tip iger, kjer igralec nadgrajuje svojega karakterja v igri medtem ko napreduje skozi predpripravljeno zgodbo. V takšnih igrah je svet sestavljen iz različnih delov, kot so mesta in vasi ali gozdovi in votline, kjer se igralec lahko bojuje z sovražniki ali drugimi igralci. RPG igre lahko razvrstimo kot: »Fantasy« igre, »Rougelike« igre, »Action RPG« igre in »MMORPG« igre.

Slika 3: Primer »MMORPG« igre, Spiral Knights

VIR: http://www.gamerevolution.com (2017)

2.3.3 Strateška igra (»Strategy Game«)

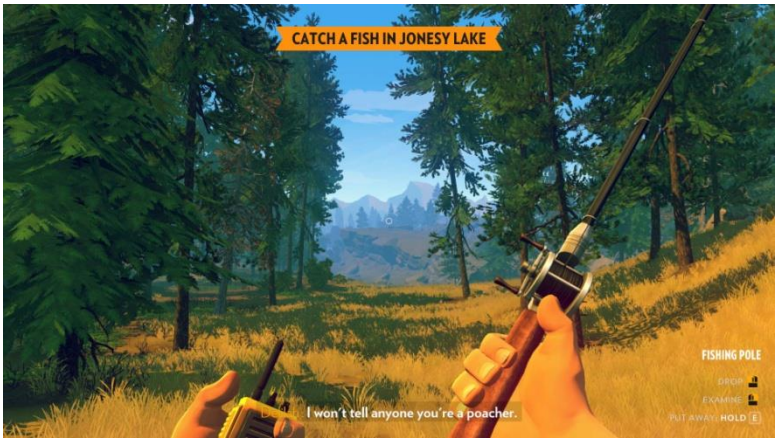
Strateške igre imajo poudarek na previdno razmišljanje in načrtovanje, če hočeš osvojiti zmago. V takšnih igrah je pričakovano videti ves svet, ki ga lahko spremeniš z svojim karakterjem ali z nečem drugim. Te igre se lahko igrajo v pravem času ali v potezah, ter se lahko osredotočijo na strategijo ali taktiko. Strateške igre razvrstimo kot: »Real Time Strategy« igre, »Multiplayer Online Battle Arena (MOBA)« igre, »Tower Defense« igre in »4X« igre.

Slika 4: Primer »MOBA« igre, Dota 2

VIR: http://imgur.com (2017)

2.3.4 Pustolovska igra (»Adventure Game«)

Pustolovske igre nimajo potrebe po hitrih refleksih, temveč poudarjajo na raznorazne uganke, ki jih mora igralec rešiti. Te uganke se po navadi rešijo tako, da igralec uporabi svojo okolico in ljudi v igri, da lahko najde rešitev tem ugankam. Pustolovske igre imajo tudi večji poudarek na zgodbo in pripovedovanje ali celo izgled okolice. Lahko jih razvrstimo kot: »Text Adventure« igre, »Graphic Adventure« igre, »Real-time 3D Adventure« igre in »Visual Novel« igre.



Slika 5: Primer »Real-time Adventure« igre, Firewatch

VIR: http://www.killapenguin.com (2017)

2.3.5 Športne igre

Športne igre so igre, ki simulirajo raznorazne vrste športa. Ekipe lahko krmiliš ti, računalnik ali drugi ljudi preko spleta. Športne igre razvrstimo kot: »Racing« igre, »Sports« igre in »Competitive« igre.

Slika 6: Primer »Sports« igre, FIFA 13

VIR: https://www.trueachievements.com (2017)

Strojna oprema

Za igranje računalniških iger potrebujemo napravo, ki je zmožna igro upravljat. Po navadi je ta naprava osebni računalnik, ampak dandanes lahko igramo igre na različnih napravah, kot so igralne konzole ali mobilni telefoni. Ena izmed prvih iger je bila narejena in igrana na računalniku.

Računalnik je naprava, ki je sposobna izvajati zaporedja aritmetičnih ali logičnih operacij. Podatke in informacije procesira skozi vhodno-izhodne enote, ki so v tem primeru tipkovnica ali miška. Ob tem še lahko priključimo več vhodno-izhodnih naprav kot so igralne palice ali igralni ploščki, ki so uporabljeni pri igralnih konzolah. Novejše igre potrebujejo zelo zmogljivo opremo, saj izvajajo veliko več procesov hkrati v primerjavi z igrami, ki so bile izdelane pred dvajsetimi leti. Svoj izdelek bom izdelal na računalniku in na njem ga tudi igral.

Igralna konzola je naprava, ki je namenjena igranju iger. Namesto tipkovnice in miške se uporablja igralni plošček in namesto računalniškega zaslona se uporablja televizor, ki je priključen na konzolo z HDMI priklopnikom. Za igranje iger na konzoli potrebujemo posebno vrsto CD-jev, ki imajo igro nameščeno. Igre lahko tudi pridobimo preko spleta in jih namestimo na samo konzolo.

Današnji telefoni so veliko zmogljivejši v primerjavi z osnovnimi telefoni ter so lahko uporabljeni za veliko več stvari kot so, preverjanje vremena, pisanje beležk ali igranje iger. Ker so pametni telefoni manj zmogljivejši kot osebni računalniki ali igralne konzole, so igre na telefonih bolj preproste in nimajo velike vizualne kvalitete.

Slika 7: Igranje iger na telefonu

VIR: http://www.techteria.com

Programska oprema

Glede izdelave igre sem potreboval programsko opremo oziroma razvojno okolje, v katerem sem jo naredil. Če hočemo narediti igro, lahko najdemo veliko različnih razvojnih okolij, pri katerih lahko naredimo svoj izdelek.

Imamo razvojna okolja, ki so namenjena samo izdelavi iger, kot so GameMaker Studio Pro, RPG Maker VX ali Clickteam Fusion. Ta okolja ne potrebujejo veliko programerskega znanja, saj so namenjena le za izdelavo iger in vsak program ima svoj sistem, da lahko svojo igro narediš. Po tem obstajajo razvojna okolja, ki potrebujejo več programerskega znanja, ter so namenjena izdelavi multimedijskih programov, kot so Allegro, PyGame ali XNA. Ti progami imajo vmesnike za zvok ali grafiko, da si uporabnik ne potrebuje izdelati tega sam. Na koncu še imamo razvojna okolja, ki potrebujejo ogromno programerskega znanja, saj v njih lahko izdelamo karkoli. Razvijalec le ima programske knjižice, ki mu zelo malo olajšajo delo, saj mora vsako stvar napisat sam. Primer takšnega okolja je C++ z knjižicami OpenGL, DirectX, GLEW in drugi.

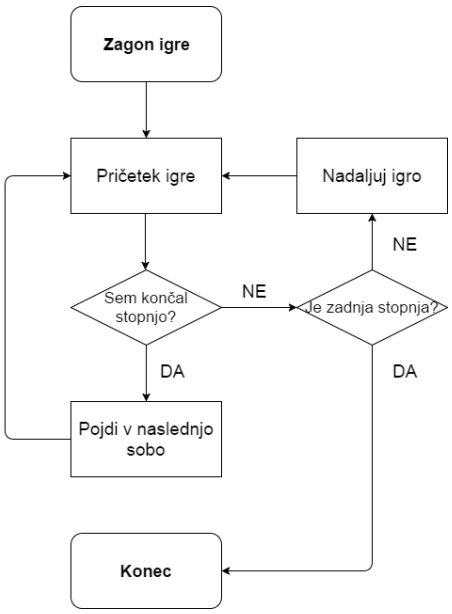
Odločil sem se, da bom uporabil GameMaker Studio Pro, saj je dober program za začetnike in je lahek za uporabo. GameMaker Studio je naredil Mark Overmars v Delphi programskem jeziku. Z tem programom lahko ustvariš igro popolnoma brez kode, samo z kodo ali mešanica obeh možnosti. GameMaker Studio ima svoj edinstven sistem »Zgrabi in Spusti«, kjer lahko objektu nastaviš predpripravljeno kodo, ki je že v samem programu. Ko ustvariš projekt v tem razvojnem okolju, ti program ustvari več map, ki loči ključne dele igre kot so liki, objekti, sobe in podobno. V samem programu lahko pišeš kodo, rišeš like in ustvarjaš sobe. Ima tudi svoj sistem, ki lahko zažene in poganja tvojo igro.

Slika 8: Logotip programa GameMaker

VIR:http://en.wikipedia.org

Arhitektura igre

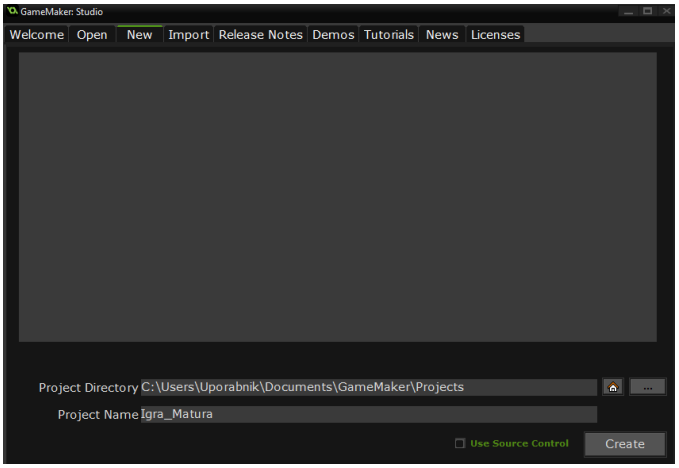
Ko uporabnik zažene igro, se prikaže prva soba in je njegov lik postavljen na začetek te sobe. Če je igralec prišel do konca sobe oziroma se dotaknil vijoličastega objekta, bo igralca prestavilo v naslednjo sobo. Ta proces se ponovi, dokler igralec ne pride ko konca igre. Če pa igralec še ni prišel do konca sobe, mora nadaljevati z igranjem dokler tega ne doseže. Če se igralec dotakne sovražnika, ga bo prestavilo na začetek sobe, drugače pa lahko nadaljuje z igranjem, če se sovražnikov izogne ali jih premaga.



Slika 9: Diagram poteka igre

Izdelava igre

Ko odpreš GameMaker Studio, se ti pojavi glavno okno. Od tega lahko izbereš meni, kjer lahko ustvariš novi projekt. Svoj projekt sem poimenoval Igra_Matura in sem ga shranil v osebno datoteko na računalniku.



Slika 10: Primer glavnega okna

Ko ustvariš novi projekt se ti polje za izdelavo pojavi in GameMaker Studio ti ustvari mape, kjer shranjuješ vse svoje datoteke in kodo. Mape, ki jih bom uporabljal so:

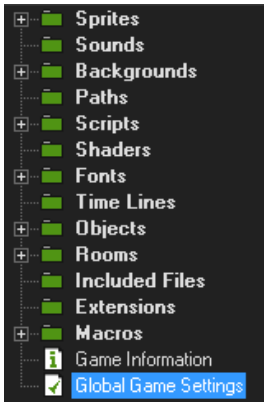
-Sprites (Mapa, v kateri se shranjuje ves slikovni material, ki se uporabljajo na objektih in jim daje videz),

-Backgrounds (Mapa, kjer se shranjuje slikovni material za slikovno ozadje igre),

-Scripts (Mapa, v kateri se shranjujejo skripti oziroma posebni deli kode),

-Objects (Mapa, ki shranjuje objekte igre, katerim nastaviš fizikalnost in jim določiš pravila ter vlogo v igri z kodo ali »Zgrabi in Spusti« metodo),

-Rooms (Mapa, ki shranjuje sobe igre ter vse objekte, ki so v vsaki posamezni sobi).



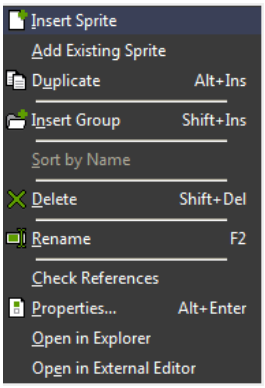
Slika 11: Prikaz vseh map v programu

6.1 Izdelava glavnega lika in ovire

6.1.1 Izdelava risbe glavnega lika

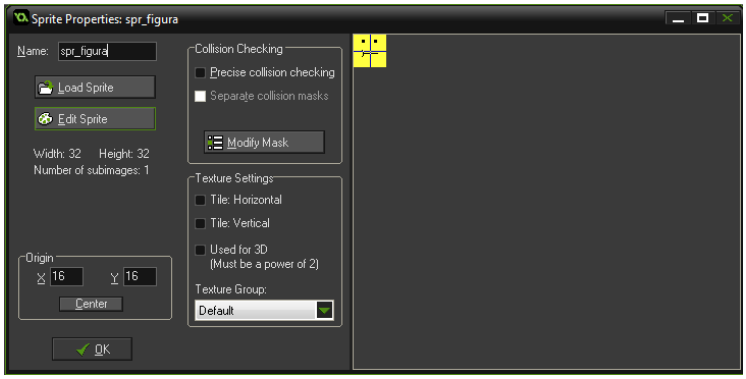
Preden sem se lotil pisanja kode za glavnega lika, sem ga moral narisat v programu. Če hočemo narisat katerikoli lik, moramo sledit tem navodilom:

-Sprva, moramo z desnim klikom klikniti na mapo z imenom »Sprites« in izbrati »Insert Sprite«,



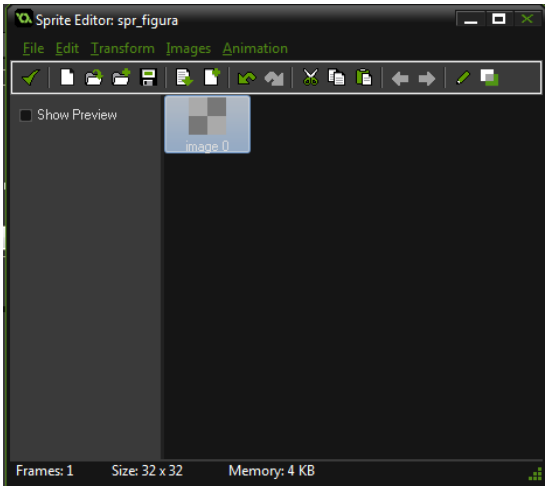
Slika 12: Navodila za dodajanje risbe

-Po izbiri se pojavi novo okno, kjer imaš nastavitve za risbo. V tem oknu lahko nastavimo ime risbe (spr_figura). Nato izberemo opcijo »Edit Sprite«, kjer bomo lahko izdelovali naš lik,



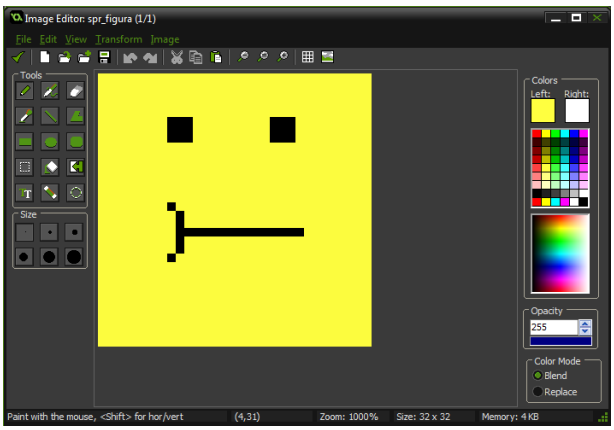
Slika 13: Glavno okno za spremembo risbe

-V novem oknu kliknemo Gumba Ctrl+N da naredimo novo in prazno datoteko. Po tem dvokliknemo na prazno datoteko,



Slika 14: Okno za dodajanje risbe

-V tem oknu imamo risalna orodja, barvno paleto in območje, kjer lahko narišemo naš lik.

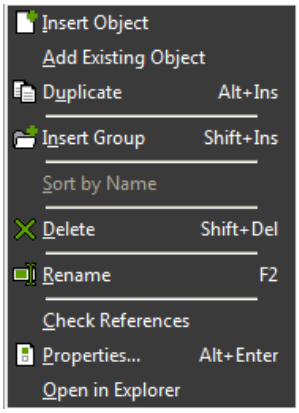


Slika 15: Okno za risanje

6.1.2 Programiranje glavnega lika

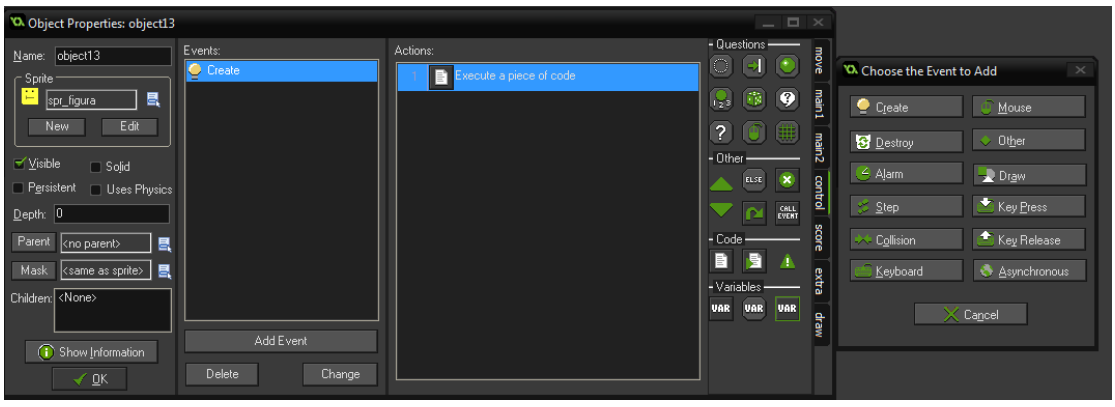
Ko sem lik narisal, sem ga začel programirati. Za programiranje vsakega elementa v igri moraš narediti objekt zanj z naslednjimi navodili:

-Z desnim klikom kliknemo na mapo z imenom »Objects« in izberemo »Insert Object«.



Slika 16: Navodila za dodajanje objekta

-Po izbiri se pojavi novo okno, kjer lahko spremenimo ime objekta, dodamo nove dogodke itd. Preimenujemo objekt v »obj_figura« in na levi strani določim objektu sličico »spr_figura«. Nato kliknemo na gump »Add Event« in izberemo »Create« dogodek. Ko imamo izbran ta dogodek, kliknemo na »control« na desni strani in potegnemo prvo risbo za »Code« v polje.



Slika 17: Navodila za dodajanje novega dogodka in akcije

Ko dvokliknemo na akcijo, lahko končno začnemo pisati svojo kodo. V Create dogodku bo akcija, ki bo ustvarila vse potrebne spremenljivke za igro, kot so vrednosti za skok, težnost, hitrost itd. Tukaj tudi ustvarimo za »Checkpoint«.

action

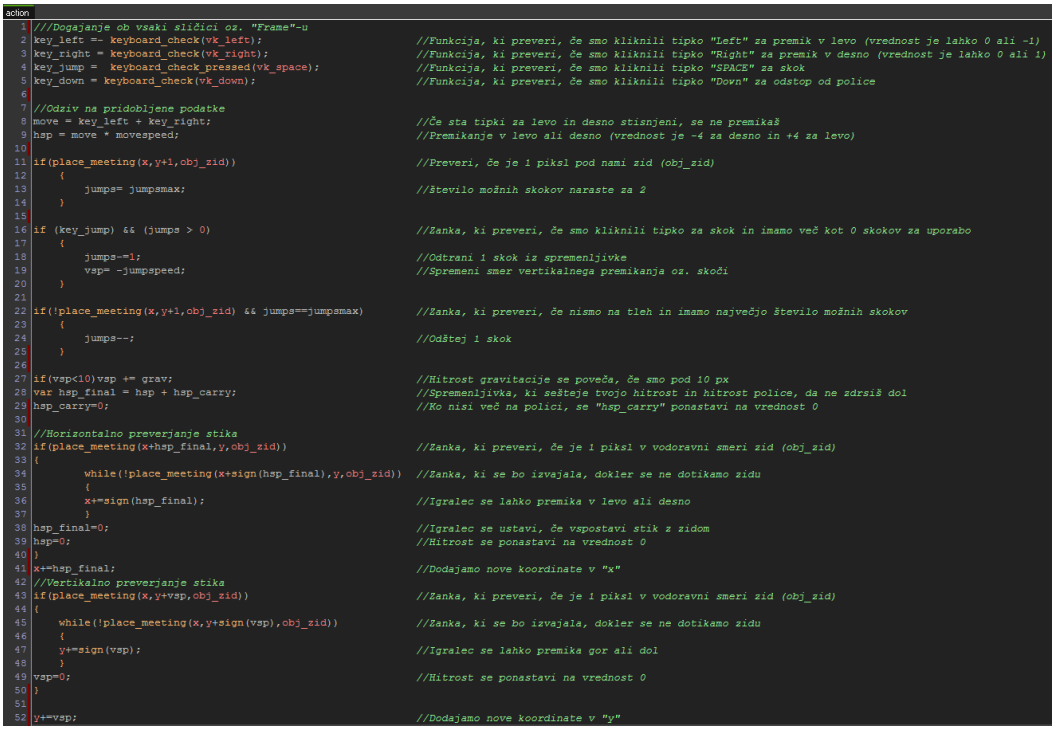
```
///Ustvarjanje spremenljivk

grav = 0.2;                 //Spremenljivka za gravitacijo

hsp = 0;                    //Spremenljivka za horizontalno hitrost
hsp_carry=0;                //Spremenljivka za hitrost na polici
vsp = 0;                    //Spremenljivka za vertikalno hitrost

jumpspeed = 4.5;            //Spremenljivka za hitrost skoka
movespeed = 4;              //Spremenljivka za hitrost premikanja

jumps=0;                    //Spremenljivka za število skokov
jumpsmax=2;                 //Spremenljivka za največjo število skokov

jumpspeed_normal=7;         //Spremenljivka za višino skoka
jumpspeed_powerup=14;       //Spremenljivka za višino skoka z močjo za skok

movespeed_normal=4;         //Spremenljivka za hitrost premikanja
movespeed_powerup=8;        //Spremenljivka za hitrost premikanja z močjo za premikanje

jumpspeed=jumpspeed_normal; //"Jumpspeed" je "jumpspeed" v kodi
movespeed=movespeed_normal; //"Movespeed" je "movespeed" v kodi

key_down=0; //Spremenljivka za, ko stistemo tipko za smer Dol

if(global.checkpointR==room)//Zanka, ki preveri, če je "checkpoint" v isti sobi kot je vrednost
{
    x=global.checkpointx;   //Našo koordinato "x" shrani v spremenljivko "checkpointx"
    y=global.checkpointy;   //Našo koordinato "y" shrani v spremenljivko "checkpointy"
}
```

Slika 18: Spremenljivke glavnega lika

Po ustvarjanju spremenljivk sem dodal novi dogodek z imenom »Step«. Ta dogodek se izvede pri vsaki sličici igre (v tem primeru 60 krat na sekundo).

Sprva sem napisal funkcije za premikanje in jim določil, na katero tipko se bo funkcija odzvala. Če bi kliknil tipko »A«, bi se začel premikati v levo smer. Nato sem dodal zanki, ki preverjata, če je možno narediti dvojni skok. Če ga hočemo narediti v igri, moramo začeti od tal in nam igra odšteje največjo število skokov.

Eden od najpomembnejših delov igre je preverjanje stika z drugimi objekti oziroma z zidovi. Izvedel sem to tako, da igra preveri, če je 1 piksel pred nami zid. Če je to res, se hitrost ponastavi in se s tem ustavimo. Če pa igra ne zazna nobenega zida pred nami, lahko nadaljujemo pot. To se posebej napiše za vertikalno in horizontalno smer. Na koncu sem še dodal možnosti skoka, če si ob zidu na levi in desni strani.



```
action
///Dogajanje ob vsaki sličici oz. "Frame"-u
key_left =- keyboard_check(vk_left);                    //Funkcija, ki preveri, če smo kliknili tipko "Left" za premik v levo (vrednost je lahko 0 ali -1)
key_right = keyboard_check(vk_right);                   //Funkcija, ki preveri, če smo kliknili tipko "Right" za premik v desno (vrednost je lahko 0 ali 1)
key_jump =  keyboard_check_pressed(vk_space);           //Funkcija, ki preveri, če smo kliknili tipko "SPACE" za skok
key_down = keyboard_check(vk_down);                     //Funkcija, ki preveri, če smo kliknili tipko "Down" za odstop od police

//Odziv na pridobljene podatke
move = key_left + key_right;                            //Če sta tipki za levo in desno stisnjeni, se ne premikaš
hsp = move * movespeed;                                 //Premikanje v levo ali desno (vrednost je -4 za desno in +4 za levo)

if(place_meeting(x,y+1,obj_zid))                        //Preveri, če je 1 piksl pod nami zid (obj_zid)
    {
        jumps= jumpsmax;                                //Število možnih skokov naraste za 2
    }

if (key_jump) && (jumps > 0)                            //Zanka, ki preveri, če smo kliknili tipko za skok in imamo več kot 0 skokov za uporabo
    {
        jumps-=1;                                       //Odtrani 1 skok iz spremenljivke
        vsp= -jumpspeed;                                //Spremeni smer vertikalnega premikanja oz. skoči
    }

if(!place_meeting(x,y+1,obj_zid) && jumps==jumpsmax)     //Zanka, ki preveri, če nismo na tleh in imamo največjo število možnih skokov
    {
        jumps--;                                        //Odštej 1 skok
    }

if(vsp<10)vsp += grav;                                  //Hitrost gravitacije se poveča, če smo pod 10 px
var hsp_final = hsp + hsp_carry;                        //Spremenljivka, ki sešteje tvojo hitrost in hitrost police, da ne zdrsiš dol
hsp_carry=0;                                            //Ko nisi več na polici, se "hsp_carry" ponastavi na vrednost 0

//Horizontalno preverjanje stika
if(place_meeting(x+hsp_final,y,obj_zid))                //Zanka, ki preveri, če je 1 piksl v vodoravni smeri zid (obj_zid)
{
        while(!place_meeting(x+sign(hsp_final),y,obj_zid))  //Zanka, ki se bo izvajala, dokler se ne dotikamo zidu
        {
        x+=sign(hsp_final);                             //Igralec se lahko premika v levo ali desno
        }
hsp_final=0;                                            //Igralec se ustavi, če vspostavi stik z zidom
hsp=0;                                                  //Hitrost se ponastavi na vrednost 0
}
x+=hsp_final;                                           //Dodajamo nove koordinate v "x"
//Vertikalno preverjanje stika
if(place_meeting(x,y+vsp,obj_zid))                      //Zanka, ki preveri, če je 1 piksl v vodoravni smeri zid (obj_zid)
{
    while(!place_meeting(x,y+sign(vsp),obj_zid))        //Zanka, ki se bo izvajala, dokler se ne dotikamo zidu
    {
    y+=sign(vsp);                                       //Igralec se lahko premika gor ali dol
    }
vsp=0;                                                  //Hitrost se ponastavi na vrednost 0
}

y+=vsp;                                                 //Dodajamo nove koordinate v "y"
```

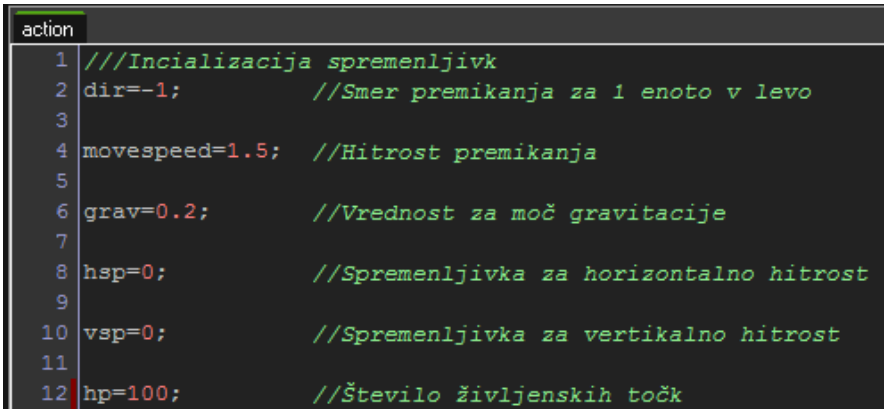
Slika 19: Delovanje glavnega lika

Ko sem napisal vso kodo za igranega lika, sem dodal novi objekt z imenom »obj_zid«, ki se šteje kot ovira, zid in tla v igri. Ta objekt nima nobene kode, saj je le potreben za interakcijo z drugimi objekti v igri. Če bi hotel en objekt imeti drugačni vpliv na tla ali zid, bi se koda napisala v tistem objektu in ne v objektu za zid. Za vsak objekt, ki sem ga naredil sem mu določil starša (ang. »Parent«) ta zid. To je namenjeno temu, da se bo vsak objekt enako odzval dotiku zida.

6.2 Izdelava sovražnikov

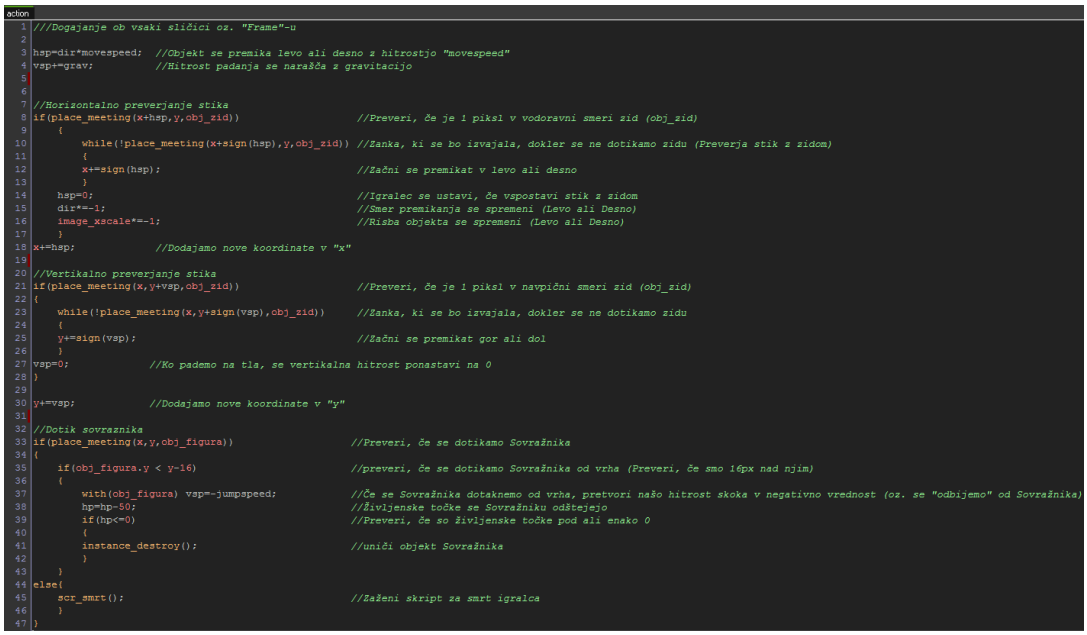
Zid ne bo edina ovira za igralca. Odločil sem se, da bom dodal nekaj sovražnikov v igro, ki lahko ubijejo igralca in ga prestavijo na začetek sobe. Ti sovražniki bodo med seboj podobni, le da bodo lahko sprejeli več škode, preden umrejo. Eden od teh sovražnikov lahko ostane na višini in ne pade dol. Vsaki objekt ima sliko narejeno po istem postopku, kot ga ima glavna figura in je tudi na isti način programiran.

Sovražniki imajo podobno kodo kot igralec. V »Create« predelu imajo podobno kodo, le da imajo spremenljivke za življenske točke.



Slika 20: Spremenljivke za objekte sovražnikov

V »Step« predelu je tudi marsikaj podobnega. Primer tega je preverjanje stika z objektom za tla in zid. V sovražnikovi kodi je zanka, ki preveri, od katere strani smo se sovražnika dotaknili. Če smo se ga dotaknili od vrha, se od sovražnika odbijejo in ga ranimo za 50 življenjskih točk. Če pa se ga dotaknemo od leve ali desne strani, se bo zagnal skript, ki igralca prestavi na začetek sobe. Vsi sovražniki se začnejo premikati v drugo smer, če se dotaknejo zidu, ki jih ovira pot.



Slika 21: Delovanje sovražnikov

Zeleni sovražnik je malo drugačen od modrega in rdečega, saj oba lahko padeta iz višin , medtem ko zeleni sovražnik ne mora. To je narejeno tako, da igra preverja, ali so pred

sovražnikom tla, na katerih se on lahko premika. Če igra opazi, da tal ni, se obrne v drugo smer.

```
//Vertikalno preverjanje stika
if(place_meeting(x,y+vsp,obj_zid))                  //Preveri, če je 1 piksl v navpični smeri zid (obj_zid)
{
    while(!place_meeting(x,y+sign(vsp),obj_zid))    //Zanka, ki se bo izvajala, dokler se ne dotikamo zidu
    {
    y+=sign(vsp);                                   //Začni se premikat gor ali dol
    }
vsp=0;              //Ko pademo na tla, se vertikalna hitrost ponastavi na 0

    if (robovi) && !position_meeting(x+(sprite_width/2)*dir, y+(sprite_height/2)+8, obj_zid) //Preveri 2 lokaciji ob Sovražniku: Polovico njegove višine pod njim in smer premikanja
    {
    dir*=-1;                                        //Če naj ne bi bilo tal pred sovražnikom, se smer premikanja spremeni (Levo ali Desno)
    }

}

y+=vsp; //Dodajamo nove koordinate v "y"
```

Slika 22: Dodatna kode zelenega sovražnika

6.3 Izdelava pripomočkov (ang. »Powerup«)

Igro sem popestril tako, da sem dodal dva tipa pripomočkov, ki lahko olajšajo igralčevo nalogo. Ti pripomočki lahko nadgradijo igralčeve sposobnosti v obliki pospešene hitrosti in višjega skoka. Učinek pripomočkov traja le par sekund, preden izgubijo svoj vpliv na igralca. Tile objekti so manjši od drugih objektov v igri.

Za izdelavo obeh pripomočkov sem jim narisal obliko in ustvaril posebej objekt zanj. Oba pripomočka imata drugačen dogodek v igri, ki se imenuje »Collision«. Ta dogodek se uporabi tako, da sprva izbereš objekt, ki se bo odzval na učinek pripomočka.

Sprva sem določil, da se bo predel kode izvajal na samem pripomočku in na določen objekt (v tem primeru je ta objekt igralec). Ko se bo igralec dotaknil pripomočka, se bo mu višina skoka povečala in se bo obarval v rdečo barvo. Na koncu se bo pripomoček uničil. Nastavil sem tudi čas, ki določa, kako dolgo bo ta učinek trajal. V tem primeru bo učinek trajal 300 milisekund oziroma 5 sekund. Ker sem uporabil stavek »alarm«, se mora nekaj zgoditi igralcu, ko čas poteče. Nato sem dodal nov dogodek igralcu, ki se imenuje »Alarm«. Ko bo zmanjkalo časa na pripompčku, se bo njegova hitrost in videz povrneta na prvotno stanje.

action

```
///Interakcija ob dotiku z Igralcem

with(obj_figura)                //Naslednja koda se bo izvajala na tem objektu in na objektu Igralca
{
jumpspeed=jumpspeed_powerup;    //Nastavi hitrost v hitrost Moči za hitrost
sprite_index=spr_figura_moc;    //Sliko Igralca spremeni v sliko za Moč za hitrost
alarm[0]=300;                   //Nastavimo dolžino trajanja te kode, preden se ponastavijo spremembe
}

instance_destroy();             //Uniči ta objekt
```

action

```
///Koda, ki se izvaja ob nekem dogodku

jumpspeed=jumpspeed_normal; //Ponastavitev hitrosti skoka igralca
sprite_index=spr_figura;    //Ponastavitev izgled igralca
```

Slika 23: Delovanje "Collision" in "Alarm" dogodka na pripomočku za hitrost

Podobno je pri drugem pripomočku, le da je vrednost za višino skoka zamenjana z vrednostjo za hitrost.

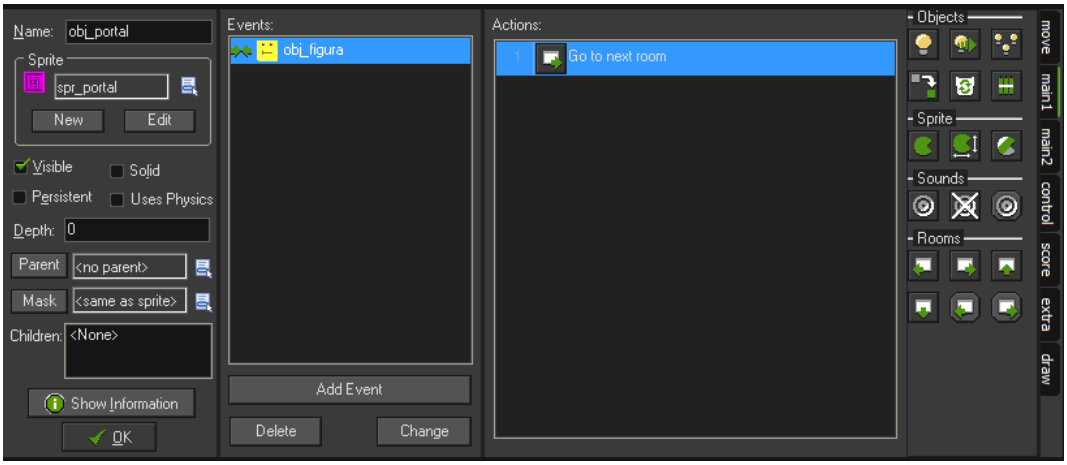
```
action
///Interakcija ob dotiku z Igralcem

with(obj_figura)                //Naslednja koda se bo izvajala na tem objektu in na objektu Igralca
{
movespeed=movespeed_powerup;    //Nastavi hitrost v hitrost Moči za hitrost
sprite_index=spr_figura_hiter;   //Sliko Igralca spremeni v sliko za Moč za hitrost
alarm[1]=300;                   //Nastavimo dolžino trajanja te kode, preden se ponastavijo spremembe
}

instance_destroy();             //Uniči ta objekt
```

```
action
///Koda, ki se izvaja ob nekem dogodku

movespeed=movespeed_normal; //Ponastavitev hitrosti premikanja igralca
sprite_index=spr_figura;    //Ponastavitev izgled igralca
```

Slika 24: Delovanje pripomočka za skok

6.4 Izdelava izhoda

Glavni cilj posamezne je, da igralec pride do izhoda. To se ponovi, dokler ne pride do zadnje sobe oziroma do konca igre. Za izdelavo izhoda nisem potreboval nobenega programiranja, saj že program ponuja akcije, ki sem jih potreboval. Sprva sem narisal lik za izhod, ustvaril objekt zanj in dodal dogodek, ki se bo izvedel, ko se bo igralec tega objekta dotaknil. V tem dogodku sem ob desni strani izbral »main1« in dodal blok »Go to next room«.



Slika 25: Delovanje objekta za prestop v naslednjo sobo

6.5 Izdelava polic

Da igralec ne bo hodil samo po tleh, sem se odločil, da bom še dodal dve vrsti polic oziroma površin: Polica, od katere lahko sestopiš in polico, ki se premika levo in desno. S temi policami lahko naredim bolj različne ter težje sobe.

6.5.1 Izdelava sestopne police

Polico sem izdelal malo drugače. Prva stvar, ki je opazna glede tega objekta je, da je za polovico manj visoka v primerjavi z vsemi ostalimi objekti. Če sem hotel narediti, da igralec lahko sestopi iz te police, sem se moral lotiti programiranja na drugačen način.

Z »Create« akcijo sem ustvaril spremenljivko, ki naredi objekt neviden (če se objekt ne vidi, je isto, kot da ga sploh ni v sobi). Nato sem uporabil novo akcijo z imenom »Draw«, ki nariše sliko police tam, kjer je postavljen objekt.

```
action
1  ///Inicializacija spremenljivk
2
3  sprite_index=-1; //Naj ne pokaže objekta oz. ne nariše slike le-tega
   action
   1  ///Risanje objekta
   2  draw_sprite(spr_polica,0,x,y); //Nariši sliko objekta
```

Slika 26: Delovanje dogodkov "Create" in "Draw" na objektu za sestopno polico

V »Step« dogodku sem napisal dve zanki. Prva zanka preverja, ali je igralec sploh v sobi, kjer je sestopna polica. Ta koda je vstavljena za vsak primer, če bi zrušili program zaradi funkcij, ki bi iskale informacije od igralca. Druga zanka pa preverja, ali stojimo na polici in smo kliknili tipko za sestop. Če sta izpolnjena oba pogoja, polica izgubi svojo trdnost in pademo iz nje. Skočimo lahko skozi polico na vrh nje, saj ima svojo trdnost samo takrat, ko stojimo na njej.

```
action
1  ///Dogajanje ob vsaki sličici oz. "Frame"-u
2  if(instance_exists(obj_figura))         //Zanka, ki preveri, če igralec sploh obstaja v instanci
3  {
4
5  if (round(obj_figura.y + (obj_figura.sprite_height/2)) > y) || (obj_figura.key_down)
6      {
7      mask_index= -1;
8      }
9      else
10     {
11     mask_index=spr_polica;
12     }
13 }
```

Slika 27: Delovanje sestopne police

6.5.2 Izdelava premikajočih polic

Ta polica je podobna prejšnji, le da ima svojo trdnost ves čas in da se premika levo in desno. Koda za to polico je ista kot koda za sovražnike in koda za prejšnjo polico, le da težnost nima vpliva na ta objekt. Premika se tako, da ko zadene zid na katerikoli strani, se bo smer premikanja spremenila.

```
action
1  ///Dogajanje ob vsaki sličici oz. "Frame"-u
2
3  mask_index=spr_policaP_X; //Nariši sliko za ta objekt
4  hsp=dir*movespeed;        //Objekt se premika levo ali desno z hitrostjo "movespeed"
5
6
7
8  //Horizontalno preverjanje stika
9  if(place_meeting(x+hsp,y,obj_zid))                    //Preveri, če je 1 piksl v vodoravni smeri zid (obj_zid)
10     {
11         while(!place_meeting(x+sign(hsp),y,obj_zid)) //Zanka, ki se bo izvajala, dokler se ne dotikamo zidu (Preverja stik z zidom)
12         {
13         x+=sign(hsp);                                //Začni se premikat v levo ali desno
14         }
15     hsp=0;                                           //Igralec se ustavi, če vspostavi stik z zidom
16     dir*=-1;                                         //Smer premikanja se spremeni (Levo ali Desno)
17     image_xscale*=-1;                                //Risba objekta se spremeni (Levo ali Desno)
18     }
19 x+=hsp;                    //Dodajamo nove koordinate v "x"
20
21
22
23 if(instance_exists(obj_figura))                      //Zanka, ki preveri, če igralec sploh obstaja v instanci
24 {
25 if place_meeting(x,y-1,obj_figura)                   //Zanka, ki preveri, če je 1px pod nami platforma
26          {
27          obj_figura.hsp_carry = hsp;                //Nastavi našo hitrost na hitrost police + našo prvotno hitrost.
28          }
29 }
```

Slika 28: Delovanje premikajoče police

6.6 Izdelava shranjevalnih točk

Za shranjevalno točko (ang. »Checkpoint«) je bilo potrebno narediti veliko stvari. Sprva sem narisal dve sličici zanj – ena risba, ki je vidna, ko točka ni vklopljena in risba, ki je vidna, ko je vklopljena. Le ena shranjevalna točka je lahko vklopljena ob kateremkoli času. Če se igralec dotakne naslednje točke, se prejšnja izklopi in se nova prižge. V »Create« dogodku sem le ustvaril spremenljivki, ki določita, katera risba je trenutno vidna in če se risba premika:

image_speed=0; //Slika se ne premika

image_index=0; //Slika se naj nastavi na prvo sličico

Nato sem v »Step« dogodku določil delovanje teh točk. Če se igralec dotakne shranjevalne točke, se položaj in število zaporedne točke shranita v globalne spremenljivke, ki so bile ustvarjene v prvi sobi (več o tem v poglavju 5.7). Naslednja zanka preveri, če je ta točka v isti

sobi, kjer je igralec. Nato preveri, katera točka je vklopljena in mu spremeni videz in hitrost vrtenja.

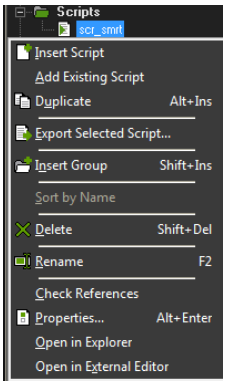
action

```
///Dogajanje ob vsaki sličici oz. "Frame"-u
image_angle -= 1;         //Slika se obrne za 1 stopinjo v smeri urinega kazalca

if(place_meeting(x,y,obj_figura)) //Preveri, če se igralec dotika tega objekta
{
    global.checkpoint=id; //id je trenuten id tega "checkpointa"
    global.checkpointx=x; //x koordinata je koordinata trenutnega "checkpointa"
    global.checkpointy=y; //y koordinata je koordinata trenutnega "checkpointa"
    global.checkpointR=room; //vrednost spremenljivke "room" je soba, v kateri je trenutno igralec
}

if (global.checkpointR==room) //Zanka, ki preveri, če je ta trenuten aktiven "checkpoint" v trenutni sobi
{
    if(global.checkpoint==id)
    {
        image_index=1;      //Spremeni indeks števila na drugo sliko objekta
        image_angle-=12;    //Naj spremeni hitrost premikanja slike
    }
    else image_index=0;     //Če je globalen "checkpoint" trenuten, mu naj ostane prvotna slika
}
```

Slika 29: Delovanje shranjevalne točke

Slika 30: Izgled shranjevalne točke (leva je vklopljena, desna je izklopljena)

Če sem hotel, da se igralec vrne do shranjevalne točke, sem moral narediti naredit skript, ki to izvede. Skript se naredi tako, da z desnim klikom klikneš na mapo z imenom »Script« in izbereš »Insert Script«. Svojo sem poimenoval scr_smrt, saj se izvede, ko igralec umre.



Slika 31: Navodila za izdelavo skripte

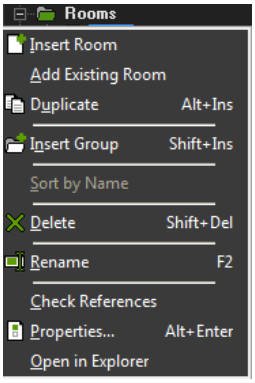
Skript ima preprosto kodo. Sprva preveri, če imamo prižgano shranjevalno točko. Če je ta pogoj dosežen, premakne igralca do prižgane točke. Če pa pogoj ni dosežen, nas vrne na začetek sobe. Ta skript je v kodi za igralca in se izvede, ko umre.

```
scr_smrt
1  if (global.checkpointR !=0)
2  {
3  room_goto(global.checkpointR);
4  }
5  else
6  {
7    room_restart();
8      }
```

Slika 32: Koda v skripti

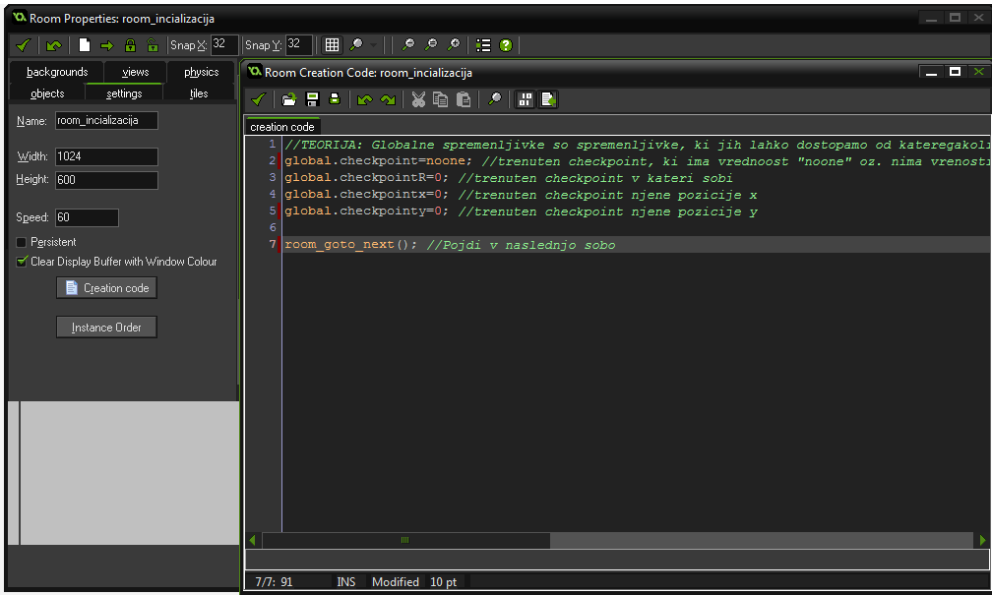
6.7 Izdelava sob

Za izdelavo sobe klikneš z desnim klikom na mapo z imenom »Rooms« in izbereš »Insert Room« opcijo.



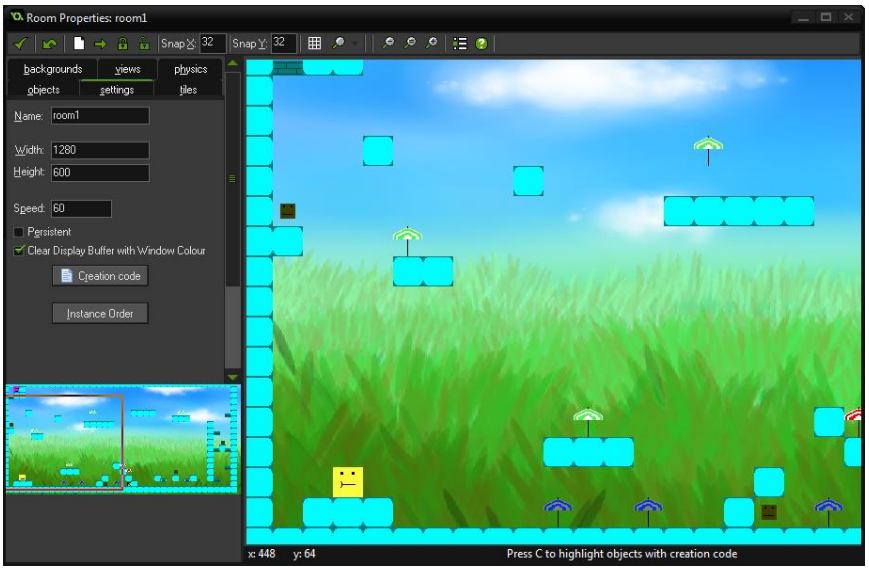
Slika 33: Navodila za dodajanje sobe

Pojavi se ti okno, kjer imaš na izbiro veliko stvari. Preden sem začel postavljati objekte in nastavljati ozadje, sem prvo sobo poimenoval »room_incializacija« in jo pustil prazno. Nato sem kliknil na gumb »Creation Code«, kjer se ti pojavi novo okno za vstavljanje kode. V tej sobi se nastavijo spremenljivke za shranjevalne točke in ko se igra vklopi takoj prestavi na naslednjo sobo.



Slika 34: Okno za izdelavo sobe in okno, ki prikazuje kodo sobe

V tem oknu imamo veliko različnih opcij in nastavitev, ampak bom le v tem primeru uporabljal »objects« za postavljanje objektov in »backgrounds« za vnos ozadja. Ustvaril sem več novih sob v igro in sem postavil naokrog ovire in zidove. Če sem hotel dodati ozadje, sem prva moral to ozadje postaviti v mapo »Backgrounds«. Nato greš v okno za izdelavo sob, izbereš področje »backgrounds« indodaš sliko kot ozadje. Za postavitev objektov izbereš področje »objects«, izbereš objekt in klikneš na polje, da ga postaviš. V področju »settings« nastaviš ločljivost oziroma velikost vsake sobe in hitrost sličic na sekundo.



Slika 35: Primer izdelane sobe

Testiranje

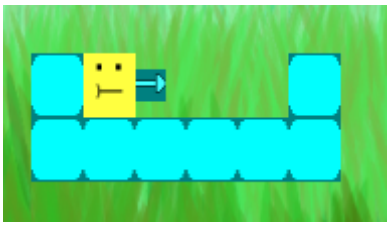
Testiranje sem izvedel med izdelavo izdelka in na koncu, ko sem ga izdelal v celoti. Vso testiranje sem naredil v igri, ki sem jo pred tem zagnal. Igro sem še podelil svojim prijateljem, da lahko najdejo napake in da te napake lahko popravim.

7.1 Testiranje odnosa med glavnim likom in drugimi objekti

Ta del testiranja je najpomembnejši, saj ima največji vpliv na delovanje igre. Testiranje sem izvedel tako, da sem zagnal igro in sem se dotaknil vsakega drugega objekta v igri.

7.1.1 Zid, sestopna polica, premikajoča polica in portal

Sprva sem se poskusil vsakega objekta, ki se šteje kot glavna ovira med tabo in koncem sobe. Zid me je ustavil na primeren način, sestopil sem lahko iz police in portal me je prestavil v naslednjo sobo. Težava je nastala s premikajočo polico, če si postavljen med zidom in to polico. Zgodi se, da se tvoje premikanje ustavi, dokler se te polica dotika od strani.



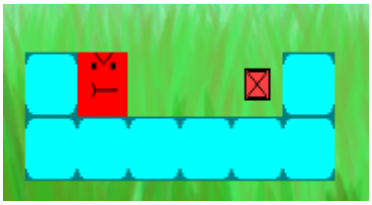
Slika 36: Primer težave z premikajočo polico

7.1.2 Sovražniki

Igralec se lahko odzove na sovražnike na dva različna načina. Če igralec skoči na zgornji del sovražnika, se od njega odbije in se sovražniku odštejejo življenjske točke. Ko sovražniku zmanjka življenjskih točk, ga premagamo in izgine. Če ima sovražnik večje število življenjskih točk, se od njega odbijemo in lahko ponovno skočimo nanj, dokler ga ne premagamo. Ko se glavni lik dotakne sovražnika iz leve ali desne strani, igralec umre in je premeščen na začetek sobe. Zeleni sovražnik se pravilno premika, saj ne pade iz višin.

7.1.3 Pripomočki

Ko se igralec dotakne zelenega ali rdečega pripomočka, se njegov videz pravilno spremeni in se mu njegove vrednosti za skok in hitrost primerno zvišajo. Ko poteče čas pripomočka, se videz in vrednosti glavnega lika ponastavijo.



Slika 37: Primer spremembe videza glavnega lika

7.2 Testiranje hitrosti delovanje igre

Testiral sem hitrost delovanje igre na namiznem računalniku, ki uporablja operacijski sistem Windows 7. Igro sem zagnal z GameMaker Studio in jo z istim programom tudi igral. V vsaki sobi sem nastavil, da najvišja hitrost delovanja ne sme presegati 60 sličic na sekundo. Ko sem igral igro, je delovanje ostalo na najvišjem nivoju ves čas in nisem zaznal, da bi se ta vrednost kdajkoli znižala. Ta preizkus je pomemben, saj če hitrost delovanja pade med igranjem, lahko igro upočasni.

7.3 Raziskovalna vprašanja

Izdelek sem izdeloval približno 3 tedne, ampak mi ga je uspelo izdelati do konca. Igro sem lahko naredil z manjšim znanjem programiranja in z nobenim znanjem jezika, ki ga program uporablja. Menim, da lahko katerikoli človek naredi najpreprostejšo igro, ampak samo, če si vzame čas in najde primerna navodila za izdelavo le-tega.

Po približno 20 urami uporabe programa GameMaker Studio, sklepam, da je program preprost za uporabo in sem se učil ta program uporabljati za zelo kratek čas. Menim, da lahko program vsaka oseba uporablja, saj je zelo dostopen in lahek za uporabo.

Zaključek

Za svoj izdelek sem naredil računalniško igro. Ta igra ni popolna, saj ji lahko dodam veliko več elementov, ki bi popestrilo zabavo, ki jo že trenutno ponuja. Od tega sem se naučil, da lahko kdorkoli naredi svojo preprosto igro, čeprav nima veliko izkušenj v programiranju. Programi, kot so GameMaker Studio, so preprosti za uporabo in vedno lahko dobiš pomoč od svetovnega spleta, ne glede, na katero vrsto igre izdeluješ.

Že od nekdaj sem hotel narediti svojo računalniško igro. Že od otroštva me je to področje zelo zanimalo, ampak nisem vedel, kje bi moral začeti in kaj uporabiti. S tem izdelkom sem zdaj pridobil veliko izkušenj glede izdelave računalniških iger, ter sem pridobil spodbudo, da naj na tem področju svoje znanje razširim.

Za konkurenčno igro je dandanes treba dodati mnogo različnih stvari, da lahko uporabnika zabava za čim več časa. Ideja za igro je lahko tudi preprosta, ampak mora imeti elemente, ki privabi ljudi, da bi tvojo igro igrali. Ti elementi so lahko raznoliki, saj so odvisni od osebe, ki igro izdeluje in si izmisli nove ideje zanj.

Najbolj pomemben del izdelave računalniške igre je, da vztrajaš in da se držiš načrta. Če hočeš nekaj narediti izven načrta, je najbolje, da to stvar dodaš na koncu le-tega. Pomembno je tudi, da izdelek dokončaš, da lahko začneš uporabljati svoje pridobljeno znanje na nove izdelke.

Viri

[1] Computer – Wikipedia, obiskano 23.2.2017, dosegljivo na:

https://en.wikipedia.org/wiki/Computer

[2] History of video games – Wikipedia, obiskano 24.2.2017, dosegljivo na:

https://en.wikipedia.org/wiki/History_of_video_games

[3] Video game – Wikipedia, obiskano 23.2.2017, dosegljivo na:

https://en.wikipedia.org/wiki/Video_game

[4] GameMaker | YoYo Games, obiskano 20.2.2017, dosegljivo na:

http://www.yoyogames.com/gamemaker