Izrada računalne igre koristeći Gamemaker:Studio

Omrčen, Luka

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:907550

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET OSIJEK

Sveučilišni studij

IZRADA RAČUNALNE IGRE KORISTEĆI GAMEMAKER:STUDIO

Završni rad

Luka Omrčen

Osijek, 2016

SADRŽAJ

1. UVOD 1
1.1. Zadatak završnog rada 1
2. GLAVNI DIO RADA 2
2.1. Opis i analiza razvojnog okruženja GameMaker:Studio 2
2.1.1. Uvod u programsko sučelje GameMaker:Studio 2
2.1.2. Programski jezik „GameMaker Language“ 6
2.2. Postupak izrade 2D računalne igre 11
2.2.1. Kreiranje i upravljanje grafičkim resursima 11
2.2.2. Izrada 2D grafičkog sučelja igre 14
2.2.3. Objekti i deklariranje događaja 16
2.2.4. Konstruiranje smislene logike i implementacija algoritama 19
2.3. Izvoz igre na željenu razvojnu platformu 27
3. ZAKLJUČAK 29
LITERATURA 30
SAŽETAK 31
ABSTRACT 32
ŽIVOTOPIS 33
PRILOZI 34
Izvorni GameMaker:Studio .gmx kod 34

1. UVOD

GameMaker je softverska platforma orijentirana za stvaranje 2D igara i ostalih grafičkih aplikacija razvijena od strane YOYO Games-a. GameMaker je dizajniran u cilju kako bi se svima omogućilo lako kreiranje i razvijanje svoje ideje i projekata bez potrebe za učenjem ili potrebnim predznanjem za nekim od kompleksnih programskih jezika kao što je C++ , jednostavnom metodom „Drag and drop“ odnosno „ Povuci i postavi“. „Povuci i postavi“ metoda omogućava korisniku kreiranje igre ili neke druge grafičke aplikacije tako da vizualno organizira i postavi ikone na zaslonu, koja će softveru predstavljati niz funkcija i logičkih uvjeta koji se pojavljuju tijekom izvođenja aplikacije. GameMaker također ima ugrađeni programski jezik nazvan „GameMaker Language“ ili skraćeno GML. GML omogućava korisniku pisanje programskog koda koji će biti izvršen za vrijeme pokretanja i izvođenja aplikacije. Postoji veliki broj verzija GameMaker-a na raspolaganju, ali većina njih je zastarjela ili nije podržana. GameMaker:Studio je prva verzija GameMakera koja je stekla popularnost na tržištu upravo zbog kompatibilnosti da korisnik svoju kreiranu igru ili aplikaciju izvede na više različitih uređaja i operacijskih sustava uključujući PC, Mac, Linux ili Android na mobilnim ili desktop verzijama. GameMaker:Studio se većinom koristi za izradu složenijih igara i ostalih grafičkih aplikacija korištenjem GML-a pri pisanju programskog koda, uz slabiju primjenu „Drag and drop“ metode. U glavnom dijelu rada objašnjene su prednosti i karakteristike GameMaker:Studio-a pri razvoju aplikacija u odnosu na druge razvojne programe, detaljan uvod i opis programskog okruženja GameMaker:Studio-a kao i osnovni postupak izrade 2D računalne igre sa odgovarajućom smislenom logikom polazeći od osnovne ideje, rješavanje problema prilikom samoga razvoja sve do izdavanja gotovog proizvoda opisujući pritom sve tehnologije i alate koji se dodatno mogu koristiti.

1.1. Zadatak završnog rada

Zadatak završnog rada je opisati i analizirati GameMaker:Studio kao platformu za razvoj 2D igara i ostalih sličnih grafičkih aplikacija, prikazati karakteristike i prednosti GameMaker:Studio-a pri razvoju aplikacija u odnosu na druga razvojna okruženja, napraviti uvod i opis programskog okruženja, napraviti primjer računalne 2D igre polazeći od osnovne ideje i rješavanje problema prilikom procesa razvoja sve do izdavanja gotovog proizvoda opisujući pritom sve tehnologije i alate koji su korišteni.

2. GLAVNI DIO RADA

2.1. Opis i analiza razvojnog okruženja GameMaker:Studio

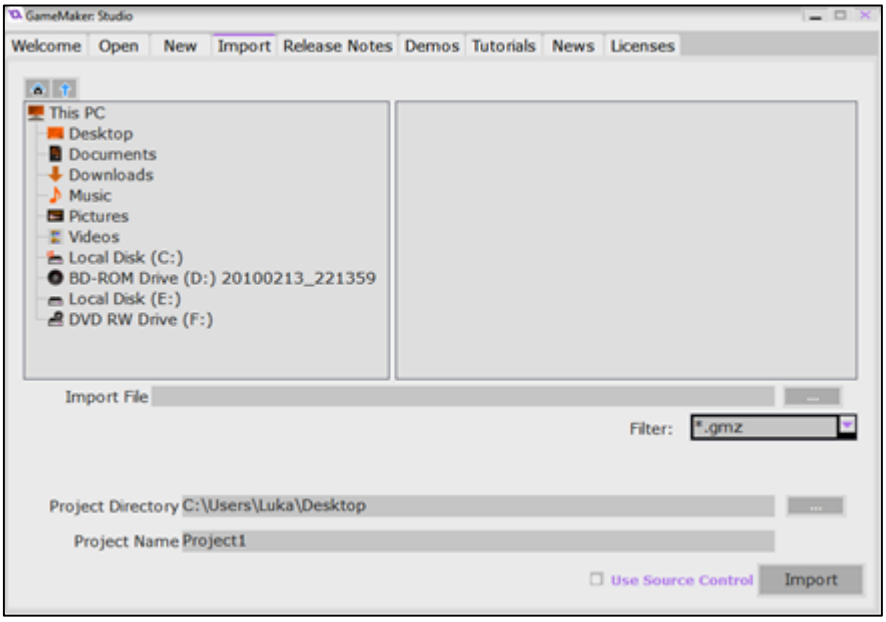
GameMaker:Studio je prva verzija GameMaker-a koja omogućava svojim korisnicima izvoz svojih proizvode na više različitih operativnih sustava ili uređaja. GameMaker:Studio prvenstveno služi za izradu 2D grafičkih aplikacija i igra, omogućuje korištenje i kreiranje ograničene 3D grafike koja još nije potpuno optimizirana. GameMaker:Studio je dizajniran tako da je njegov razvojni programski jezik „GameMaker Language“ (GML) univerzalan i kompatibilan prilikom izvoza na bilo koji operacijskih sustav današnjice s minimalnim promjenama izvornog koda prilikom izvoza. To znači da autor aplikacije ne mora utrošiti dodatno vrijeme za uređivanje i pisanje nove sintakse programa koja bi odgovarala željenom operacijskom sustavu. GML je snažno integriran u programsko okruženje GameMaker:Studio-a te daje mnogo veću fleksibilnost i kontrolu od standardnih programskih jezika koje poznajemo (C++,Java..), upravo iz razloga što postoji jako velika baza biblioteka i funkcija, kao i mogućnost organiziranja i stvaranja korisničkih biblioteka i resursa koje će prilikom izvoza na bilo koju operacijsku platformu biti kompatibilne i potpuno funkcionalne. Postoje nekoliko paketa GameMaker:Studio, a može se preuzeti i besplatna verzija programa s ograničenim funkcijama na web stranici http://www.yoyogames.com/get koja sadrži sve potrebne informacije i dokumentaciju u vezi instaliranja i korištenja razvojnog okruženja.

2.1.1. Uvod u programsko sučelje GameMaker:Studio

Nakon svakog pokretanja GameMaker:Studio pojavljuje se tzv. „Welcome window“ odnosno „Prozor dobrodošlice“ na kojemu se mogu vidjeti razni savjeti kako najbolje iskoristiti sve pogodnosti koje nudi GameMaker:Studio. Također mogu se vidjeti i nedavno zatvoreni projekti na kojima korisnik radi. Da bi se otvorio željeni projekt potrebno je dva puta kliknuti na naziv projekta. Osim što pokazuje posljednje korištene, prozor dobrodošlice ima i druge ugrađene značajke koje se mogu koristiti. Na vrhu prozora postoji devet malih izbornika *(sl. 2.1)* na temelju navedenih funkcija koje pružaju korisniku, a nazivi izbornika su navedeni engleskim jezikom kao i cijelo programsko sučelje po zadanim početnim postavkama. Trenutno u ovoj fazi ne postoji jezična podrška programa na Hrvatskom jeziku, ali je u fazi razvoja.

Sl. 2.1 *Prozor dobrodošlice GameMaker:Studio-a*

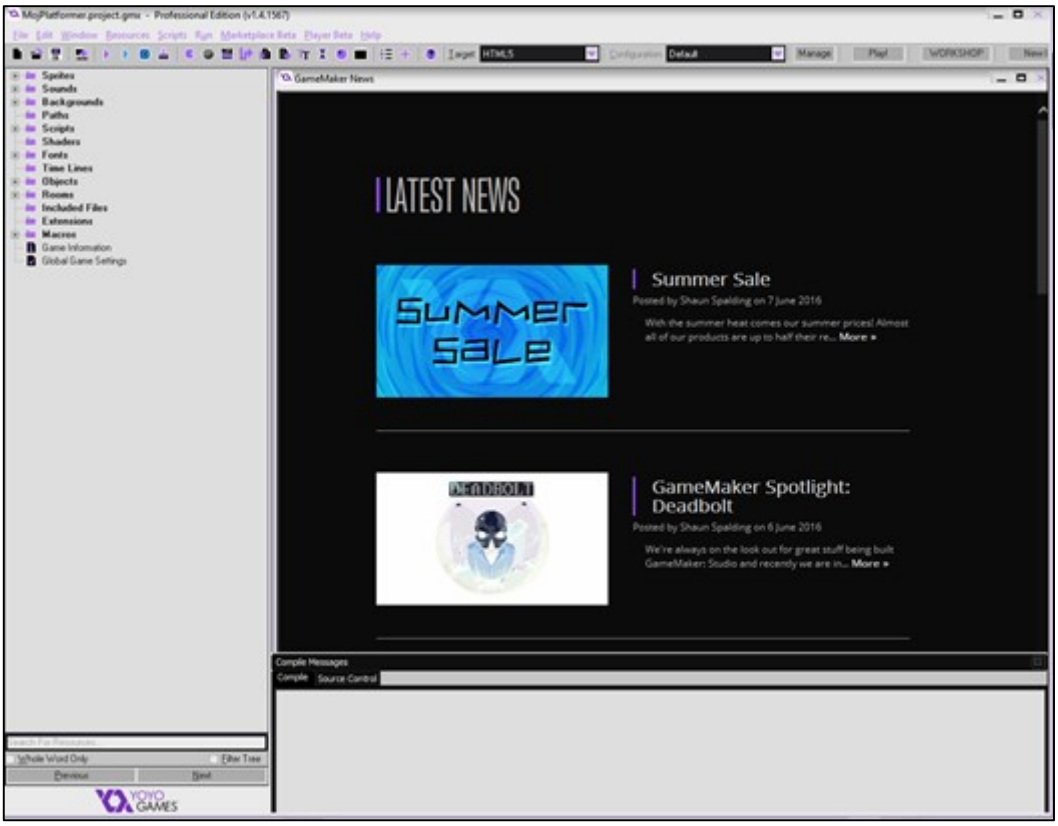
Izbornik **Open** omogućava pregledavanje i otvaranje projekata pod ekstenzijom .gmx. Izbornik **New** omogućava pokretanje i otvaranje novog projekta kojemu je potrebno navesti. **Import** (*sl. 2.2*) je izbornik koji omogućuje uvoz projekta iz prijašnjih verzija GameMaker-a ili iz stvorenih sigurnosnih kopija. Osim sigurnosnih kopija omogućava i uvoz GameMaker ZIP (GMZ) datoteka koje su nešto drugačije od sigurnosnih kopija jer zbog sažimanja datoteka zauzima manje memorije.



Sl. 2.2 *GameMaker:Studio prikaz import naredbe*

Pomoću izbornika **Release Notes** možemo vidjeti trenutne novosti i nadogradnje u vezi GameMaker-a. **Demos** je izbornik pomoću kojega možemo preuzimati demo projekte s dostupnim izvornim datotekama, koje pomažu novim korisnicima da lakše započnu sa kreiranjem njihovih prvih projekata. Izbornik **Tutorials** sadrži veliku bazu dostupnih materijala preko kojih se može naučiti različite programske tehnike i vještine u cilju lakšeg korištenja razvojnog okruženja. Izbornik **News** prikazuje najnovije vijesti i događaje koje se nalazi na stranici YOYO Games-a, i na kraju izbornik **Licenses** prikazuje sve korisničke podatke i licencu razvojnog okruženja. Ovi izbornici osiguravaju brzi pristup svim GameMaker informacijama i funkcijama.

Nakon kreiranja novog projekt pomoću izbornika **New** pokreće se glavni programski zaslon (*sl. 2.3.)*



Sl. 2.3 *GameMaker:Studio glavni programski zaslon*

Glavni programski zaslon sastoji se od nekoliko dijelova. Najvažniji i najčešće korišteni dio glavnog izbornika je **glavna alatna traka** koja predstavlja mjesto gdje se mogu pronaći svi bitni elementi, odnosno ona predstavlja brz i jednostavan način pristupa željenim funkcijama koje su bitne za stvaranje resursa u projektu, kao i za spremanje te izvoz gotovih projekata**. Glavnoj alatnoj traci** može se pristupiti s glavnog zaslona *(sl. 2.4)* i pomoću padajućeg izbornika.

Sl. 2.4 *GameMaker:Studio glavni programski zaslon*

U prvom dijelu glavne alatne trake s plavim obrubom, nalaze se funkcije vezane za otvaranje, kreiranje, izvoz te spremanje novih projekata. U drugom dijelu glavne alatne trake s crvenim obrubom nalaze se funkcije za ispitivanje projekta. U ovom dijelu nalaze se gumbi za pokretanje te testiranje grafičke aplikacije u normalnom i „debuq" okruženju, pokretanje i zaustavljanje web servera te gumb za čišćenje predmemorije. „Debuq" okruženje omogućuje informacije korisniku o radu grafičke aplikacije tijekom pokretanja, izvođenja te završetka njezinog rada prilikom kojeg se mogu vidjeti sredstva i razne varijable za praćenje rada aplikacije. WebServer je poseban GameMaker-ov mikro poslužitelj koji se koristi najčešće za testiranje HTML5 modula. U trećem dijelu alatne trake s žutim obrubom nalaze se gumbi koje koji omogućavaju pristup globalnim postavkama projekta kao i sa postavkama upravljanja dodatnih proširenja. Globalne postavke sadrže sve važne podatke i informacije s kojima se upravlja i opisuje izrađena grafička aplikacija, koje su prvenstveno bitne za sami izvoz aplikacije na neku od željenih razvojnih platformi. Dodatna proširenja su nastavci kreirani od strane korisnika koji se mogu pridodati funkcionalnost izraženim grafičkim aplikacijama, koji se dodaju gotovim funkcijama obuhvaćenim od strane GameMaker-a.

Drugi važan dio glavnog programskog zaslona su **padajući izbornici** *(sl. 2.5)* koji omogućavaju pristup svim funkcijama kao što pruža i **glavna alatna traka** uz nekoliko dodatnih izbornika i funkcija.

Sl. 2.5 *GameMaker:Studio padajući izbornik*

Dodatni izbornici koji se ne mogu naći u glavnoj alatnoj traci su **Marketplace_Beta**, i **Help** izbornik. Izbornik **Marketplace_Beta** omogućuje korisniku pristup „YOYO games" stranici za razmjenu, kupovinu i prodaju svih materijala i resursa namijenjeni za GameMaker platformu. Izbornik **Help** omogućuje korisniku da pristupi sadržaju u kojem su objedinjene sve naredbe, funkcije, biblioteka, linkovi za GameMaker stranicu i njezine sadržaje, informacije o korisničkim podatcima i licenci kao i dio za vijesti i forum.

Posljednji dio glavnog programskog zaslona obuhvaća Stablo resursa *(sl. 2.6)*, koji predstavlja jedan od glavnih aspekata u GameMaker-ovom programskom sučelju. U njemu se nalazi svaki pojedini resurs koji se koristi u projektu, poredano prema vrsti kao i najčešće korištenim resursima. Stablo resursa pohranjuje i ostvaruje trajnu vezu između bilo koje vanjske datoteke i unutrašnjosti programa koje je nužna za pokretanje grafičke aplikacije, kao što su pojedine DLL datoteke ili slike, koja se ne mora nužno nalaziti unutar datoteke projekta.

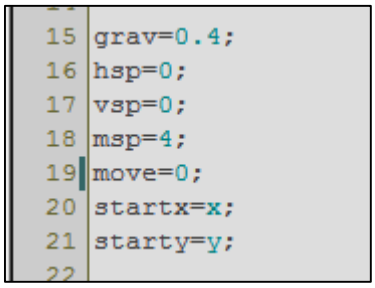
Sl. 2.6 *GameMaker:Studio stablo resursa*

2.1.2. Programski jezik „GameMaker Language“

GameMaker Language (GML) je programski jezik koji dolazi u paketu s GameMaker:Studio-om te je u potpunosti optimiziran i integriran za navedeno razvojno sučelje. Upravo zbog toga GameMaker Language programski jezik (skraćeno GML) daje veću fleksibilnost i kontrolu korisniku pri radu u svojem projektu koje ne bi imao da koristi neki drugi programski jezik koji nije optimiziran za sva razvojna okruženja. GML posjeduje veliki spektar gotovih klasa i biblioteka koje olakšavaju korisniku rad, te smanjuju vrijeme koje bi morao provesti kreiranjem dodatnih linija koda za određenu radnju, a jednostavno može pozvati funkciju ili klasu koja mu je potrebna.

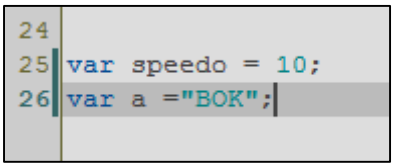
Kao i u svakom programskom jeziku, u GameMaker-u osnovni element programskog jezika je **varijabla**. U GameMakeru, postoje četiri vrste glavne vrste varijabli: konstruktori, lokalne varijable, globalne varijable i polja.

Konstruktori *(sl. 2.7)* su varijable koje ne sadrže početnu vrijednost, korisnik je inicijalizira u određenoj instanci odnosno pojedinom objektu. Početna vrijednost se ostvaruje pomoću operatora pridruživanja *(=)*. Ova vrsta varijabla je vezana za određenu instancu i nije dostupna nijednom drugom dijelu projekta. Dodijeljena vrijednost može biti brojčana ili niz znakova. Ako se radi o znakovnom nizu potrebno je nakon operatora jednakosti dodati navodnike, a nalazi se u memoriji sve dok instanca postoji.



Sl. 2.7 *Primjer deklaracije konstruktora unutar instance*

Lokalne varijable *(sl. 2.8)* su varijable koje se deklariraju na određenom dijelu programskog bloka ili skripti koja se stvara u trenutku pokretanja aplikacije. Nakon što se programski blok ili skripta izvrši, one se brišu iz memorije. Lokalne varijable su prepoznatljive u GameMaker-u po tome ispred svoga imena imaju oznaku **var.** Razlika između lokalne varijable i konstruktora je ta da se lokalna varijabla može koristiti tijekom cijelog programskog bloka ili skripte u kojoj se nalazi, dok konstruktor ne.



Sl. 2.8 *Primjer deklaracije lokalne varijable*

Globalne varijable *(sl. 2.9)* su varijable koje mogu biti deklarirane u svakom dijelu programskog bloka ili skripte, a može se koristiti također u svakom dijelu programskog koda. Globalne varijable su trajne i zauzimaju određeno mjesto u memoriji sve dok se ručno ne izbrišu. U GameMakeru-u da bi inicijalizirali globalnu varijablu potrebno je upisati prvo **global.** ispred imena varijable, a zatim pomoću operatora jednakosti (=) pridružiti određenu vrijednost.

```
1  ///Pocetne varijable
2  global.deda_stani=1;
3  global.provjera1=0;
4  global.provjera2=1;
5  global.provjera3=1;
6  global.provjera4=1;
7  global.provjera5=1;
8  global.provjera6=1;
```

Sl. 2.9 *Primjer deklaracije globalne varijable*

Polja ili nizovi su poseban tip varijable koji se sastoji od više elemenata različitih vrijednosti koje se vode pod zajedničkim imenom. U GameMaker:Studio postoje jednodimenzionalna i dvodimenzionalna polja ili nizovi.

Jednodimenzionalna polja ili nizovi *(sl. 2.10)* su polja koja sadrže i grupiraju vrijednosti elemenata u jednom stupcu. Inicijalizacija jednodimenzionalnog polja se vrši tako da se upiše ime polja a zatim je potrebno otvoriti uglatu zagradu i upisati vrijednost elementa (indeks) pod kojim se želi upisati određena vrijednost, zatvorit uglate zagrade i poslije znaka jednakosti (=) unijeti željenu vrijednost polja.

```
5 msg[0] = " Hmmm ... nisam skupio sve novcice, bolje da se vratim";
6 max_text = 0;
```

Sl. 2.10 *Primjer deklaracije jednodimenzionalnog polja*

Dvodimenzionalna polja ili nizovi *(sl. 2.11)* su polja koja sadrže i grupiraju vrijednosti elemenata u dva stupca. Za razliku od jednodimenzionalnih polja dvodimenzionalna polja imaju dva indeksa, što znači veću kategorizaciju i organiziranost u polju. Da bi deklarirali dvodimenzionalno polje potrebno je navesti ime polja i napraviti uglatu zagradu u kojima će se unijeti vrijednosti elemenata (indeks) odvojeni zarezom i poslije znaka jednakosti (=) potrebno je unijeti vrijednost koja će se pohranit pod određenim indeksima u polje. Prvi indeks se može gledati kao redak u tablici, dok drugi indeks se općenito gleda kao stupac.

```
11 polje[0,0]="Riki martine";
12 polje[0,1]="dobices batine";
13 polje[1,1]=3;
```

Sl. 2.11 *Primjer deklaracije dvodimenzionalnog polja*

Kao i u svakom programskom jeziku, pa tako i u GameMaker:Studio postoji podržana sintaksa za grananje programske logike u kodu. Grananje programske logike u GameMaker:Studio ostvarujemo pomoću poznatih **if**, **else** i **else if** naredbi koje često nazivamo uvjeti. Osim uvjeta postoje i petlje koje omogućavaju korisniku da smanji količinu programskog koda koji se ponavlja određeni ili neodređeni broj puta, a ostvaruju se pomoću **while** i **for** petlji.

Programska naredba **if** *(sl 2.12)* omogućuje korisniku da se ispita neki uvjet koji je postavljen unutar zagrada. Ako je zadani uvjet istinit, izvršava se programski kod koji se nalazi unutar vitičastih zagrada. U suprotnom, ako je zadani uvjet lažan, programski kod koji se nalazi unutar vitičastih zagrada se preskače i izvršava se sljedeća linija koda poslije zagrada.

```
if(global.provjera1==0)
{
    instance_create(x,y,textbox1);
    global.provjera1=1;

}
```

Sl. 2.12 *Primjer IF uvijeta*

Else i **Else if** *(sl 2.13)* su uvjeti koje se izvršavaju u kombinaciji s **if** uvjetom. Naredba else prilikom deklaracija ne sadrži nikakav uvjet i izvršava se ukoliko je uvjet unutar if-a lažan, dok kod else if ukoliko je if uvjet neistinit, ispituje se uvjet else if naredbe. Ako je uvjet unutar else if naredbe istinit, izvršava se programski dio koda unutar vitičastih zagrada, u suprotnom programski kod unutar vitičastih zagrada se preskače.

```
if(snowrandom==1)
{
   effect_create_below(ef_cloud,random(room_width),view_yview[0],1,c_white)
   effect_create_below(ef_snow,random(room_width),view_yview[0],1,c_white)
}
else if(snowrandom==2)
{
    effect_create_below(ef_snow,random(room_width),view_yview[0],2,c_white)
}
else
{
effect_create_below(ef_snow,random(room_width),view_yview[0],3,c_white)
effect_create_below(ef_cloud,random(room_width),view_yview[0],1,c_white)
}
```

Sl. 2.13 *Primjer ELSE i ELSE IF uvijeta*

While *(sl 2.14)* petlja je vrsta petlje u kojoj se programski kod koji se nalazi unutar vitičastih zagrada ponavlja se dok je uvjet koji se nalazi unutar while petlje istinit. Ipak ova vrsta petlje je specifična zbog mogućnosti postavljanja beskonačne petlje. Beskonačna petlja je petlja koja nema izlaza. To uzrokuje izvoženje koda u petlji zauvijek, odnosno program nikada neće izaći iz te petlje.

```
while(x<100)
{
    x=x+1;
}
```

Sl. 2.14 *Primjer WHILE petlje*

For *(sl 2.15)* petlja je jedna od najčešće korištenih petlja u GameMaker:Studio-u. Prilikom korištenja for petlje moramo deklarirati novu ili postojeću varijablu, postaviti joj uvjet koliko puta će se ponavljati i kao treći uvjet postaviti deklariranu varijablu da se prilikom svakog ponavljanja povećava vrijednost deklarirane varijable ili smanjuje. Kada uvjet u for petlji postane lažan, for petlja se prekida izvršavati.

```
1 for (i=0; i<10; i++){
2     array[i]=5;
3 }
```

Sl. 2.15 *Primjer FOR petlje*

Funkcije *(sl 2.16)* unutar GameMaker:Studio-a mogu se promatrati kao posebne logičke cjeline unutar razvojnog programa. Funkciju možemo definirati kao dio programskog koda koji se može pozvati u bilo kojem dijelu razvojnog programskog okruženja koji sadrži prethodno napisane i definirane postupke koji se izvršavaju unutar funkcije. Da bi se pozvala funkcija potrebno je napisati valjano ime funkcije, postaviti željene vrijednosti odnosno argumente funkcije odgovarajućeg tipa npr. ime_funkcije (argument1,argument2). Funkcija je zamišljena nakon izvođenja programskog koda da vrati vrijednost koja odgovara postavljenom tipu (pomoću naredbe return). Ako funkcija umjesto tipa ima oznaku void, tada ne vraća vrijednost, odnosno radi se o nepovratnoj funkciji. U GameMaker:Studio postoji niz ugrađenih funkcija koje se mogu pronaći u korisničkim priručniku.

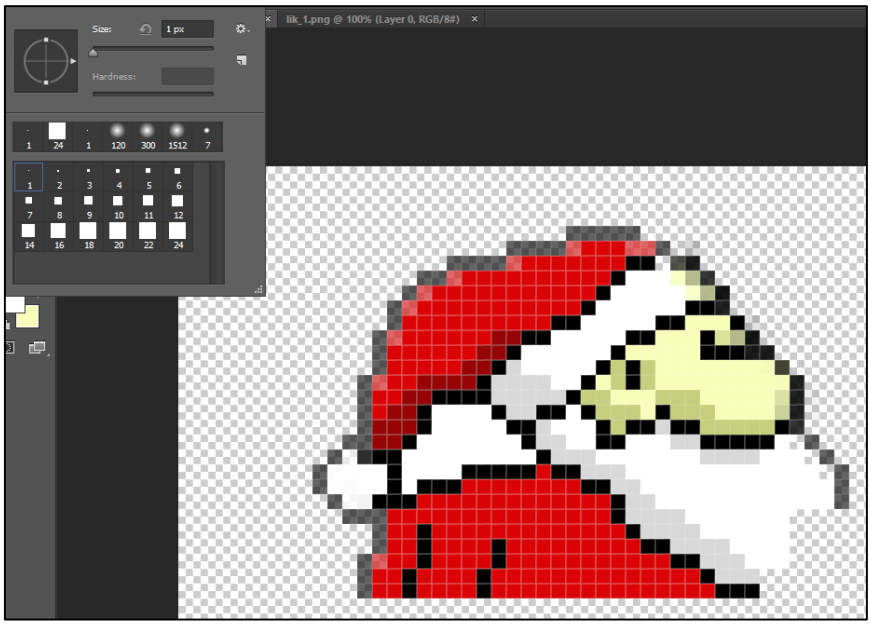
```
3 draw_text_ext(x+137,y+4,output,string_height("W"),640-127);
4 draw_sprite(face_spr_deda,0,x,y);
```

Sl. 2.16 *Primjer pozivanja funkcije*

2.2. Postupak izrade 2D računalne igre

2.2.1. Kreiranje i upravljanje grafičkim resursima

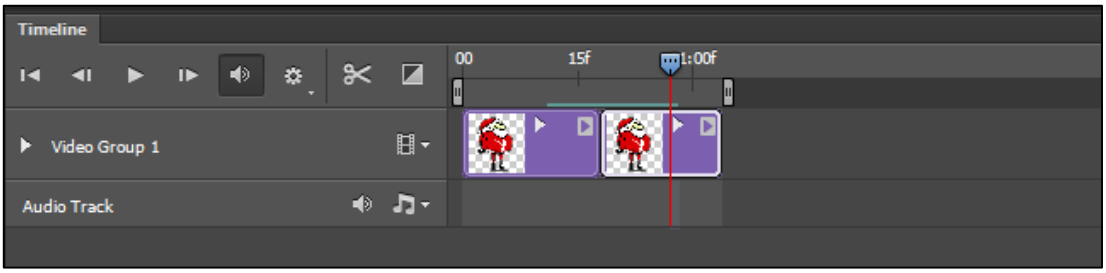
Prilikom kreiranja 2D računalne igre, osnovno polazište poslije same ideje je izrada i korištenje grafičkih resursa. Grafički resursi su važan čimbenik u izradi grafičke aplikacije koji predstavljaju jedinstveni identitet po kojemu je aplikacija prepoznatljiva od strane raznih korisnika. Za izradu grafičkih resursa postoje razne aplikacije za obradu grafičkih materijala, a najpoznatiji i najčešće korišten program je Adobe Photoshop. Pošto se radi o 2D računalnoj igrici, najčešća i najpopularnija korištena metoda grafičke obrade je „Pixel Art“. „Pixel Art“ predstavlja oblik grafičke umjetnosti gdje se stvara slika u obliku malih kvadratića odnosno „pixela“. Ovakav način grafičke obrade predstavlja stare odnosno ograničene verzije grafičkih aplikacija ili igrica koje su su bile korištene sredinom devedesetih godina, a nedavno su stekle takozvanu „retro“ popularnost.



Sl. 2.17 *Grafička izrada glavnog lika*

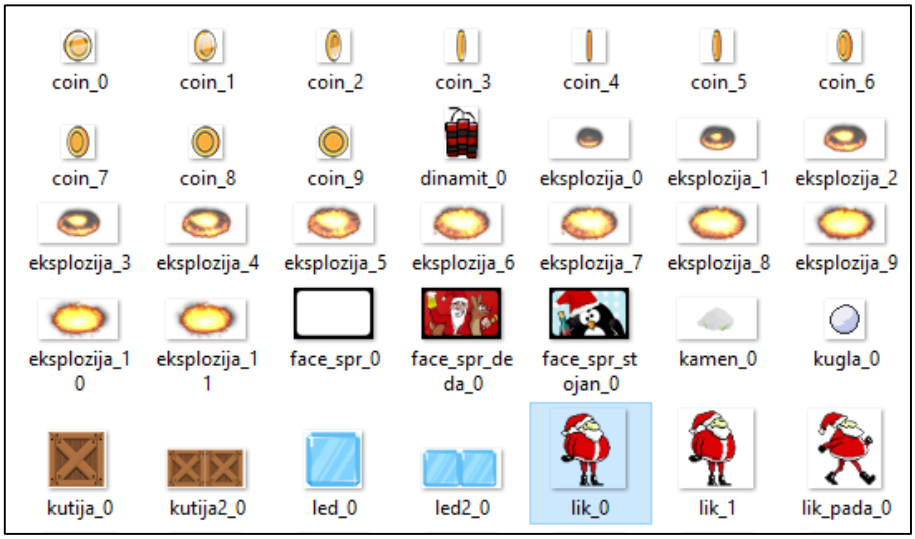
Osim kreiranja grafičkih materijala u grafičkim razvojnim okruženjima, postoje razne grafičke web-stranice orijentirane raznim materijalima koji se mogu iskoristiti za grafički identitet a trenutno najpopularnija po broju skinutih grafičkih materijala je http://opengameart.org/. Važno je pogledati upute i dokumentaciju autora grafičkog djela, odnosno je li potrebno otkupiti licencu za korištenje grafičkog materijala u aplikaciji ili se autor odriče svoje licence i slobodno daje na korištenje cijeli predani grafički materijal.

Kako bi ostvarili željenu animaciju unutar 2D igre, potrebno je reproducirati niz slika kako bi se stvorio dojam nekakve radnje ili kretanja. Da se to ostvariti potrebno je koristiti opciju **Motion** u grafičkom okruženju Adobe Photoshop koji nam daje alate za izradu reprodukcije naziva **Timeline**. Potrebno je postaviti prvu sliku i kliknuti na opciju **cut** koja će stvoriti prvi okvir, odnosno spremljenu prvu sliku unutar izbornika Timeline. Nakon toga moguće je modificirati trenutnu sliku te nakon završetka potrebno je ponovno kliknuti na opciju cut koja će stvoriti drugi okvir, odnosno drugu sliku unutar izbornika Timeline *(sl 2.18)*. Prilikom pritiska gumba za pokretanje videa moguće je vidjeti izmjenu prve i druge slike pri kojom se dobiva dojam 2D animacije. Animaciju je potrebno spremit u .png format i odabrati opciju da svaki okvir Timeline-a predstavlja jednu sliku .png formata.



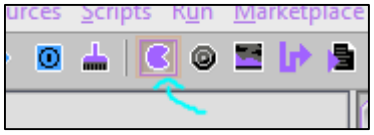
Sl. 2.18 *Izrada 2D animacije pomoću Timeline-a*

Nakon što su realizirani svi potrebni grafički resursi koji su potrebni za izradu 2D igre potrebno ih je komplimentirati pod valjanim nazivom i formatom, kako bi se moglo što lakše upravljati sa grafičkim resursima po primjeru na *sl 2.19*.



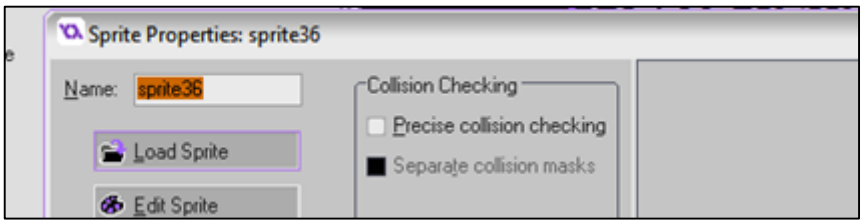
Sl. 2.19 Primjer *upravljanja i objedinjavanja grafičkih materijala*

U GameMaker:Studio postoji poseban naziv za grafičke resurse, a oni se nazivaju **sprite**. Sprite je uglavnom vizualni prikaz objekta koji se nalazi unutar grafičke aplikacije, iako se mogu koristiti za razne druge svrhe. Sprite se može sastojati od jedne ili više slika, koje se mogu reproducirati jedna za drugom pri čemu dobivamo grafičku 2D animaciju. GameMaker:Studio podržava razne grafičke formate izrade sprite-ova ali je preporučljivo kao „tradicionalni" pristup koristiti .png radi transparentnosti slike. Da bismo kreirali novi sprite potrebno je kliknuti na ikonicu u alatnoj traci po primjeru na *sl. 2.20* koji će biti pohranjeni u stablo resursa.



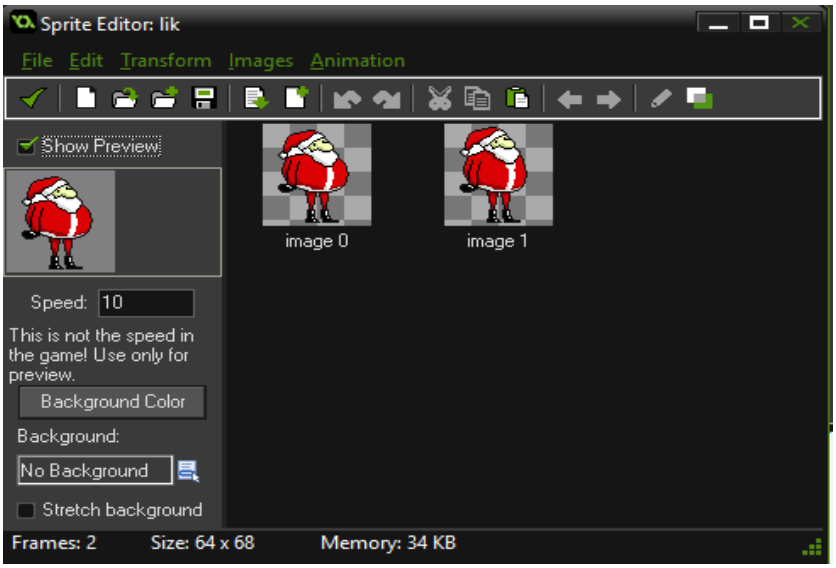
Sl. 2.20 *Gumb za kreiranje novog sprite-a*

Nakon toga pojavljuje se izbornik s novokreiranim sprite-om u kojem je moguće odabrati naziv, željenu sliku ili niz slika kao reprodukciju pritiskom na dugme „load sprite" po primjeru na *sl. 2.21.*



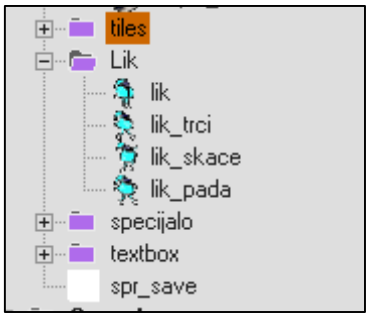
Sl. 2.21 *Gumb za odabiranje slike*

Nakon otvaranje i odabiranja željene slike, pritiskom na gumb „edit sprite" ,po primjeru na *sl.2.21,* pokreće se novi izbornik gdje je moguće vršiti dodatnu obradu slike ukoliko je to potrebno. Osim izbornika za dodatnu obradu slike nalazi se i mali preglednik s lijeve strane koji pokazuje trenutnu animaciju s odgovarajućom kontrolom brzine ispod njega *(sl.2.22)*. Pri dnu izbornika prikazane su informacije o dimenziji slika, broj slika kao i prostoru koji zauzima trenutni sprite u radnom memoriji prilikom pokretanja igre *(sl. 2.22)*. Ukoliko je izvršena izmjena nad trenutnim slikama u sprite-u potrebno je spremiti željene promjene na gumbić koji ima oznaku zelene kvačice, u suprotnom neće se spremiti izmjene koje su izvršene nad slikama.



Sl. 2.22 *Izbornik za pregled i dodatnu obradu slike*

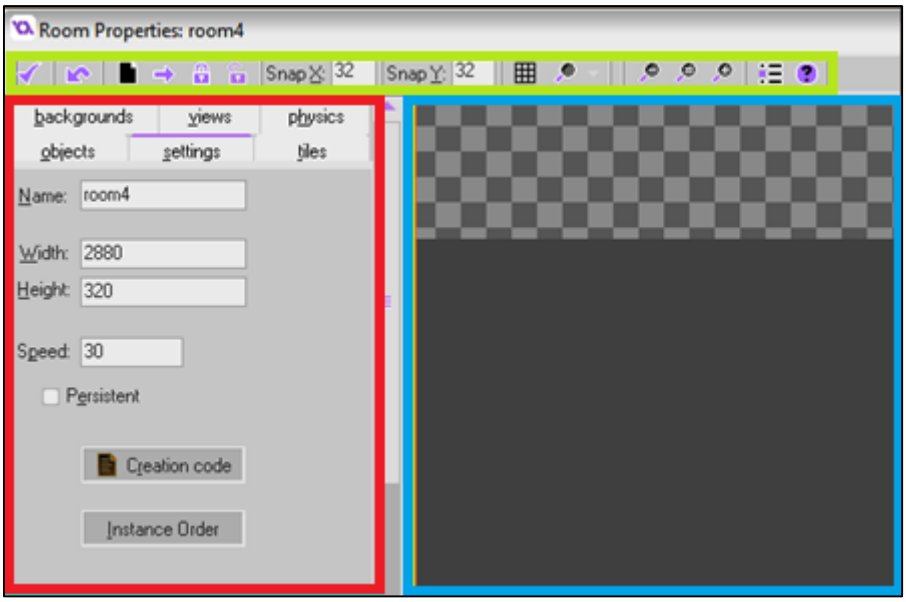
Sve kreirane i pohranjene sprite-ove moguće je pronaći u stablu resursa *(sl 2.23)* na lijevoj strani razvojnog okruženja, a da bi lakše upravljali i koristili novonastale sprite-ove potrebno je dati jasne nazive i organizirati ih u grupe radi jednostavnosti i preglednosti.



Sl. 2.23 *Organiziranje i grupiranje sprite-ova u stablu resursa*

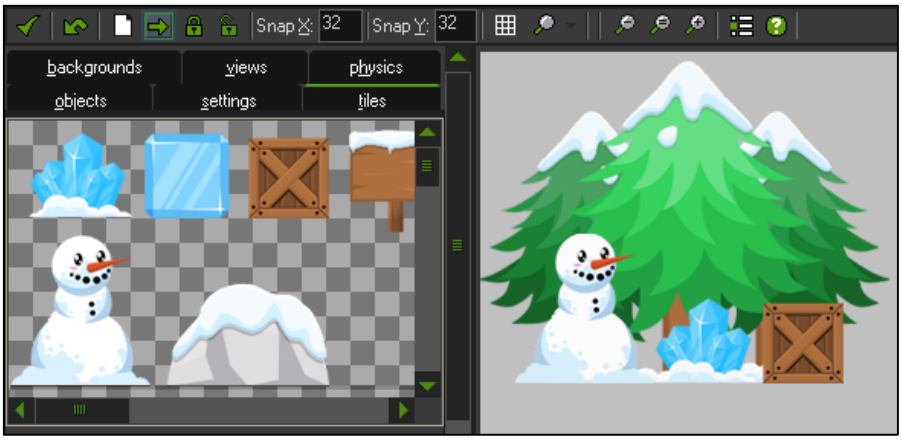
2.2.2. Izrada 2D grafičkog sučelja igre

Za izradu 2D grafičkog sučelja u GameMaker:Studio koristi se potprogramsko okruženje implementirano u GameMaker:Studio koje se naziva „Rooms“ odnosno sobe. Sobe predstavljaju mjesto u kojemu se implementiraju sva grafička okruženja, kreirani objekti i njihova interakcija s okolinom kreiranim u programskom kodu. GameMaker:Studio dozvoljava kreiranje jedne ili više sobe s mogućnosti uređivanja svake sobe posebno, u kojemu se mogu odrediti razne grafičke karakteristike kao rezolucija, grafički detalji,broj slika u sekundi i sl. Prilikom kreiranja nove sobe prema u stablu resursa pod grupom „Rooms“ pojavljuje se novo kreirana soba i otvaranjem sobe dvostrukim klikom nude se razne funkcije prilikom uređivanja novonastale sobe *(sl. 2.24).*



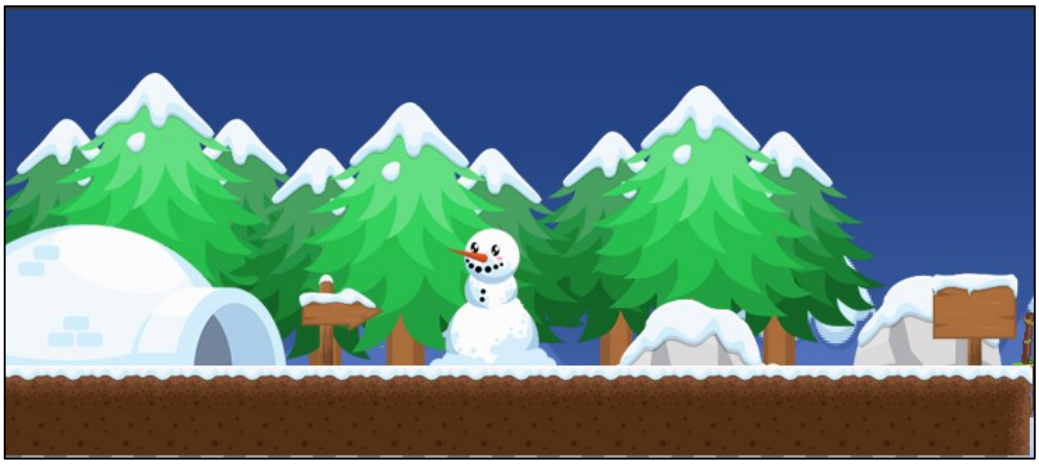
Sl. 2.24 *Prikaz uređivača sobe i njegovih izbornika*

Dio uređivača sobe sa zelenim obrubom *(sl,. 2.24)* predstavlja alatnu traku uređivača u kojem se nalaze osnovne funkcije pri radu s uređivanjem sobe. Dio uređivača s crvenim obrubom nalaze se šest izbornika sa svojim podizbornicima u kojima se kreiraju osnovni podatci o sobi kao što su implementacija grafičkih materijala i pozadina, veličina sobe, širina sobe, broj reprodukcije sličica u sekundi, rezolucija igre koja će biti prikazana korisnikom zaslonu itd. Dio uređivača s plavim obrubom predstavlja prostor u koji se smještaju i dizajniraju svi grafički materijali, objekti i strukturi koje se objedinjuju u interaktivno grafičko sučelje 2D igre. Da bi implementirali grafičke materijale potrebno je otvoriti izbornik „tiles“ *(sl. 2.24)*. Da bi GameMaker:Studio prepoznao grafičke materijale potrebno je postaviti putanju direktorija u kojem se nalaze kreirani grafički podatci. Ukoliko su svi grafički materijali odvojeni i spremljeni u .png format materijali se nalaze jedan do drugog s lijeve strane uređivača *(sl. 2.25)* .



Sl. 2.25 *Postavljanje grafičkih materijala u sobu*

Prilikom postavljanja grafičkih materijala u prostor sobe potrebno je s kombinacijom lijevog klika miša i držanja „alt“ tipke na tipkovnici preći preko željenog grafičkog materijala kako bi ga se označilo i da bi ga se smjestilo u prostor sobe potrebno je kliknuti na željenu poziciju unutar prostora sobe *(sl. 2.25)*. GameMaker:Studio omogućava postavljanja grafičkih materijala u slojeve, odnosno ukoliko se neki materijal nalazi u nižem sloju od drugoga, on će prilikom dizajniranja sobe biti ispred materijala koji se nalazi u višem sloju. Važno je napomenuti da grafičko sučelje unutar sobe nema nikakvu interakciju s objektima i s programskim kodom, te jedino služi u svrhu stvaranja jedinstvenog grafičkog identiteta *(sl. 2.26)*.

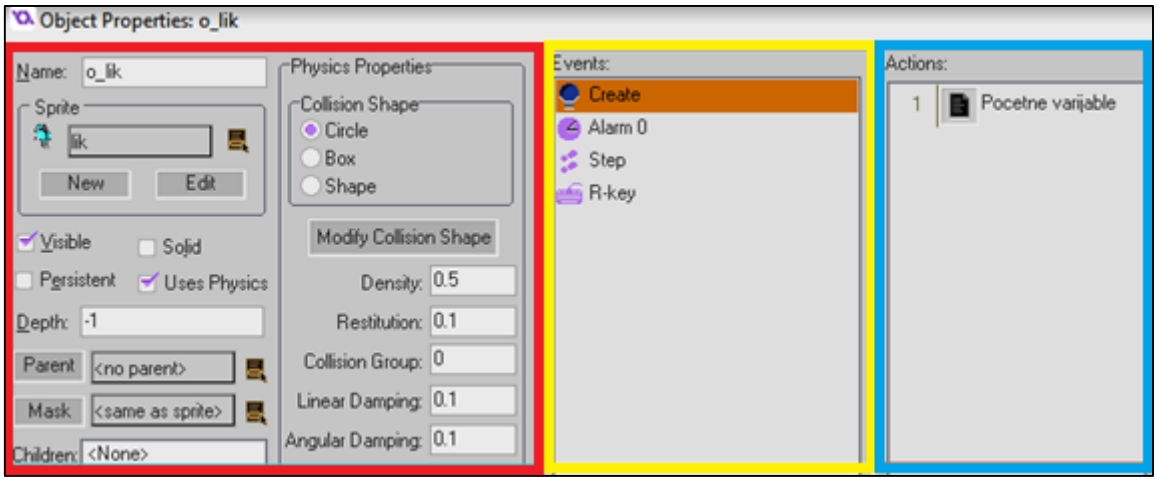


Sl. 2.26 *Prikaz gotovog grafičkog sučelja sobe*

2.2.3. Objekti i deklariranje događaja

Objekti predstavljaju specijalni resurs u GameMaker:Studio-u pomoću kojih se kontrolira svaki dio igre ili druge grafičke aplikacije kako bi izvršila neka specifična zadana radnja. Objektu možemo dodijeliti specifično ponašanje, omogućiti mu reakciju na određene događaje s kojima se susreće kao i interakciju s okolinom u kojoj se nalazi. Objekte možemo promatrati kao nacrte u kojima se nalazi napisani programski kod. Objekt u GameMaker:Studio omogućava grafičku vizualizaciju krajnjem korisniku i prikazuje što se događa prilikom izvođenja programskog koda, što čini programiranje u GameMaker-u puno lakše u odnosu na druga programska okruženja koja nemaju mogućnost grafičke interakcije objekata unutar izvođenja programskog koda. Ponašanje objekata vršimo kreiranjem vlastitih ili ugrađenih događaja koji su ugrađeni u GameMaker:Studio. Događajima definiramo ponašanje i postupke pomoću kojega se kontrolira određeni objekt. Postoje nekoliko vrsta ugrađenih događaja kao što su : **Create, Step, Draw, Alarm** koji se razliku po redoslijedu izvršavanja i svojim posebnim karakteristikama. Kako bi kreirali novi objekt

potrebno je iz izbornika Resources kliknuti na opciju „New object“ te će se otvoriti prozor prikazan na *sl. 2.27.*



Sl. 2.27 *Prikaz izbornika novog objekta*

Dio označen crvenim obrubom *(sl.2.27)* predstavlja opće informacije o objektu. Objektu se može postaviti ime, postaviti mu odgovarajući s**prite** odnosno pridodati grafički materijal koji smo pohranili u stablo resursa i koje će predstavljati objekt unutar grafičkog sučelja, odrediti mu vidljivost unutar grafičkog sučelja, deklarirat ga kao dijete koji nasljeđuje događaje nekog drugog objekata ili ga deklarirati kao roditelj nekih drugih objekata i razne ugrađene fizičke karakteristike ako se koristi fizički modul unutar GameMaker:Studio-a. Dio označen šutim obrubom *(sl.2.27)* predstavlja događaje koje karakteriziraju ponašanje i radnju objekta. Dio označen plavim obrubom *(sl.2.27)* predstavlja jednu ili više akcija u kojem se nalazi programski kod koji se izvršava ovisno o događaju u kojem se nalazi jer se svaki događaj razlikuje po svojem redoslijedu izvršenja i karakteristikama.

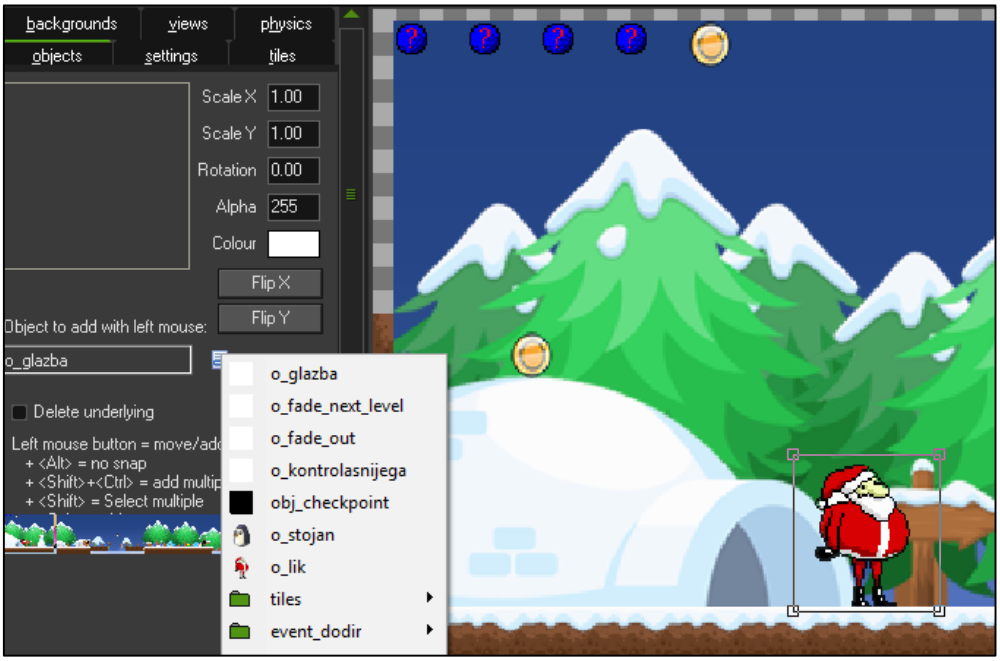
Create je događaj u kojemu se najčešće deklariraju i inicijaliziraju varijable i atributi objekta jer je njegova karakteristika da se po redoslijedu izvršavanja on izvršava prvi, prije svih ostalih događaja.

Step događaj je jedan od najkorisnijih događaja unutar GameMaker:Studio poslije Create događaja. Step događaj je poseban upravo zbog toga što izvršava zadanu akciju odnosno programski kod unutar nje nakon reprodukcije određenog broja slika u sekundi, drugim riječima ako je u sobi postavljeno reprodukcija 60 slika u sekundi, step događaj će izvršiti zadani programski kod svakih 60 reproduciranih slika. Step događaj se koristi kada imamo određenu računalnu logiku koju je potrebno konstantno ispitivati i provjeravati, na primjer prilikom pritiska određene tipke treba se izvršiti određena radnja, interakcije između objekata i uvjeti koji se trebaju

izvršit ako dođu u konflikt jedan s drugim, praćenje statistike, odnosno ako neki objekt ima određeni broj života i ako izgubi sve živote potrebno ga je uništiti.

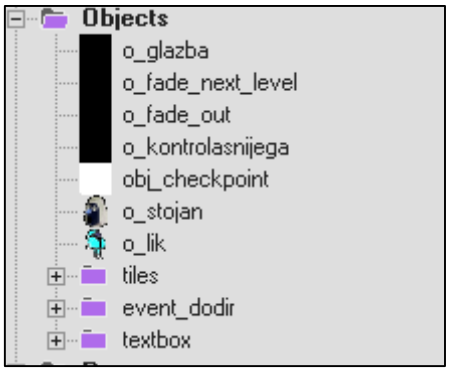
Alarm je događaj koji ne radi ništa, ukoliko nije ranije postavljen. Njegova je funkcija ta da sadrži određeni programski kod, a poziva se tako da ga se pozove kao polje s imenom alarm i u uglatim zagradama postavi vrijednost. Ta vrijednost prikazuje vrijeme odbrojavanja. Kada to odbrojavanje dođe do kraja izvršava se programski kod unutar alarm događaja, te kada dođe do kraja vraća se na liniju programskog koda gdje je bio pozvan od strane nekod drugog događaja te se izvršava dalje programski kod.

Objekti kao i sprite-ovi, imaju mogućnost postavljanja u sobu. Razlika između objekta i sprite je ta da sprite-ovi su samo grafički materijal koji je prikazan u sobi u svrhu grafičkog identiteta, dok objekti kada se postave u sobu imaju određena svojstva i ponašanja koja će se izvoditi tijekom izvođenja programa unutar same sobe. Objektima može ali i ne mora biti pridodan grafički materijal prema kojima će biti vidljivi u sobi. Prema *sl. 2.28* objektima kojima je pridodan grafički materijal biti će vidljivi u sobi pod tim grafičkim materijalom, dok objekti koji nemaju pridodan grafički materijal na sebi će imati plavi kružić s upitnikom. Prilikom pokretanja igre ili grafičke aplikacije svi objekti koji imaju definirani grafički materijal biti će vidljivi korisniku, dok ostali koji nemaju će biti nevidljivi ali će biti svejedno prisutni. Da bismo dodali objekt u sobu potrebno je samo otvoriti listu objekata s lijeve strane unutar izbornika sobe, označiti koji objekt želimo i jednostavnim klikom miša pozicionirati gdje želimo da se objekt nalazi prilikom pokretanja.



Sl. 2.28 *Prikaz postavljanja objekta u sobu*

Na primjeru organiziranja sprite-ova, preporučljivo je da se organiziraju objekti *(sl.2.29)* u grupe radni lakšeg upravljanja i korištenja novonastalih objekata.



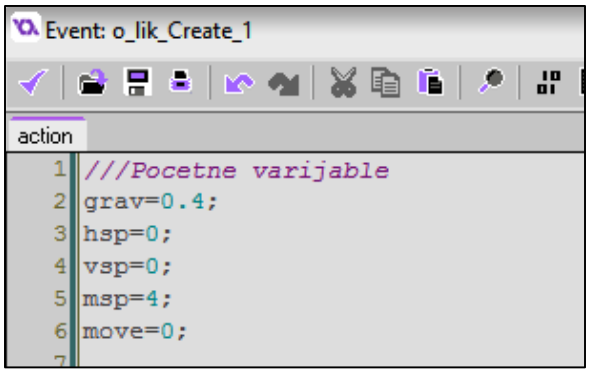
Sl. 2.29 *Organiziranje i grupiranje objekata u stablu resursa*

2.2.4. Konstruiranje smislene logike i implementacija algoritama

Nakon kreiranja i implementacije grafičkih materijala koji predstavlja početnu fazu prilikom kreiranja igre ili neke druge grafičke aplikacije, potrebno je usmjeriti se na drugi dio faze izrade koji se odnosi na pisanje programskog koda i implementiranju algoritama koji će nam pomoći da ostvarimo odgovarajuću smislenu računalnu logiku kako bi igra ili druga grafička aplikacija imala svrhu koja će privući i voditi korisnika pri ostvarivanju postavljenog cilja. Kako bi napisali odgovarajući programski kod i konstruirali smislenu logiku potrebno je odrediti svojstva i parametre svakog pojedinog objekta, njegovo ponašanje i interakciju s okolinom koja će biti popraćena s odgovarajućim programskim kodom, kao i određivanje njegove svrhe pri ostvarivanju zadanog cilja. U primjeru ovog završnog rada kreiramo igru po vrsti zvanoj „Platformer“. Glavna ideja ovog tipa igre predstavlja postojanje lika koji ima danu slobodu kretanja u prostoru kojim upravlja krajnji korisnik s ciljem da ostvari zadani zadatak pri kojem nailazi na razne prepreke. Kako bismo ostvarili pojedine elemente ovog žanra, navesti ćemo i implementirati algoritme za osnovnu strukturu kretanja (hodanje,skakanje,padanje,itd), kao i interakciju glavnog lika s okolinom, koji će imati odgovarajuću smislenu računalnu logiku pri rješavanju prepreka na putu.

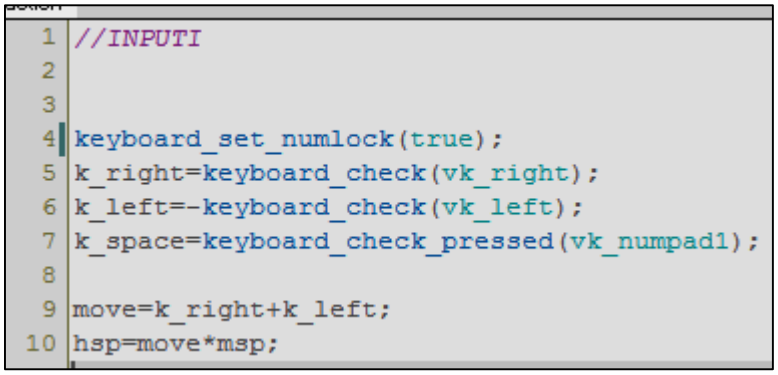
Prvi korak ovoga procesa predstavlja deklariranje varijabli za horizontalnu, vertikalnu i maksimalnu varijablu *(sl 2.30)*, kao i deklariranje varijable za gravitaciju koji će predstavljati osnovne parametre kretanja s potrebnom fizikom. Varijable ćemo deklarirati tako da otvorimo kreiramo „create“ događaj unutar o_lik i unutar „create“ događaja otvorimo programski kod i deklariramo potrebne varijable. Prilikom deklaracija varijabli potrebno je samo prije imena varijabli pridodati oznaku „var“ a razvojno okruženje GameMaker:Studio prilikom dodavanja

vrijednosti odrediti će dali je varijabla cjelobrojnog ili drugog tipa. Popis objekata i sprite-ova nalazi se u prilogu radi lakšeg snalaženja.



Sl. 2.30 *Deklariranje osnovnih varijabli objekta o_lik*

Nakon što su kreirani početne varijable, nakon „create“ događaja potrebno je deklarirati „step“ događaj unutar kojega će se implementirati svi potrebno algoritmi kako bi dobili odgovarajuću računalnu logiku. Kako se radi o „Platformer“ tipu igre, potrebno je odrediti pomoću kojih tipki na tipkovnici će se izvršavati određeni smjerovi kretanja. To ostvarujemo pomoću ugrađenih funkcija: **keyboard_set_numlock(), keyboard_check()** i **keyboard_check_pressed(),** a za vrijednost unutar zagrada postavljamo naziv željene tipke *(sl 2.31)*.



Sl. 2.31 *Definiranje tipki za kontrolu kretanja*

Funkcija **keyboard_set_numlock()** omogućuje nam automatsko paljenje numlock-a na tipkovnici ukoliko od strane korisnika nije upaljeno. Keyboard_check() i keyboard_check_pressed() predstavljaju funkcije koje ispituju dali je određena tipka aktivna. Ukoliko je određena tipka aktivna kao vrijednost funkcije vraća se jedinica, ukoliko nije, funkcija vraća nulu. Razlika između ove dvije funkcija je ta keyboard_check() neprestano provjerava je li tipka aktivna i vraćat će vrijednost sve dok je tipka stisnuta, dok keyboard_check_pressed vraća jedinicu nakon što je tipka stisnuta i puštena. Potrebno je deklarirati posebne varijable za lijevu i desnu stranu, kao i za skok. Kako bi razlikovali lijevu od desne strane potrebno je prilikom vraćanje vrijednosti funkcije za

lijevu stranu dodati minus predznak, što znači da ako korisnik pritisne lijevu tipku varijabla će zabilježit vrijednost sa negativnim predznakom. Tada će **move** varijabla koja služi za horizontalno kretanje *(sl 2.31)* imati negativni predznak, u suprotnom slučaju imat će pozitivan. Varijabla **hsp** koja služi za horizontalnu brzinu prilikom pritisnute desne tipke vratit će jedinicu i kao umnožak s maksimalnom brzinom dati će određenu vrijednost, isto vrijedi za lijevu tipku s drugim predznakom. U slučaju niti jedne pritisnute tipke vrijednost će biti nula. Nakon što su osigurane sve potrebne varijable potrebno je implementirati algoritam za osnovu pokreta prema *sl. 2.32.*

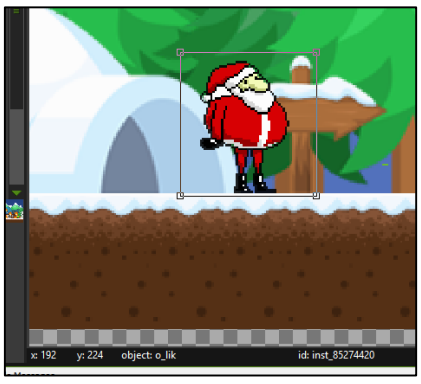
```
if(vsp<10)
        {
          vsp=vsp+grav;
        }
if(place_meeting(x,y+1,o_podloga1))
        {
          vsp=k_space*-7;
        }

 if(place_meeting(x+hsp,y,o_podloga1))
         {
          while(!place_meeting(x+sign(hsp),y,o_podloga1))
                    {
                      x=x+sign(hsp);
                    }
         hsp=0;
         }
if(place_meeting(x,y+vsp,o_podloga1))
         {
           while(!place_meeting(x,y+sign(vsp),o_podloga1))
                   {
                     y=y+sign(vsp);
                   }
         vsp=0;
         }

      if(global.pause != true)
      {
        x=x+hsp;
        y=y+vsp;
      }
```

Sl. 2.32 *Implementacija algoritma za osnovu pokreta*

Algoritam (sl. 2.32.) prvo provjerava je li zadana vertikalna brzina manja od 10. Prilikom skoka lika vertikalna brzina je u negativnoj brojčanoj vrijednost, dok prilikom pada ona dobiva pozitivnu brojčanu vrijednost. Kako ne bi lebdjeli po zraku, potrebno je pri određenoj vertikalnoj brzini varijabli vsp dodati zbroj varijable grav koja je pozitivna kako bi usmjerili vertikalno kretanje prema dolje. Kako bi omogućili o_lik mogućnost skoka potrebno je kreirati sljedeći uvjet. Pomoću funkcije place_meeting kojega ispitujemo ponašanje u prostoru s drugim objektima. Funkcija

place_meeting uzima 3 parametra: koordinatu mjesta x s pripadajućom vrijednosti horizontalne brzine, koordinatu mjesta y i objekt s kojim se dodiruje. U ovom slučaju koristimo place_meeting kako bi stvorili dojam da lik stoji na zemlji. Ako unutar if-a su zadovoljeni svi parametri odnosno lik stoji na podlozi varijabli vsp se pridodaje vrijednost k_space. Ukoliko je varijabla k_space pritisnuta ona će imati vrijednost 1 i pomnoženo s -7 daje negativnu vrijednost. U tom trenutku o_lik ima negativnu vertikalnu brzinu što znači da će se kretati prema gore, a zatim se aktivira prvi uvjet koji ga pomoću varijable grav vraća na pozitivnu vrijednost, odnosno spušta dolje.

Ukoliko nije pritisnuta tipka za skakanje varijabla k_space iznosi nula i time je varijabla vsp jednaka nuli što znači da nema pomaka. U sljedećem uvjetu ispituje se dali o_lik dodiruje neki objekt odnosno podlogu po vertikalnoj ili horizontalnoj osi. Ukoliko su svi parametri zadovoljeni pokreće se s programski blok i ispituje se uvjet u while-petlji. While se izvršava sve dok o_lik ne dodiruje podlogu po horizontalnoj koordinati. Ukoliko je pritisnuta lijeva ili desna tipka postoji vrijednosti na varijabli hsp, a pomoću funkcije sign zaokružuje se vrijednost 1,-1 ili 0 ovisno o kojem broju i predznaku se radi (ukoliko je vrijednost -542 funkcija sign će zaokružiti na vrijednost -1). Koordinata x je uvećana za vrijednost varijable hsp i time mijenja položaj lika u sobi. Ukoliko nije pritisnuta ni lijeva ni desna tipka i ako lik dodiruje podlogu na određenim horizontalnim koordinatama, while petlja se preskače i varijabla hsp jednaka nuli. Isto ispitivanje uvjeta u algoritmu vrijedi i za ispitivanje vertikalne osi samo što u ovom slučaju u place_meeting funkciji pridodaje se vrijednost varijable vsp y koordinati. Na kraju imamo završni uvjet u kojemu se provjerava je li igra u trenutnoj pauzi. Ako nije pridodaju se vrijednosti vsp i hsp varijablama x i y koje predstavljaju ugrađene varijable unutar GameMaker-a za položaj objekta unutar sobe koje omogućavaju kretanje kroz prostor. Da bismo završili ovaj proces s osnovama kretanja potrebno je zatvoriti sve trenutne prozore programskog koda unutar događaja te unutar uređivača sobe potrebno je odabrati objekt na kojemu smo kreirali algoritam, te ga postaviti unutar sobe (sl 2.33) . Da bi smo izvršili testiranje potrebno je pokrenite igru u debuq modu pritiskom tipke F5.



Sl. 2.33 *Postavljanje objekta o_lik u trenutnu sobu*

Da bi stekli dojam kako lik zapravo hoda i skače, potrebno je ujediniti grafičke i zvučne materijale jedinstvenim programskim kodom. Kao i pri implementiranju algoritma za osnovnu pokreta, zvučne i grafičke resurse objedinit ćemo u „step“ događaju o_lik-a prema *sl 2.34*. ispituje se prvo

```
if(move!=0)
        {
         image_xscale=move;
        }
if(place_meeting(x,y+1,o_podloga1))
{
            if(move!=0)
            {
              image_speed=0.25;
              sprite_index=lik_trci;
              if (!audio_is_playing(sound2))
                audio_play_sound(sound2,10,false);
            }
            else
            {
               image_speed=0.08;
               sprite_index=lik;
            }
}
if (vsp<0)
 {
 sprite_index=lik_skace;
 if (!audio_is_playing(sound1))
     audio_play_sound(sound1,10,false);
 }
 else
 {
 sprite_index=lik_pada;
 }
```

Sl. 2.34 *Implementacija animacije i zvučnih resursa*

uvjet varijable move. Ako je varijabla move različita od nule znači da se trenutni objekt kreće u prostoru. Ukoliko je move pozitivan, sprite koji se nalazi pod objektom imat će orijentaciju slike jednaku prema desnoj strani, u suprotnom biti će okrenut prema lijevo. Sljedeći uvjet provjerava dali se objekt o_lik nalazi na podlozi. Ukoliko se objekt nalazi na podlozi i njegova brzina nije nula, ugrađenoj funkcija image_speed daje brzinu reprodukcije slika unutar sprite-a te sprinte_index koji određuje trenutni ili novi sprite koji će objekt koristiti. U ovom slučaju izmijenit će se 25 slike unutar sprite u jednoj sekundi te će objekt imati sprite lik_trci (sl. 2.35). U suprotnom brzina reprodukcije slika će biti 8 slika unutar sprite-a u jednoj sekundi i objekt će imati postavljen sprite lik. Posljednji uvijet provjerava dali je varijabla vsp manja od nule. Ukoliko je vrijednost manja od nule sprite objekta o_lik postavit će se na lik_skace i istovremeno testirat će se još jedan uvjet s ugrađenom funkcijom audio_is_playing koja provjerava dali postoji trenutno aktivan zvuk. Ako ne postoji, uvjet će pomoću funkcije audio_play_sound izvršiti zadani zvuk skoka. U suprotnom, postavit će se sprite lik_pada ako je vertikalna brzina pozitivna.

Sl. 2.35 *Animacije pri određenim ispunjenim uvjetima*

Nakon što je postavljena osnovna konstrukcije igre, potrebno je implementirati zadatak, odnosno korisnik mora skupiti određeni broj novčića kako bi mogao završiti nivo. Unutar o_coin koji se nalazi na popisu objekata u prilogu potrebno je kreirati step događaj i implementirati programski kod prema *sl. 2.36.* Ukoliko o_coin dođe u kontakt s o_lik-om, globalnoj varijabli bodovi uvećat će se vrijednost za 1. Nakon uvećana vrijednost pomoću funkcije instance_destroy() uništit će se trenutni objekt kako ne bi došlo do stalnog uvećavanja vrijednosti na istom objektu. Nakon toga

```
1 image_speed=0.25;
2 if(place_meeting(x,y,o_lik))
3 {
4     global.bodovi=global.bodovi+1;
5     instance_destroy();
6 }
```

Sl. 2.36 *Programski kod unutar o_coin*

potrebno je unutar step događaja objekta **o_coinb** definirati koordinate x i y objekta koje se moraju stalno ažurirati da bi se stekao dojam kako rezulat u gornjem desnom kutu stoji ravno čak i onda kada se objekt o_lik kreće u bilo kojem smijeru (sl. 2.37). Unutar draw događaja istog objekta potrebno je definirati dodatne linije koda (sl 2.37), te pomoću funkcija draw_self(), draw_text_colour() i draw_set_font() kako bi ostvarili dinamički prikaz brojanja novčića unutar sobe.

```
1  x = view_xview+560;
2  y = view_yview+8;
3  image_speed=0.25;
```

step događaj

action

```
1  draw_self();
2  draw_set_font(font2);
3  draw_text_colour(x+13,y-3,"  " +
4  string(global.bodovi) + "/10",c_white,
5  c_white,c_white,c_white,100);
6
```

draw događaj

Sl. 2.37 *Programski kod step i draw događaja o_coinb*

Nakon što o_lik dođe u konflikt s o_coin i globalna varijabla bodovi se uveća za jedan, istovremeno unutar step događaja o_coinb koji se odvija svake sekunde promijeniti će se ispis na ekranu koji pokazuje rezultat *(sl. 2.38)*. O_coinb potrebno je priložiti na sva mjesta gdje se želi prikupiti novčić, dok o_coinb moguće je postaviti na bilo koje mjesto, jer programskim kodom unutar step događaja uvijek će biti postavljen na zadane x i y koordinate. Kako bi onemogućili korisniku da dođe do kraja nivo-a postaviti ćemo objekt „nevidljivizid“ na kraju nivo-a.

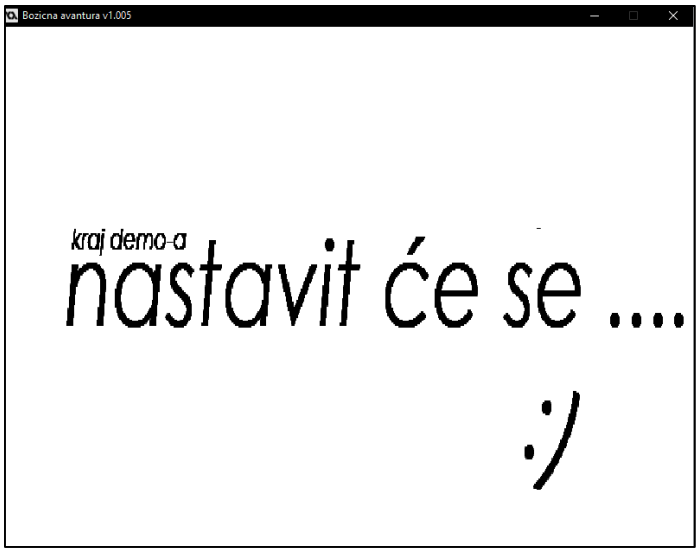
Sl. 2.38 *Prikaz rezultata skupljenih novčića*

Potrebno je ponovo unutar step događaja o_lika definirati uvjet koji će stalno provjeravati dali je korisnik ispunio i skupio određeni broj novčića. Ukoliko je skupio, pomoću funkcije with() *(sl. 2.39)* koja omogućava za svaki istoimeni objekt unutar sobe se primjeni programski blok unutar funkcije. Stoga ako korisnik skupio traženi broj novčića, svi objekti „nevidljivizid“ će biti uništeni te korisnik može slobodni doći do kraja nivoa.

```
85 if(global.bodovi==10)
86 {
87     with(nevidljivizid)
88     {
89         instance_destroy();
90     }
91 }
92
93 if(place_meeting(x,y,o_kraj)
94 {
95     room_goto_next();
96 }
```

Sl. 2.39 *Prikaz rezultata skupljenih novčića*

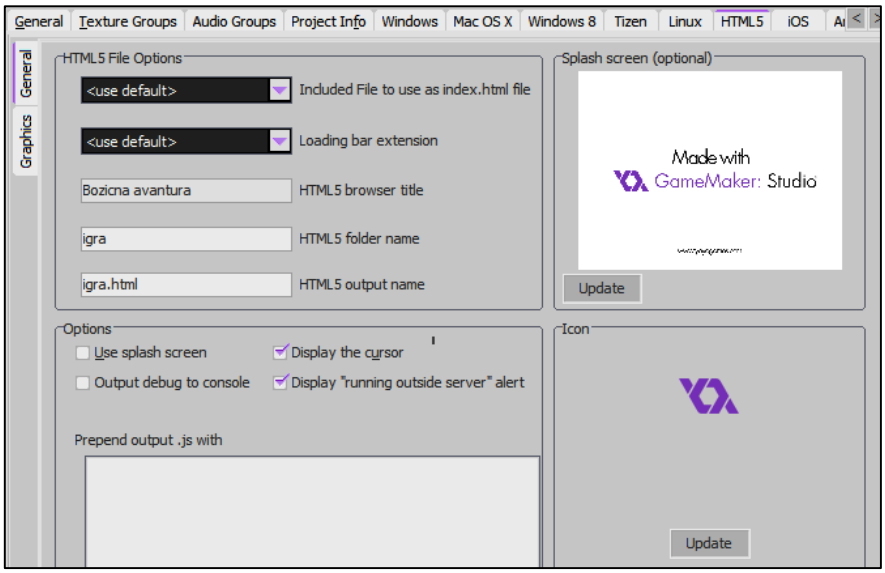
Poželjno je staviti objekt o_kraj na iza nevidljivog zida, te nakon što objekt „nevidljivizid“ bude uništen, te nakon što o_lik dođe u kontakt s objektom o_kraj *(sl. 2.39)* aktivira se ugrađena GameMaker-ova ugrađena funkcija za automatski prebacivanje zaslona igre u novu sobu *(sl 2.40)*.



Sl. 2.40 *Prijelaz u sljedeću sobu*

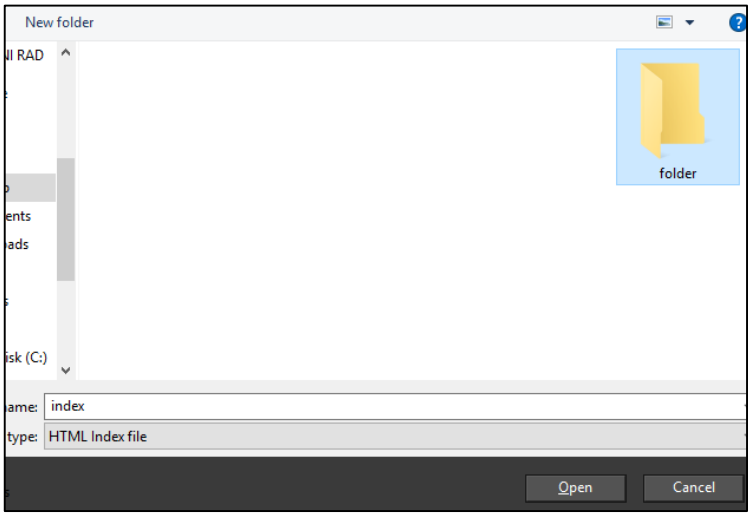
2.3. Izvoz igre na željenu razvojnu platformu

Zadnji korak u procesu izrade 2D računalne igre u GameMaker:Studio predstavlja izvoz na željenu razvojnu platformu. Da bismo to ostvarili potrebno je unutar stabla resursa kliknuti na opciju „Global Game Settings“. Nakon toga pojavljuje se izbornik u kojem su ponuđena razna razvojna okruženja. Jedna od najpopularnijih platforma za izvoz igre je HTML 5 s kombinacijom JS-a koji se nalazi na jednoj od kartica izbornika. Prema *sl. 2.41* moguće je definirati opće i grafičke postavke koje želimo da se spreme prilikom izvoza igre na odgovarajuću platformu.



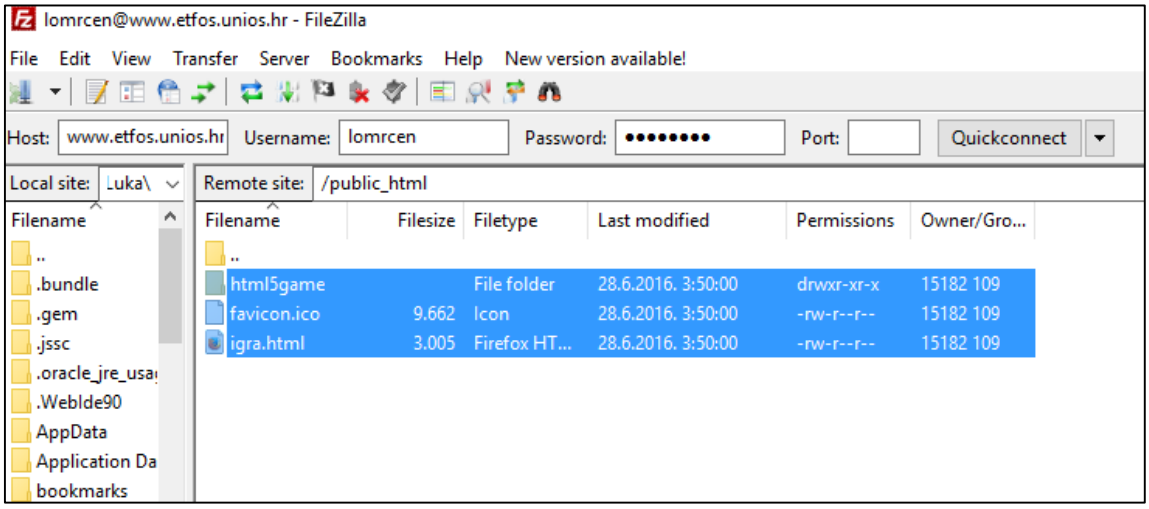
Sl. 2.41 *Opće i grafičke postavke unutar izvoza kartice HTML5*

Da bi izvezli igru potrebno je otvoriti alatnu traku i izabrati opciju **Create application** te prema sl. 2.42 odabrati mjesto gdje želimo pohraniti igru u .html formatu.

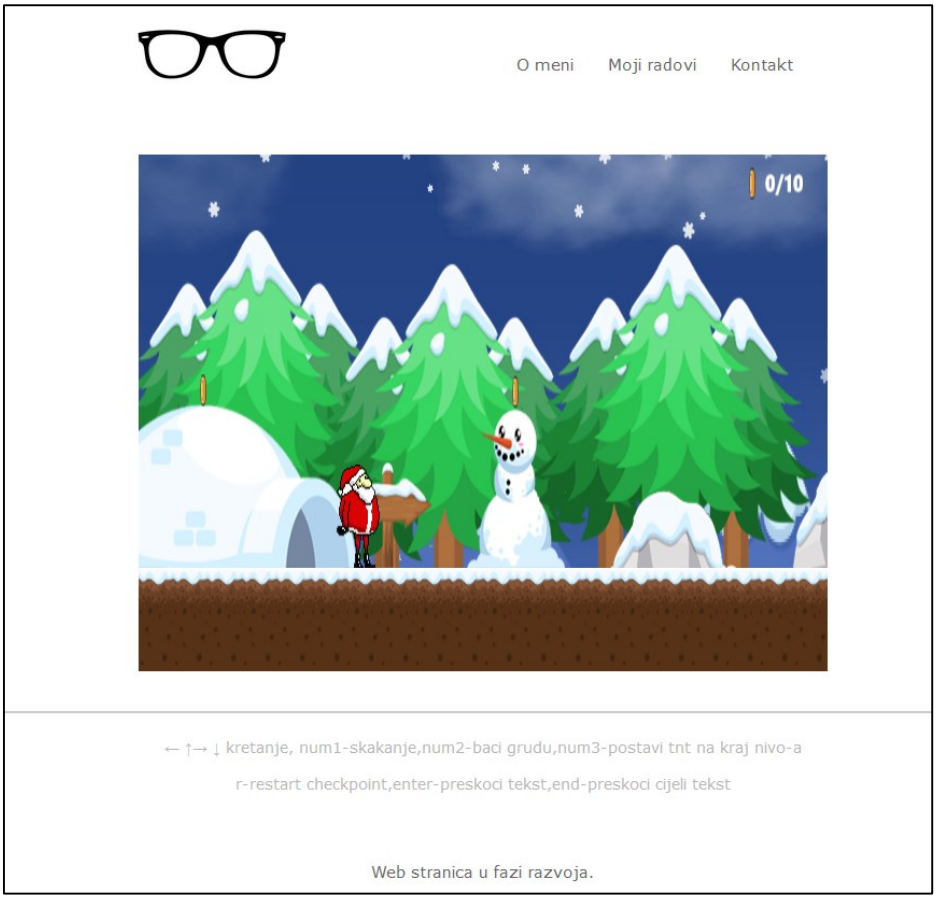


Sl. 2.42 *Odabiranje mjesta za spremanje igre*

Kada je pohrana završena, potrebno je pokrenuti jedan od FTP programa, po mogućnosti **FileZilla**, prijaviti se na korisničkim podatcima na web server i prema *sl. 2.43* pohraniti spremljene datoteke na web server. Nakon što su podatci od igre pohranjeni na web server, u ovom završnom radu, igru je moguće otvoriti na sljedećem linku : http://www.etfos.unios.hr/~lomrcen/igra.html *(sl. 2.44)*.



Sl. 2.43 *Pohrana igre na web server*



Sl. 2.44 *Prikaz gotove igre unutar web stranice*

3. ZAKLJUČAK

GameMaker:Studio predstavlja vrlo fleksibilan i jednostavan alat za korištenje, u svakom smislu idealan po svojim mogućnostima u kreiranju 2D igra i grafičkih aplikacija ili određenih prototipa projekata. Prednost ovog razvojnog okruženja je da omogućuje samim početnicima koji možda nemaju potrebno programersko predznanju da realiziraju svoje željene ideje i projekte zahvaljujući vrlo jednostavnim metodama, dok iskusnim programerima pruža niz alata i metoda s kojima obavljaju posao brže, bolje i efikasnije. U ovome završnim radu objašnjene su temeljne karakteristike, prednosti i mane navedenog razvojnog okruženja, kao i uvod u programski jezik GML. Kako bi realizirali određeni projekt, odnosno u ovom završnom radu 2D igru, potrebno je raspolagati s osnovnim grafičkim i programerskim vještinama kao i posjedovanjem određenih programskih alata osim GameMaker:Studio-a. Nakon određivanja ideje projekta, pomoću programa Adobe Phoshop kreirani su grafičkim materijali, koja su zatim implementirani unutar razvojnog okruženja GameMaker:Studio u svrhu kreiranja grafičkog sučelja i grafičkog identiteta objekata. Definirani su sprite-ovi, objekti, sobe i događaji u kojima su implementirani određeni algoritmi i ostali programski kod bez kojih nije moguće ostvariti smislenu računalnu logiku i zadani cilj igre. Na kraju, u zadnjem poglavlju objašnjen je postupak prilikom izvoza projekta na željenu razvojnu platformu ,u ovom slučaju HTML5 te pomoću programa Filezilla postavljanje na željeni web server. GameMaker:Studio predstavlja kompletan alat za nezavisne korisnike koji realiziraju sve grafičke, zvučne i programerske dijelove projekta i samim time povećava konkurentnost malih timova u odnosu na velika razvojna studija na tržištu.

LITERATURA

[1] https://forum.yoyogames.com/index.php (25.svibanj.2016)

[2] M.Overmars, J.Habgod, *The Game Maker's Apprentice: Game Development for Beginners*, 2007.

[3] E.Adams, *Fundamentals of Game Design,* (3rd Edition), 2014.

SAŽETAK

U ovome završnom radu cilj je bio opisati i analizirati GameMaker:Studio kao platformu za razvoj 2D igara i grafičkih aplikacija te napraviti primjer računalne 2D igre polazeći od osnovne ideje i rješavanje problema prilikom procesa razvoja sve do izdavanja gotovog proizvoda opisujući pritom sve tehnologije i alate koji su korišteni. U prvoj fazi ovog završnog rada koristi se Adobe Photoshop u svrhu kreiranja grafičkih materijala. Pomoću GameMaker:Studio-a implementiraju se napravljeni grafički materijali i kreiraju se sprite-ovi, objekti, sobe, događaji i ostali elementi bitni za razvoj 2D igre. S programskim jezikom „GameMaker Language“ (skraćeno GML) implementiramo potrebne algoritme i odgovarajući programski kod kako bi kreirali smislenu računalnu logiku i zadani cilj igre. Prilikom izvoza 2D igre biramo odgovarajuću razvojnu platformu (HTML5) i pomoću programa FileZilla postavljamo odgovarajuće materijale na željeni web server.

Ključne riječi: GameMaker:Studio, 2D igra, 2D grafička aplikacija, Adobe Photoshop, GameMaker Language, GML, sprite, objekti, sobe, događaji, HTML5, FileZilla

ABSTRACT

Development of computer game using Gamemaker : Studio

The main goal of this thesis was to describe and analyze GameMaker: Studio as a platform for the development of 2D game and graphics applications, making an example of 2D computer game starting from the basic ideas and solving problems in the process of development to the issuance of the final product describing all the technology and tools that were used. In the first step of this thesis we use program Adobe Photoshop to create graphic materials. GameMaker:Studio is used to implement the designed graphic materials and to create the sprite's, objects, rooms, events and other elements essential to the development of 2D game. The programming language "GameMaker Language" (abbreviated GML) is used to implement the necessary algorithms and corresponding source code to create a meaningful computational logic and goal of the game. When exporting 2D games we choose an appropriate development platform (HTML5) and using FileZilla we can set appropriate materials to the desired web server.

Keywords: GameMaker:Studio, 2D game, 2D graphics applications, Adobe Photoshop, GameMaker Language, GML, sprite, objects , rooms, events, html5, FileZilla

ŽIVOTOPIS

Luka Omrčen rođen je 20.09.1994.g. u Osijeku. Osnovnu školu završio u Josipovcu, a srednju školu u Osijeku. Student je treće godine računarstva na Elektrotehničkom fakultetu u Osijeku. Govori engleski jezik. Posjeduje izvrsno znanje C-a, C++-a, HTML-a, CSS-a i proceduralnog PHP-a. Osim studiranja, tri godine aktivno radi preko studenskog servisa u S.Oliver trgovini, pritom stječući dobre komunikacijske vještine i rad u timu. U slobodno vrijeme zanima se za nove popularne programske jezike kao C# i pyton, aktivno se bavi sportom, razvojem 2D igara i web dizajnom.

Potpis:

_____

PRILOZI

Izvorni GameMaker:Studio .gmx kod

http://www.etfos.unios.hr/~lomrcen/dokumenti/MojPlatformer.gmx.rar