Izrada računalne igre u razvojnom okruženju Construct 2

Šag, Antonio

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:238234

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

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

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

ANTONIO ŠAG

Izrada računalne igre u programskom okruženju Construct 2

Završni rad

Pula, 2018.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

ANTONIO ŠAG

Izrada računalne igre u programskom okruženju Construct 2

Završni rad

JMBAG: 0303046575, redoviti student

Studijski smjer: Informatika

Predmet: Napredne tehnike programiranja

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujan 2018. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

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

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, Antonio Šag dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom Izrada računalne igre u programskom okruženju Construct 2 koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu sa Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

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

U Puli, _____

Potpis

_____

Sadržaj

1. Uvod 1

1.1. Kritička analiza igara 2

1.2. Zadatak završnog rada 4

2. Generalno o alatu Construct 2 5

2.1. Kreiranje novog projekta 6

2.2. Kreiranje početnog zaslona 7

2.3. Scene 8

2.4. Slojevi 8

3. Lista događaja 9

2.4. Događaji 11

4. Objekti 12

4.1. *Plug-inovi* 12

4.2. Tipovi objekata 13

4.3. Instance tipova objekata 13

4.4. Ponašanja 14

4.5. Efekti 15

4.6. Familije 15

5. Razvoj igre 17

5.1. Pozadina i izgled *levela* 17

5.2. Glavni lik 20

5.3. Neprijatelji 23

5.4. Uništavanje neprijatelja 25

5.5.Globalne varijable, *HUD* i izbornik *levela* 26

5.6.Razine 36

5.7.Prelazak razine 38
5.8.Glavni izbornik 39
6. Poslovni plan, ciljna skupina i strategija monetizacije igre 41
7. Zaključak 43
Literatura 45
Popis slika 47
Sažetak 50
Summary 51

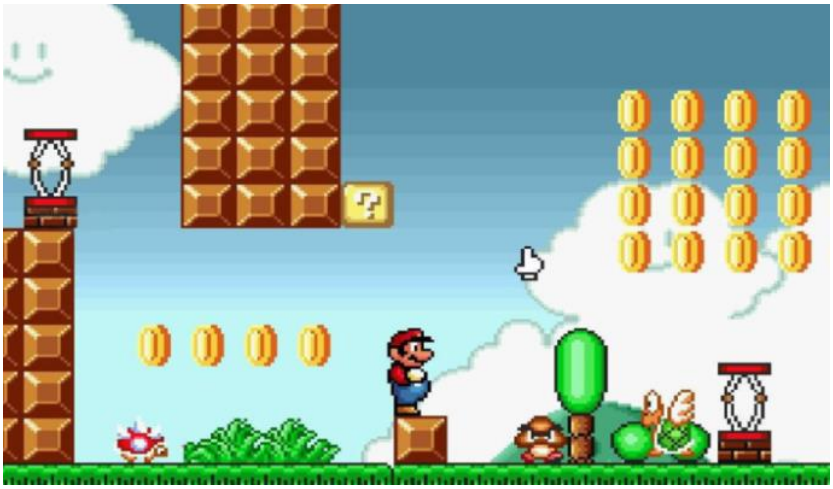
1. Uvod

U dvadeset prvom stoljeću računalne igre predstavljaju jedan od najpopularnijih oblika zabave. Igre su ušle u sve pore svakodnevnog života i ne poznaju dobne granice. Razvoj takvog oblika zabave vrlo je zahtjevan jer multimedijski oblik obuhvaća integraciju i adaptaciju videa, animacije, zvuka, teksta, pokretne i nepokretne slike. Razvoj multimedijske računalne igre vrlo je složen posao i zahtijeva tim stručnjaka koji poznaju teoriju i praktične metode koje se koriste u razvoju, koji se služe profesionalnim alatima za razvoj multimedijskih računalnih igri. U završnom radu „Izrada računalne igre u programskom okruženju Construct 2" motivacija za kreaciju igre proizašla je iz detaljnog istraživanja te analiziranja takozvanih „oldschool" 2D igara koje su obilježile mnoga djetinjstva. Kroz rad će se govoriti o izradi 2D platform igre namijenjenoj jednom igraču. Naziv igre je „Funky Student", a temelj je izradbe svođen na korištenje Construct 2 arhitekture. Specifičnost Construct 2D programa je u prilagodljivosti na različitim platformama, kao što su preglednici koji podržavaju HTML5 i operacijski sustavi: Mac, Linux i Windows te interaktivni pregled igre (engl. *compile*) u bilo kojem trenutku, što omogućava programeru realni pregled funkcionalnosti igre. Grafičko je korisničko sučelje zamišljeno po principu pustolovne (engl. *adventure*) platform igre „Super Mario". Svi vizualni elementi predstavljeni su u obliku dvodimenzionalnosti, i to u obliku vektora. Rad se sastoji od pet cjelina te uvoda i zaključka. U prvom dijelu rada govorit će se generalno o aplikacijskom sučelju Construct 2 i njegovom korisničkom sučelju. Sljedeći dio, točnije treća i četvrta cjelina fokusiraju se na detaljnu razradu funkcionalnosti liste događaja te objekata s kojima se programer susreće prilikom kreacije igre. Nakon toga slijedi razrada igre u kojoj se korak po korak razjašnjava pojedini tehnički proces same izrade. Dodatno je razrađen i pristup monetizaciji gdje se govori o ciljanim skupinama te strategijama i tehnikama korištenja monetizacije unutar igre. Tehnička je problematika igre riješena korištenjem svih funkcionalnosti aplikacije Construct 2 te prije same razradbe igre korištenjem potrebno je znati osnovne elemente i mogućnosti koje nam programsko sučelje nudi[1].

[1]Scirra Ltd, 2015, Construct 2 Manual [Online]. Dostupno na: https://www.scirra.com/manual/1/construct-2., [pristupljeno 17. rujna 2018.]

1.1. Kritička analiza igara

Zamisao je igre generalno percipirana po takozvanom platform načinu igranja gdje korisnik upravlja glavnim likom, a razine predstavljaju platforme koje sadržavaju različite objekte, a to mogu biti neprijatelji, ključevi za otključavanje „vrata“, kako bi mogli prijeći na sljedeću razinu i slično. Bazni pristup igre „Funky Student“ lako je povezati s dobro nam znanom igrom „Super Mario“. Također su, principi, sučelje te pristup ostalih 2D igara pridonijele samom planu i razradbi procesa igre, a na tom popisu bi se svakako našle igre: Icy Tower, Flappy Bird i Angry Birds, koje su svojom jednostavnošću te prilagodljivim korisničkim sučeljem predstavljali „zaraznu eru“ 2D igara.



Slika 1. - Super Mario

Slika 2. - Icy Tower

Slika 3. - Flappy Bird

Slika 4. – Angry Birds

1.2. Zadatak završnog rada

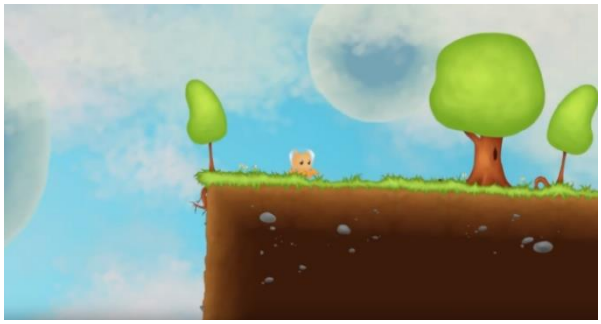
Detaljniji opis zadatka završnog rada, točnije igre, predstavljanje je igrača (studenta) koji je glavni lik u cijeloj priči. On u desnoj ruci nosi indeks (u svrhu prikupljanja potpisa) koji se nalaze na različitim razinama (engl. *level*). Kada student skupi tri potpisa, zadovoljava tražene uvjete *levela* te na taj način otvara vrata sveučilišta, koja mu omogućavaju put na sljedeću razinu. Na tom putu, sprječavaju ga različiti neprijatelji (na zemlji te u zraku), a vodilju kroz razine predstavljat će *gemovi* (engl. *Gem-s or Diamonds*). Pozadinu prvog i drugog *levela* čine motivi koji asociraju na grad Pulu (pulski divovi te gradilište), a treća razina generirana je prema izmišljenoj inicijativi. Elementi vezani uz grafiku, objekti i pozadina kreirani su unutar programa Adobe Illustratora te su za primjere kreiranja materijala platformi korišteni izvori Art of Game Designea[2]. U konačnici, kako bi sama igra postigla emocionalnu poveznicu igrača i korisnika, svaka interakcija i kolizija objekata obogaćena je zvukom, gdje su pojedini dijelovi dorađeni unutar Adobe Auditiona (*softwarea* koji pruža visoke performanse uz intuitivne alate za audio editiranje, miksanje i razne efekte[3]).

Zvučni specijalni efekti i pozadinska glazba postižu se dodavanjem preuzete glazbe sa stranica s besplatnim zvučnim specijalnim efektima: *FreeSFX[4] i FreeSound[5]*.

[2] Schell, J. (2008) The Art of Game Design, Amsterdam; Boston : Elsevier/Morgan Kaufmann Publishers.,[pristupljeno 17. rujna 2018.]
[3] Adobe Audition, Opis proizvoda [Online]. Dostupno na: https://www.hsm360.com/proizvod/adobe-audition-cc-3/., [pristupljeno 17.rujna 2018.]
[4] FreeSFX,Stranica sa za specijalne zvučne efekte [Online]. Dostupno na: http://www.freesfx.co.uk/., [pristupljeno 17.rujna 2018.]
[5]FreeSound, Background 8-bit Audio [Online]. Dostupno na: https://freesound.org/people/ProjectsU012/packs/18837/., [pristupljeno 17.rujna 2018.]

2. Generalno o alatu Construct 2

Construct 2 vrlo je jednostavna komercijalna aplikacija za kreiranje i dizajniranje dvodimenzionalnih igara, razvijena početkom 2011. godine. Ona ne zahtijeva klasično kodiranje, nego kodiranje putem odabira grafičkih ikona. Ikone se razvrstavaju tehnikom pritisni–povuci (engl. *drag and drop*). Kod je baziran na kombinaciji HTML5 (engl. *Hypertext Markup Language*, kod koji opisuje web stranice) u kojemu se implementira C++ i Javascript kod. Kompatibilan je za rad u više operacijskih sustava: Windows, Mac OS. Također, unutar aplikacijskog sučelja Construct 2, razvijeno je nekoliko komercijalnih računalnih igara, a neke od njih dostupne su i na Steamu[6], od kojih su primjeri: Airscape: The Fall of Gravity[7], The Next Penelope[8].



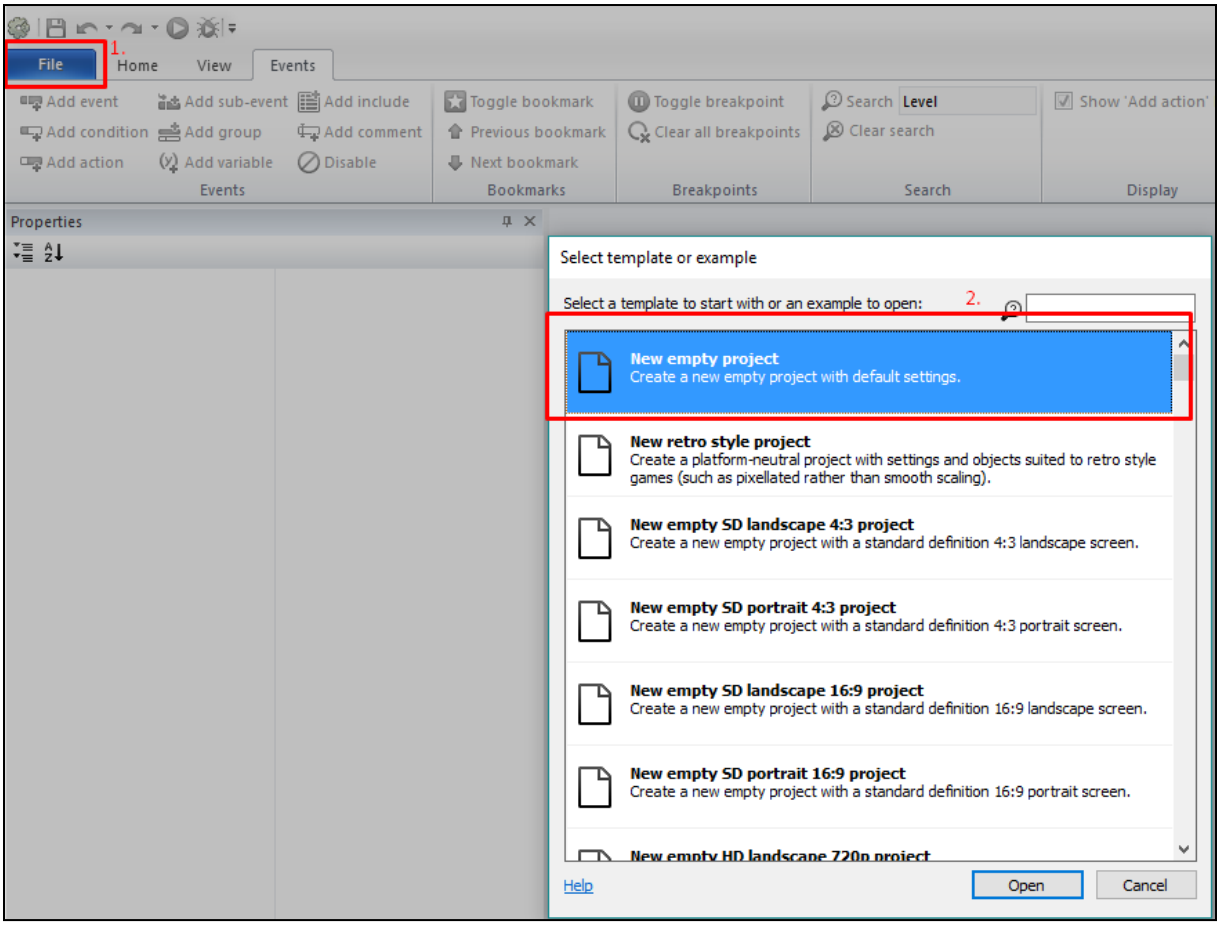
Slika 5. - Airscape: The Fall of Gravity

Slika 6. – The Next Penelope

[6] Steam, What is Steam? [Online]. Dostupno na: https://www.youtube.com/watch?v=QjWoSr_hZeY., [pristupljeno 18. rujna 2018.]
[7] Steam Game, Airscape: The Fall of Gravity [Online]. Dostupno na: https://store.steampowered.com/app/317250/Airscape__The_Fall_of_Gravity/., [pristupljeno 18. rujna 2018.]
[8] Steam Game, The Next Penelope [Online]. Dostupno na: https://store.steampowered.com/app/332250/The_Next_Penelope/., [pristupljeno 18. rujna 2018.]

2.1. Kreiranje novog projekta

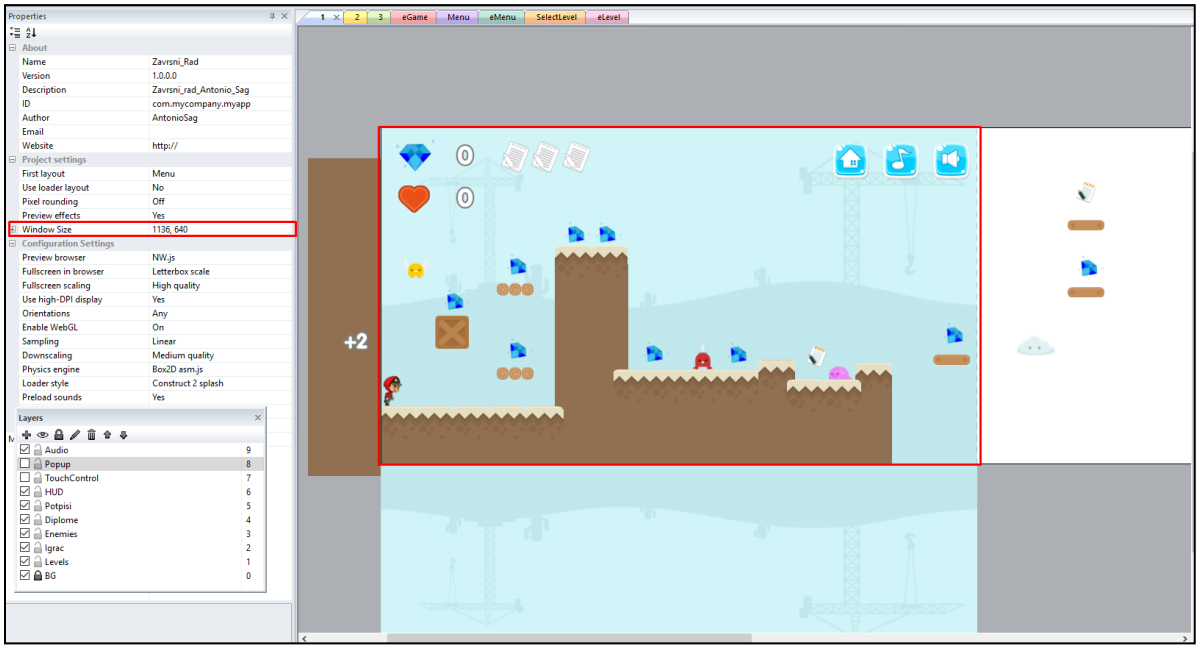
Početni korak izrade igre vođen je kreiranjem novog praznog projekta (engl. *New empty project*). Uz spomenutu opciju kreiranja praznog projekta, postoje i različiti predlošci koji unaprijed sadržavaju zadane postavke i funkcionalnosti, a primjeri su: *New retrostyle project, New empty HD landscape 1080p project, Template: Flappingbird, Template: Physical Catapult* i drugi.



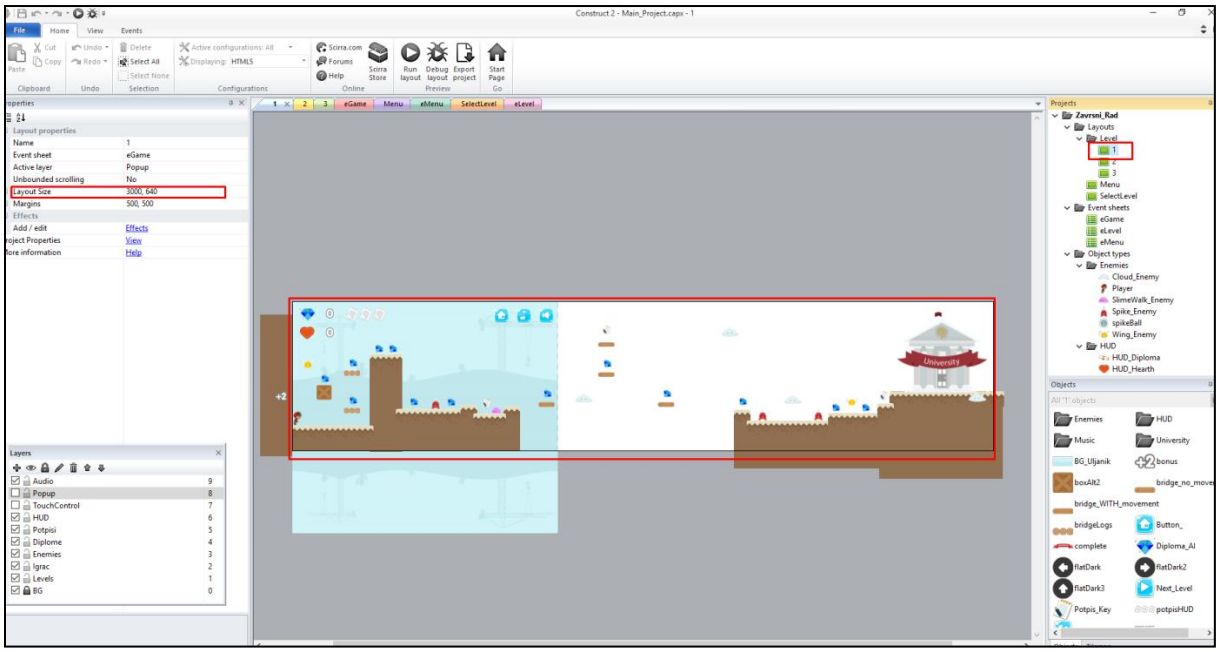
Slika 7. - Proces kreiranja novog projekta

2.2. Kreiranje početnog zaslona

Odabirom naredbe *New project* dolazimo do sljedećeg koraka, a to su postavke projekta koje se nalaze s lijeve strane *interfacea*. U početku je dovoljno postaviti veličinu prozora (engl. *Windows Size*) te veličinu scene (engl. *Layout Size*).



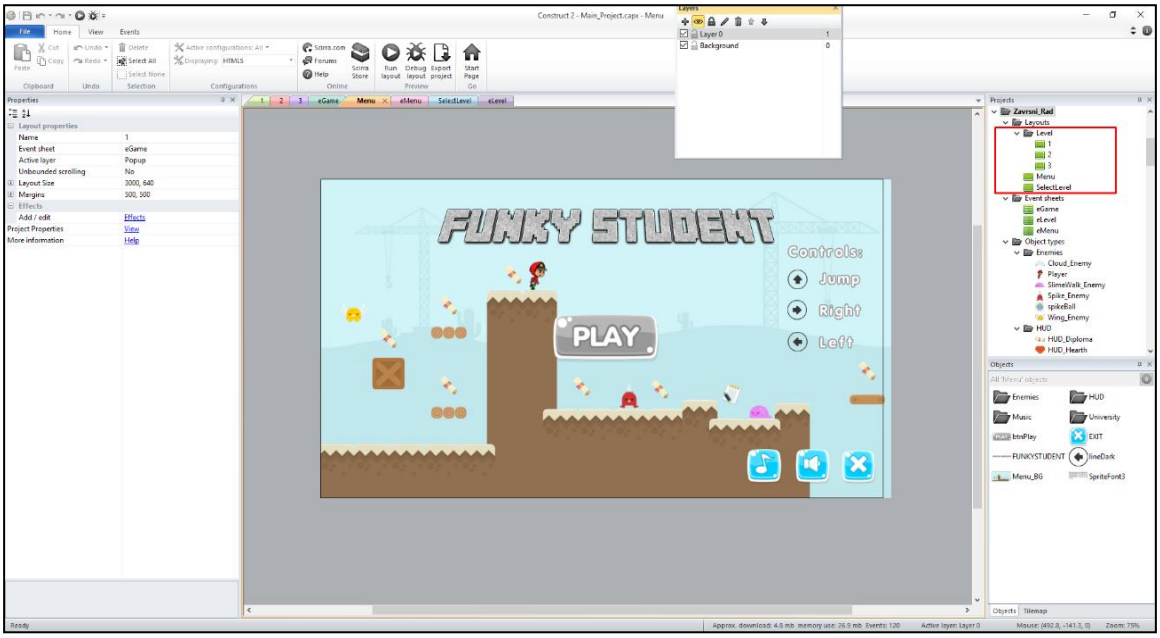
Slika 8. - Postavke zaslona igre



Slika 9. - Postavke veličine scene

2.3. Scene

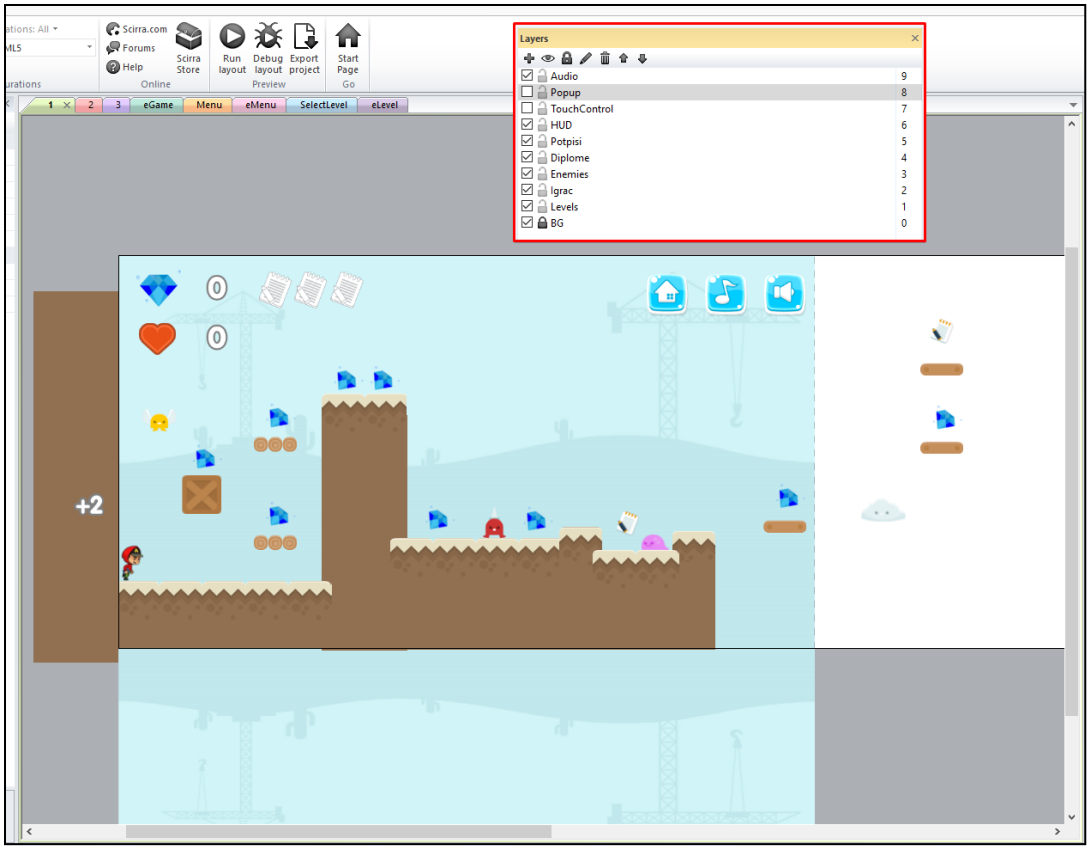
Scene (engl. *layouts*) predstavljaju razine ili nivoe u igrici. Važno je napomenuti da pojedina scena može sadržavati posebnu listu događaja koja će opisivati njena ponašanja te na taj način biti ovisna ili neovisna o ostalim scenama. Također sav sadržaj te objekti koje unosimo na aktivnu scenu, automatski će biti povezani sa zadanom scenom.



Slika 10. - Grubi prikaz scena

2.4. Slojevi

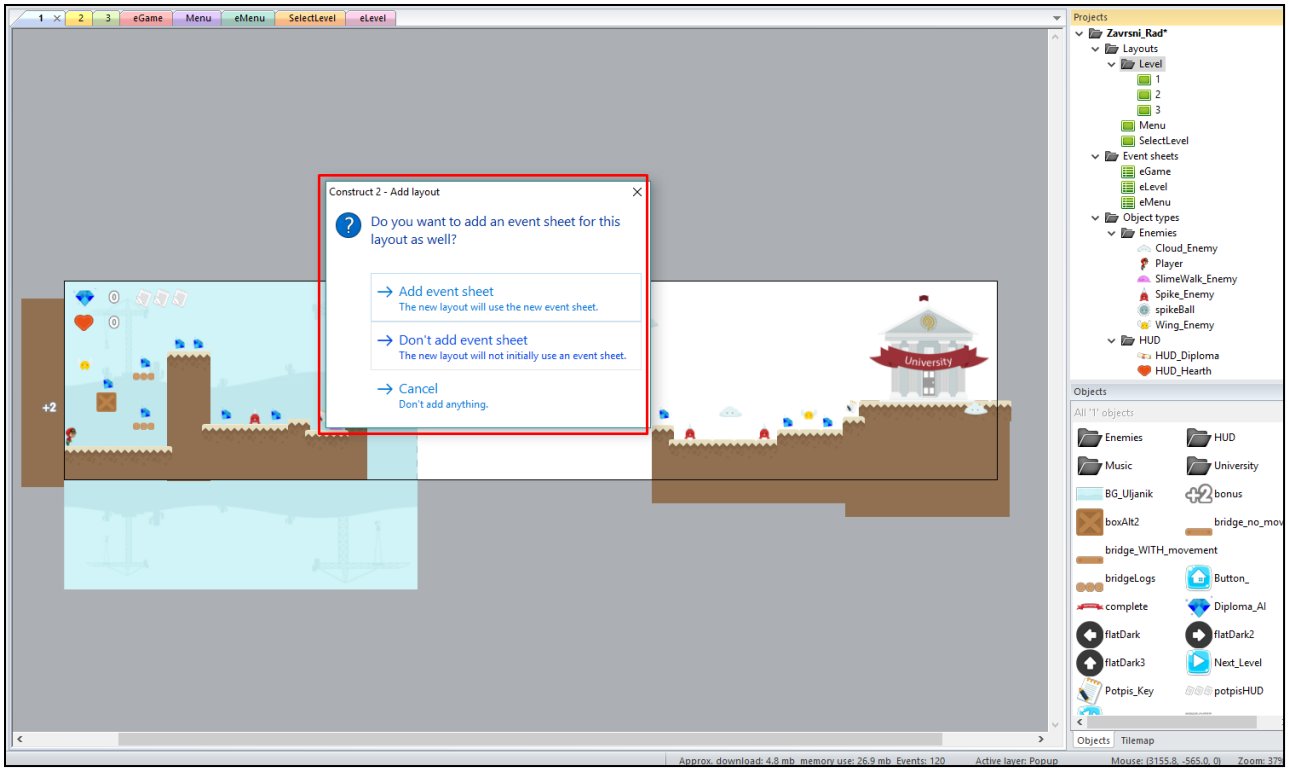
Scene se sastoje od takozvanih slojeva (engl. *layers*). Slojevi nam služe kako bi sve uvezene (engl. *import*) materijale mogli podijeliti po grupama. Zbog lakšeg rada, bolje organizacije te preglednosti samog projekta, važno je prilikom uvoza materijala i objekata vršiti proces sortiranja po slojevima, koje se provodi u specificiranom prozoru za slojeve. U tom prozoru mogu se dodavati, brisati, preimenovati i mijenjati redoslijedi prikazivanja pojedinih slojeva.



Slika11. - Grubi prikaz prozora sa slojevima

3. Lista događaja

Svaka od scena posjeduje pridruženu listu događaja (engl. *event sheet*) u kojoj razrađujemo sve korake vezane za dodavanje *eventa*, dodavanje akcija te ostalih događaja. Prilikom kreacije nove scene važno je naglasiti da ćemo dobiti opciju kreiranja nove liste događaja.



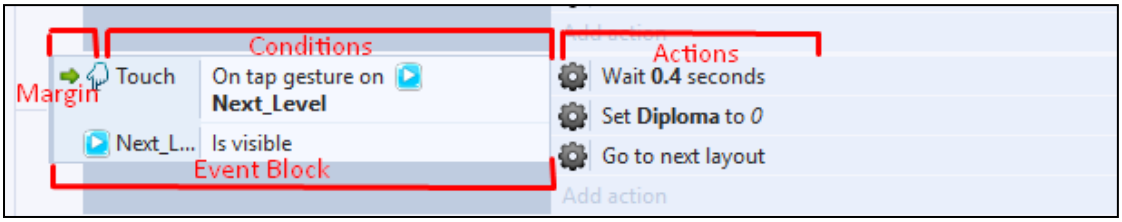
Slika 12. - Kreiranje nove liste događaja

Slika 13. - Grubi prikaz liste događaja

3.1. Događaji

Događaji (engl. *events*) su glavna značajka Construct 2. Umjesto kompliciranog skriptiranja programskih jezika s fiksnom sintaksom te svladavanja prepreka za koje je potrebna određena količina strasti i strpljenja te pomnog čitanja ogromne količine tekstova. Da bi shvatili koji je najbolji način ostvarenja zadanog problema Construct 2 nam nudi programiranje pomoću logičkih blokova sustava. Ukratko, blokovi se zajednički nazivaju događaji. Osnovni koncept događaja jest da uvjeti filtriraju slučajeve koji ispunjavaju stanje, a zatim se radnje izvode samo za te slučajeve. To omogućuje kontrolu primjeraka samostalno posebno kada se koristi s primjerom varijabli. Temeljni je način rada događaja filtriranje pojedinačnih slučajeva i pokretanje akcija samo onih koji su ispunjavali uvjete. Uz događaje su također važni sljedeći pojmovi, a to su:

- uvjeti – testiraju da su određeni kriteriji ispunjeni
- akcije – pokreću stvari, točnije utječu samo na slučajeve koji su filtrirani uvjetima događaja
- izrazi - mogu predstavljati iznose za izračunavanje vrijednosti ili preuzimanje vrijednosti iz objekata
- poddogađaji – uvjeti poddogađaja mogu dodatno filtrirati slučajeve i pokrenuti više akcija
- grupe – generalno se koriste za bolje organiziranje događaja
- komentari - omogućuju pisanje bilješki i opisa događaja te se nalaze iznad događaja kojega opisuju
- varijable događaja - mogu pohraniti brojeve i/ili tekst na dva načina: globalno (za sve *layoute*) ili lokalno (za određeni raspon događaja).



Slika 14. - Primjer jednog događaja

4. Objekti

Objekt (engl. *object*) je prikaz određene ideje. U Constructu 2 ta se definicija i dalje primjenjuje. Mentalno, kada zamislite određenu ideju to može biti nešto poput knjige ili kugle. Iako dobro znate kako knjiga izgleda, također znate da postoje mnoge jedinstvene knjige.

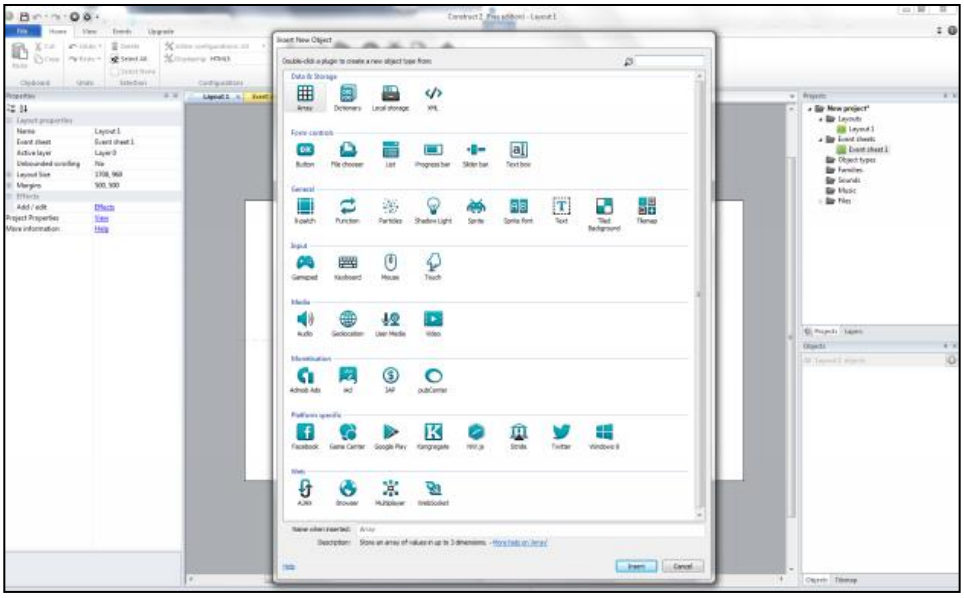
Iako je moguće imati mnogo instanci vrsti objekata, u Constructu 2 ne postoje dva objekta. Na primjer, moguće je imati 30 objekata knjiga u jednom rasporedu, no svaka će knjiga biti drugačija. To je jasno jer svaka instanca bilo kojeg objekta u Constructu 2 dodjeljuje jedinstvenu vrijednost poznatu kao UID (engl. *Unique ID*)[9].

4.1. *Plug-inovi*

Plugin-ovi (engl. *plug-in*) utvrđuju vrstu objekta kao što je npr. *Sprite* jedna vrsta objekta, a recimo video, *keyboard*, *button* su potpuno druge vrste objekata. Postoje tri glavne vrste dodataka:

- Vizualni *plug-inovi* (npr. *Sprite*) pojavljuju se u izgledu i iscrtavaju nešto na zaslonu.
- Skriveni plugin-*ovi* (npr. *Array*) postavljeni su u određenim scenama, ali se ne iscrtavaju na zaslonu.
- Projektni *plug-inovi* (npr., *Miš, audio*) dodaju se u cijeli projekt i mogu se dodati samo jednom. Ne može biti više od jedne vrste objekta ili instance proširenja na razini projekta. Jednostavno ostvaruju novu mogućnost sustava (kao što je mogućnost unosa miša).

[9]Scirra Ltd, 2015 Manual, Construct 2 Objects [Online], Dostupno na: https://www.scirra.com/tutorials/803/construct-2-basics-objects., [pristupljeno 19.rujna 2018.]



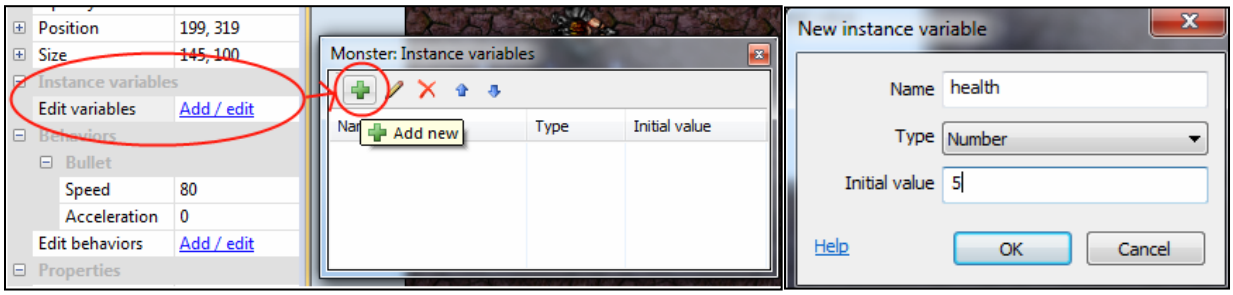
Slika 15. - Prikaz prozora s postojećim Construct 2 *plug-inovima*

4.2. Tipovi objekata

Jedan od glavnih elemenata dizajniranja igara u Construct 2 jest tip objekta. Oni definiraju klasu nekog objekta, na primjer imamo tipove objekata „Igrač“ i „Neprijatelj“ koji dolaze od istog *plug-ina*, a to je *Sprite plug-in*. Oni imaju različite animacije i pomoću događaja svaki za sebe će se drugačije ponašati makar dolazili od *Sprite* objekta.

4.3. Instance tipova objekata

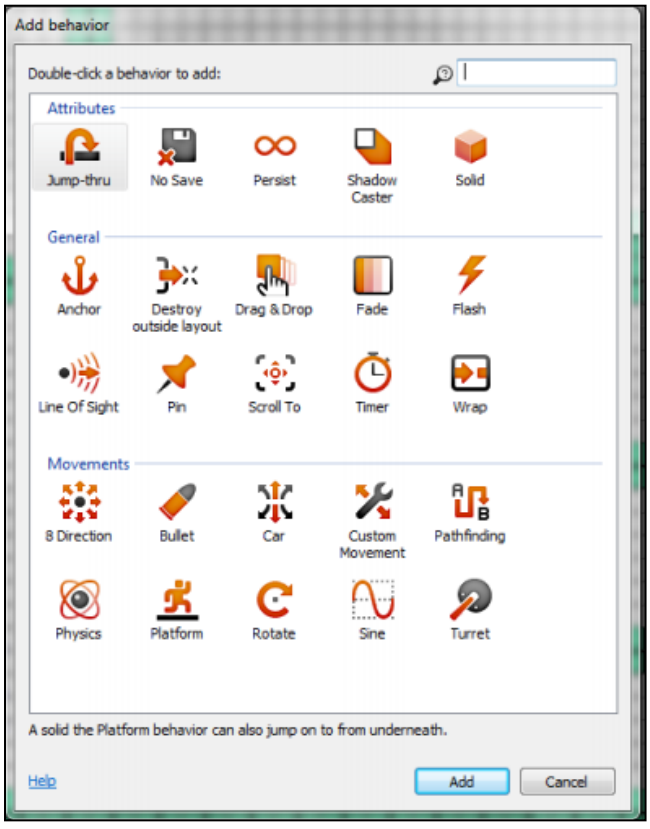
Instance tipova (engl. *instance variables*) objekata omogućuju svakom objektu (npr. neprijatelju) da spremi vlastitu vrijednost života. Kratko rečeno, varijabla je vrijednost koja se može promijeniti (ili varirati) i pohranjuje se zasebno za svaku instancu.



Slika 16. – Primjer kreiranja *instance variable*

4.4. Ponašanja

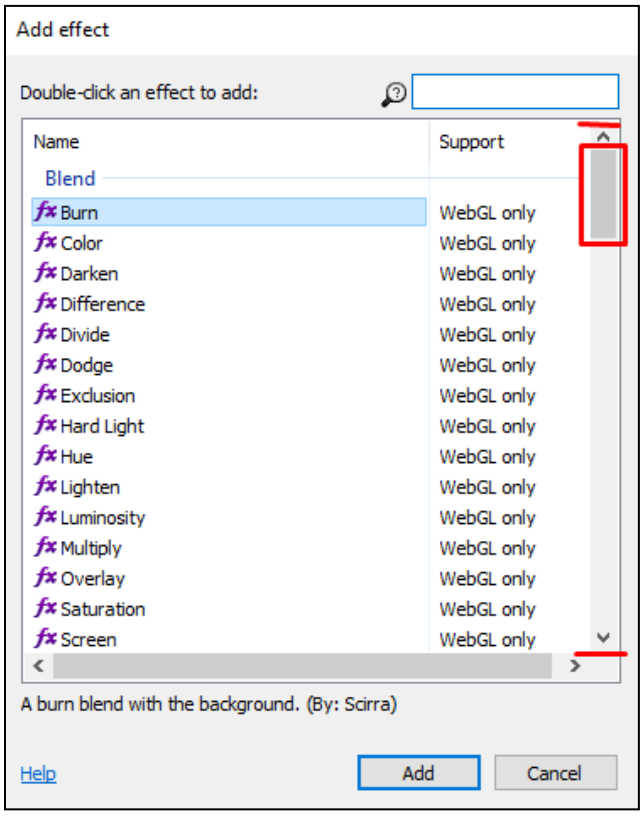
Ponašanja (engl. *behaviors*) daju predefiniranu funkcionalnost tipovima objekata**.** Može ih se dodati pomoću dijaloškog okvira „*Object Behaviors*“. Jedan primjer ponašanja može biti „8 direction behavior“ koji omogućuje kretnju objekta u svim smjerovima (gore, dolje, lijevo, desno i na dijagonalama prema zadanim postavkama tipaka sa strelicama).



Slika 17. - Prozor s postojećim ponašanjima unutar Construct 2

4.5. Efekti

Efekti (engl. *effects*) mijenjaju vizualni izgled objekata. Mogu se dodati u dijaloškom okviru *Effects*. Efekti se mogu dodati i slojevima i scenama, iako se efekti koji se stapaju s pozadinom ne mogu koristiti na sceni. Oni se ponekad svrstavaju i pod *shadere* ili *WebGL shadere*, samo zato što ta tehnologija omogućava prikaz tih efekata. Construct 2 ima u svojoj biblioteci malo više od 70 različitih efekata koji se mogu koristiti.



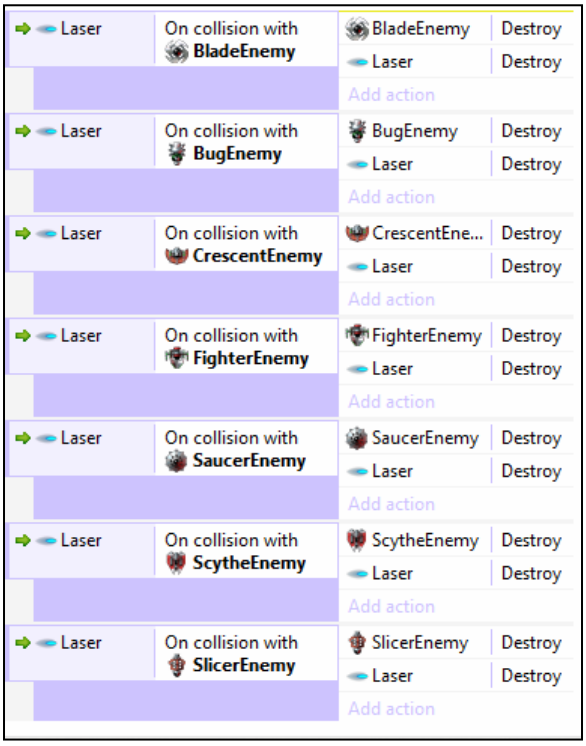
Slika 18. - Prozor s postojećim efektima

4.6. Familije

U Construct 2, familije (engl. *families*) su grupe ili skupine objekata. Sve vrste objekata u obitelji moraju biti iz istog *plug-ina*, npr. svi *Sprite* objekti (a ne kombinacija *Sprite* i *Tiled Background* objekata). Familije nam pomažu da izbjegnemo ponavljanje događaja, na primjer umjesto da imamo iste događaje za objekte Enemy1, Enemy2 i Enemy3 sve ih

možemo dodati u familiju „*Enemies*“ i izvršiti događaje jednom za familiju. Zatim se događaji automatski primjenjuju na sve vrste objekata unutar familije.

- Primjer korištenja familija:



Slika 19. - Primjer događaja bez korištenja familija

- Korištenjem familija svih sedam događaja može se zamijeniti jednim događajem što olakšava stvaranje i održavanje projekata s puno objekata koji trebaju raditi na sličan način.

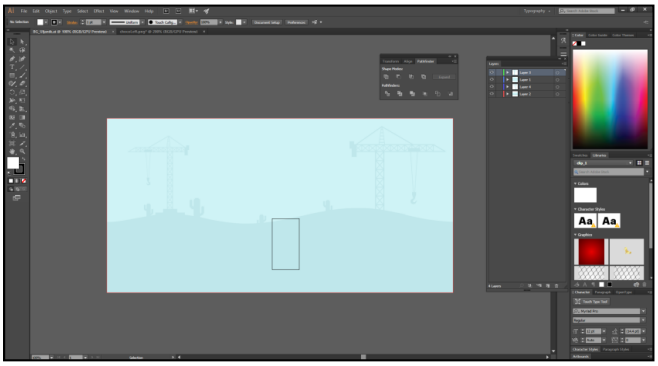
Slika 20. - Primjer događaja ukoliko koristimo familije

5. Razvoj igre

U petom poglavlju prolazi se kroz detaljnu izradu igre, točnije od njene početne do završne verzije. Prije nego što se krenulo u izradu igre, bilo je potrebno smisliti koncept same igre i sve njene dijelove. Sama igra smišljena je na način da se igrač (engl. *player*) nalazi u 2D platformi gdje mu je cilj skupiti tri potpisa kako bi uspješno završio *level* i otvorio vrata sveučilišta za sljedeću razinu. Zadatak igraču otežavaju različiti neprijatelji koji se nalaze na različitim pozicijama *levela*. Također, na platformama se nalaze *gemovi* koji služe kao vodič kroz *level*. Igrač posjeduje moć koju možemo povezati s popularnom igrom „Super Mario" gdje skokom, točnije padanjem, može uništiti objekt neprijatelja. Igra je sastavljena od tri *levela*, gdje pojedini *level* sadrži različite elemente te se na posljednjem *levelu* pojavljuju i dodatni neprijatelji koji će nastojati otežati skupljanje finalnih potpisa i *gemova*. Kompletnu igru prate specijalni zvukovi, koji još više naglašavaju pojedinu akciju, te glazba u pozadini koja daje dodatnu čar.

5.1. Pozadina i izgled *levela*

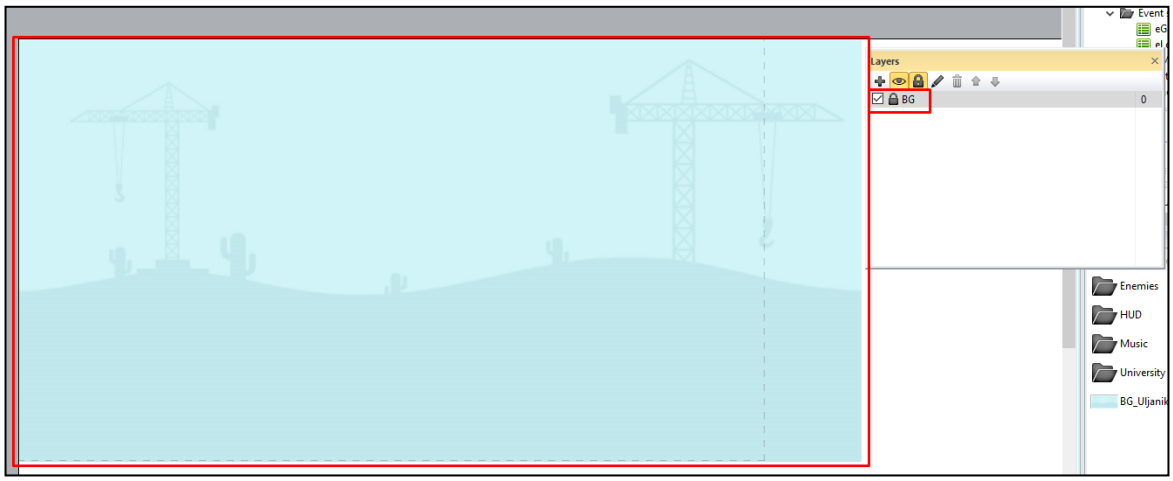
Pozadina (engl. *Background*) *levela* kreirana je unutar programa Adobe Illustrator. Referencu motiva pozadine predstavljaju karakteristike vezane uz grad Pulu, a za referencu dizajna same pozadine korišteni su primjeri s *freepik* stranice[10]. U konkretnom slučaju igre radi se o poznatim pulskim divovima.



Slika 21. - Prikaz pozadine igre kreirane u Adobe Illustratoru

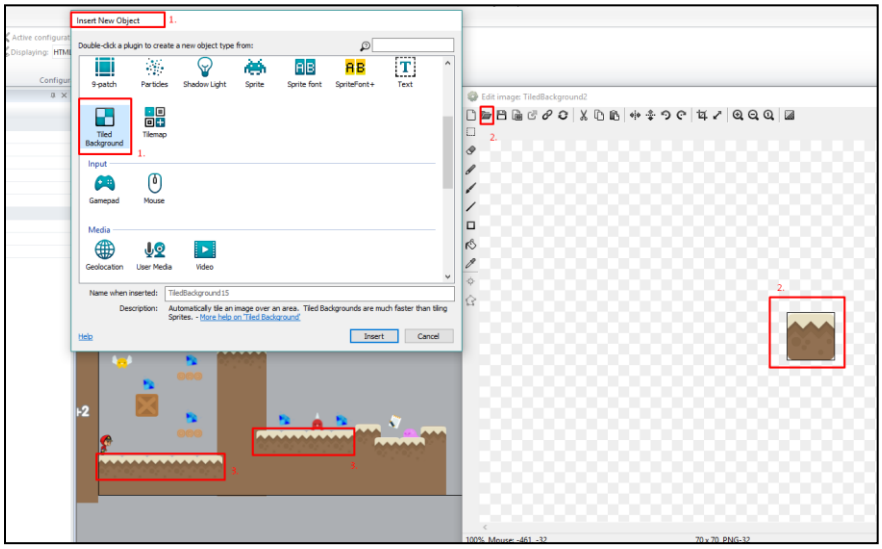
[10] Freepik.com, 2D Game Background Images [Online]. Dostupno na: https://www.freepik.com/index.php?goto=2&searchform=1&k=2d+game+background [pristupljeno 20.rujna 2018.]

Nakon što je pozadina izrađena potrebno je napraviti novi *layer*, u ovom slučaju „BG“ te unutar njega ubaciti izrađenu pozadinu. Poželjno je zaključati *layer* u kojemu se nalazi pozadina kako ju slučajnim klikom miša ne bi pomaknuli.



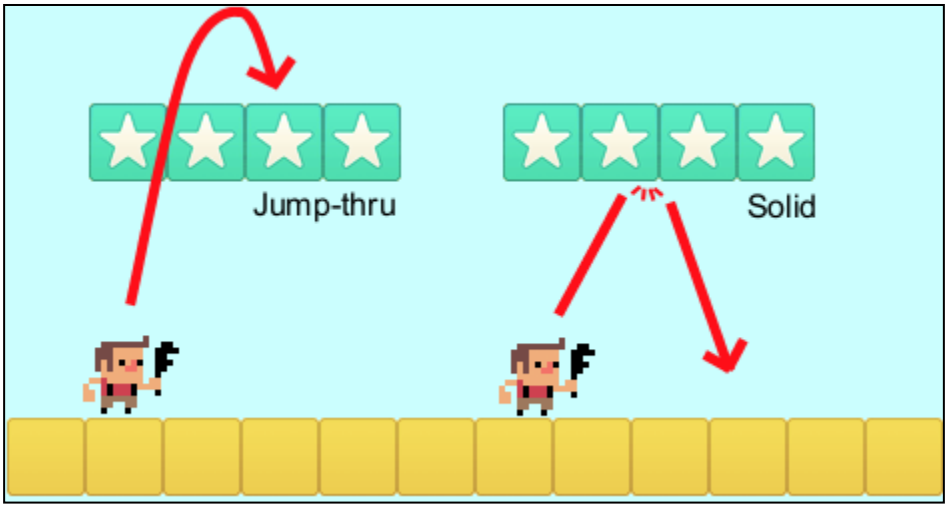
Slika 22. - Prikaz pozadine unutar *Constructa*

Sljedeći korak vezan je uz kreiranje tla platforme po kojemu će igrač hodati. On se radi na način da dva puta kliknemo na pozadinu te unutar prozora tražimo *plug-in Tiled background* u kojemu lociramo sliku koju zatim možemo pravilno rasporediti po cijeloj površini. Važno je tlu platforme dodijeliti ponašanje (engl. *behavior*) *Solid*, koji čini objekt neprohodnim, tako da ostali objekti ne mogu proći ili pasti kroz njega.



Slika 23. - Prikaz procesa kreiranja *Tiled Backgrounda*

Pored navedenog dodijeljenog ponašanja *Solida,* posjedujemo i *behavior* po imenu *Jump-thru*, koji omogućava *Platform* behavioru dvije opcije: da stoji na objektu i da može skočiti na njega odozdo. To se razlikuje od *Solid* behaviora, jer na njemu *Platform behavior* može samo stajati, ali ne može skočiti na njega odozdo.



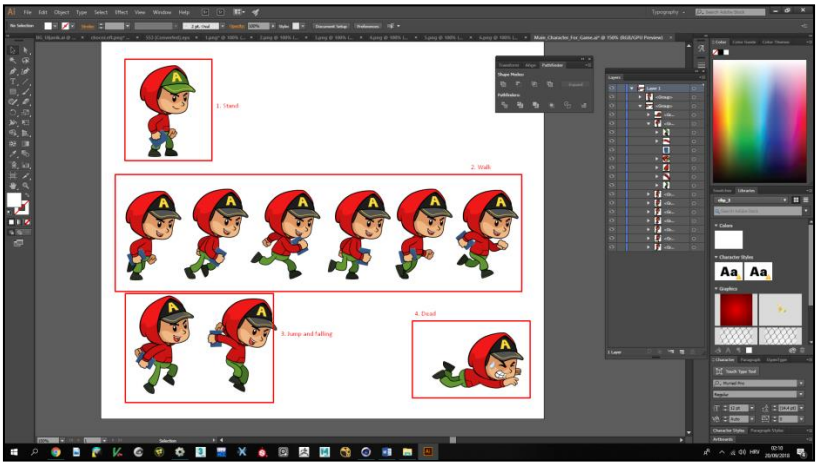
Slika 24. - Razlika *Jumpthru* i *Solid* behaviora

U konkretnom slučaju *Jump-thru behavior* je korišten na platformama koje se nalaze u „zraku“.

Slika 25. - Prikaz *Jump-thru* behaviora

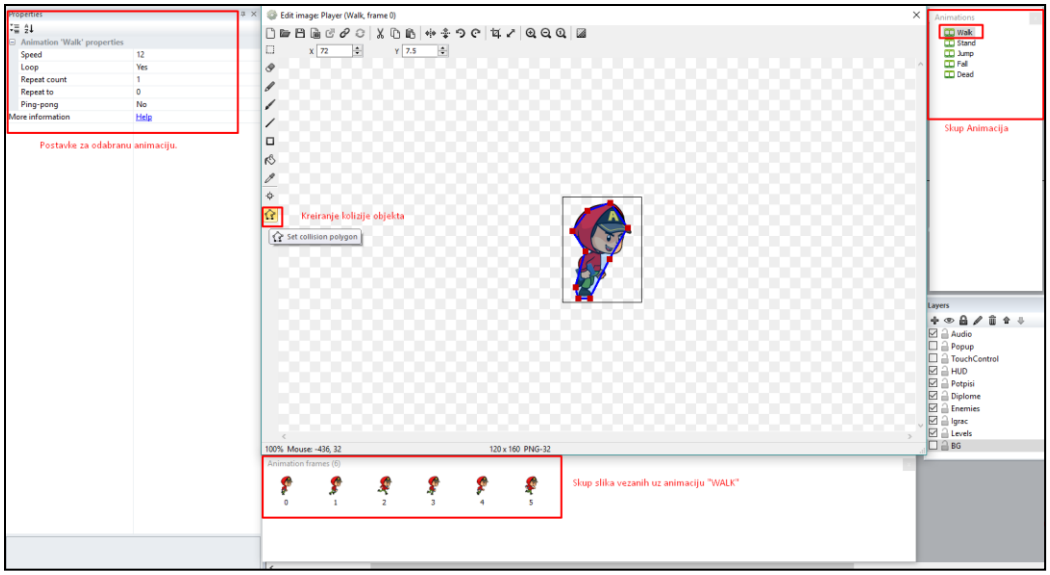
5.2. Glavni lik

Kreacija samog lika je također rađena unutar Adobe Illustratora. Unaprijed je potrebno specificirati za što će se točno određeni objekt koristiti. Budući da se radi o glavnom liku, unaprijed je potrebno specificirati pokrete, točnije *rig* animacije koje će nam prilikom animiranja kretnji, dodjeljivanja događaja i akcija igrača uvelike olakšati.



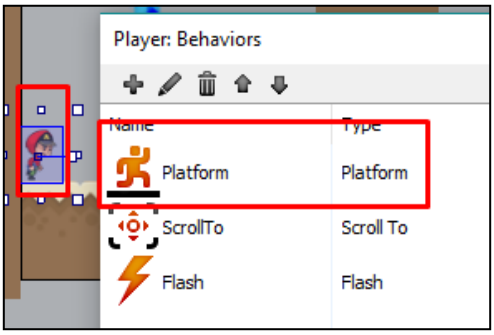
Slika 26. - Prikaz kreacije igrača unutar Adobe Illustratora

Proces se svodi na *drag and drop* skupa slika animacije unutar *Sprite* objekta. Na taj način dolazimo do kreiranja početne animacije, kojoj se treba specificirati kolizija na pojedinoj ubačenoj slici. Važno je napomenuti da kolizija predstavlja margine uvezenog objekta, točnije objekt će komunicirati s okolinom samo unutar označene kolizije. Prilikom *drag and dropa* slika, početnoj se animaciji automatski dodjeljuje ime „*Default*“, što je potrebno izmijeniti sukladno referentnoj animaciji. Za glavnog igrača kreirano je pet animacija, točnije: *walk, stand, jump, fall* i *dead*.



Slika 27. - Prikaz animacija i kreiranja kolizije igrača

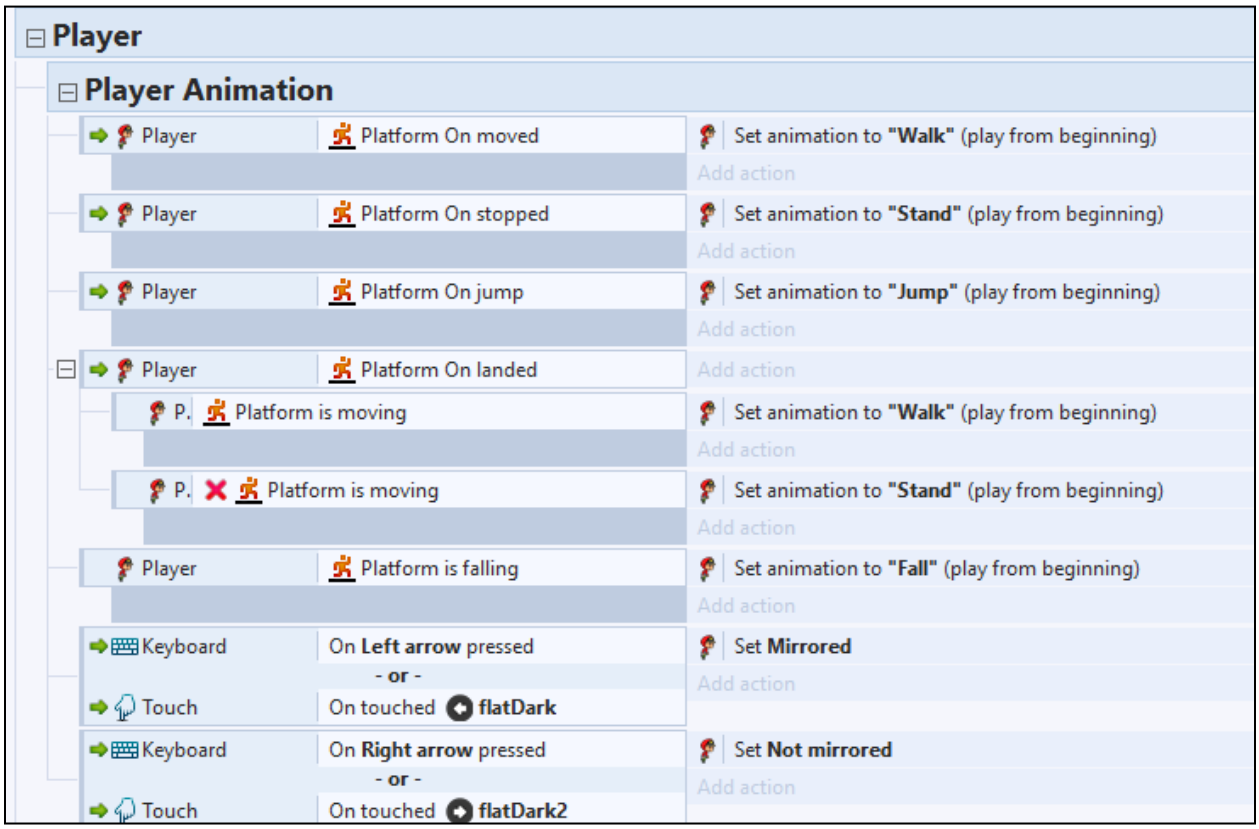
Kako bi se igrač mogao kretati po platformama, potrebno mu je dodijeliti „jump and run" stil kretnji, a upravo se to izvodi *behaviorom Platform,* čije ponašanje daje objektu mogućnosti kretanja lijevo, desno te mogućnost skoka. Najčešće se koristi u *platformerima* za igrače i neprijatelje. Ponašanje *Platform* podržava nagibe, platforme koje posjeduju izmjenu pozicija, platforme s *jump-thru* ponašanjem i određene vrste gravitacije.



Slika 28. - Prikaz *behaviora Platform*

Ponašanje platforme bit će usmjereno na bilo kakve objekte sa *Solid* ili *Jump-thru behaviorima*. U ovom je projektu odlučeno da će se igrač moći kretati samo u tri osnovna smjera (lijevo, desno i skok). Svakom od tri smjera kretanja pridružena je prethodno izrađena animacija za kretanje u tome smjeru. Dodatno je odrađen i dio vezan za

zrcaljenje (engl. *Mirror effect*) igrača, kako bi prilikom promjene na x-osi igrač uvijek bio okrenut prema pravom smjeru.

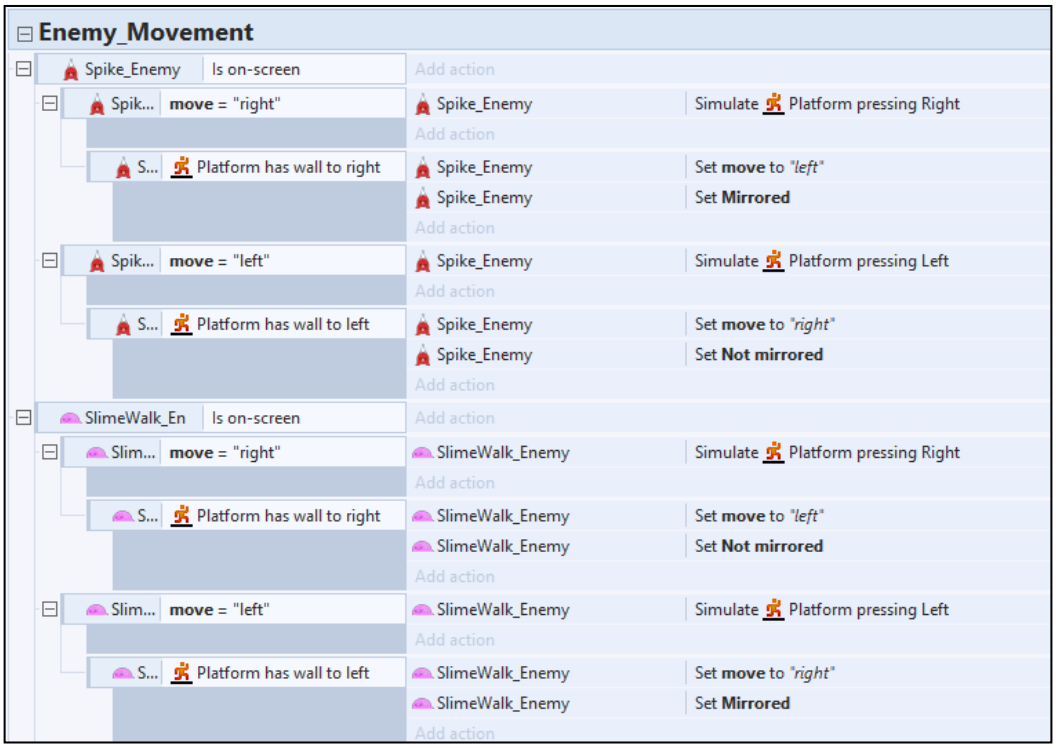


Slika 29. - Prikaz događaja i animacija vezanih za kretnju igrača

Konačno je ponašanje koje se dodalo glavnom liku to da ga kamera konstantno drži u fokusu zbivanja. Što znači da se na početku *levela* igrač stvori na poziciji koju smo prethodno postavili te naravno ta lokacija ne smije biti zapreka ili zid. Kamera se automatski fokusira na njega i prati ga kroz njegovo kretanje. Ponašanje *Scroll To Object* omogućuje fokusiranje na određeni objekt, u ovome slučaju to je igrač.

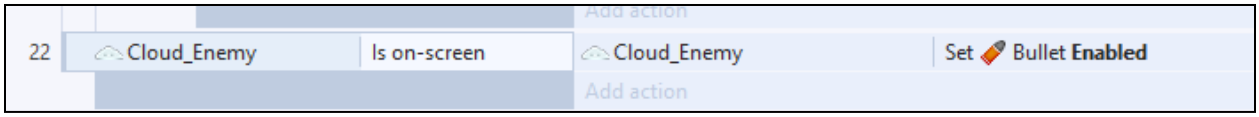
5.3. Neprijatelji

Neprijatelji su objekti kojima je potrebno pomoću događaja kontrolirati kretanje po radnom prostoru neovisno o korisniku ili prilagođavati kretnje ovisno o igraču. Kreiraju se različiti objekti koji predstavljaju neprijatelje kako bi se mogle koristiti razine različitih težina. Proces je kreacije neprijatelja jednak opisu kreiranja igrača. Prikazani na slici 30. su događaji za jednostavno automatsko kretanje neprijatelja, gdje će objekt prilikom realizacije *Solid* behaviora, u ovom slučaju zida, izvršiti rotaciju za 180 ili -180 stupnjeva na x-osi te se nastaviti kretati.



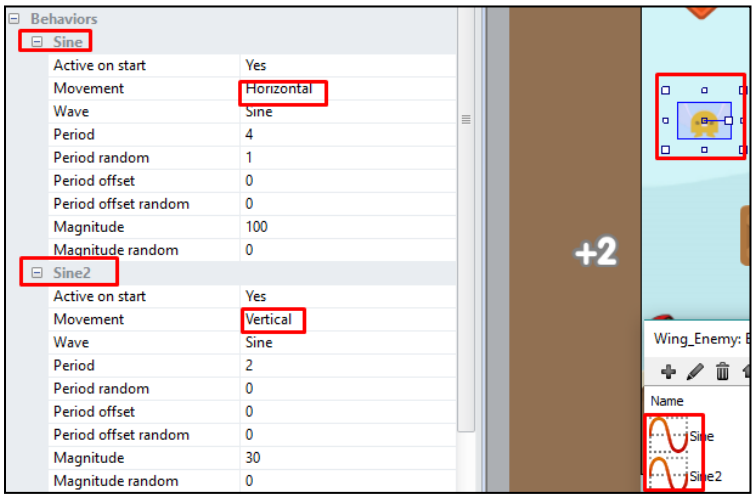
Slika 30. - Prikaz događaja i animacija vezanih uz kretnje neprijatelja

Također, imamo primjer objekta „Cloud_Enemy“ koji ne ovisi o korisniku nego mu se dodaje svojstvo pravocrtnog kretanja naziva *Bullet* prikazano na slici 31., a događajima se prilagođava tako da se može kretati samo kada se pojavi unutar radnog prostora i to specifično po x-osi.



Slika 31. - Prikaz neprijatelja koji je vođen *Bullet* svojstvom

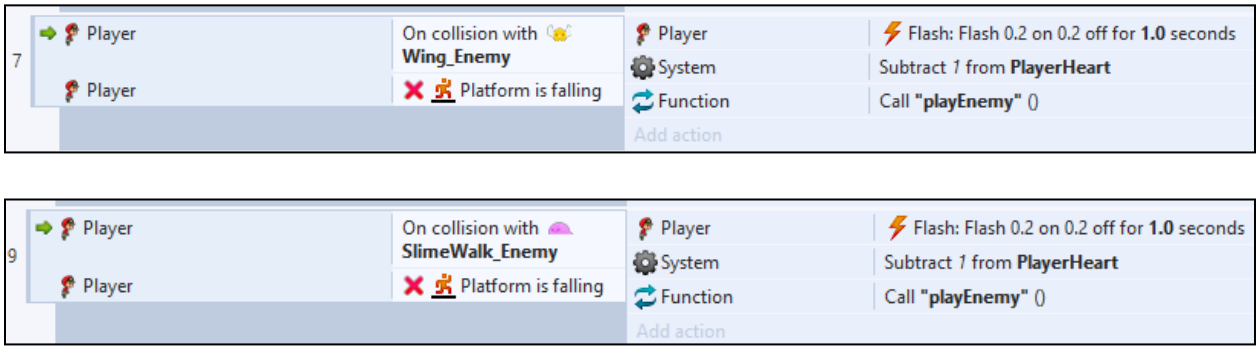
Još jednu razliku predstavlja i objekt „Wing_Enemy“, koji je animiran putem *Sine behaviora* te se nalazi u zraku. Pomoću *Sine behaviora* moguće je postaviti neke karakteristike objekta, kao što su položaj, veličina, kut i to na način da oscilira naprijed-natrag. Od funkcija koje se koriste, za takvo kretanje, postoje sinusna, trokutasta, pila ili obrnuta pila, a od karakteristika koje se mijenjaju po određenoj funkciji mogu se mijenjati: vertikalni i horizontalni položaj, prozirnost, kut, veličina, širina i visina. U konkretnom slučaju radilo se o kombinaciji dva *Sine behaviora*, jedan za izmjenu horizontalnog položaja i drugi za izmjenu vertikalnog položaja. Na taj način odrađena je simulacija leta objekta.



Slika 32. - Prikaz neprijatelja koji je animiran putem *Sine behaviora*

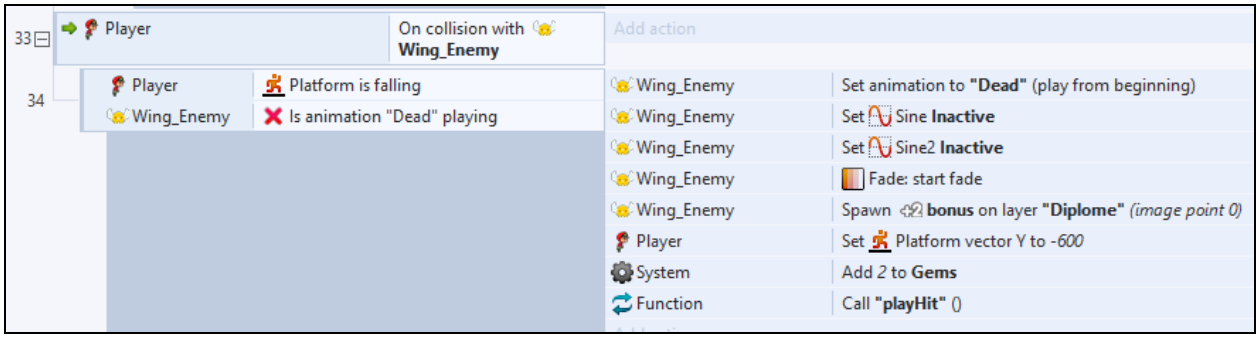
5.4. Uništavanje neprijatelja

Zajedničko je za sve neprijatelje da prilikom „TRUE“ kolizije s igračem budu uništeni. Prilikom „FALSE“ kolizije igraču se oduzima jedan život te se pokreće *Flash behavior* koji daje objektu mogućnost bljeskanja tako da jako brzo uključuje i isključuje vidljivost tog objekta. Naposljetku se pokreće zvučni zapis koji naglašava sudar, točnije „FALSE“ koliziju s neprijateljem.

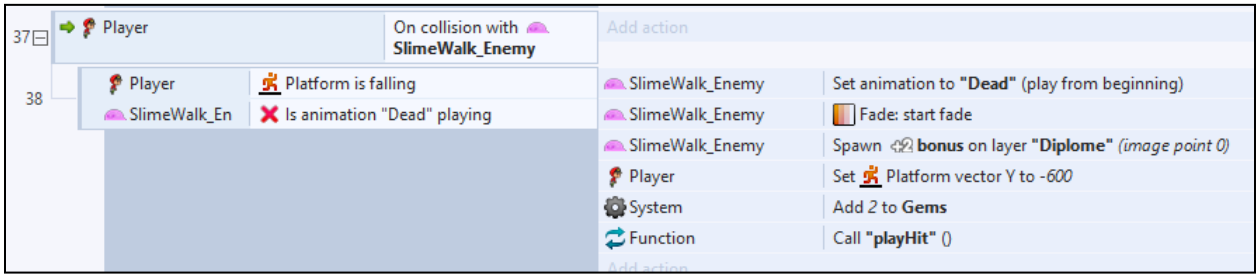


Slika 33. - Prikaz dvaju „FALSE“ kolizija između igrača i neprijatelja

Kako bi igrač uništio neprijatelja, on mora skočiti te pasti na neprijatelja kako bi se izvršilo uništavanje neprijateljskog objekta. Na slici 34. vidimo prikaz jedne kolizije s neprijateljem koji se nalazi u zraku. Detaljnije, kada objekt koji posjeduje *behavior Platform* padne na neprijatelja, u ovom slučaju „*Wing Enemya“,* animacija neprijatelja stavlja se na „Dead“, zatim se ponašanja *Sine* i *Sine 2* isključuju, a nakon toga pokreće se *behavior* „*Fade*“ kojemu je svrha da objekt u podešenom vremenskom okviru polako dobiva prozirnost i nestane. Tome još dodajemo animaciju *+2 bonus* koja ima samo vizualnu ulogu. Nakon toga, igraču se dodjeljuje skok za -600 *pixela* na y-osi te se dva *gema* pridodaju globalnoj varijabli „*Gems*“ koja je vezana za zbroj *gemova*. Na kraju se pokreće zvučni zapis koji naglašava akciju uništavanja neprijatelja.



Slika 34. - Prikaz „TRUE“ kolizije između igrača i neprijatelja **u zraku.**



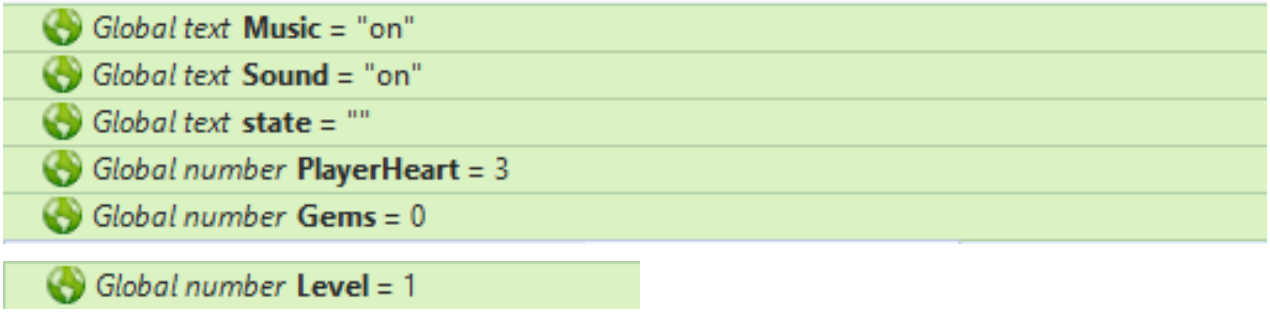
Slika 35. - Prikaz „TRUE“ kolizije između igrača i neprijatelja **na tlu.**

5.5. Globalne varijable, HUD i izbornik *levela*

Riječ varijabla, dolazi iz engleske riječi *variable što* znači promjenjiva vrijednost. To može biti broj, riječ ili nešto drugo.

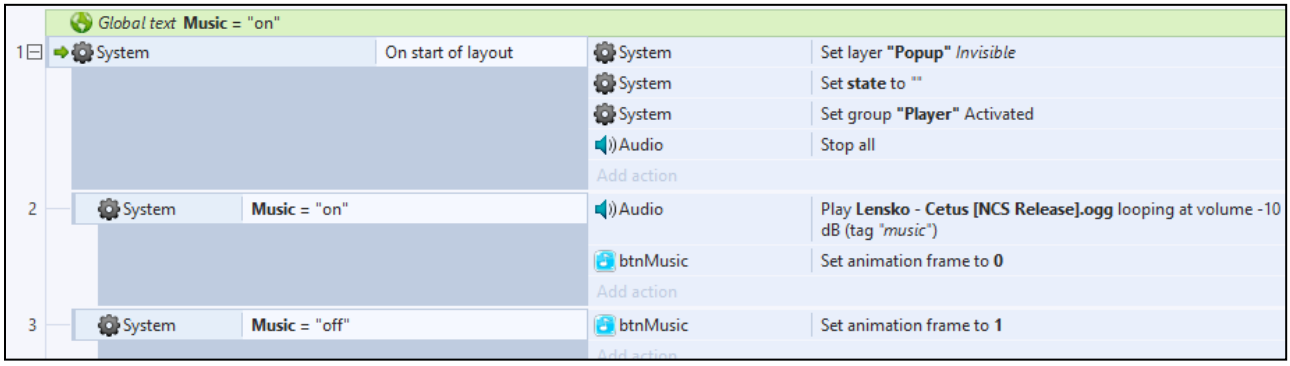
Globalne varijable pohranjuju svoje vrijednosti između scena. Događaji iz bilo koje scene mogu pristupiti bilo kojoj globalnoj varijabli, čak i ako je stvorena na drugoj listi događaja koja nije uključena. Moguće ih je premjestiti na drugu listu događaja „rezanjem“ (engl. *cut*) i lijepljenjem (engl. *paste*). Nakon rezanja, reference na globalnu varijablu nestat će jer je uklonjena; to je normalno i nema potrebe za brigom jer kada zalijepite globalnu varijablu reference koje su nestale ponovo će se pojaviti.

U ovome projektu postoje dijelovi koji se prate i mijenjaju u vremenu. To su globalne varijable vezane za paljenje i gašenje pozadinske glazbe te specijalnih efekata, također varijable vezane uz odnose, točnije status igre (engl. *state*: *win or lose*) te također brojčane varijable koje se odnose na živote igrača te zbroj *gemova* kroz *levele*.



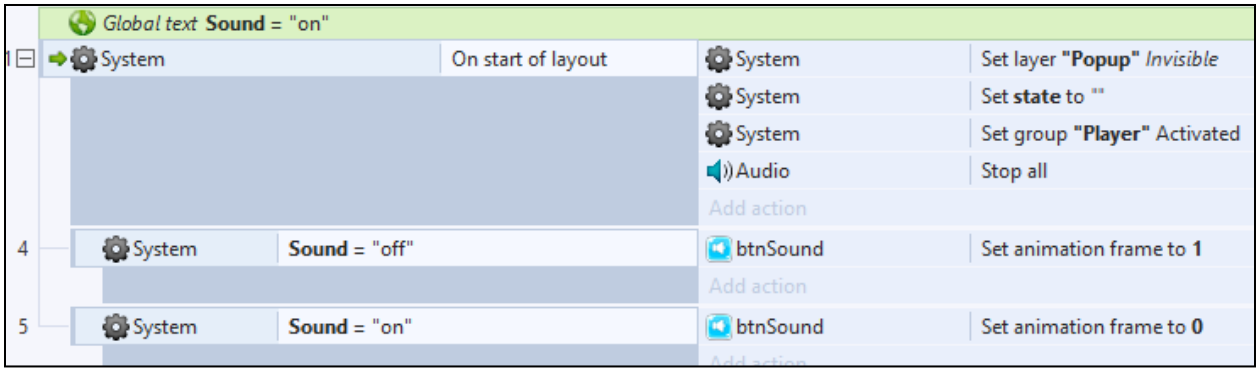
Slika 36. - Prikaz svih globalnih varijabli unutar igre „Funky Student“

Globalna tekstualna varijabla *Music* služi kako bi korisnik prilikom igranja, u bilo kojem trenutku, mogao isključiti pozadinsku glazbu te ju ponovno uključiti ukoliko on to želi. Radnja se izvršava pritiskom na plavi gumb koji posjeduje ikonu osminke (nota).



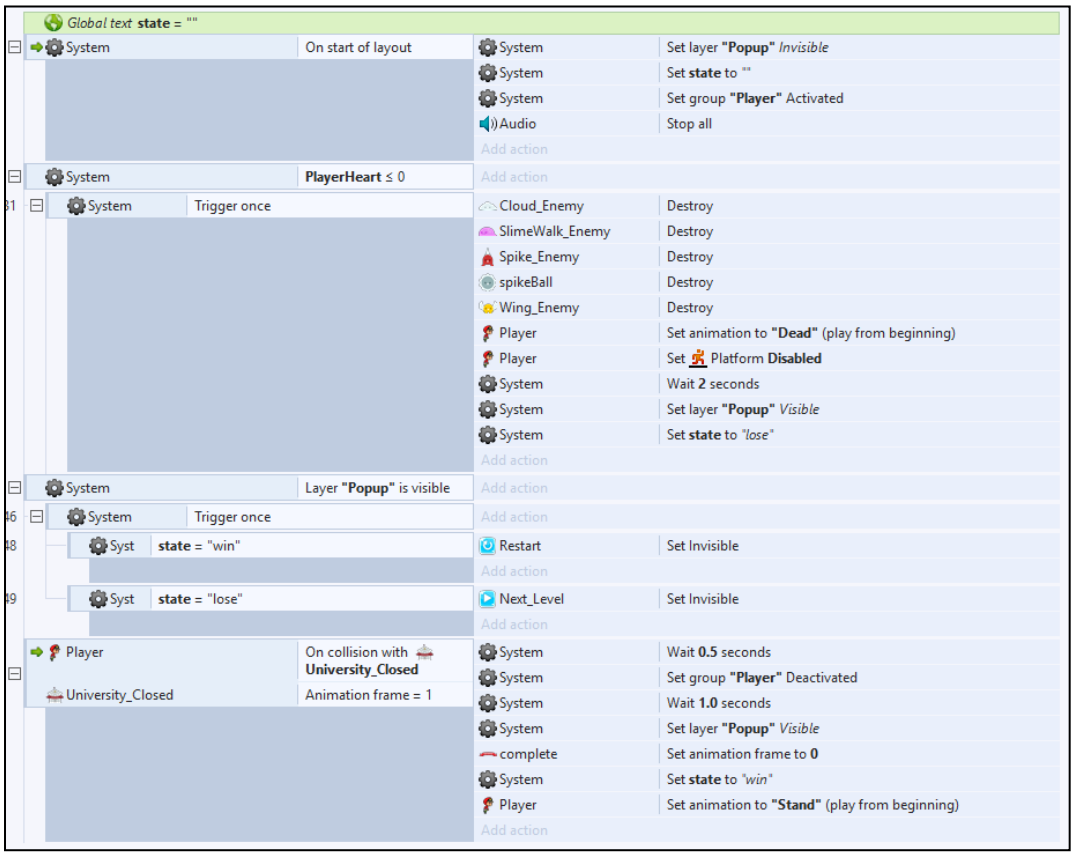
Slika 37. - Prikaz globalne varijable *Music*

Globalna tekstualna varijabla *Sound*, služi kako bi korisnik prilikom igranja, u bilo kojem trenutku, mogao isključiti zvučne specijalne efekte (zvuk klika, zvuk kolizije igrača s objektima, zvuk uspješno završenog *levela* itd.) te također, kako bi ih ponovno mogao uključiti ukoliko on to želi. Radnja se izvršava pritiskom na plavi gumb koji posjeduje ikonu „zvučnika“.



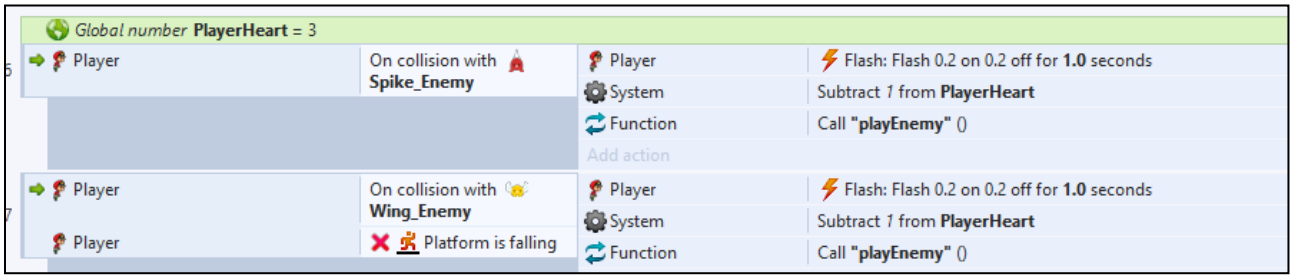
Slika 38. - Prikaz globalne varijable *Sound*

Globalna tekstualna varijabla *state* služi kako bi mogli specificirati status igre, točnije ukoliko je igrač izvršio sve potrebne zadatke za prelazak *levela* (pokupio tri potpisa), *state* će biti jednak *win*. Ukoliko je igrač, npr. bio u sudaru s neprijateljem te pri tome izgubio sve živote, *state* će biti jednak *lose*.



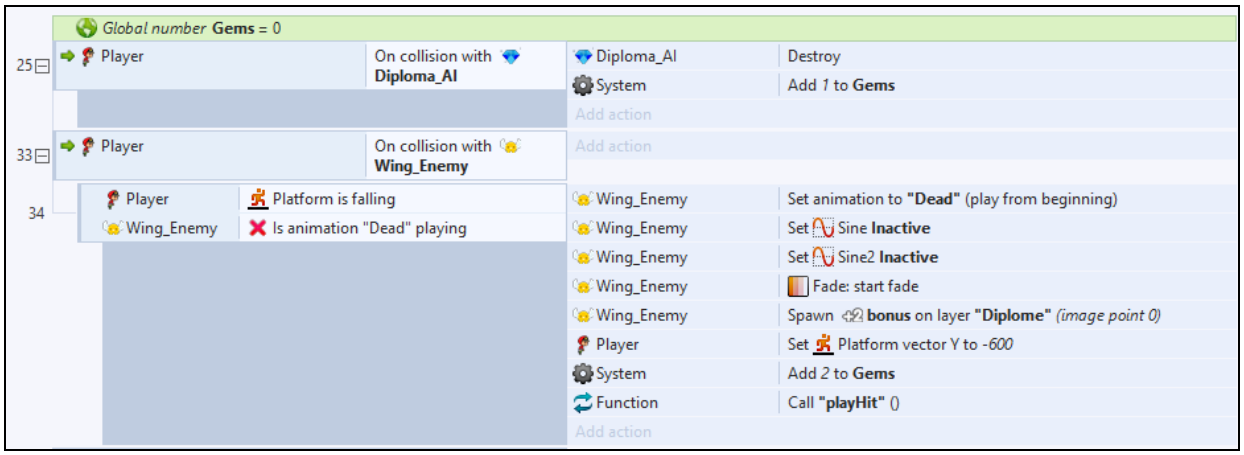
Slika 39. - Prikaz globalne varijable *state*

Globalna brojčana varijabla *PlayerHeart*, služi kako bi korisnik prilikom igranja, interaktivno bio obaviješten koliko „života“ trenutno posjeduje. Igraču se odmah dodjeljuje vrijednost tri, točnije igrač će prilikom pokretanja igre automatski posjedovati tri života. Ukoliko igrač uništi neprijatelja, zbroj će se pridodati, no ukoliko dođe u sudar s neprijateljem, oduzet će mu se jedan život.



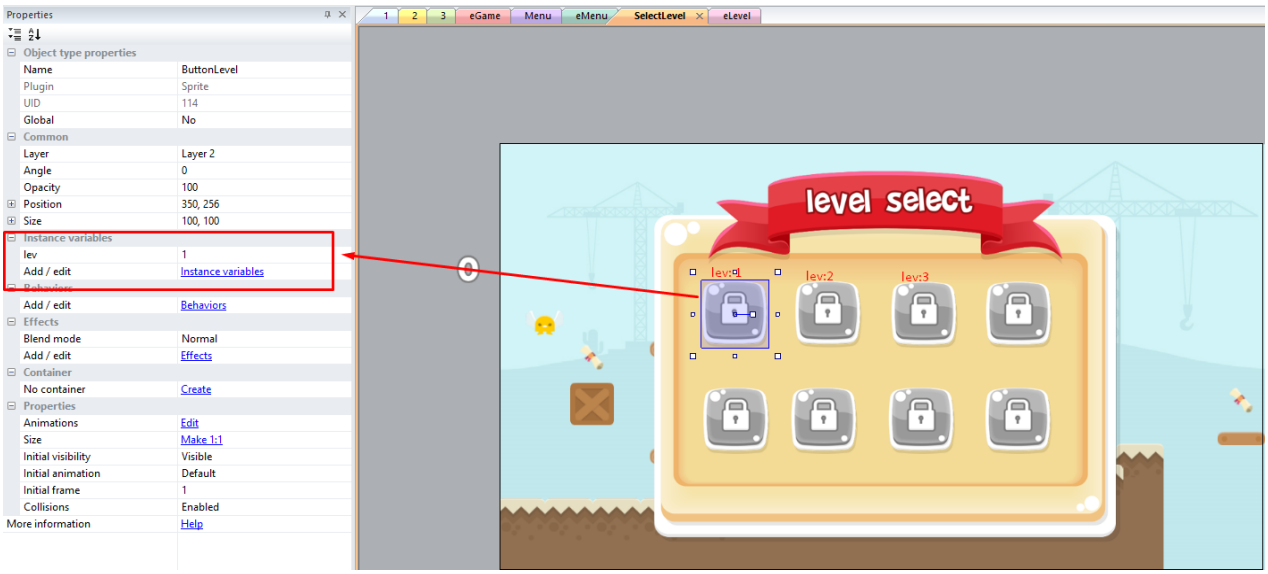
Slika 40. - Prikaz globalne varijable *PlayerHeart* u sudaru s neprijateljem

Globalna brojčana varijabla *Gems*, služi kako bi korisnik prilikom igranja interaktivno bio obaviješten koliko *gemova* trenutno posjeduje. Varijabli se dodijeljuje vrijednost nula, točnije igrač će prilikom pokretanja igre automatski posjedovati nula *gemova*. Ukoliko igrač pokupi *gem*, jedan će se pridodati zbroju te ukoliko uništi neprijatelja dva će se *gema* pridodati zbroju globalne varijable.



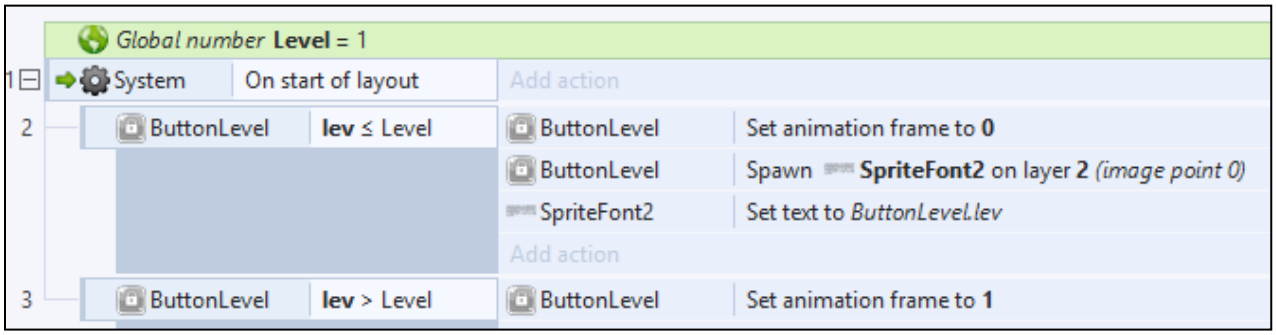
Slika 41. - Prikaz globalne varijable *Gems* prilikom kolizije s igračem

Globalna brojčana varijabla *Level* važna nam je kod izbora *levela* (engl. *Level Select*), po *defaultu* postavljena je na jedan. Točnije, operator *System* na početku svake scene izvršava usporedbu globalne varijable s brojčanim instancama koje su dodijeljene *buttonima* za pojedini *level* (engl. Level Buttons), koje prikazuje slika 42.



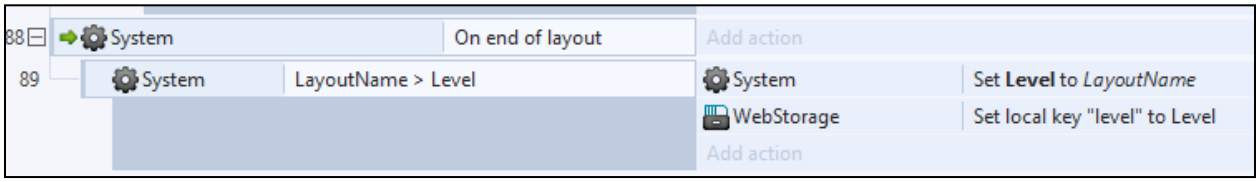
Slika 42. - Prikaz brojčanih varijabli koje su dodijeljene pojedinom *levelu*

Razlog zbog kojega je varijabli dodijeljena vrijednost jedan je kako bi od samog početka prvi *level* bio otključan.



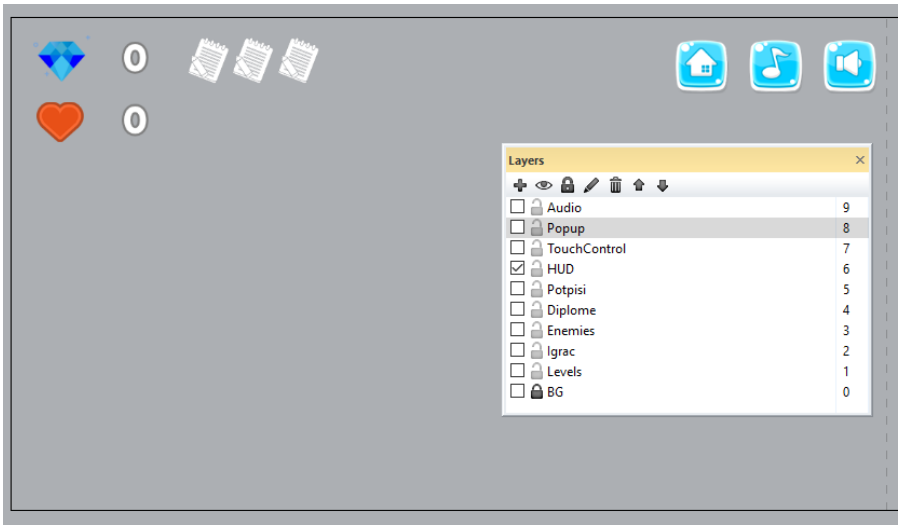
Slika 43. - Prikaz događaja vezanih uz globalnu varijablu *Level*

Na kraju svake scene odvijat će se usporedba dvaju vrijednosti. Prva će vrijednost biti da ime scene (engl. *LayoutName*) mora biti veća od globalne varijable *Level*. Ukoliko je rezultat istinit *System* će u varijablu *Level* postaviti vrijednost imena scene *LayoutName* te otključati sljedeću razinu.



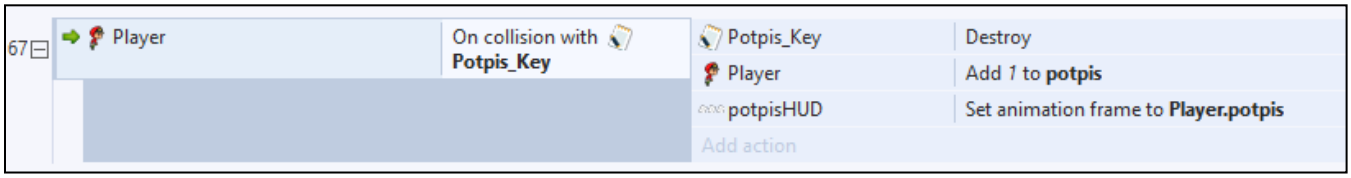
Slika 44. - Prikaz događaja usporedbi vrijednosti, koji se javljaju na kraju scene

Riječ HUD (engl. *head-up display*) predstavlja pogled na igračev napredak, stvari koje trenutno posjeduje (životi) te stvari koje napretkom prikuplja (potpisi i *gemovi*). Za HUD je potrebno napraviti poseban sloj (engl. *layer*), kako se ne bi pomicao ovisno o *levelu* već da bi bio „uglavljen“ na istoj poziciji. Najčešće je napravljen od tekstualno-brojevnog objekta povezanog s nekom varijablom. U ovome projektu HUD je upotrijebljen kako bi igrač mogao pratiti koliko ima života, *gemova* i potpisa.

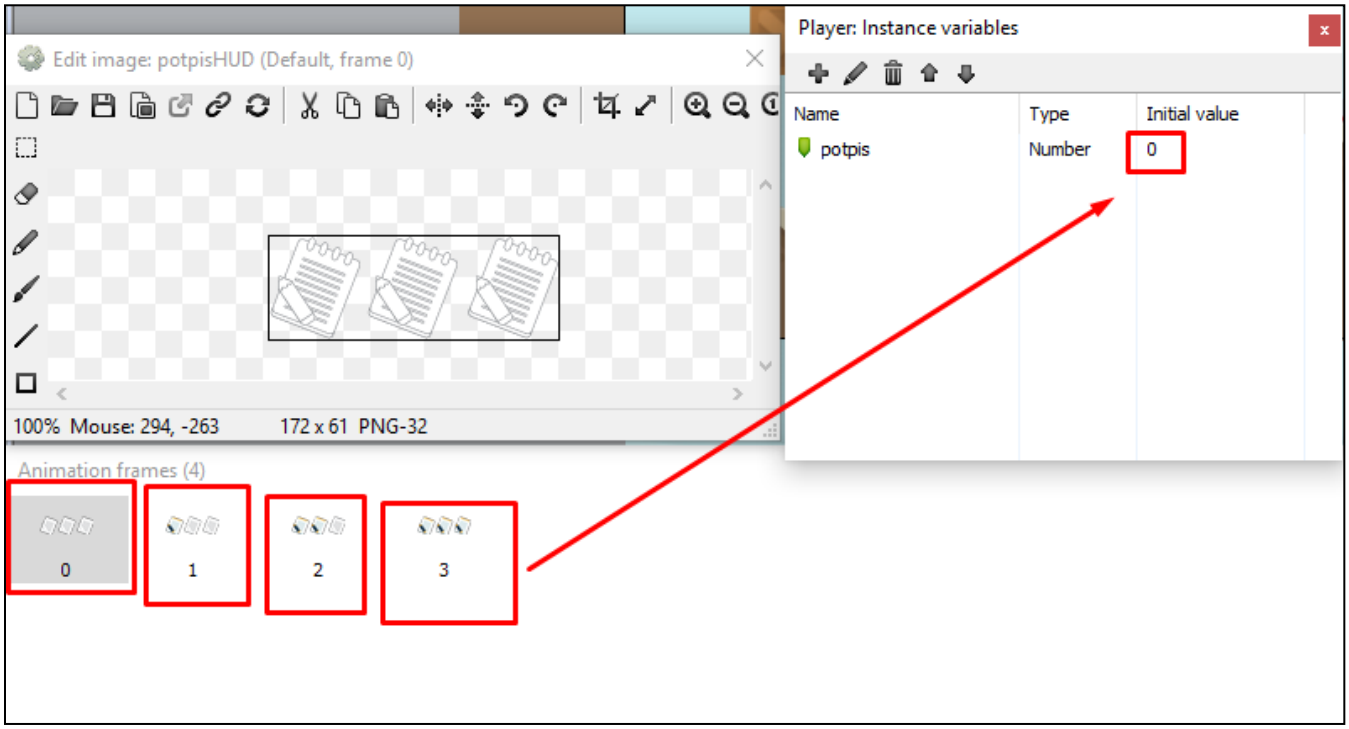


Slika 45. - Prikaz HUD-a

Na slici 46. moguće je vidjeti primjer jedne kolizije igrača s objektom „*Potpis_Key*“, točnije gdje igrač prilikom kolizije s *Potpis_Key*om prvo uništava objekt. Zatim se u instance varijablu „potpis“ dodaje broj jedan pa se u konačnici animacija *framea potpis HUD*-a ažurira s brojem koji trenutno posjeduje varijabla „potpis“.

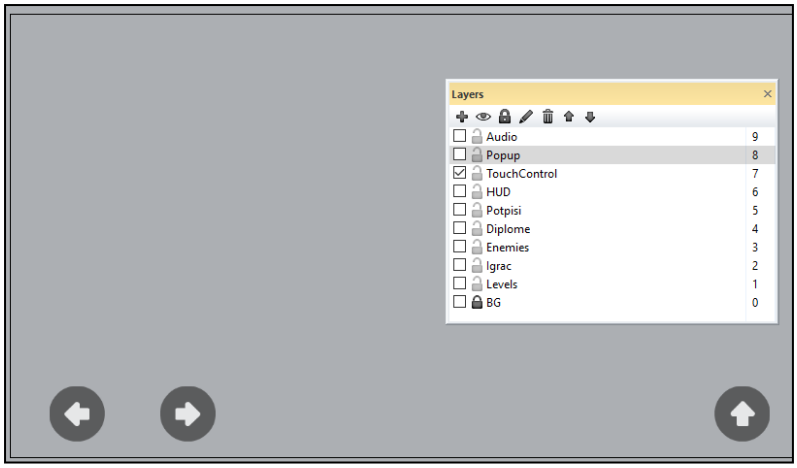


Slika 46. - Prikaz kolizije igrača s objektom *Potpis Key*



Slika 47. - Prikaz animacijskih *frameova Potpis Key*a i varijable *potpis*

Također, unutar kontrole putem tipaka odrađen je i dodatan pristup kako bi korisnik mogao kontrolirati igrača i putem miša, tj. kontrolom dodira (engl. *Touch Control*).



Slika 48. - Prikaz sloja TouchControl

Proces događaja sloja vođenog kontrolom dodira je vrlo jednostavan. Kreiran je novi sloj po imenu *TouchControl* na kojemu su pravilno raspoređene tipke kretnji. Nakon toga su tipke ispravno povezane sa smjerom događaja kretnji te je dodatno odrađen i dio za zrcaljenje igrača (engl. *Mirror Effect*).

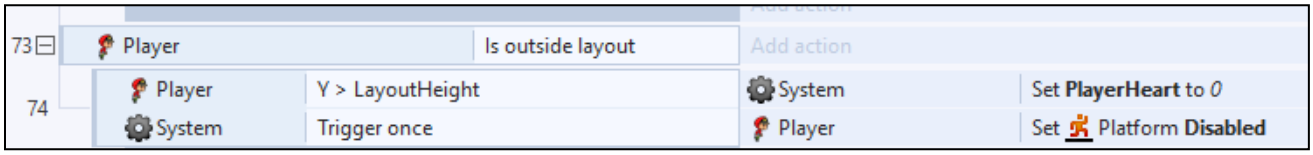
Slika 49. - Prikaz kontrola događaja vezanih uz TouchControl

Ovdje je moguće vrlo lako nadodati još jedan sloj, a to je *Popup Layer*, koji označava dva uvjeta: kraj *levela* (engl. *Level Complete*) ili kraj igre (engl. *Game Over*). Oba se uvjeta dosta razlikuju, no posjeduju jedan zajednički *event*, a to je da prilikom pojavljivanja ispišu podatke skupljenih *gemova* i potpisa.

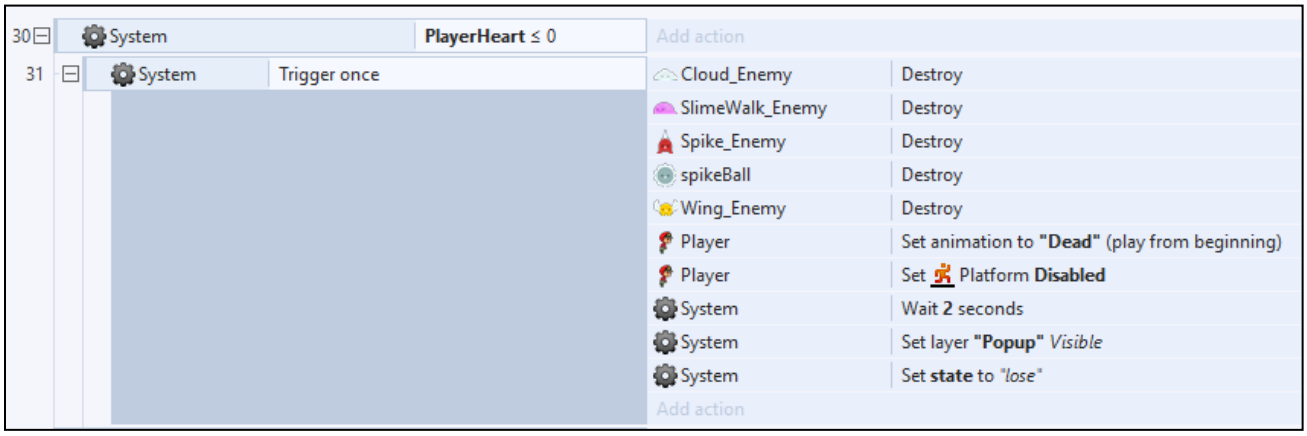
Slika 50. - Prikaz sloja Game Over

Slika 51. - Prikaz sloja Level Complete

Pop-up prozor vezan za kraj igre (engl. *Game Over*), pojavljuje se samo ukoliko igrač padne s platforme te ukoliko igrač posjeduje manje ili jednako nula života (Slika 52. i Slika 53.).



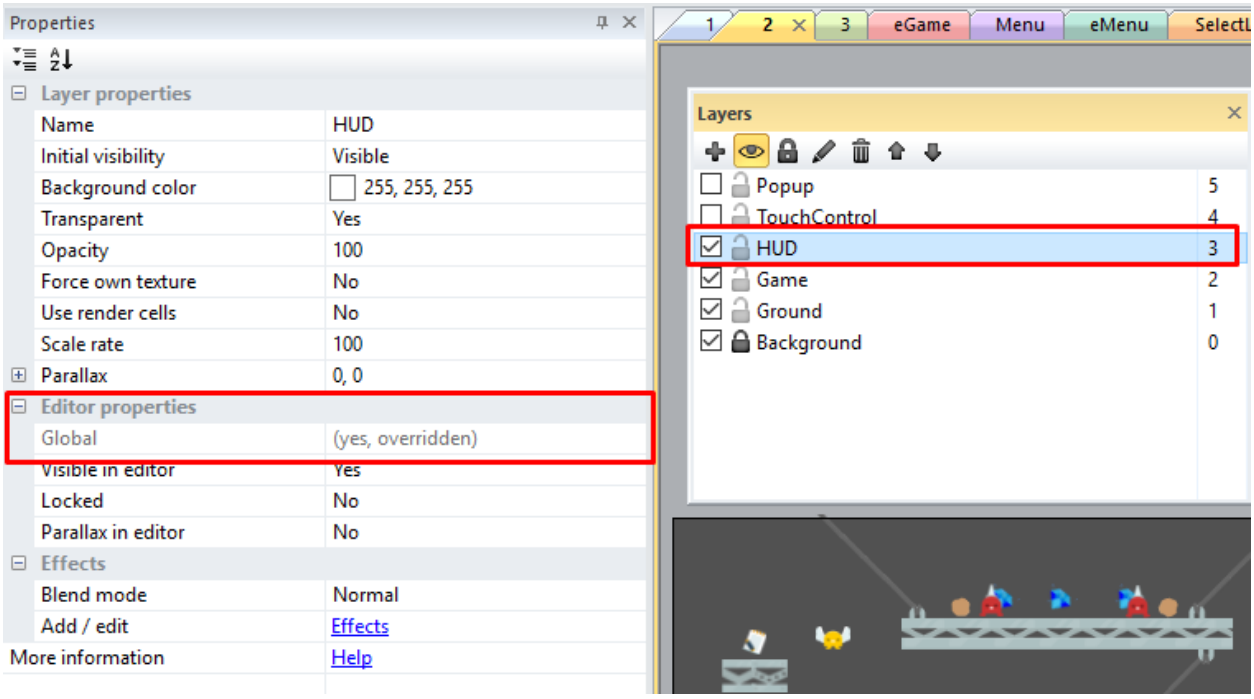
Slika 52. - Prikaz događaja *Game over* ukoliko igrač padne sa platforme



Slika 53. - Prikaz događaja *Game over* ukoliko igrač posjeduje manje ili jednako nula života

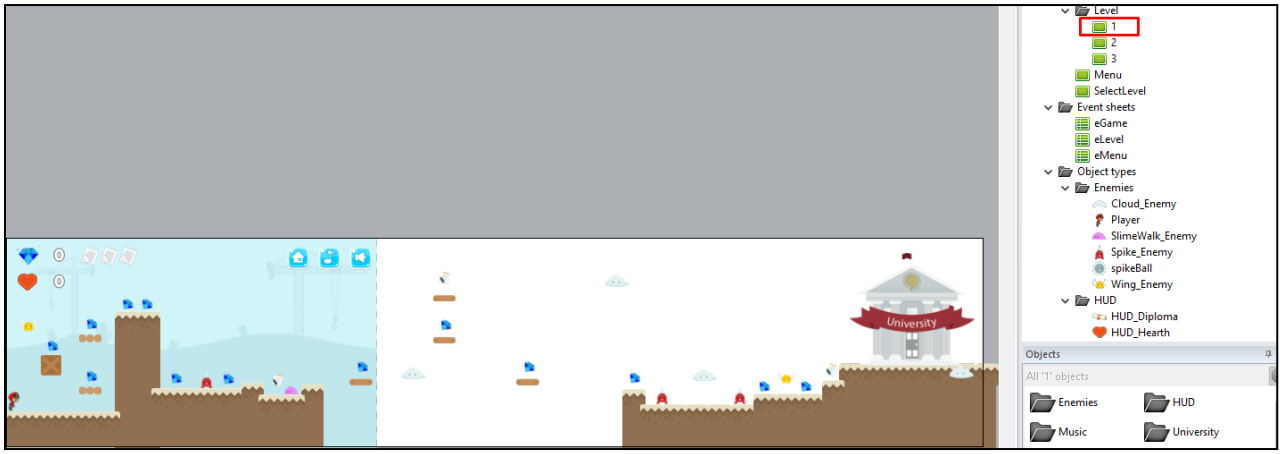
5.6. Razine

Jedan od zadataka bio je i kreiranje preostalih razina. Različite su težine razina postignute jednostavnim raspoređivanjem objekata po radnom prostoru i mijenjanjem svojstava svih objekata kako bi se dobio efekt težine. Na određenim razinama susrećemo iste neprijatelje, no pozicije i izgled platformi specifične su. Također, kao vodič smjera igre posjedujemo *gemove*, čije nam skupljanje ne predstavlja nikakav uvjet, osim vizualne vodilje kroz razinu. *HUD, Popup* i *TouchControl layeri* su naslijeđeni od prve razine, na način da prilikom kreiranja nove scene, kreiramo sloj koji mora imati isto ime kao i na prethodnim scenama. Potvrda nam je kada označimo *layer* te pogledamo postavke *editora* da je uvjet *Global = yes, overridden.*

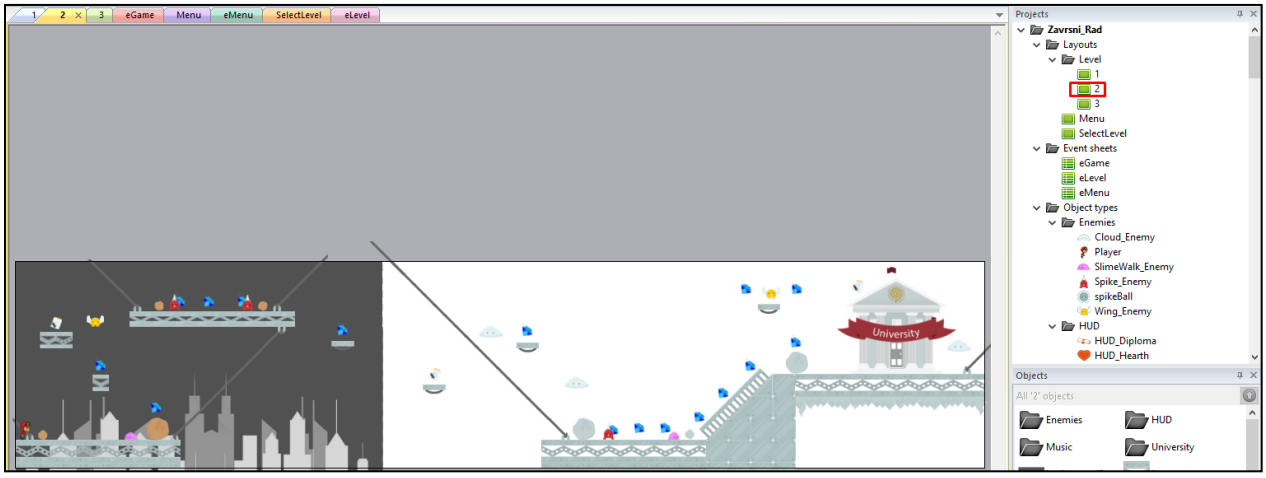


Slika 54. - Prikaz uspješno naslijeđenog „HUD“ *layera*

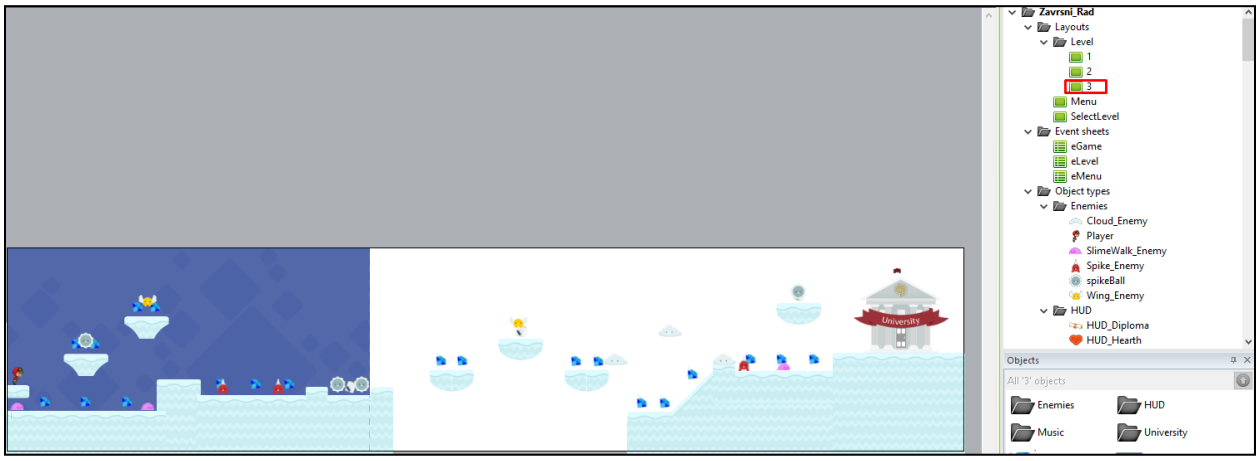
Konačni pogled na sve tri scene:



Slika 55. - Prikaz prve scene/razine



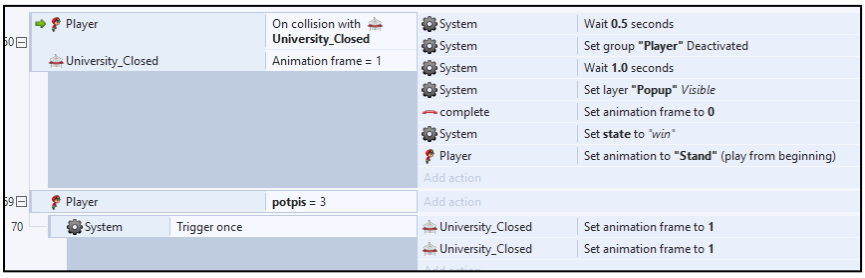
Slika 56. - Prikaz druge scene/razine



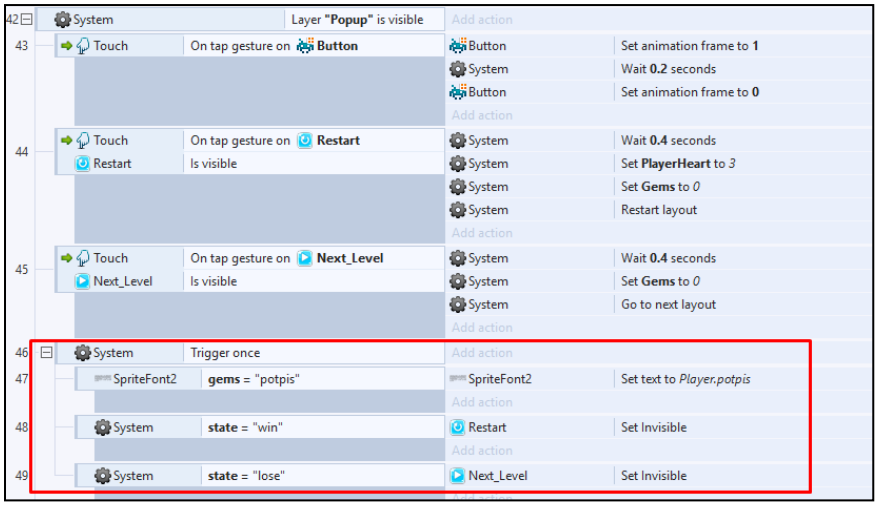
Slika 57. - Prikaz treće scene/razine

5.7. Prelazak razine

Krajnji cilj igrača, točnije korisnika, prelazak je igre, a to se ostvaruje prelaskom s razine na razinu. Svaka je razina specifična te posjeduje različitu strukturu platformi i neprijatelja. Važno je napomenuti, sa strane liste događaja, kako je proces prelaska sljedeći: kada igrač pokupi tri potpisa, sistem će pokrenuti akciju izmjene animacije na „vratima sveučilišta“ te boju zastavice iz crvene u zelenu što će rezultirati vizualnim podražajem otvaranja vrata. No s tehničke strane, igrač mora ostvariti koliziju s vratima kako bi pokrenuo događaj vezan za prelazak razine. Točnije, pri koliziji *System* će napraviti pauzu od 0,5 sekundi, zatim deaktivirati kretnje igrača i napraviti drugu pauzu od 1 sekunde. Zatim će se pojaviti *Popup layer* koji će nam u kombinaciji s varijablom *state=win* ponuditi opciju/tipku za sljedeći *level* (engl. *Next Level*). Ovaj proces vrijedi za svaku razinu, točnije svaka razina na kraju prikazuje „sveučilište“ čija vrata treba otključati kako bismo došli do kraja igre.



Slika 58. - Prikaz događaja vezanih za otključavanje vrata sveučilišta



Slika 59. - Prikaz okidača koji pomoću *plug-ina SpriteFont* ispisuje prikupljene potpise te pomoću globalne varijable *state* manipulira tipkama „Restart“ i „NextLevel“

5.8. Glavni izbornik

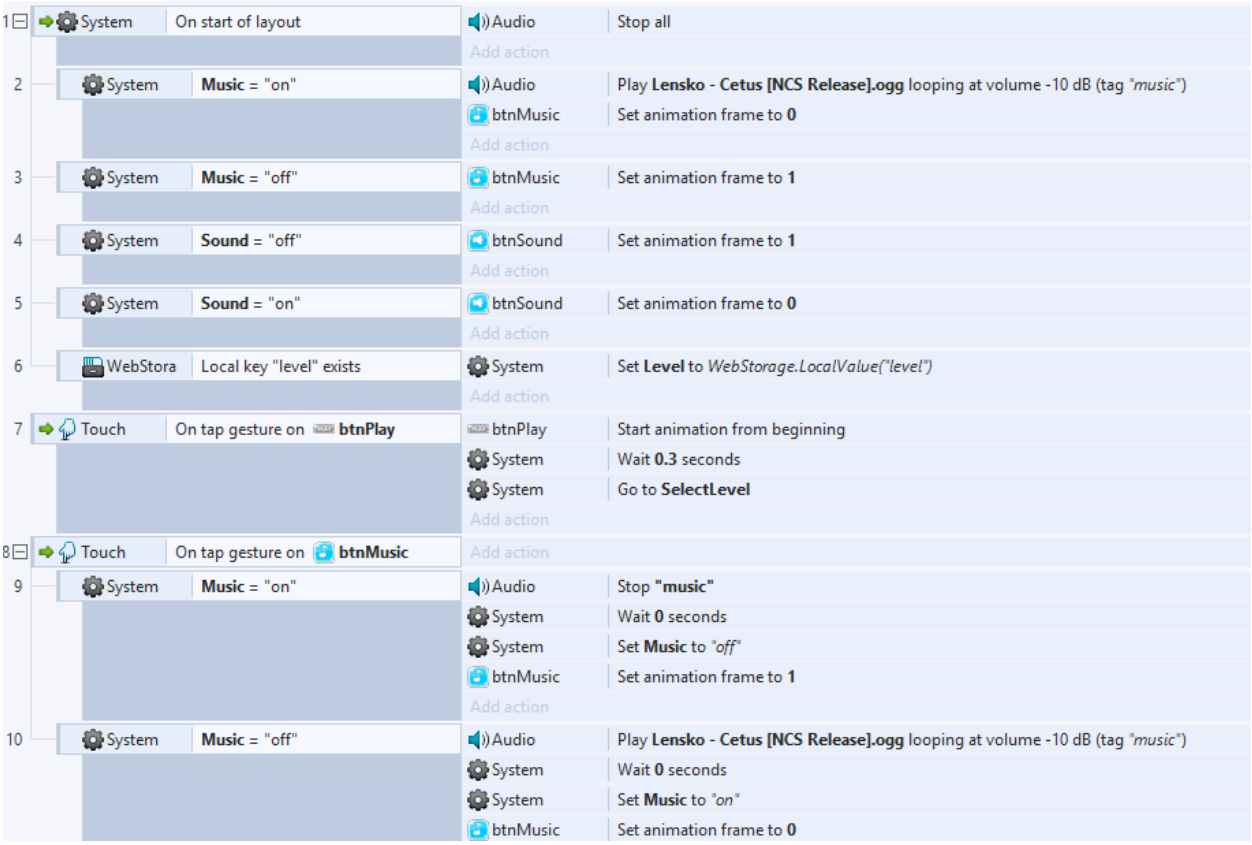
Posljednja faza igre svodi se na kreiranje glavnog izbornika. Glavni izbornik služi kako bi korisniku predstavio *home screen* igre te ga uveo u osnovne kontrole. Naravno, u bilo kojem trenutku igre pritiskom tipke *escape* ili klikom na *home button* otvara se glavni izbornik (Slika 59.).

Slika 60. - Prikaz glavnog izbornika igre

Najprije se kreiraju kontrole koje će korisnika uvesti u sami proces komunikacije s igračem. Zatim je kreiran gumb igraj (engl. *Play*) koji nas automatski povezuje s izbornikom *levela*.

Slika 61. - Prikaz događaja vezanih uz tipku *„Play“*

Pored toga, dodane su i opcije vezane za paljenje i gašenje glazbe, specijalnih zvučnih efekata te opcija za izlazak iz igre. Dodana je i ideja „intaraktivnih gumbova“ čija je referenca preuzeta sa stranice vezane uz interaktivne *buttone*[11]. Prilikom pritiska na bilo koju tipku, izvedena je malena vizualna potvrda animacije (izmjena animacijskih *frameova*). Preciznije, kada se „klikne“ na bilo koji *button*, boja će se *buttona* iz plave promijeniti u sivu te ukoliko ga ponovno upalimo, plava boja će se vratiti.



Slika 62. - Prikaz događaja vezanih za „vizualnu potvrdu animacije“

[11]TemplateNET,20+ Best Interactive Buttons for Web Developers [Online]. Dostupno na: https://www.template.net/design-templates/buttons/interactive-buttons/., [pristupljeno 20.rujna 2018.]

6. Poslovni plan, ciljna skupina i strategija monetizacije igre

Monetizacija je dobro znana kao neizostavna stavka koja se mora razmatrati već od početne faze konceptualizacije igre. Koliko god igra bila kvalitetna i zabavna, bez realnog i adekvatnog plana monetizacije, igra će teško privući širi krug korisnika. Kod procesa, fokus je na pronalasku logične, nenametljive i inovativne strategije načina monetizacije.

Prvi korak predstavlja identifikacija i razumijevanje ciljne skupine. Prilikom realizacije, potrebno je uzeti u obzir sljedeće faktore: geografski položaj, ekonomske faktore geografskih ciljanih skupina te također odnose dobnih skupina. Cijena igre te cijena dodatnih mogućnosti unutar igre trebala bi biti maksimalno optimizirana u odnosu s potrošačkim granicama ciljne grupe. Preniska ili previsoka cijena direktno će utjecati na potencijal monetizacije igre. Potrebno je i istražiti sve dostupne opcije distributera platformi za igru. *Online* distributeri najčešće uzimaju oko 30% prihoda, a ukoliko uzimaju manje, to za sobom povlači i manju probojnost na tržište. Odabir platforme i kanala distribucije igrat će značajnu ulogu u ukupnim prihodima igre.

Igra „Funky Student“ usmjerena je na mlađu dob (no ne nužno) te se svodi na strategiju besplatne igre (engl. *free to play method – F2P*), gdje bi igra u osnovi bila besplatna, no također s druge strane posjedovala bi opcije transakcija čime bi utjecala na profit čije su metode u daljnjem tekstu opisane. U F2P obliku monetizacije istovremeno se javljaju dva značajna pojma, a to su: retencija i konverzija. Retencija je pojam koji se odnosi na sposobnost igre da održi pažnju i interes igrača tijekom dužeg perioda, a konverzija predstavlja trenutak kada je igrač u tolikoj mjeri zadovoljan s atributima igre da odlučuje potrošiti novac unutar igre. Drugim riječima, neophodno je da igra drži pažnju igrača dovoljno dugo da bi se on konačno odlučio na trošak. Za ovakav tip igre, privlačni su grafički/vizualni dodaci te prilagodbe glavnog igrača jer pridonose kreativnom izražavanju kroz personalizaciju osobnosti pojedinog korisnika igre te stvaraju emotivnu vezu igre i korisnika. Radi se o takozvanim *skinovima*, točnije različitim odjevnim kombinacijama te različitim elementima i oblicima dizajna igrača koji bi svakako pridonijeli ukupnom vizualnom dojmu igre.

Kao primjer jedne kreativne tehnike monetizacije unutar igre *League Of Legends*, možemo vidjeti kako je za jednog specifičnog „heroja“ kreirano mnoštvo različitih stilova *skinova* s različitim tematikama, pritom prikazujući i usklađene vizualne animacije svakog pojedinog *skina* (Slika 62.).

Slika 63. - Prikaz monetizacije vođene kreativnom tehnikom *skinova* unutar igre *League Of Legends*

Da bi se uspješno implementirale kreativne tehnike monetizacije, potrebno je u igri identificirati ono što se u ekonomskoj teoriji naziva oskudnim resursima. Drugim riječima, specificirati resurs koji je posebno poželjan koji korisnik od početka igre nema dovoljno te resurs koji igrači žele u većoj količini od one u kojoj ga dobivaju. Tehnike monetizacije koje nastaju oko oskudnih resursa bile bi vođene metodom prelaska težeg *levela* gdje bi korisnik prilikom prelaska težeg *levela* dobivao minijaturnu dozu oskudnih resursa. Igrač u potrazi za ovim resursima imao bi samo dvije opcije: ili da konstantno igra igru i skuplja resurse sve dok ih konačno ne zasluži ili da putem transakcije kupi potrebnu količinu resursa. Zaključak je da se igraču nikada ne dopušta direktna kupnja traženog *skina,* nego da uvijek postoji još jedan korak prije, a upravo to je „oskudni resurs“ koji predstavlja uvjet kupovine.

7. Zaključak

Iz svega navedenog, može se zaključiti kako je proces izrade računalne igre uistinu kompleksan. Ovisno o potrebama i tipovima igre, potrebno je odabrati pravilnu hijerarhiju događaja, točnije rečeno, procesa kojih se treba pridržavati te ih maksimalno moguće optimizirati kako bi maksimalno smanjili utjecaj *bugova* i neželjenih akcija. Po uzoru na *old school* 2D igre, kreirani su objekti (glavni lik, neprijatelji, platforme, *gemovi*, potpisi) kojima su akcije, točnije događaji dodani unutar aplikacije Construct 2. Kada korisnik uspješno izvrši problematiku *levela* otvaraju mu se vrata sveučilišta. Upravo za ovaj dio igre, referenca je izvučena iz konteksta polaganja ispita. Iz navedenog, može se zaključiti da Construct 2, na zanimljiv i dinamičan način korisnika upoznaje s izradom korisničkog sučelja preko kojeg korisnik komunicira s igrom. Događaji upravljaju objektima, a objekti su sredstva pomoću kojih je napravljen sam kostur igre. Izrada pojedinog elementa igre postiže se korištenjem i kombiniranjem svih elemenata programskog paketa i kombiniranjem znanja objektno orijentiranog programiranja. Program je pristupačan i jednostavan za izradu, kao i sučelje koje je podložno raznim proširenjima/dodatcima s treće strane (engl. *third party plug-ins*) te plug-inovima programiranim u Javascriptu. Ubacivanjem objekata iz Adobe Illustratora dobiva se predložak igre s tri *levela* i korisničkim sučeljem. U današnje doba uistinu svatko može napraviti igru na svom osobnom računalu. Kako bi igra izgledala dobro i foto-realistično te ujedno posjedovala realistične simulacije potrebno je mnogo vremena i znanja u različitim područjima i programima. Trenutne su mane, točnije ograničenja Constructa 2 nemogućnost izrade bilo kakvih 3D igara te izrada i izvedba kolizije kompleksnijih 2D modela. Moguća je izvedba i polu-3D igre, no potrebna je dobra ideja i implementacija kako bi dobili ispravne odnose i dubinu *levela*. Kod kolizije kompleksnih modela problem predstavljaju animacije igrača, točnije za svaki *frame* animacije potrebno je kreirati vlastitu koliziju kako bi igrač posjedovao realistične granice kolizije. Problematika se pojavljuje kada se igrač pronađe u koliziji s objektom na sredini *loopa* određene animacije (npr. *walk*), gdje se pojavljuju čudni artefakti i *buggovi*. Jako se mora paziti i na veličine slikovnih objekata i količinu ostalih elemenata u sceni jer moguće je doći do smanjenja performansi. Može se zaključiti kako su se s razvojem igara razvili i dodatni programi kojima se izrađuju igre i koji

olakšavaju određene procese kreacije. Budući mehanizmi i *plug-inovi* trebali bi riješiti spomenute probleme kako bi sama stabilnost te mogućnosti rada s kompleksnijim modelima bile moguće. To bi u konačnici privuklo interes mnoštva drugih programera zainteresiranih za ovu vrlo specifičnu, no ujedno i zanimljivu temu. Interes bi sigurno postojao i kod mlađih generacija koje tek otkrivaju svijet programiranja pa bi time Construct 2 aplikacija bila odličan predstavnik u svijetu zanimljivog, ali i poučnog načina programiranja.

Literatura

[1] Construct 2 Manual, Scirra Ltd, 2015, https://www.scirra.com/manual/1/construct-2,(pristupljeno 17. rujna 2018.).

[2] Schell, J. (2008) The Art of Game Design, Amsterdam; Boston : Elsevier/Morgan Kaufmann Publishers (pristupljeno 8. rujna 2018.).

[3] Opis aplikacije Adobe Audition CC, https://www.hsm360.com/proizvod/adobe-audition-cc-3/, (pristupljeno 21. rujna 2018.).

[4] Glazba za specijalne efekte (kolizije, klikove, kraj levela…): http://www.freesfx.co.uk, (pristupljeno 1. rujna 2018.).

[5] Free Sound – 8 bit video game sound (pozadinska glazba): https://freesound.org/people/ProjectsU012/packs/18837/ , (pristupljeno 20. rujna 2018.).

[6] Općenito o Steam-u, https://store.steampowered.com/about/, (pristupljeno 20. rujna 2018.).

[7] Opis igre na Steamu -Airscape: The Fall of Gravity (izrađene unutar Construct 2), https://store.steampowered.com/app/317250/Airscape__The_Fall_of_Gravity/,(pristupljeno 22. rujna 2018.).

[8]Opis igre na Steamu - The Next Penelope (izrađene unutar Construct 2), https://store.steampowered.com/app/332250/The_Next_Penelope/, (pristupljeno 22. rujna 2018.).

[9] Construct 2 – Objekti, https://www.scirra.com/tutorials/803/construct-2-basics-objects,(pristupljeno 18. rujna 2018.).

[10] Stranica korištena za primjere pozadine igre, Freepik.com, https://www.freepik.com/index.php?goto=2&searchform=1&k=2d+game+background, (pristupljeno 10. rujna 2018.).

[11] 20+ Best Interactive Buttons for Web Developers(Referenca za interaktive buttone) https://www.template.net/design-templates/buttons/interactive-buttons/ , (pristupljeno 12. rujna 2018.).

Izvori slika:

Slika 1. - Super Mario, Dostupno na: https://amp.businessinsider.com/images/5571adb8eab8eacc63186f29-750-559.jpg

Slika 2. - Icy Tower, Dostupno na: https://icy-tower.en.aptoide.com/

Slika 3. - Flappy Birds, Dostupno na: http://wallpapers.ae/wp-content/uploads/2015/02/Flappy-Bird-Background-Image.png

Slika 4. - Angry Birds, Dostupno na: http://blog.mtel.ba/aplikacija-angry-birds-2/

Slika 5. - Airscape: The Fall of Gravity, Dostupno na: https://steamcdn-a.akamaihd.net/steam/apps/317250/ss_f7ad5dfba99ca631df2a78cb0e25a00b83739d88.116x65.jpg

Slika 6. - The Next Penelope, Dostupno na: https://steamcdn-a.akamaihd.net/steam/apps/332250/ss_8c0987f70ec2c27601c09c647c38048d2fbcd51a.116x65.jpg

Slika 19. - Primjer događaja bez korištenja familija, Dostupno na: https://www.scirra.com/images/articles/nofamilyevents.png

Slika 20. - Primjer događaja ukoliko koristimo familije, Dostupno na: https://www.scirra.com/images/articles/familyevent.png

Slika 24. - Razlika *Jump-thru* i *Solid* behaviora, Dostupno na: https://www.scirra.com/images/articles/jumpthru-solid.png

Slika 62. - Prikaz monetizacije vođene kreativnom tehnikom *skinova* unutar igre „League Of Legends“,Dostupno na: https://media.rankedboost.com/wp-content/uploads/2016/11/Elementalist-Lux.jpg

Popis slika:

Slika 1. - Super Mario 2
Slika 2. - Icy Tower 2
Slika 3. - Flappy Bird 2
Slika 4. – Angry Birds 2
Slika 5. - Airscape: The Fall of Gravity 4
Slika 6. – The Next Penelope 4
Slika 7: Proces kreiranja novog projekta 5
Slika 8. - Postavke veličine igraćeg ekrana 6
Slika 9. - Postavke veličine scene 7
Slika 10. - Grubi prikaz scena 7
Slika 11. - Grubi prikaz prozora sa slojevima 8
Slika 12. - Kreiranje nove liste događaja 9
Slika 13. - Grubi prikaz liste događaja 9
Slika 14. - Primjer jednog događaja 10
Slika 15. - Prikaz prozora s postojećim Construct 2 plug-inovima 12
Slika 16. - Primjer kreiranja instance variable. 12
Slika 17. - Prozor s postojećim ponašanjima unutar Construct 2. 13
Slika 18. - Prozor s postojećim efektima. 14
Slika 19. - Primjer događaja bez korištenja familija 15
Slika 20. - Primjer događaja ukoliko koristimo familije 15
Slika 21. - Prikaz pozadine igre kreirane u Adobe Illustratoru 16
Slika 22. - Prikaz pozadine unutar Constructa 17
Slika 23. - Prikaz procesa kreiranja Tiled Backgrounda 17

Slika 24. - Razlika *Jump-thru* i *Solid* behaviora 18
Slika 25. - Prikaz *Jump-thru* behaviora 18
Slika 26. - Prikaz kreacije igrača unutar Adobe Illustratora 19
Slika 27. - Prikaz animacija i kreiranja kolizije igrača 20
Slika 28. - Prikaz *behaviora* Platform 20
Slika 29. - Prikaz događaja i animacija vezanih za kretnju igrača 21
Slika 30. - Prikaz događaja i animacija vezanih uz kretnje neprijatelja 22
Slika 31. - Prikaz neprijatelja koji je vođen *Bullet* svojstvom 23
Slika 32. - Prikaz neprijatelja koji je animiran putem *Sine behaviora* 23
Slika 33. - Prikaz dvaju „FALSE“ kolizija između igrača i neprijatelja 24
Slika 34. - Prikaz „TRUE“ kolizije između igrača i neprijatelja u zraku 25
Slika 35. - Prikaz „TRUE“ kolizije između igrača i neprijatelja na tlu 25
Slika 36. - Prikaz svih globalnih varijabli unutar igre „Funky Student“ 26
Slika 37. - Prikaz globalne varijable *Music* 26
Slika 38. - Prikaz globalne varijable *Sound* 27
Slika 39. - Prikaz globalne varijable *state* 27
Slika 40. - Prikaz globalne varijable *PlayerHeart* u sudaru s neprijateljem 28
Slika 41. - Prikaz globalne varijable *Gems* prilikom kolizije s igračem 28
Slika 42. - Prikaz brojčanih varijabli koje su dodijeljene pojedinom *levelu* 29
Slika 43. - Prikaz događaja vezanih uz globalnu varijablu *Level* 29
Slika 44. - Prikaz događaja usporedbi vrijednosti, koji se javljaju na kraju scene 30
Slika 45. - Prikaz HUD-a 30
Slika 46. - Prikaz kolizije igrača s objektom „Potpis Key“ 31
Slika 47. - Prikaz animacijskih *frameova* Potpis Keya i varijable „potpis“ 31
Slika 48. - Prikaz sloja TouchControl 32

Slika 49. - Prikaz kontrola događaja vezanih uz *TouchControl* 32

Slika 50. - Prikaz sloja *Game Over* 33

Slika 51. - Prikaz sloja *Level Complete* 33

Slika 52. - Prikaz događaja *Game over* ukoliko igrač padne s platforme 33

Slika 53. - Prikaz događaja *Game over* ukoliko igrač posjeduje manje ili jednako od nula života 33

Slika 54. - Prikaz uspješno naslijeđenog „*HUD*“ *layera* 34

Slika 55. - Prikaz prve scene/razine 35

Slika 56. - Prikaz druge scene/razine 35

Slika 57. - Prikaz treće scene/razine 35

Slika 58. - Prikaz događaja vezanih za otključavanje vrata sveučilišta 36

Slika 59. - Prikaz okidača koji pomoću plug-ina *Sprite Fon*t ispisuje prikupljene potpise te pomoću globalne varijable state manipulira tipkama *Restart* i *NextLeve*l 36

Slika 60. - Prikaz glavnog izbornika igre 37

Slika 61. - Prikaz događaja vezanih uz tipku *Play* 37

Slika 62. - Prikaz događaja vezanih za „vizualnu potvrdu animacije“ 38

Slika 63. - Prikaz monetizacije vođene kreativnom tehnikom *skinova* unutar igre *League Of Legends* 42

Sažetak

Priprema procesa programiranja računalnih igara predstavlja veliku problematiku i izazov jer za kreirati jedan konačan proizvod potrebno je mnogo više znanja, iskustva i vještina nego što bi se reklo. Optimizacija je igre ključna točka kako bi uspjeli GUI grafičko korisničko sučelje (engl. *graphical user interface*) povezati s tehničkom (programerskom) stranom igre. U slučaju Construct 2, sam proces vrlo je jednostavan te ne zahtjeva klasično programiranje. Metodom pritisni–povuci (engl. *drag and drop*) vizualne materijale unosimo u našu igru, a princip programiranja sveden je na događaje, pomoću kojih unesenim materijalima dodjeljujemo akcije putem odabira grafičkih ikona. Aplikacija nam u bilo kojem trenutku omogućava interaktivni pregled odrađenog dijela igre kako bi uvijek mogli vidjeti kako nam igra stvarno reagira na određene akcije, uvjete i događaje. Kao završnu točku, uz dobru organizaciju i razumijevanje mogućnosti koje pruža aplikacijski paket Construct 2, moguće je odraditi bilo kakav zamišljeni 2D projekt, i to na vrlo pregledan proceduralan način (u bilo kojem trenutku, unutar koda moguće je promijeniti hijerarhiju događaja ili npr. izmijeniti objekt koji predstavlja temelj događaja).

Ključne riječi: Izrada računalnih igara, Construct 2, optimizacija igre, GUI, interaktivni pregled, 2D projekt, proceduralan način

Summary

The process and the preparation of programming a computer game presents a big issue and challenge because creating a single product requires much more knowledge, experience, and skill than you would think. Game optimization is a key point to connect a graphical user interface (GUI) with a technical (programmer) side of the game. In the case of using Construct 2, the process itself is very simple and does not require classical programming. Using the drag and drop method, we can implement visual material into our game, and the principle of "programming" is reduced to the events option, which helps us to assign actions through selected graphics icons. Construct 2 application allows us to use interactively preview of the game - so we could always see how the game actually responds to certain actions, conditions, and events. As a final point, with a good organization and understanding of the capabilities provided by the Construct application package, it is possible to do any conceivable 2D project, in a very thoroughly procedural way (at any time, within the code, it is possible to change the hierarchy of events or eg change the object which is the „basic“ of the event tree).

Key words: Creation of the Computer Games, Construct 2, Game Optimization, GUI, Interactive Review, 2D Project, Procedural Mode