SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4363

# Izrada igre upotrebom WebGL-a

Mladen Gelemanović

Zagreb, lipanj 2016.

# Sadržaj

# Uvod

Razvoj video igara jedno je od najzanimljivijih područja računarstva, koje koristi znanja iz mnogih disciplina, kao što su matematika, fizika, umjetnost, povijest i psihologija, da stvori program koji služi kao izvor zabave korisniku. Igre se najbolje može opisati kao generatore iskustava [ 1 ].

Kako su ljudi vizualna bića, ne iznenađuje da je značajan dio šarma igara baš u grafičkom prikazu. Danas većina igara koristi jedan od dva (OpenGL, DirectX) popularna programska sučelja za korištenje grafičke kartice i iscrtavanje na ekran. Igre napravljene sa ovim tehnologijama mogu se pokretati na računalima ili na konzolama. Međutim, postoji i velika potražnja za igranjem igara preko Interneta. Iako su postojali načini za razvoj igara koje bi se pokretale na internetskim stranicama (Flash), takvi proizvodi bili su često ograničeni u funkcionalnostima i brzini izvođenja. Zato je 2011. objavljena prva verzija WebGL-a, programskog sučelja koje omogućava iskorištavanje snage grafičke kartice za brzi 2D i 3D prikaz na stranicama.

Cilj ovoga rada bio je upoznavanje sa programskim sučeljem WebGL i pokušaj izrade jednostavne 2D platformerske igre sa njime. Za izradu ove igre, autor je kao inspiraciju imao klasične igre kao što su Super Mario Bros. (Nintendo, 1985), Sonic the Hedgehog (Sega, 1991) i Mega Man X (Capcom, 1993).

# 1. Korištene tehnologije

## 1.1. Programsko sučelje WebGL

WebGL (eng. *Web Graphics Library*) je JavaScript API niske razine za brzi prikaz interaktivne 2D i 3D računalne grafike na web stranicama. Njegove karakteristike su neovisnost o operacijskom sustavu, podrška u većini modernih preglednika i omogućavanje 3D prikaza bez dodataka za preglednik (eng. *plugin*). WebGL dizajnira i održava neprofitna udruga Khronos Group.

Programsko sučelje WebGL-a bazirano je na OpenGL ES 2.0, grafičkom sučelju niske razine namijenjenom za ugrađene sustave. WebGL programi sastoje se od kontrolnog koda napisanog u JavaScriptu i malih programa koji se izvode direktno na grafičkoj kartici, napisani u jeziku GLSL (eng. *Graphics Library Shader Language*).

WebGL kontekstu pristupa se preko posebnog HTML5 elementa <canvas>. Zbog toga je prikaz dobiven WebGL-om moguće slobodno kombinirati s ostalim elementima internetskih stranica.

## 1.2. Programski jezik JavaScript

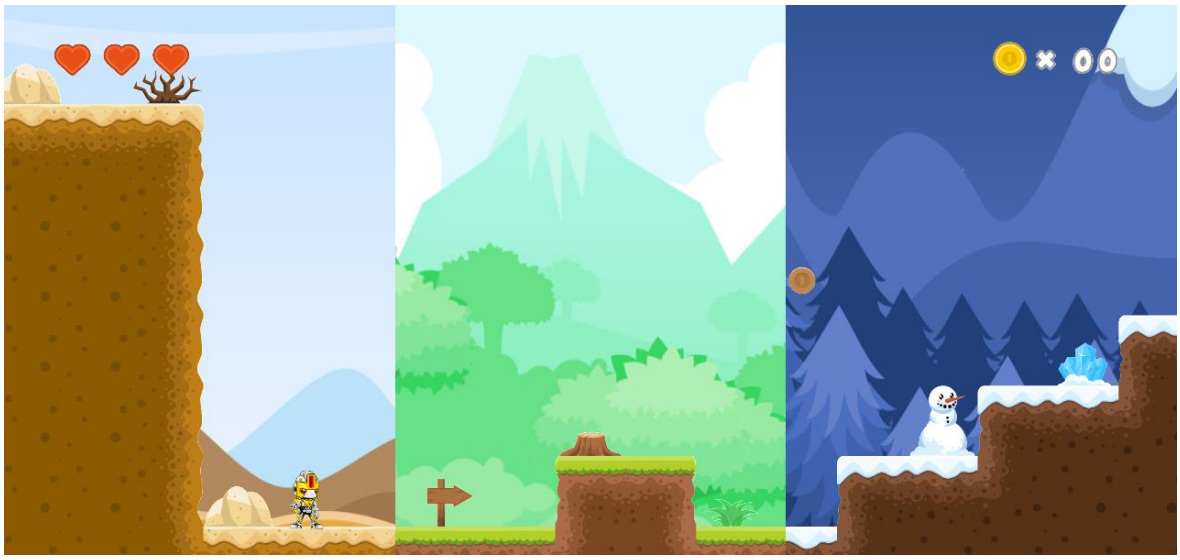
JavaScript je interpretirani dinamički programski jezik visoke razine bez tipiziranja. Jezik je baziran na prototipovima s funkcijama kao objektima, što ga čini jezikom s više programskih paradigmi, podržavajući objektno orijentirani, imperativni i funkcijski programski stil. Uz HTML i CSS čini osnovu tehnologija za razvoj sadržaja na internetskim stranicama.

## 1.3. Zapis JSON

JSON (eng. *JavaScript Object Notation*) je otvoreni format proizašao iz jezika JavaScript, koji koristi ljudski razumljivi tekst za prijenos podataka sastojeći se od parova atribut i vrijednost. To je najčešći format podataka kod asinkrone komunikacije između klijenta i servera i neovisan je o programskom jeziku.

# 2. Struktura programa

Rad razvijenog programa može se podijeliti na dvije faze. U prvoj fazi, koja započinje učitavanjem stranice, instancira se vršni objekt igre i prikazuje se početni izbornik. Vršni objekt tijekom stvaranja mora inicijalizirati grafički kontekst (WebGL i programe za sjenčanje), učitati potrebne teksture i stvoriti ostale objekte nužne za ispravan rad igre. Po završetku instanciranja, igra još nije pokrenuta. U izborniku je moguće promijeniti neke od korištenih tekstura (Slika 1) i pokrenuti igru.



*Slika 1. Različita okruženja u igri*

U drugoj fazi programa, igra je pokrenuta. Nakon odabira nove igre učitava se početna scena i početni izbornik nestaje. Nakon učitavanja scene započinje glavna petlja programa (eng. *game loop*). Preglednik pokušava izvoditi petlju brzinom od 60 prolazaka u sekundi. U svakom prolasku petlje (eng. *frame*) izvršavaju se tri radnje: obrada korisnikovog unosa, osvježavanje stanja igre i iscrtavanje.

Postoje dva načina rada programa. Prvi način je igra (Isječak koda 1), gdje se korisnikove akcije pretvaraju u akcije igraćeg lika, objekti se gibaju pod utjecajem fizikalnih zakona te se konačno iscrtava scena, igrač i grafičko sučelje.

```
loop: function () {                     // Standard game loop
    this.inputManager.handleInput();    // Handle player input

    this.player.update();               // Update player
    this.scene.update();                // Update scene
    this.scene.camera.followPlayer();

    this.scene.render();                // Render game world
    this.player.render();               // Render player
    this.hud.render();                  // Render in-game HUD
}
```

*Isječak koda 1. Prolaz petlje u načinu igre*

Drugi način rada je uređivač scena (Isječak koda 2), gdje se unos s tipkovnice koristi za pomicanje kamere i mijenjanje opcija uređivača, ne računa se fizika i iscrtava se samo scena i trenutno korišteni objekt. Mijenjanje načina rada programa ostvaruje se preko izbornika u igri. Izvođenje programa privremeno se zaustavlja dok su izbornici aktivni.

```
loop: function () {                     // Editor game loop
    this.handleInput();                 // Handle editor input

    game.scene.render();                // Render game world
    if (this.selectOn)
        this.drawObjectSelection();     // Render all objects
    this.drawUsedObject();              // Render currently used object
}
```

*Isječak koda 2. Prolaz petlje u načinu rada uređivača*

# 3. Grafičke komponente

Prvi korak u izradi video igre je pisanje koda za iscrtavanje objekata na ekran. Obzirom da ovaj rad koristi WebGL za crtanje, prvo je potrebno dobiti WebGL kontekst. Pozivom funkcije *getContext* nad HTML5 elementom <canvas> s argumentom „*webgl*“ ili „*experimental-webgl*“, traži se od preglednika WebGL kontekst za crtanje. Sljedeće je potrebno definirati vidljivi dio prozora (eng. *viewport*), inicijalizirati programe za sjenčanje, skupove podataka o vrhovima i teksturama (eng. *vertex and texture buffer*) te uključiti stapanje tekstura (eng. *blending*).

Rad je napravljen s dva minimalna skupa podataka (eng. *buffer*). U prvom skupu podataka nalazi se 12 vrijednosti koje definiraju vrhove četverokuta sa središtem u (0, 0) i duljinama stranica od 1 m. Drugi skup podataka sadrži 8 vrijednosti koje preslikavaju čitave teksture na vrhove objekta.

Svaki objekt u igri sadrži informacije o vlastitoj poziciji četverokuta i korištenu teksturu. Prilikom crtanja na ekran, vrhovi četverokuta se translatiraju na odgovarajuće mjesto na sceni te se trenutno aktivna tekstura postavlja na teksturu trenutno iscrtavanog objekta. Nakon postavljanja, podaci se šalju grafičkoj kartici na iscrtavanje. Zbog ovakvog pristupa iscrtavanju, objekti se crtaju pojedinačno, jedan za drugim, što nije optimalni način iscrtavanja jer se za jednu sličicu scene podaci šalju na crtanje u prosjeku 50 puta, umjesto da se svi potrebni podaci pošalju u jednom koraku. Međutim, ovakav pristup je izabran zbog jednostavnosti scena i malog broja objekata koji se mogu nalaziti na ekranu u svakom trenutku. Također, crtanje objekata je stoga iznimno jednostavno za programera, koji mora brinuti o samo tri stvari: poziciji, veličini i teksturi objekta za trenutni prolazak petlje.

## 3.1. Programi za sjenčanje

Kako je WebGL baziran na OpenGL 2.0 ES tehnologiji, za ispravno iscrtavanje potrebno je napisati i programe za sjenčanje (eng. *shader*), male programe koji se izvode direktno na grafičkoj kartici. U ovom radu korišteni su jednostavni program za vrhove (eng. *vertex shader*) i fragmente (eng. *fragment*

*shader*). Korišteni program za vrhove radi transformaciju pogleda i šalje podatke o teksturama programu za fragmente koji boja piksele obzirom na predanu teksturu.

Pri svakom pokretanju igre, programi za sjenčanje moraju se dinamički prevesti i povezati u jedan program koji WebGL onda može koristiti.

## 3.2. Teksture

U igrama, teksture su jedan od najvažnijih čimbenika kako igrač doživljava igru. To je posebno važno u 2D igrama, gdje nema potrebe za iznimno detaljnim modelima, već se svi objekti sastoje od teksturiranih površina. Drugim riječima, teksture određuju izgled igre. Za ovaj rad korištene su besplatne teksture dostupne sa Interneta [ 2 ][ 3 ].

*Slika 2. Jedna od slika korištena za spremanje tekstura igrača*

Prvi problem koji se javlja kod korištenja tekstura je učitavanje slika. Na sreću, HTML5 ima tu funkcionalnost već ugrađenu i kvalitetno napravljenu, što smanjuje opterećenje na programera i omogućava bržu izradu prototipova. Drugo pitanje koje se javlja je organizacija korištenih tekstura. Naime, ovaj rad koristi preko 70 različitih sličica za teksture i bilo bi iznimno neučinkovito kad bi program morao učitavati toliko velik broj različitih datoteka. Kao rješenje, korištene su veće slike (eng. *spritesheet*) (Slika 2) koje se sastoje od više srodnih tekstura (eng. *sprite*). Sada, nakon

učitavanja slike, potrebno je odvojiti pojedinačne teksture i spremiti ih u odgovarajuće strukture podataka za daljnje korištenje.

## 3.3. Animacije

Animacije su izvedene mijenjanjem trenutno aktivne teksture objekta. Za pravilan rad potrebno je znati polje tekstura koje se želi koristiti u animaciji i broj prolazaka petlje igre nakon kojeg se ciklično mijenja indeks korištene teksture. Podršku za animiranje ima svaki objekt koji se može mijenjati. Većina animacija koristi jedno animacijsko stanje, dok se animacije glavnog lika mogu podijeliti u tri različita stanja: mirno, trčanje (Slika 3) i skakanje. Za određivanje animacijskog stanja zadužen je objekt koji se animira.

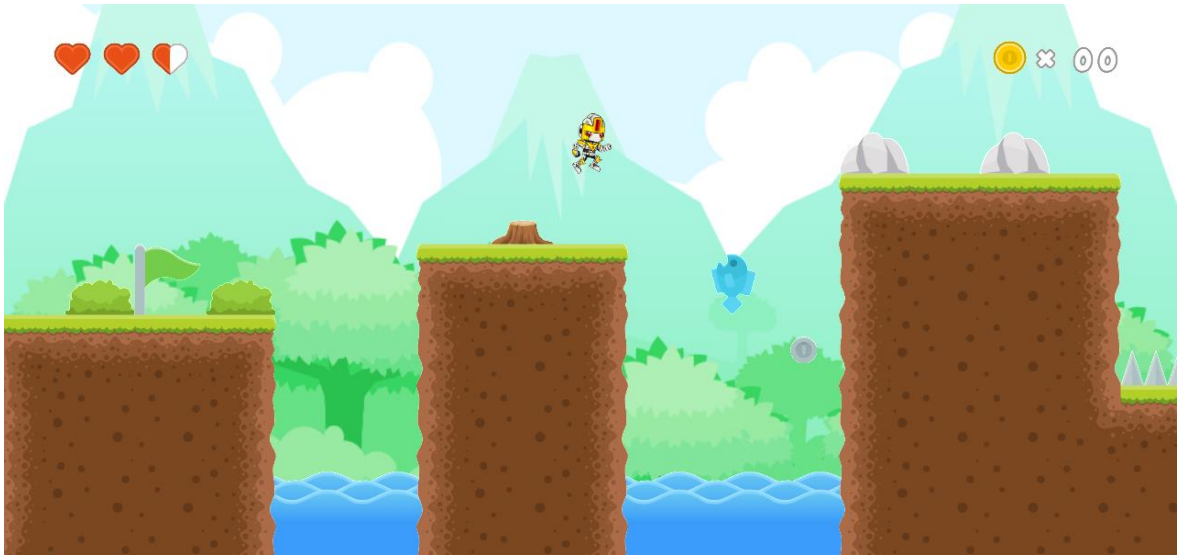
*Slika 3. Sličice korištene u animaciji trčanja igrača*

# 4. Fizika

Sustav fizike u igri sastoji se isključivo od primjene gravitacije na određene objekte i otkrivanje sudara između dva objekta [ 4 ].

Iako je igra u početku izrade podržavala definiranje mase objekta i primjene sila radi realističnog ponašanja, to je odbačeno u korist jednostavnijeg modela koji se pokazao znatno ugodniji za igrati. Napravljena igra je platformer, žanr igara u kojem se očekuje precizna kontrola lika radi prelaženja pojedine razine. Silama upravljani model pokazao se nezgrapan i težak za upravljati. Očuvanje momenta pokazalo se iznimno problematičnim, gdje igrač nije bio u stanju stati točno kad je on htio, nego trenutak ili dva kasnije, što je često rezultiralo neočekivanim padom u smrt i frustracijom.

Korišteni model, iako jednostavan i ograničen u mogućnostima, pokazao se kao iznimno dobro rješenje. Igrač uvijek skače u vis jednako visoko (Slika 4), što omogućava bolje planiranje sljedećeg poteza. Kretanje igrača je preciznije, s trenutnim prelaskom iz stanja trčanja u stanje mirovanja. Na kraju, otvorile su se mogućnosti za novim načinima kretanja lika, kao što je dupli skok i odbijanje od drugih objekata.



*Slika 4. Igrač i neprijatelj pod utjecajem gravitacije*

## 4.1. Gravitacija

Fizikalni model sastoji se samo od gravitacije i komponente brzine u x i y smjeru. U svakom trenutku igre, svi objekti koji se mogu pokretati izračunavaju novu vrijednost vertikalne brzine, po formuli *( 1 )*, dok se horizontalna brzina mijenja ovisno o interakcijama unutar igre. Nakon brzine, računa se nova pozicija objekta po formuli *( 2 )*. Vremenski korak ($\Delta t$) je razlika vremena između dva prolaska petlje igre, s ograničenjem na 30 ms.

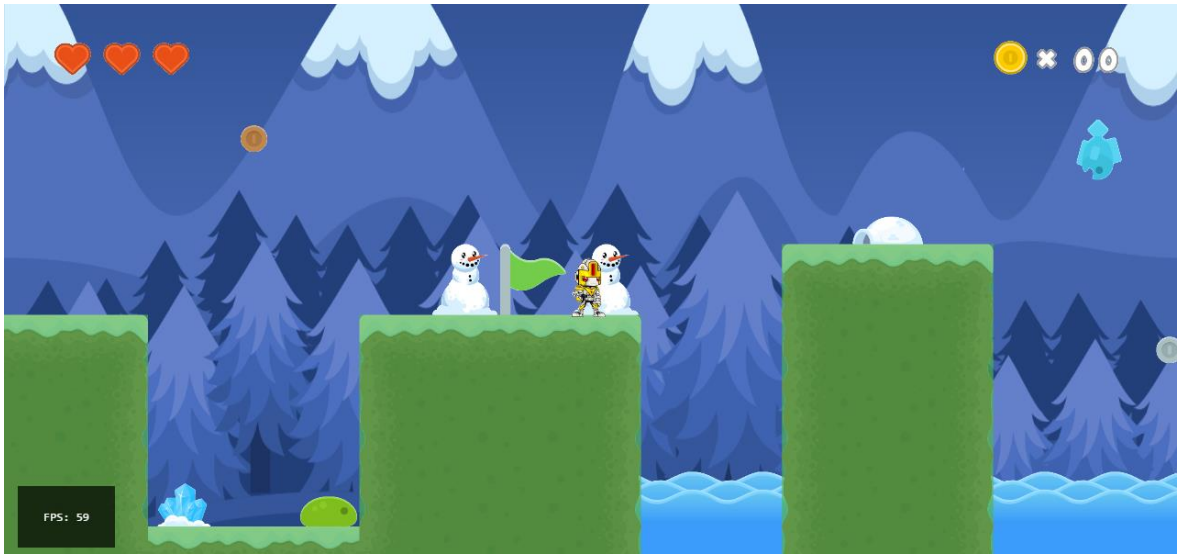
$$v_{novo} = v_{staro} - g \cdot \Delta t \qquad (1)$$

$$(x, y)_{novo} = (x, y)_{staro} + v(x, y) \cdot \Delta t \qquad (2)$$

Ovo ograničenje nametnuto je radi sprječavanja prevelikih pomaka koji bi mogli smjestiti jedan objekt unutar drugoga, pritom zaobilazeći provjere sudara.

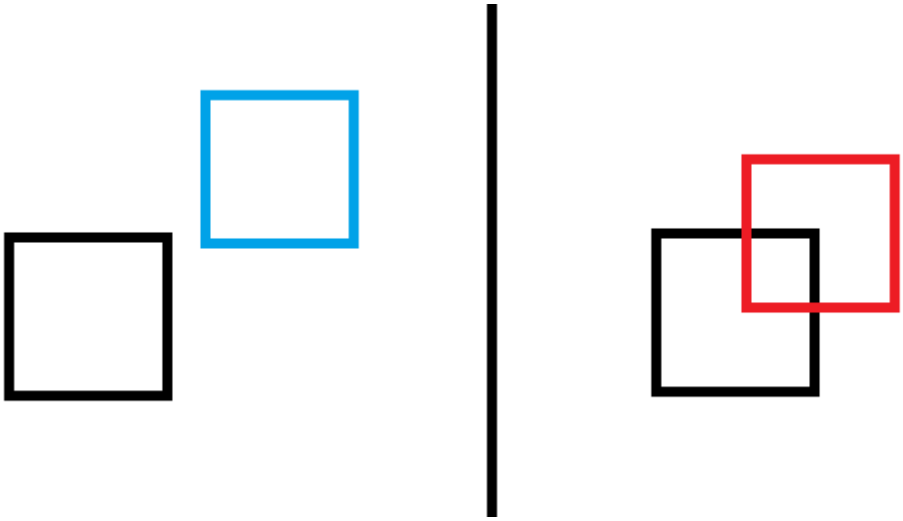
## 4.2. Detekcija sudara

Sva interakcija između objekata na sceni bazirana je na otkrivanju sudara (Slika 5). U slučaju kad se dva objekta sudare, započinje obrada sudara koja koristi dva podatka za rad: tipove objekata u sudaru (želimo da se različiti objekti različito ponašaju) i smjer sudara (želimo drugačije ponašanje ovisno o poziciji objekata).




*Slika 5. Vidljivi sudarači poda*

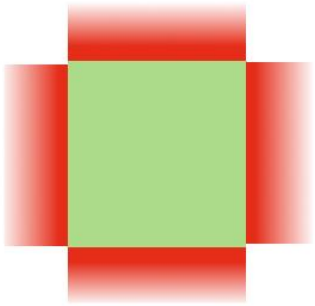
Prije obrade sudara, potrebno je ustanoviti koji se objekti uopće dodiruju. Kako se ne bi morao svaki objekt provjeravati sa svakim drugim objektom na sceni, provjera se izvodi samo na igraćem liku i nekolicini vrsti neprijatelja. Druga optimizacija napravljena je kod odabira objekata koji su potencijalno u dodiru s trenutno promatranim objektom. Naime, u obzir se uzimaju samo objekti koji su udaljeni od promatranog objekta za manje od 1 m. Razlog ovakvog odabira je u činjenici da nijedan objekt u igri nema dužine stranica veće od 1 m, pa sa sigurnošću možemo zaključiti da su objekti međusobno predaleko da bi uopće došlo do sudara.



*Slika 6. Postoji razmak između objekata, nema sudara (lijevo);*
*Nema razmaka između objekata, postoji sudar (desno)*

Kako su svi objekti u igri napravljeni od četverokuta i ne dopušta se njihova rotacija, za provjeru sudara odabran je AABB (eng. *Axis Aligned Bounding Box*) algoritam. Na svaki objekt dodaje se sudarač (eng. *collider*), kojem je moguće dodatno mijenjati poziciju i veličinu, neovisno od stvarnog položaja i veličine objekta. Algoritam je iznimno jednostavan i sastoji se od jedne provjere koja određuje da li se dva kvadrata poravnata s koordinatnim osima preklapaju. Određivanje se izvodi sa dokazivanjem da ne postoji razmak između objekata s nijedne strane (Slika 6). Ako se ustanovi sudar, sljedeći korak je određivanje smjera iz kojega je sudar nastao.

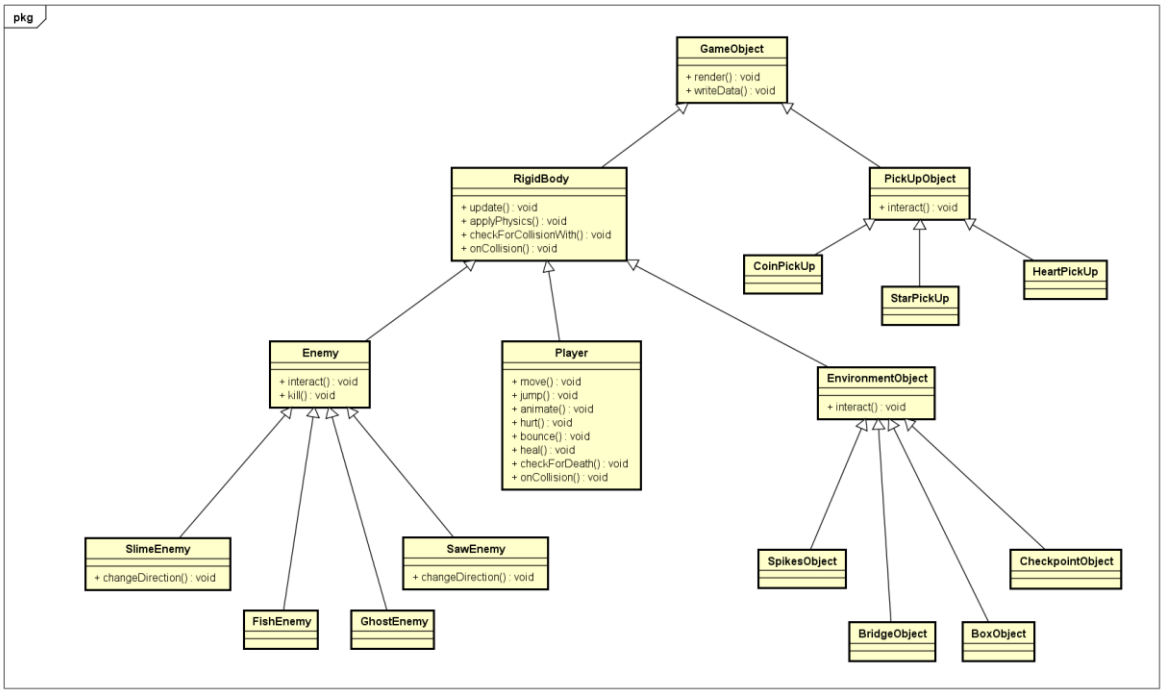
Smjer se određuje temeljem položaja centra objekta (Slika 7). Zbog odabranih područja za određivanje smjera, postoji greška u izvođenju (eng. *bug*) kada se sudar događa na samome vrhu objekta. U tom slučaju, iako je prvotno utvrđen sudar, kasnijim odabirom smjera algoritam javlja da nije bilo sudara. Za posljedicu, u rijetkim situacijama kad se točno pogodi vrh, jedan objekt (najčešće igrač) će jednostavno propasti u drugi objekt. U tom slučaju, dovoljno je samo skočiti da se izađe iz zaglavljenog objekta.



*Slika 7. Korištena područja za određivanje smjera sudara*

Interakcije koje se odvijaju između različitih objekata bit će detaljno razrađene u kasnijim poglavljima o samim vrstama objekata koji se mogu pronaći u igri.

# 5. Elementi igre




*Slika 8. Dijagram razreda objekata u igri*

## 5.1. Stvaranje objekata

Prije detaljnog opisa raznih vrsta objekata koji se koriste u igri (Slika 8), potrebno je objasniti kako se novi objekti dodaju u program. Za početak, potreban je način razlikovanja različitih objekata bez velikog broja provjera. U tu svrhu napravljen je sustav oznaka. Svaki objekt ima vlastitu oznaku i svi objekti iz iste porodice objekata koriste istu oznaku, specifičnu za tu vrstu. S time na umu napravljen je pomoćni objekt u programu, nazvan Creator, koji zna za svaku vrstu objekata kako se instanciraju objekti te vrste, u koje polje objekata se dodaju i kako se objekti te vrste pojavljuju u popisu svih blokova u uređivaču scena. Ovo je napravljeno uz mogućnost jezika JavaScript da se objektima nakon instanciranja dodaju novi atributi. Stoga, uvođenje nove vrste objekata potrebno je obaviti samo na jednom mjestu u kodu. Treba napomenuti da uvođenje novih tekstura nije podržano na ovaj način, odnosno potrebno je mijenjati postojeći kod.

## 5.2. Osnovni objekti

Svi objekti u igri nasljeđuju ovaj vršni razred objekata. Instancirani objekti sadrže osnovne informacije o oznaci, položaju, udaljenosti od kamere, veličini, teksturi i sudaraču. Pružena funkcionalnost, potrebna za sve objekte u igri, sastoji se od crtanja na ekran i zapisa podataka u JSON format, koji se koristi kod stvaranja i spremanja scena.

*Slika 9. Korištene teksture za pod i ukrase*

Postoje dva tipa objekata koji se stvaraju kao primjerci ovoga razreda (Slika 9). Prvi tip objekata, kojima je potrebno jedino iscrtavanje, su ukrasni objekti bez ikakvog utjecaja na igru. Osim ukrasa, ovaj razred stvara i blokove po kojima igrač može hodati. Karakterističnost oba tipa je statičnost, odnosno stvoreni objekti nisu utjecani fizikom i nije ih moguće animirati. Razlika između njih je u korištenju dodijeljenih sudarača.

## 5.3. Dinamički objekti

Ovaj razred uvodi objekte koji se mogu pokretati i mijenjati tijekom izvođenja igre. Nove funkcionalnosti su brzina objekta, mogućnost osvježavanja stanja, primjena fizike te provjera i obrada sudara s drugim objektima.

Osvježavanje stanja (eng. *update*) poziva se jednom za svaki vidljivi objekt tijekom jednog iscrtavanja scene i sastoji se od primjene gravitacije na objekt, pomicanje objekta ovisno o brzini i provjere sudara s podom. Ukoliko se ustanovi sudar, objekt se pomiče na novu poziciju u kojoj više neće biti u sudaru. Dodatno, objekt se uzemljava (dopušta se skakanje) ako je ustanovljen smjer sudara od gore

(objekt se nalazi na podu), dok se vertikalna brzina postavlja na početnu vrijednost od 0 m/s (trenutni skok se prekida) kada je smjer sudara od dolje (objekt je lupio u donju stranu poda).

Iako bitan zbog dodane funkcionalnosti u igru, ovaj razred nema vlastite primjerke stvorenih objekata, već služi kao sučelje za sve druge tipove interaktivnih objekata. Interaktivni objekti mogu se podijeliti u 4 kategorije: objekti koje igrač sakuplja, objekti okoline, neprijatelji i igrač.

## 5.4. Objekti za sakupljanje

U igri se nalazi nekoliko vrsta objekata koje igrač može pokupiti (Slika 10). Svi takvi objekti nasljeđuju vršni razred sakupljivih objekata, koji implementira funkcionalnost sakupljanja. Jednom kad igrač pokupi nešto, taj objekt se mora prvo maknuti sa scene i zatim staviti u polje maknutih objekata. Na objekte koje igrač sakuplja ne djeluje gravitacija, sa ciljem postavljanja nagrada za igrača na teže dohvativa mjesta.

*Slika 10. Objekti koje igrač može pokupiti*

Postoje tri vrste objekata koji se mogu naći u igri i svaki od njih definira vlastitu funkcionalnost nakon što se objekt pokupi. Najčešći objekti koje će igrač nalaziti su novčići. Nakon sakupljanja, igračev rezultat (eng. *game score*) povećava se za vrijednost pokupljenog novčića. Postoji nekoliko varijanti novčića s različitim vrijednostima: brončani (1), srebrni (2), zlatni (5) i dijamant (15). Nakon 100 sakupljenih novčića, povećava se maksimalni broj života igrača.

Sljedeći objekti koji se mogu pronaći na sceni su srca. Ovi objekti vraćaju jedan život igraču, pod uvjetom da je igrač uopće ranjen. Ukoliko igrač nije ranjen, srce se neće niti pokupiti.

Konačno, postoji objekt zvijezde, koji označava kraj trenutne scene. Sakupljanjem zvijezde, igra se pauzira, igraču se javlja poruka o uspjehu i započinje učitavanje sljedeće scene.

## 5.5. Objekti okoline

Pod objekte okoline smatraju se svi objekti koje igrač ne može pokupiti ili koji se pokreću tek nakon interakcije s igračem (Slika 11). Ovakvi objekti mogu biti i statični i dinamički, ovisno o konkretnom objektu. Namjena ovakvog razreda je popunjavanje scene dodatnim izazovima ili pomaganjem igraču na neki način. Pod objekte okoline spadaju šiljci, zastavice, mostovi i kutije, i sa svim vrstama je omogućena interakcija.

Najjednostavniji objekti skupine su šiljci, statični predmeti koji će ozljediti igrača, pod uvjetom da padne na njih. Inače, igrač smije slobodno prolaziti pokraj njih.

Zastavice služe za postavljanje nove lokacije za oživljavanje igrača (eng. *respawn point*) kad padne sa scene. Prije aktiviranja, zastavica je spuštena. Nakon aktiviranja započinje animacija dignute zastavice i postavlja se novo mjesto oživljavanja. Igrač može zastavicu aktivirati samo jedanput.

Most se ponaša kao normalan objekt poda, dokle god igrač ne stane na njega. Jednu sekundu nakon što igrač dotakne most, on će početi padati. Igrač je i dalje sposoban stajati na padajućem mostu, međutim to će rezultirati padom sa scene i gubitkom jednog života. Mostovi se ne vraćaju na originalno mjesto nakon što padnu.

*Slika 11. Objekti okoline*

Zadnja vrsta objekata u ovoj skupini su kutije iznenađenja. Kutije se isto ponašaju kao pod, međutim ako igrač lupi kutiju od ispod, ona će nestati i na njenom

mjestu pojavit će se novi objekt. Pojavljeni objekt nasumično se bira sa sljedećim vjerojatnostima: srebrni novčić (40%), srce (30%), duh (25%), dijamant (5%).

## 5.6. Neprijatelji

U igri postoji i razred objekata koji predstavljaju opasnost za igrača (Slika 12). Karakteristika tih objekata je kretanje neovisno od akcija igrača. Neprijatelji moraju implementirati vlastito kretanje, interakciju s igračem, originalnu poziciju na sceni i ponašanje u slučaju pada sa scene. Trenutno je implementirano 4 različite vrste neprijatelja: ljigavci, pile, ribe i duhovi. Sve vrste će ozlijediti igrača za pola života.

Ljigavci su prva vrsta neprijatelja koja je bila dodana u igru. Oni se pomiču horizontalno, odbijajući se od zidova, igrača i drugih ljigavaca. Ljigavci će pasti sa scene ako naiđu na rub poda. Ako igrač skoči na njih, ljigavac će se spljoštiti i prestati micati. Nakon toga ljigavac ne nestaje sa scene, već služi kao trampolin za igrača.

Tip neprijatelja sličan ljigavcima su pile. One se isto pomiču horizontalno i odbijaju od zidova, međutim pile neće pasti s ruba nego će promijeniti smjer. Također, pile su brže od ljigavaca i nije ih moguće onesposobiti. Igrač će lagano odskočiti ako pokuša skočiti na pilu, no i dalje će biti normalno ozlijeđen.

*Slika 12. Teksture neprijatelja koji se mogu pronaći u igri*

Ribe su vertikalna vrsta neprijatelja koja periodično skače u vis. Nakon pada sa scene, riba čeka 1.5 s prije novog skoka. Svaka riba skače na istu visinu, osim kada igrač skoči na nju. U tom slučaju, igrač će lagano odskočiti, skok od ribe bit će

prekinut i riba će početi padati natrag. Treba napomenuti da se riba tada ne miče sa scene, nego normalno ponavlja čitav skok.

Zadnja vrsta neprijatelja koji se mogu pronaći u igri su duhovi. Duhovi su posebni objekti u igri jer ne rade provjeru sudara s podom i nisu pokretani fizikom. Umjesto toga, duhovi se kreću po formulama *( 3 )* i *( 4 )*.

$$x_{novo} = x_{staro} + A \cdot \cos(korak / 60) \qquad (3)$$

$$y_{novo} = y_{staro} + B \cdot \sin(2 \cdot korak / 60) \qquad (4)$$

Vrijednosti *A* i *B* odabiru se nasumično prilikom svakog instanciranja objekta duha. Ovi objekti sadrže dodatni brojač koraka koji se povećava pri svakom pomaku duha. Duhovi se ne mogu onesposobiti i ozljedit će igrača pri dodiru.

# 6. Igrač

Najvažniji element svake igre je lik kojega igrač kontrolira (Slika 13). To je slučaj i s ovim radom, gdje je glavni lik objekt sa najrazrađenijim skupom funkcionalnosti.

*Slika 13. Likovi sa kojima igrač može igrati*

## 6.1. Kretanje igrača

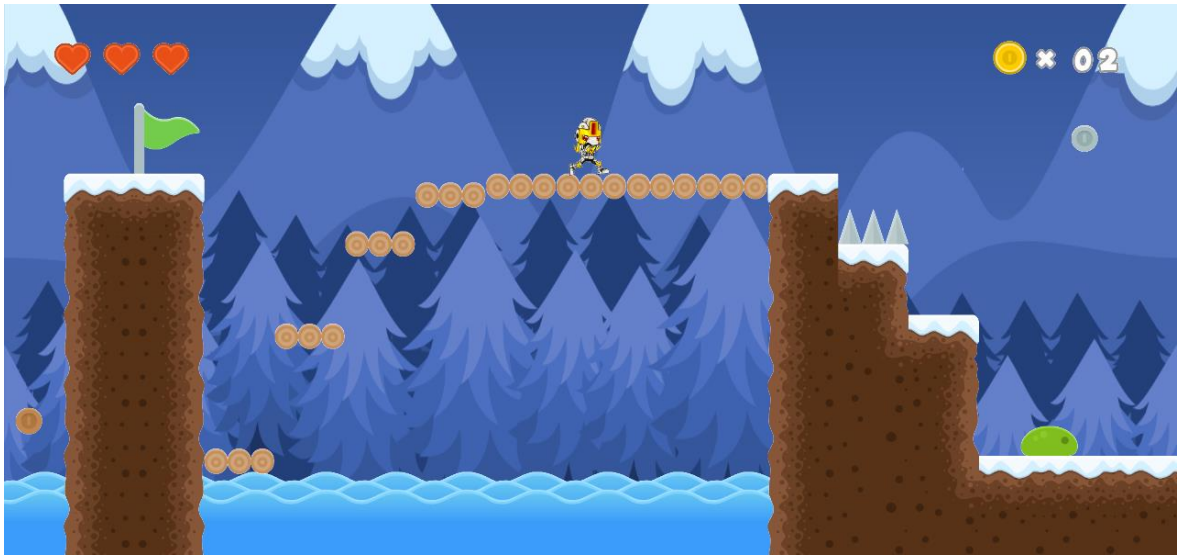
Glavni lik pokreće se unosom igrača preko tipkovnice. Kretanje lika može se podijeliti na horizontalno i vertikalno kretanje. Horizontalno kretanje napravljeno je jednostavnije, s postavljanjem horizontalne komponente brzine na ±4 m/s, ovisno o željenom smjeru gibanja, ili na 0 m/s ukoliko igrač prestane unositi naredbe za kretanjem. Vertikalno kretanje sastoji se od skoka, duplog skoka ili odbijanja u posebnim prilikama. Igrač je u mogućnosti skakati samo kada se nalazi na podu i

jednom u zraku nakon prvog skoka. Odbijanje se događa prilikom interakcije s određenim vrstama neprijatelja. I skokovi i odbijanja su implementirana postavljanjem vertikalne brzine na određenu vrijednost, koja se potom mijenja zbog utjecaja gravitacije. Time je postignuto jednoliko skakanje, neovisno o prethodnim uvjetima.

Animacije glavnog lika određene su brzinom kretanja. Ukoliko postoji vertikalna brzina (odnosno igrač se nalazi u zraku), koristi se animacija skoka. Inače, ako postoji horizontalna brzina prikazuje se animacija trčanja. Konačno, ako igrač stoji mirno na mjestu korištena je animacija mirovanja.

## 6.2. Interakcije u igri

Osim s podom, igrač radi provjere sudara i s ostalim tipovima interaktivnih objekata (Slika 14, Slika 15). Pritom je odgovornost obrade sudara prenesena na te objekte, dok igrač samo zahtjeva od sudarenog objekta obradu. Time je omogućena interakcija između igrača i ostalih objekata, bez da igrač zna za njih (modularni dizajn).

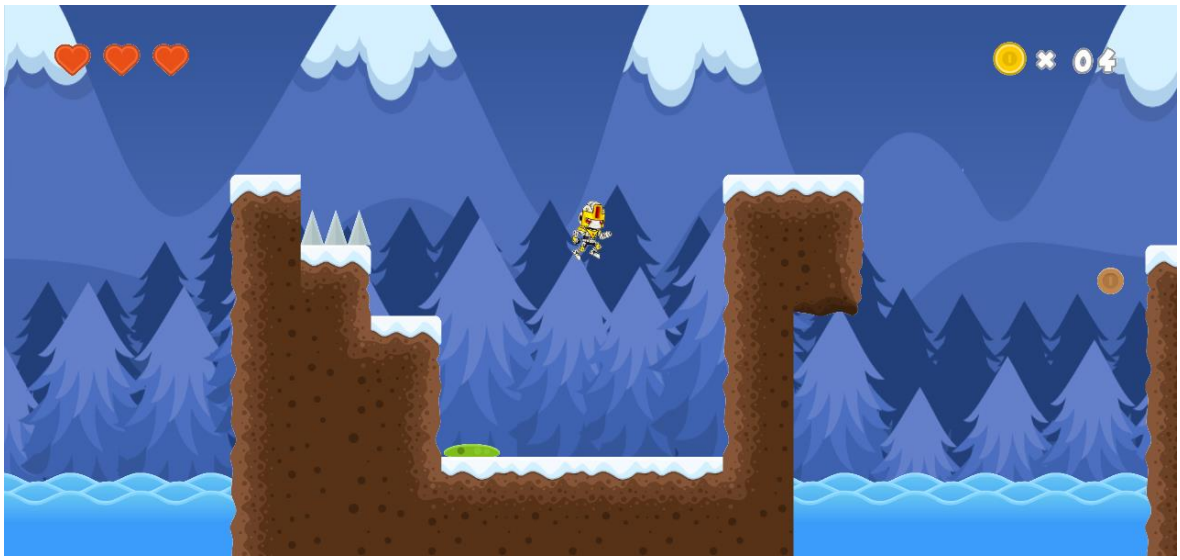



*Slika 14. Igrač u interakciji sa okolinom*

## 6.3. Sustav života

Igrač je jedini objekt u igri koji ima živote. Trenutni i maksimalni broj života postavlja se prilikom stvaranja objekta na tri. Kasnijim interakcijama s drugim

objektima, broj trenutnih života može rasti (igrač je pokupio srce) ili padati (neprijatelj je ozlijedio igrača). Ako igrač padne sa scene, oduzima se jedan život i igrač se pozicionira na lokaciju oživljavanja. Nakon što trenutni broj života padne ispod 0, igra se pauzira, pokazuje se poruka o neuspjehu i trenutna scena se ponovno učitava. Nakon toga, briše se broj sakupljenih novčića i trenutni i maksimalni životi igrača postavljaju se opet na početnu vrijednost.



*Slika 15. Igrač u interakciji sa neprijateljima*

## 7. Scena

Igra se sastoji od više, međusobno neovisnih scena. Svaka scena drži informacije o svim objektima koji se nalaze na njoj. Inicijalizacija scene odvija se u dva koraka. U prvom koraku, koji se izvodi pri pokretanju programa, kreira se scenska kamera. U ovom trenutku scena još ne sadrži nikakve informacije o objektima.

Razlog za podjelom inicijalizacije scene je asinkrono učitavanje podataka o sceni iz JSON datoteke koja se dohvaća sa poslužitelja. Nakon učitavanja i provjere ispravnosti datoteke, moguće je prijeći na drugi korak, čitanje podataka o objektima. Podaci se sastoje od oznake objekta, pozicije na sceni i indeksa korištene teksture. Temeljem oznake, program zna koju vrstu objekta mora stvoriti i u koje polje objekata scene ga mora smjestiti.

Scena ima 5 polja objekata (statični, ukrasni, neprijatelji, okolina, objekti za pokupiti), koja se koriste za odvajanje provjera koje se izvode na različitim tipovima objekata. Primjerice, na nepokretnim objektima nije potrebno računati utjecaje gravitacije i animiranje tekstura, dok se na pokretnim objektima (neprijatelji, okolina) to mora računati. Polja su također iskorištena za ubrzavanje provjera sudara, tako što se objekti bez sudarača nikad ne uzimaju u obzir (ukrasni objekti). Postoji dodatno polje za objekte koji su maknuti sa scene tokom igranja (npr. neprijatelj je pao sa scene, novčić je pokupljen) i ono služi za vraćanje tih objekata natrag na scenu pri ulasku u uređivač scena.

Prelaskom trenutne scene, započinje učitavanje datoteke koja sadrži informacije o sljedećoj sceni na redu. Nakon uspješnog učitavanja ponavlja se prije navedeni drugi korak inicijalizacije scene. Scena se također ponovno učitava i prilikom umiranja igrača.

*Slika 16. Koraci iscrtavanja scene*

Iscrtavanje scene prva je od tri funkcije crtanja koje se pozivaju prilikom igranja, i jedina funkcija crtanja korištena u oba načina rada programa. Crtanje započinje s pozadinskom slikom nakon čega se crtaju polja objekata po prioritetu važnosti (Slika 16). Stoga se prvo crtaju ukrasni objekti, budući da oni imaju najmanji utjecaj na igranje. Sljedeće se crtaju statični objekti po kojima je moguće hodati, objekti koje je moguće pokupiti i na kraju objekti okoline i neprijatelji, obzirom na opasnost koju predstavljaju igraču.

## 7.1. Kamera

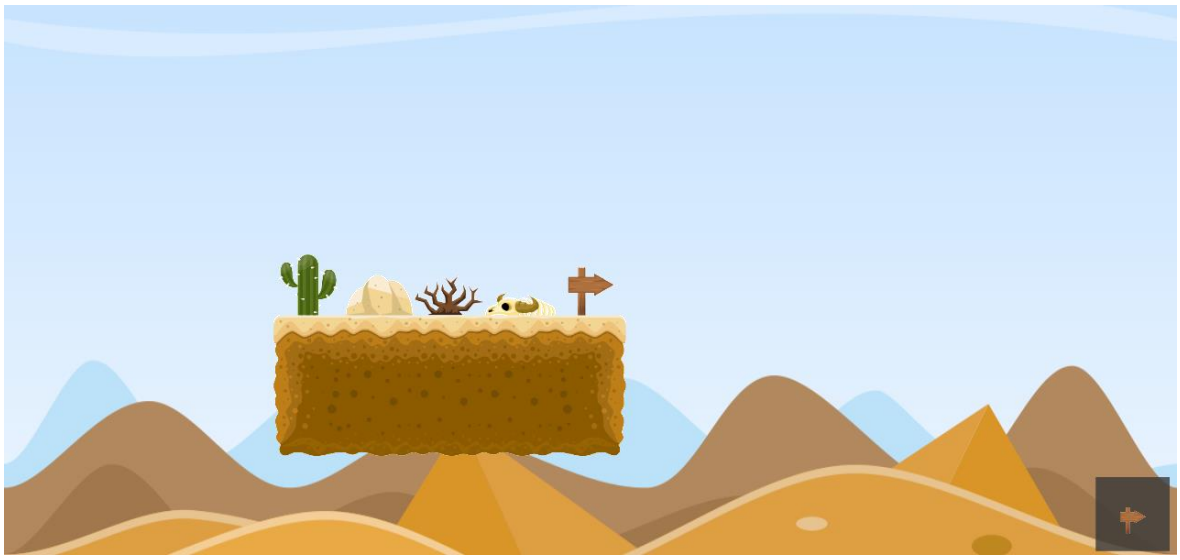
Kako bi se objekti na sceni uopće mogli prikazati, potrebna je kamera. Iako je igra napravljena u dvije dimenzije, za što je uobičajeno koristiti ortografsku kameru, ovaj rad je napravljen s perspektivnom kamerom. Razlog za ovakvim odabirom bio je u skaliranju svih objekata na sceni na istu veličinu. Naime, kako je većina objekata kvadrat stranica dužine 1 m, umjesto pojedinačnog mijenjanja veličine, korišteno je udaljavanje od kamere da se postignu manji objekti na sceni.

Kamera se pomiče na različite načine, ovisno o trenutnom načinu rada programa. Tijekom rada u uređivaču scena, kameru je moguće proizvoljno pomicati u obje dimenzije pomoću tipki smjera. Za vrijeme igre, kamera prati igrača uz neka ograničenja. Prvo ograničenje je nemogućnost kamere da se nalazi lijevo od $x = 4$ i niže od $y = 0$. Dodatno, kamera prati igrača vertikalno samo kada je igrač iznad $y = 2$ i tada se igrač više ne nalazi u samom centru ekrana nego je lagano pomaknut gore. Ovaj pomak je napravljen radi lakšeg gledanja poda na koji je potrebno doskočiti.

Obzirom na veličinu scena, kamera je u nemogućnosti prikazivanja svih objekata na trenutnoj sceni. Stoga, objekte koji se nalaze izvan vidokruga kamere nije potrebno niti pokušavati iscrtavati, jer bi se nepotrebno trošili resursi grafičke kartice na posao koji nije vidljiv u konačnici. Objekti koji se crtaju moraju se nalaziti unutar radijusa od 9 m horizontalno i 5 m vertikalno.

# 8. Uređivač scena

Radi jednostavnije izrade scena i mogućnosti da igrači rade vlastite scene, odlučeno je dodati uređivač scena (eng. *editor*) u program. Uređivač se pokreće iz glavnog izbornika. Prije ulaska u uređivač nudi se mogućnost rada na trenutnoj sceni ili stvaranja nove scene. Nakon odabira, program prelazi u uređivanje scene (Slika 17). U ovom načinu rada, umjesto igraćeg lika i grafičkog sučelja crta se trenutno korišteni objekt.



*Slika 17. Uređivač scena*

Trenutno korišteni objekt postavlja se na scenu pritiskom miša. Mijenjanje trenutnog objekta izvodi se odabirom iz popisa svih objekata koji postoje u igri (Slika 18). Uređivač koristi tri zastavice za rad, jednu za prikazivanje popisa objekata, drugu za kontroliranje je li objekt ukrasni ili ne (ima učinka samo na objekte po kojima igrač hoda), te treću za brisanja objekata. Ukoliko je uključeno brisanje objekata, pritiskom miša brišu se svi objekti koji se nalaze na lokaciji miša.

*Slika 18. Popis objekata u uređivaču scena*

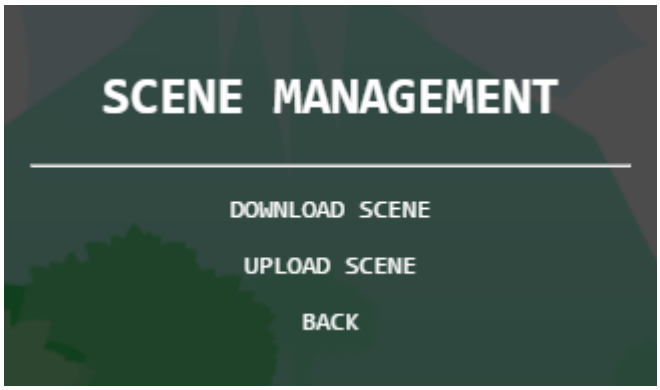
Za razlikovanje uključenih zastavica, korišteno je kodiranje bojom (Slika 19). U normalnom radu, pozadinska boja korištenog objekta i popisa je siva. Ako je uključena zastavica za ukrasne objekte, korištena boja bit će plava, dok uključivanjem opcije za brisanje boja će postati crvena.

*Slika 19. Načini rada uređivača (normalni način rada, ukrasni način, način brisanja)*

## 8.1. Spremanje i učitavanje scene

Igrači koji će koristiti uređivač, najvjerojatnije će htjeti i mogućnost lokalnog spremanja scena, radi ponovnog igranja vlastith scena i pokazivanja svojih radova drugim igračima. Također, potrebno je omogućiti i učitavanje lokalnih scena. U izbornicima igre nalazi se podrška za obje operacije (Slika 20).

Kod spremanja scena, stvara se JSON datoteka koja sadrži informacije o svim objektima na sceni. Podaci koji se spremaju o objektu su oznaka, pozicija objekta i indeks teksture. Dobivanje tih podataka ostvareno je pozivom funkcije *writeData*, koju je moguće i nadjačati radi zapisa dodatnih informacija o objektu. Nakon zapisa svih podataka započinje skidanje datoteke.




*Slika 20. Izbornik za rad sa scenama*

Učitavanje lokalnih scena slično je učitavanju scena koje se nalaze na poslužitelju. Razlika je u dohvaćanju potrebne datoteke. Dok se za normalne scene koristi HTTP zahtjev za potrebnom datotekom, kod lokalnih scena otvara se prozor za odabir datoteka (eng. *file selector*) u pregledniku, s ciljem pretrage računala i odabirom željene scene. Povremeno se javlja greška kod odabira scena, gdje se nakon odabira datoteke, scena ne učitava i korisnik mora ponoviti postupak. Razlog ovoga problema nije poznat, zbog nemogućnosti kontroliranog ponavljanja i naizgled nasumičnog pojavljivanja problema.

# 9. Sučelje prema korisniku

Igre su interaktivni medij i potrebno je omogućiti igraču da vrši željenu interakciju na što je moguće jednostavniji i intuitivniji način. Pošto se ova igra sastoji pretežito od trčanja po platformama, preciznog skakanja i izbjegavanja protivnika, nezgrapne kontrole (ekran na dodir, kontrola pokretima) nisu dobro rješenje. Iako je igru moguće pokrenuti na mobilnom uređaju koja ima podršku za WebGL, igranje nije podržano (nedostatak kontrola). Igra trenutno ima podršku samo za tipkovnicu i miš.

## 9.1. Tipkovnica i miš

Tipkovnica se koristi kad program radi u načinu igre. Korištene tipke su:

- lijevo/desno za kretanje,
- razmaknica (eng. *space* bar*)* za skakanje,
- strelice za pomicanje kamere (uređivač scene),
- q za prikazivanje svih objekata od kojih se može napraviti scena (uređivač scene),
- w za ukrasne objekte (uređivač scene),
- e za brisanje objekata (uređivač scene),
- miš se koristi u izbornicima za odabir stavki i u uređivaču scena za postavljanje, odabir i micanje objekata.

## 9.2. Grafičko sučelje

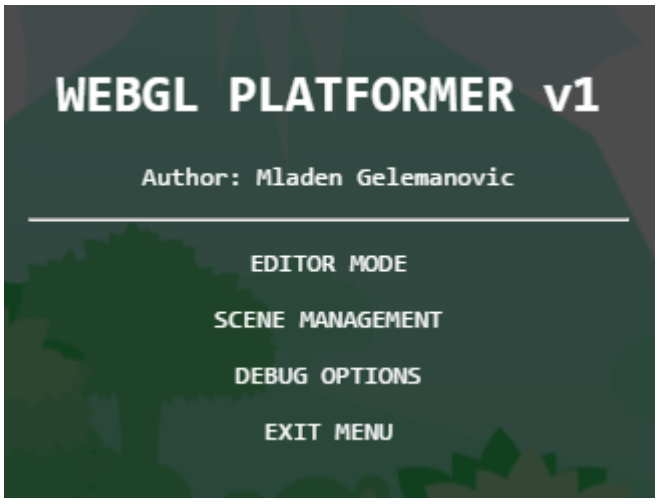
Grafičko sučelje prikazuje igraču potrebne informacije o trenutnom stanju igre (Slika 21). Od informacija prikazuju se trenutni životi glavnoga lika i broj do sada skupljenih novčića u igri. Grafičko sučelje crta se zadnje na ekran, što znači da će se uvijek nalaziti iznad svih ostalih elemenata igre.


*Slika 21. Izgled grafičkog sučelja u igri*

## 9.3. Izbornici

Izbornici u igri napravljeni su pomoću HTML5  elemenata. Prikazivanje i zatvaranje izbornika postignuto je mijenjanjem vidljivosti elementa, dok se potrebne akcije pozivaju preko događaja uzrokovanih pritiskom miša. Većina izbornika (osim početnog i glavnog u igri) generira se unutar programa dinamički. Igra je pauzirana dok su izbornici vidljivi. Iznimka je pomoćni izbornik za prikazivanje broja prolazaka kroz petlju igre u sekundi (eng. *frames per second*).

U igri postoji tri skupa podizbornika (Slika 22). Prvi od njih je izbornik za rad s uređivačem, drugi je izbornik za spremanje i učitavanje scena, te treći koji služi za uključivanje prikaza sudarača.




*Slika 22. Glavni izbornik igre*

# Zaključak

Razvoj igre sa programskim sučeljem WebGL pokazao se kao jedno ugodno iskustvo. Ovaj rad započeo je iz ničega i nakon tri mjeseca rada dobivena je verzija igre koja je ugodna i zanimljiva za igrati, unatoč neispoliranosti. Bitno je napomenuti da se igra konstantno izvodi zadovoljavajućom brzinom od 60 FPS-a, čak i pri srednjim opterećenjima računala. Još jedna prednost ovog rada je dostupnost. Naime, za pokretanje igre nije potrebna nikakva instalacija, već jedino preglednik koji može prikazivati WebGL (većina modernih preglednika).

Veliku značajnost ima i obavezni programski jezik, JavaScript. Mnogi programeri neće odabrati WebGL samo zbog toga. Karakteristike JavaScrpita ne čine ga najboljim odabirom za igre. U odnosu na C++, jezik industrije igara, JavaScript nema tipiziranje varijabli i privatne članove objekata. Iako ovaj rad nije imao problema sa ovim nedostacima, u većim projektima ovo bi se moglo pokazati kao preveliki problem.

Međutim, jezik ima i svoje prednosti. Neke već gotove funkcionalnosti i naknadno dodavanje atributa objektima pokazalo se iznimno korisnim kod izrade rada. Dodatno, i JavaScript i WebGL su poprilično jednostavni za naučiti, iako se preporuča da programer već ima prijašnja iskustva sa programiranjem. U konačnici, WebGL, u kombinaciji sa HTML5, je dobar alat za brzu izradu prototipova igre, iako možda nije primjeren za izradu gotovog proizvoda.

# Literatura

[ 1 ] Sylvester T., Designing Games, O’Reilly 2013

[ 2 ] Royalty Free 2D Game Assets Store, www.gameart2d.com

[ 3 ] Kenney Game Assets, www.kenney.nl

[ 4 ] Bourg D., Bywalec B., Physics for Game Developers, O'Reilly 2013

# Izrada igre upotrebom WebGL-a

## Sažetak

Ovaj rad razmatra primjenu programskog sučelja WebGL za izradu video igara koje se izvode na internetskim stranicama. Rezultat rada je 2D platformerska igra. Napravljena igra podržava korištenje programa za sjenčanje, tekstura, animacije, primjenu gravitacije na objekte, detekciju sudara, izbornike i izradu vlastitih scena igre. Implementiran je igrač koji može trčati i skakati po sceni, razne vrste neprijatelja koje pokušavaju ozlijediti glavnog lika, objekti koje igrač može sakupljati radi određene dobiti i objekti okoline koji uvode dodatnu razinu izazova u igru. Teksture korištene u igri dostupne su besplatno sa Interneta.

## Ključne riječi

WebGL, JavaScript, Grafika, 2D, Video igra, Platformer

# WebGL game development

## Summary

This paper takes WebGL programming interface into consideration when developing video games for web. Resulting work is a 2D platformer game. Developed game offers support for shader programs, textures, animations, objects affected by gravity, collision detection, menus and user created levels. Game implements a playable character that can run and jump, various types of enemies that try to hurt the player, pickup object for player that give some sort of benefit and environmental object that make levels more complex. Textures used in the game are available for free from the Internet.

## Key words

WebGL, JavaScript, Graphics, 2D, Video game, Platformer