Šolski center Ptuj - Elektro računalniška šola Ptuj

Računalniška igra Labirint

Računalniško programiranje

Raziskovalna naloga

Autorji: David Murko
Benjamin Vesenjak
Mentorja : Marjan Čeh
Franc Vrbančič

Ptuj, 2017

Povzetek

Za to temo sem se odločil, ker me že nekaj časa zanima programiranje, sploh na področju računalniških iger. V to igro sem vključil moje najljubše elemente iz iger, ki sem jih igral. Zastavil sem si cilj, da izdelam zanimivo igro, v kateri bo uporabnik imel možnost igrati že izdelane stopnje ali izdelati svojo stopnjo. Najprej sem naredil osnovo igre (enostaven meni, play in create mode). Nato sem dodal enostavne ovire (zid, ključavnica, ključ). Zaradi premajhne težavnosti stopenj sem dodal zahtevne ovire (skala, led, razpoka) in sovražnike, ki se lahko premikanjo navpično ali vododravno. Pri kreiranju stopenj se je pojavil problem, saj je uporabnik izdelal stopnjo in jo lahko igral, ko pa je zapustil stran pa so se podatki o njej izgubili. Zato sem uporabniku dal možnost, da shrani izdelano stopnjo v datoteko in jo prenese na računalnik, preneseno datoteko lahko pozneje naloži v igro in nadaljuje z izdelovanjem njegove stopnje. Da bi izboljšal uporabniško izkušnjo, sem dodal možnost prijave ali registracije uporabnika. S tem sem omogočil uporabniku da shrani svoj napredek pri igranju igre in da lahko shrani izdelane stopnje v bazo podatkov in do njih pozneje dostopa. Pri tem sem imel dosti težav, saj podatki ob slabši internetni lahko potujejo od uporabnika do baze tudi več sekund, kar pa je povzročalo probleme pri nalaganju shranjenih stopenj. Problem sem rešil tako, da aplikacija počaka, da dobi odgovor s podatki in takrat nadaljuje.

Kazalo vsebine

1 UVOD

Računalniške igre so programi namenjeni zabavi uporabnikov. Ponavadi se izvajajo na osebnem računalniku. Delujejo tako, da program, ki se izvaja s pomočjo strojne in programske opreme na zaslonu prikazuje grafiko in različne like, ki jih uporabnik kontrolira z vhodnimi enotami. V zadnjem času se igre hitro razvijajo in izboljšujejo. V nekaterih igrah uporabnik igra proti računalniku, oz. single player. Vse bolj pogoste pa postajajo tudi online multiplayer igre v katerih uporabnik igra z ali proti drugim ljudem.

2 ZVRSTI VIDEO IGER

Zvrst video igre se ponavadi ne opredeljuje glede po dejanski vsebini igre ali načinu igranja(PC, Xbox, Playstation) temveč po skupnem cilju.

Zvrsti obsegajo široko paleto iger, kar vodi do še bolj specifičnih razvrstitev imenovanih podzvrsti. Ker je zvrsti mnogo, z novo nastajajočimi, bodo predstavljene le nekatere.

2.1 Zvrsti:

- FPS First person shooter (Igralec ima pogled prve osebe)
- RPG Role playing game (Igralec je v vlogi lika)
- MMORPG Massively multiplayer online RPG (RPG igra v večigralnem načinu)
- TBS Turn based strategy (Stratežka igra, v potezah)
- TPS Third person shooter
- Arkadne igre (2D)

2.2 First person shooter (FPS)

FPS je zvrst video igre, ki se osredotoča na strelsko orožje in uporabo le tega. Igralec doživlja igro skozi oči protagonist. Tovrstne igre uporabljajo 3D grafiko, zato so včasih bile izziv za strojno opremo, dandanes več ni tako. Večigralski način je pri teh igral zelo popularen.

Slika 1: Prva FPS igra

Vir: www....

Maze War, je izšla leta 1973.

Slika 2: FPS igre danes

Vir: www....

Tako FPS igre zgledajo danes.

2.3 Role-Playing game (RPG)

RPG je igra v kateri igralci prevzemajo vlogo likov v izmišljenem okolju. Igralci like igrajo direktno skozi strukturirano sprejemanje odločitev ali skozi razvoj svojega lika (nabiranje izkušenj(Experience) in stopenj(Levels)). Dejanja storjena v tovrstnih igrah so lahko uspešna ali ne (zadajanje kritične škode(Critical strike)). To določajo razni faktorji, ki so zapisani v formalnem sistemu pravil igre.

Slika 3: Začetek RPG iger(še brez računalnika)

Vir: www....

Dungeons and Dragons, ena izmed prvih RPG iger. Bila igrana brez računalnika

Slika 4:Dark Souls III, ena najnovejših RPG-jev

Vir: http://www.godisageek.com/wp-content/uploads/dark-souls-3-3.jpg

2.4 Massively Multiplayer Online Role-Playing Game (MMORPG)

Je RPG vendar v večigralskm načinu. Dobeseden prevod iz angleščine je “Masivno Večigralko Spletno Igranje Namišljenih Vlog”. Igralec ustvari lik(Character), kateremu izbere vlogo(Kaj bo njegov lik zmogel početi), podobo in ime. Nato se poda na postulovščino v namišljen svet. Tam nabira izkušnje(Experience), predmete(Items), točke večih vrst, opravlja misije(quests), sklepa prijateljstva in premaguje ovire vseh vrst. Navadno se tovrstne igre igrajo na Računalniku.

Slika 5: MMORPG igre včasih

Vir: https://i.ytimg.com/vi/rawv4wPYDOs/maxresdefault.jpg

Slika 6: MMORPG danes.

Vir:

http://www.guildwars2portal.com/wp-content/uploads/2016/07/1070062-amazing-guild-wars-2-wallpaper.jpg

2.5 Turn Based Strategy

Turn based strategy je strateška igra ponavadi neke vrste vojna igra ki se igra v potezah to se razlikuje od Real Time Strategy. Ponavadi zahteva zastopanje vojaških taktik in operacij

Slika7: Stara Turn Based game(1997).

Vir: https://msgpwebcdn.azureedge.net/ageofempires/wp-content/uploads/2015/07/3.jpg

Slika 8: Novejša Turn Based Strategy Generations VI.

Vir: http://gameabout.com/files/attach/images/115/679/750/003/368ff3b3c465

f878c3690da08a24f10c.jpg

2.6 Third Person Shooter

Third person shooter je igra, ki temelji na streljanju. Je zelo podobna First person shooterju(FPS), le da lahko tukaj igralec vidi svojega avatarja s pogleda tretje osebe.

Slika 9: Ikoničen Third Person Shooter, Max Payne.

Vir: https://cyberpowerpc.files.wordpress.com/2015/06/max-payne.png

Slika 10: Današnji TPS-ji.

Vir: https://i.ytimg.com/vi/VRMQ-DfnULI/maxresdefault.jpg

2.7 Arkadne igre (2D)

Arkadne igre so preproste igre, ponavadi 2D. Sestavljene iz nivojev (leveli) v katerih se ob napredovanju težavnost veča. Stopnje preizkušajo spretnost igralca, lahko pa vsebujejo tudi uganke (puzzle).

Slika 5: Super Mario, ena izmer prvih arkadnih iger

Vir:

http://images.complex.com/complex/image/upload/c_limit,w_680/fl_lossy,pg_1,q_auto/super-mario-brothers-game_nea7ms.jpg

Slika 6: Geometry rush, arkadna igra danes

Vir: https://s-media-cache-ak0.pinimg.com/236x/25/91/2b/25912bf2b26576463b9a400650e2227a.jpg

3 STROJNA OPREMA

Za igranje iger potrebujemo tudi strojno opremo. Večina iger se igra na osebnem računalniku, ali na igralnih konzolah. Obstajajo pa tudi igre za mobilne telefone, tablice ipd.

3.1 Osebni računalnik

Večina grafično zahtevnih iger je narejenih za osebni računalnik. Igre je potrebno pred igranjem namestiti, med izvajanjem pa upobljajo velik del procesorske moči in RAM-a. Ponavadi se za igranje uporablja miška in tipkovnica.

3.2 Igralna konzola

Igralne konzole so namenjene igranju iger, ampak jih večina podpira tudi različne multimedijske vsebine (filmi, glasba ...). Za igranje se uporabljajo kontrolerji.

3.3 Mobilni telefon

Igre na pametnih telefonih niso tako zahtevne kot na osebnih računalnikih ali konzolah. Za igranje se uporablja zaslon na dotik.

3.4 Brskalniške igre

Te igre so manj zahtevne in se izvajajo v spletnih brskalnikih. Ni potrebne namestitve. Ponavadi se igrajo na osebnih računalnikih, ampak se lahko igrajo tudi na kateri drugi napravi, ki ima dostop do spleta in ima ustrezne vhodne elemente. Ponavadi se za igranje uporablja miška in tipkovnica.

4 PROGRAMSKA OPREMA

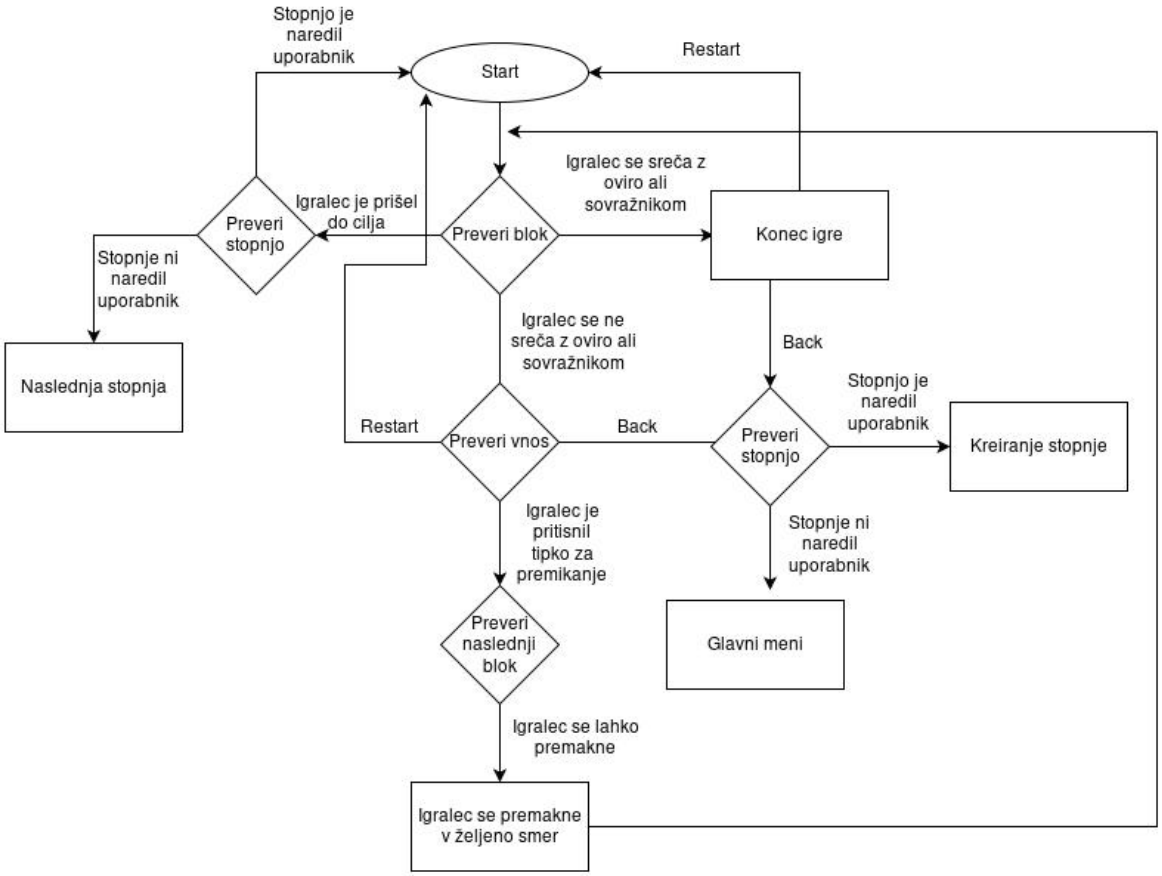
Igra je večinoma napisana v programskem jeziku Javascript, uporablja pa tudi HTML (elementi na spletni strani, gumbi, igralno okno,...), CSS(postavitev elementov na strani, barva, pisava,...) in PHP (login in povezavo z bazo). Za prikaz igre sem uporabil HTML5 element canvas. Canvas omogoča prikazovanje grafike in njeno manipuliranje z javascriptom.

4.1 Programsko okolje

Za izdelavo igre nisem uporabil programskega okolja. Igro sem napisal tekstovno. Cilj tega je, da bi bila igra delovala na vsakem brskalniku brez dodatnih vtičnikov npr. Flash player. Prednost tega je, da imam pregled nad celotno kodo in da vem točno kako igra deluje. Slabosti pa so, da je bilo programiranje igre težje in sem potreboval več časa.

4.2 Diagram poteka

Sledi diagram poteka načina play. Do načina lahko dostopam preko glavnega menija, če izberem možnost Play, ali če v glavnem meniju izberem možnost Create Stage in v orodni vrstici pritisnem Play.



Slika 7: Diagram poteka igre

Vir: lasten

5 PROGRAMIRANJE IGRE

Igro sem programiral po naslednjem postopku: Najprej sem si zamislil idejo in si po potrebi skiciral njeno delovanje. Nato sem kodo testiral in odpravljal napake, dokler ni delovala po mojih pričakovanjih. Pozneje sem kodo vključil v igro in nadaljeval z testiranjem. Ko je nova koda v igri delovala sem jo naložil v moj projekt na spletni strani github.com. Ta spletna stran omogoča sinhroniziranost kode, ki jo naenkrat razvija več ljudi. Preden se tvoja koda naloži preveri razlike med njo in trenutno naloženo kodo.

5.1 Kreiranje igralnega okna

Html element canvas, sem kreiral z dinamično z javascriptom. Funkcija createCanvas() se izvede takoj, ko se stran naloži. Kličem jo v datoteki main.js.

```
function createCanvas() //funkcija ki naredi canvas element
{
    canvas = document.createElement('CANVAS'); //spremenljivka brez var je globalna
    canvas.width = '800';
    canvas.height = '627';
    canvas.style.background=  '#000000';
    canvas.style.border = 'black solid 2px';
    screen = canvas.getContext('2d');
    //canvas element prikljuci bodyju spletne strani
    document.getElementsByTagName('BODY')[0].getElementsByTagName('div')[0].appendChild(canvas);
}
```

Slika 8: funkcija, ki ustvari igralno okno

Vir: lasten

5.2 Zaznavanje uporabnikovega vnosa

Da lahko uporabnik vpliva na igro, je potrebno zaznati dogodke ki se zgodijo ob določenih akcijah. Za to igro sem potreboval dogodke z miško(premikanje miške, klik) in dogodke s tipkovnico(klik, pritisne tipko, spusti tipko). Javascript za sprejemanje dogodkov podpira podpira posušalce dogodkov(eventListener), ki ob določenem dogodku izvedejo določeno metodo. eventListenerje dodam z metodo objekt.addEventListener, ki ima tri parametre - prvi parameter je ime dogodka ki ga posluša, drugi parameter je ime funkcije, ki se izvedo ob tem dogodku, tretji parameter je logični tip, ki ima, če je nastavljen na true prednost pred ostalimi eventListenerji, ki poslušajo enak dogodek. Vse eventListenerje dodam na enkrat z metodo key.addEventListeners() v datoteki key.js.

Slika 9: funkcija za poslušalce dogodkov

Vir: lasten

5.2.1 Zaznavanje tipkovnice

Za vsako tipko, ki bi jo rad zaznal sem rezerviral pomnilnik logičnega tipa v objektu key. Vse sem nastavil na false. Vrednost teh pomnilnikov se spreminja z metodama key.onDown() in key.onUp(), ki se izvedeta ko uporabnik pritisne ali spusti tipko.

```
onDown:function(event) {

    if (event.keyCode == 39) {
        key.right = true;
    }
    else if (event.keyCode == 37) {
        key.left = true;
    }
    else if (event.keyCode == 38) {
        key.up = true;
    }
    else if (event.keyCode == 40) {
        key.down = true;
    }else if(event.keyCode ==32 || event.keyCode==0){
        key.space=true;
    }

},
```

Slika 10: funkcija preverjanja pritiska tipke

Vir: lasten

```
onUp:function(event) {
    if (event.keyCode == 39) {
        key.right = false;
    }
    else if (event.keyCode == 37) {
        key.left = false;
    }
    else if (event.keyCode == 38) {
        key.up = false;
    }
    else if (event.keyCode == 40) {
        key.down = false;
    }else if(event.keyCode ==32 || event.keyCode==0){
        key.space=false;
    }
}
```

Slika 11: funkcija preverjanja spusta tipke

Vir: lasten

5.2.2 Zaznavanje miške

Za miško sem rezerviral pomnilnike v objektu key.mouse.button podobno kot pri tipkovnici in jih nastavil na false. Z metodama mouse.buttonDown() in mouse.buttonUp() se vrednost teh pomnilnikov spremenja ob pritisku na tipke miške.

```
buttonDown:function(event)
{
    if (event.which == 1) //levi gumb
    {
        mouse.button.left=true;
    }
    if (event.which == 3) //desni gumb
    {
        mouse.button.right=true;
    }
},
```

Slika 12: funkcija preverjanja pritiska tipk miške

Vir: lasten

```
buttonUp:function(event)
{
    if (event.which == 1) //levi gumb
    {
        mouse.button.left=false;
    }
    if (event.which == 3) //desni gumb
    {
        mouse.button.right=false;
    }
},
```

Slika 13: funkcija preverjanja spusta tipk miške

Vir: lasten

Potreboval sem tudi trenutno pozicijo miške, zato se je ob vsakem premiku miške znotraj canvasa izvedla metoda mouse.onMove(), ki je v objekt mouse.canvasCoord shrani trenutno pozicijo miške.

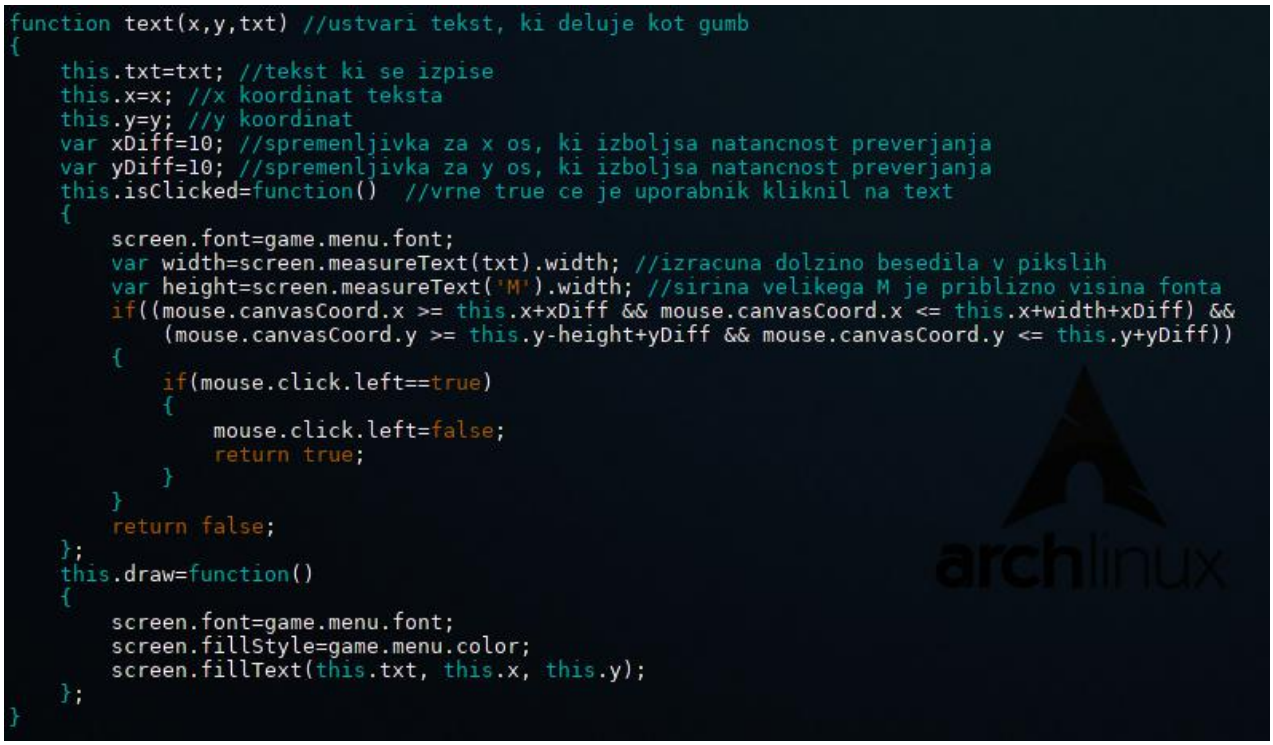
```
onMove:function(event)
{
    mouse.canvasCoord.x = event.clientX - (canvas.offsetLeft - window.pageXOffset);
    mouse.canvasCoord.y = event.clientY - (canvas.offsetTop - window.pageYOffset);
}
```

Slika 14: funkcija zaznavanja trenutne pozicije kazalca

Vir: lasten

5.3 Glavni meni

Za glavni meni in za nekatere gumbe v orodni vrstici sem uporabljal objekte tipa text.



```
function text(x,y,txt) //ustvari tekst, ki deluje kot gumb
{
    this.txt=txt; //tekst ki se izpise
    this.x=x; //x koordinat teksta
    this.y=y; //y koordinat
    var xDiff=10; //spremenljivka za x os, ki izboljsa natancnost preverjanja
    var yDiff=10; //spremenljivka za y os, ki izboljsa natancnost preverjanja
    this.isClicked=function()  //vrne true ce je uporabnik kliknil na text
    {
        screen.font=game.menu.font;
        var width=screen.measureText(txt).width; //izracuna dolzino besedila v pikslih
        var height=screen.measureText('M').width; //sirina velikega M je priblizno visina fonta
        if((mouse.canvasCoord.x >= this.x+xDiff && mouse.canvasCoord.x <= this.x+width+xDiff) &&
            (mouse.canvasCoord.y >= this.y-height+yDiff && mouse.canvasCoord.y <= this.y+yDiff))
        {
            if(mouse.click.left==true)
            {
                mouse.click.left=false;
                return true;
            }
        }
        return false;
    };
    this.draw=function()
    {
        screen.font=game.menu.font;
        screen.fillStyle=game.menu.color;
        screen.fillText(this.txt, this.x, this.y);
    };
}
```

Slika 15: objekt text

Vir: lasten

Objekte tipa text sem definiral v objektu game.menu.button.

```
button:{
    play:new text(250,250,'Play'),
    make:new text(250,300,'Create stage'),
    loadF:new text(250,350,'Load from file'),
    load:new text(250,400,'Load level'),
    logout:new text(250,450,'Logout')
},
```

Slika 16: primer rezervacije objekta text

Vir: lasten

Ko je objekt definiran, je treba v zanki klicati še metodo za izris objekta (objekt.draw()), in metodo, ki preverja če ga je uporabnik kliknil (objekt.isClicked()).

```
game.clear();
game.console.draw();
sound.drawButton();
sound.checkButton();
game.menu.button.play.draw();
game.menu.button.make.draw();
game.menu.button.loadF.draw();
```

Slika 17: primer izrisov objektov text

Vir: lasten

```
if(game.menu.button.play.isClicked())
{
    game.form.hide();
    game.menu.loop=false;
    map.make.loop=false;
    game.loop=true;
    game.clear();
    map.level=toArray(window['level_'+map.levelIndex]);
    game.reset();
    game.init();
    map.draw50();
    map.drawPanel();
    game.start();
}
```

Slika 18: primer preverjanja klikov na objektu text

Vir: lasten

Za ponavljanje funkcije glavnega menija in ostalih načinov v igri sem uporabil javascript

funkcijo window.setTimeout(). Funkcija omogoča ponovitev določene metode v željenem časovnem intervalu, kar mi je dalo možnost kontrole nad hitrostjo igre, in izrisovanja na zaslon. Funkciji podamo dva parametra - prvi parameter je ime metode, ki se bo ponavljala, drugi parameter pa je število milisekund po katerih se bo klicala ponovno. game.menu.main je ime funkcije, game.menu.tick pa so milisekunde.

```
if(game.menu.loop!=false)
{
    setTimeout(game.menu.main,game.menu.tick);
}
```

Slika 19: metoda ponavljanja glavne zanke

Vir: lasten

5.4 Stopnja(level) - mapa

Vsaka stopnja je zapisana kot dvodimenzionalno polje številskega tipa, velikosti 32(x os) in 25(y os). Zaradi velikega števila znakov, sem naredil metodo, ki dvodimenzionalno polje pretvori v niz, in metodo, ki naredi obratno.

```
function toArray(str) //pretvori string mape v dvodimenzionalno polje
{
    var arr=str.split('|');
    var n=[];
    for(var i=0;i<arr.length;i=i+1)
    {
        n[i]=JSON.parse('[' + arr[i] + ']');

    }
    return n;
}
```

Slika 20: funkcija za pretvorbo niza v polje

Vir: lasten

```
function toMapString(arr) //pretvori dvodimenzionalno polje v string
{
    var str='';
    for(var i=0;i<arr.length;i=i+1)
    {
        str=str+arr[i].join(',');
        if(i+1<arr.length)
        {
            str=str+'|';
        }

    }
    return str;
}
```

Slika 21: funkcija za pretvorbo iz polja v niz

Vir: lasten

Primer prazne stopnje v obliki polja in niza:

```
var emptyLevel = [
                [0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2],
                [0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0]
];
```

Slika 22: prazna stopnja v obliki polja

Vir: lasten

```
emptyLevel   ="0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2|0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0";
```

Slika 23: prazna stopnja v obliki niza

Vir: lasten

5.4.1 Izris stopnje

Stopnjo izrisujem z dvema metodama. Metoda map.draw50() izriše 13x13 blokov, vsak blok je velik 50x50 pikslov. Ta metoda se izvaja med načinom play. Izrisuje bloke okrog igralca, če na določenem mestu ni bloka, oz. je polja konec izriše prazen blok.

```
while(curPos.y<canLimit.y)
{
    row=1;
    mapa.x=mapaStart.x;
    curPos.x=0;
    while(curPos.x<canLimit.x)
    {
        //preveri ce polje obstaja
        if(typeof(map.level[mapa.y]) != 'undefined' && typeof(map.level[mapa.y][mapa.x]) != 'undefined')
        {
            if(map.getBlock50(mapa.x,mapa.y)=='enemy03')
            {
                screen.drawImage(enemy03.list[enemy03.findByCoord(mapa.x,mapa.y)].img, curPos.x, curPos.y);

            }else if(map.getBlock50(mapa.x,mapa.y)=='enemy02')
            {
                screen.drawImage(enemy02.list[enemy02.findByCoord(mapa.x,mapa.y)].img, curPos.x, curPos.y);

            }else if(map.getBlock50(mapa.x,mapa.y)=='enemy01')
            {
                screen.drawImage(enemy01.list[enemy01.findByCoord(mapa.x,mapa.y)].img, curPos.x, curPos.y);
            }else
            {
                screen.drawImage(map.block[map.getBlock50(mapa.x,mapa.y)], curPos.x, curPos.y);
            }
        } else
        {
            screen.drawImage(map.block['blank'], curPos.x, curPos.y); //narise prazen blok

        }
        mapa.x = mapa.x + 1;
        row=row+1;
        curPos.x = curPos.x + map.blockSize;
    }
    mapa.y = mapa.y +1;
    column=column+1;
    curPos.y = curPos.y + map.blockSize;
    player.draw();
}
```

Slika 24: glavna zanka metode za izris stopnje v velikosti 50x50 pikslov

Vir: lasten

Ko se igralec premika, se izvaja metoda player.drawMovingFrame(), ki deluje podobno kot metoda map.draw50(), le da prva vrstica v smeri gibanja postopoma izginja, zadnja vrstica v smeri gibanja pa se postopoma prikazuje.

:Slika 25: primer delovanja metode player.drawMovingFrame()

Vir: lasten

Za izrisovanje stopnje, ki jo kreira uporabnik se uporablja metoda map.draw25(), ki izriše celotno polje stopnje v velikosti bloka 25x25 pikslov.

```
draw25:function() //narise polje map.level v 25x25 velikosti blokov
{
    enemy01.resetAll();
    var curPos=new coord(0,0); //current position
    var mapa = new coord(1,1); //zacetna pozicija risanja v dvodimenzionalnem polju
    var limit = new coord(32,25); // meja polja po sirini, po visini

    while(mapa.y < limit.y) //zanka gre od 0,0 do limit.x, limit.y
    {
        mapa.x=1;
        curPos.x=0;
        while(mapa.x <= limit.x)
        {
            if(map.getBlock25(mapa.x,mapa.y)!=false)
            {
                screen.drawImage(map.block[map.getBlock25(mapa.x,mapa.y)], curPos.x,curPos.y);
            }
            mapa.x = mapa.x + 1;
            curPos.x = curPos.x + map.blockSize/2;
        }
        mapa.y = mapa.y +1;
        curPos.y = curPos.y + map.blockSize/2;
    }
},
```

Slika 26: metoda za izris stopnje v velikosti 25x25 pikslov

Vir: lasten

5.5 Igralec

Lik igralca se pomika po labirintu, kot mu ukaže uporabnik.

Slika 27: lik igralca

Vir: lasten

5.5.1 Izris igralca

Igralec se vedno izriše na enakih koordinatih, premikanje blokov za njim ustvari iluzijo gibanja. Izrišem ga z metodo player.draw(), ki v primeru da je igralec padel v luknjo(pomnilnik player.fall) ne nariše ničesar.

```
draw:function()
{
    if(player.fall==false)
    {
        screen.drawImage(player.img, player.canvasCoord.x, player.canvasCoord.y);
    }
},
```

Slika 29: metoda z izris igralca

Vir: lasten

5.5.2 Premikanje igralca

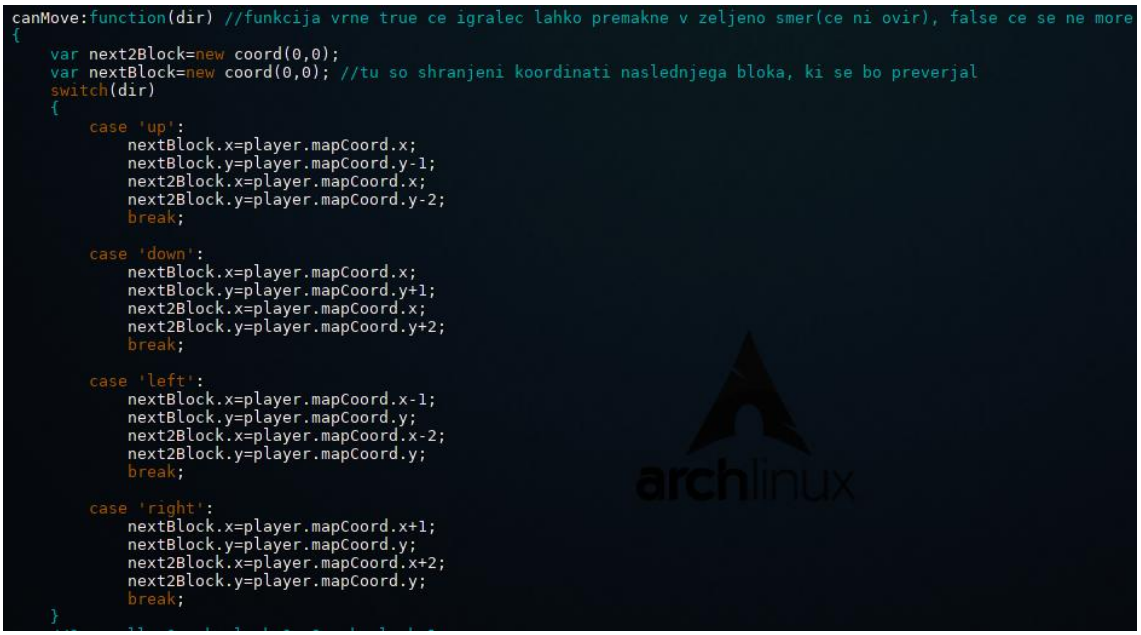
Igralec se premika z metodo player.move(). Paramter metode je niz smeri v katero se naj igralec premakne. Metoda spremeni igralčeve koordinate v player.mapCoord, nastavi player.isMoving na true (druge metode preverjajo je to spremenljivko, da se do konca premikanja v eno smer onemogoči premikanje v ostale smeri), in predvaja animacijo gibanja. Pomnilniki player.movingFrame.firstLine, player.movingFrame.midLine, player.movingFrame.lastLine so parametri, ki izrisujejo del bloka(glej poglavje 5.4.1). Na primer premikanje v desno:

```
if(dir== 'right')
{
    player.dir='right';
    player.animation.start(map.block['playerRight0'],map.block['playerRight2'],map.block['playerRight1']);

    if(player.lastDir=='down')
    {
        player.movingFrame.start.y=player.movingFrame.start.y+1;

    }
    player.lastDir='right';
    player.movingFrame.firstLine=0;
    player.movingFrame.midLine=-50;
    player.movingFrame.lastLine=-50;
    player.movingFrame.xCh=1;
    player.movingFrame.yCh=0;
}
player.movingFrame.exitCount=0;
player.mapCoord.x=player.mapCoord.x+player.movingFrame.xCh;
player.mapCoord.y=player.mapCoord.y+player.movingFrame.yCh;
player.isMoving=true;
```

Slika 30: del metode player.move - preverjanje za premik v desno

Vir: lasten

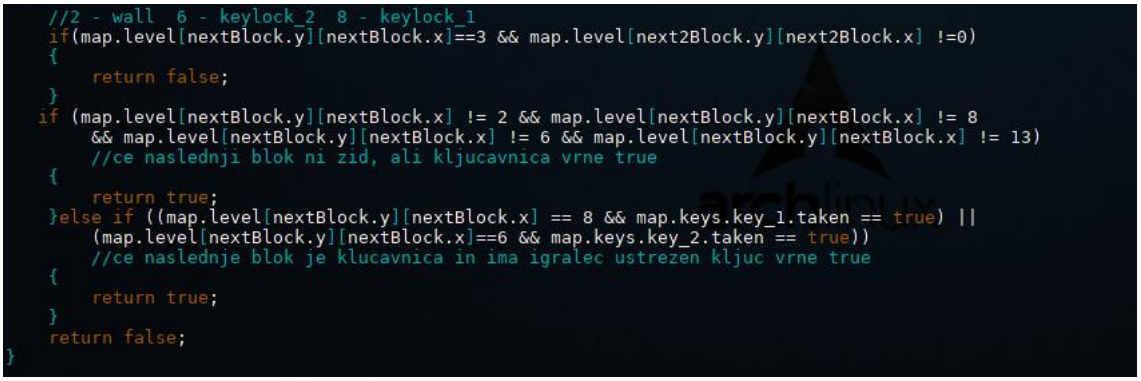
Pred premikanjem je treba preveriti ali naslednju blok ustreza igralcu, npr. če je zid ali zakljenjena ključavnica se igralec ne more premakniti v željeno smer. To preverim z metodo player.canMove(), kot parameter podam niz smeri katero preverjam. Vrne true, če se igralec lako premaken v željeno smer, drugače vrne false.



```
canMove:function(dir) //funkcija vrne true ce igralec lahko premakne v zeljeno smer(ce ni ovir), false ce se ne more
{
    var next2Block=new coord(0,0);
    var nextBlock=new coord(0,0); //tu so shranjeni koordinati naslednjega bloka, ki se bo preverjal
    switch(dir)
    {
        case 'up':
            nextBlock.x=player.mapCoord.x;
            nextBlock.y=player.mapCoord.y-1;
            next2Block.x=player.mapCoord.x;
            next2Block.y=player.mapCoord.y-2;
            break;

        case 'down':
            nextBlock.x=player.mapCoord.x;
            nextBlock.y=player.mapCoord.y+1;
            next2Block.x=player.mapCoord.x;
            next2Block.y=player.mapCoord.y+2;
            break;

        case 'left':
            nextBlock.x=player.mapCoord.x-1;
            nextBlock.y=player.mapCoord.y;
            next2Block.x=player.mapCoord.x-2;
            next2Block.y=player.mapCoord.y;
            break;

        case 'right':
            nextBlock.x=player.mapCoord.x+1;
            nextBlock.y=player.mapCoord.y;
            next2Block.x=player.mapCoord.x+2;
            next2Block.y=player.mapCoord.y;
            break;
    }
```

Slika 31: metoda, ki preveri ali se lahko igralec premakne v željeno smer (prvi del)

Vir: lasten



```
    //2 - wall  6 - keylock_2  8 - keylock_1
    if(map.level[nextBlock.y][nextBlock.x]==3 && map.level[next2Block.y][next2Block.x] !=0)
    {
        return false;
    }
   if (map.level[nextBlock.y][nextBlock.x] != 2 && map.level[nextBlock.y][nextBlock.x] != 8
        && map.level[nextBlock.y][nextBlock.x] != 6 && map.level[nextBlock.y][nextBlock.x] != 13)
        //ce naslednji blok ni zid, ali kljucavnica vrne true
    {
        return true;
    }else if ((map.level[nextBlock.y][nextBlock.x] == 8 && map.keys.key_1.taken == true) ||
        (map.level[nextBlock.y][nextBlock.x]==6 && map.keys.key_2.taken == true))
        //ce naslednje blok je klucavnica in ima igralec ustrezen kljuc vrne true
    {
        return true;
    }
    return false;
}
```

Slika 32: metoda, ki preveri ali se lahko igralec premakne v željeno smer (drugi del)

Vir: lasten

5.5.3 Srečanje igralca s sovražnikom

Za preverjanje srečanje sem uporabil metodo player.isHit(), ki preveri ali se je igralec dotaknil sovražnika. Ker se ob klicu metode player.move() igralčevi koordinati takoj spremenijo, slika igralca pa se še premika, preverjam en blok za igralčevo trenutno pozicijo. Če se igralec sreča z sovražnikom mu ta zmanjša zdravje za določeno vrednost.

```
isHit:function()
{
    var lastDir=new coord(player.mapCoord.x,player.mapCoord.y);
    if(player.isMoving==true)
    {
        switch(player.lastDir)
        {
            case 'up':
                lastDir.y=lastDir.y+1;
                break;
            case 'down':
                lastDir.y=lastDir.y-1;
                break;
            case 'left':
                lastDir.x=lastDir.x+1;
                break;
            case 'right':
                lastDir.x=lastDir.x-1;
                break;
        }
    }
     if(map.level[lastDir.y][lastDir.x]==12)
    {
        player.hp=player.hp-enemy02.dmg;
    }
    if(map.level[lastDir.y][lastDir.x]==11)
    {
        player.hp=player.hp-enemy01.dmg;
    }
},
```

Slika 33: metoda preverjanja trčenja igralca z sovražnikom

Vir: lasten

5.6 Ovire

Zid je osnovna ovira, igralec in sovražnik ne moreta dalje. V polju stopnje ima številko 2.

Slika 34: ovira - zid

Vir: lasten

Ključavnice so ovira skozi katero igralec pride samo, če ima ustrezen ključ. Če igralec pride do ključavnice in ima ustrezen ključ, se izvede metoda map.keys.key_x.unlock(). Ključavnici imata v polju stopnje številko 8 (siva ključavnica) in številko 6 (rdeča ključavnica).

Slika 35: ovira - siva ključavnica

Vir: lasten

Slika 36: ovira - rdeča ključavnica

Vir: lasten

```
unlock:function()
{
    map.level[player.mapCoord.y][player.mapCoord.x] = 0; //kljucavnica izgine
    this.num = this.num - 1; //stevec kljucev se zmanjsa
    if (this.num == 0) //ce je stevilo kljucev 0, potem iz inventorija izgine slika kljuca
    {
        player.inventory.remove(player.inventory.getIndex('Key01_25x25.png'));
        this.taken = false;
    }else
    { //ce ni nic, potem se v inventoriju narise slika kljuca in stevilo pobranih klucev
        player.inventory.update(player.inventory.getIndex('Key01_25x25.png'),map.keys.key_1.num);

    }
}
```

Slika 37: metoda map.keys.key_x.unlock()

Vir: lasten

Ključi so predmeti, ki jih igralec pobira in se shranijo v inventorij. Če igralec pobere ključ, se izvede metoda map.key.key_x.pickUp(). Ključa imata v polju stopnje številko 5 (sivi ključ) in številko 7 (rdeči ključ).

Slika 38: ovira - sivi ključ

Vir: lasten

Slika 39: ovira - rdeči ključ

Vir: lasten

```
pickUp:function()
{
    map.level[player.mapCoord.y][player.mapCoord.x] = 0; //kljuc izgine
    this.num = this.num + 1; //stevec se poveca za 1
    if (this.num == 1) //ce je to prvi kljuc, ga narise v inventoriju
    {
        this.taken = true;
        player.inventory.add(map.block['key_2_25'],map.keys.key_2.num);

    }else
    { //ce ni prvi preveri in spremeni stevilo kljucev v inventoriju
        player.inventory.update(player.inventory.getIndex('Key02_25x25.png'),map.keys.key_2.num);

    }
},
```

Slika 40: metoda map.keys.key_x.pickUp()

Vir: lasten

Skala je ovira, ki jo lahko igralec premika, če je blok za njo prazen. Ko jo igralec premika se izvaja naslednja koda. Skala ima v polju stopnje številko 3.

Slika 41: ovira - skala

Vir: lasten

```
switch(player.lastDir)
{
    case 'up':
        map.level[player.mapCoord.y][player.mapCoord.x]=0; //trenuten blok postane prazen
        map.level[player.mapCoord.y-1][player.mapCoord.x]=3; //naslednji blok postane skala
        break;

    case 'down':
        map.level[player.mapCoord.y][player.mapCoord.x]=0;
        map.level[player.mapCoord.y+1][player.mapCoord.x]=3;
        break;

    case 'left':
        map.level[player.mapCoord.y][player.mapCoord.x]=0;
        map.level[player.mapCoord.y][player.mapCoord.x-1]=3;
        break;

    case 'right':
        map.level[player.mapCoord.y][player.mapCoord.x]=0;
        map.level[player.mapCoord.y][player.mapCoord.x+1]=3;
        break;
}
```

Slika 42: delovanje ovire skala

Vir: lasten

Led je ovira, na kateri igralec nima več kontrole. Premika se v smeri v kateri je stopil nanj, dokler ne pride do konca ledu, ali do druge neprehodne ovire. Za premikanje po ledu sem uporabil podobno kodo kot za premikanje, le da se ne predvajajo animacije in da ne sprejema tipk za premikanje dokler se igralec drsa. Led ima v polju stopnje številko 4.

Slika 43: ovira - led

Vir: lasten

Razpoka je ovira, ki jo lahko igralec prečka le enkrat (slika 1). Ko jo prečka se spremeni v drugo sliko in če ponovno stopi nanjo pade v luknjo in mora začeti stopnjo znova. Prva stopnja razpoke ima v polju stopnje številko 9, druga stopnja pa 99.

Slika 44: ovira - razpoka (prva stopnja)

Vir: lasten

Slika 45: ovira - razpoka (druga stopnja)

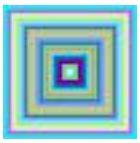
Vir: lasten

```
switch(player.lastDir)
{
    case 'up':
        if(map.level[player.mapCoord.y+1][player.mapCoord.x]==9)
        {
            map.level[player.mapCoord.y+1][player.mapCoord.x]=99;
        }
        break;

    case 'down':
        if(map.level[player.mapCoord.y-1][player.mapCoord.x]==9)
        {
            map.level[player.mapCoord.y-1][player.mapCoord.x]=99;
        }
        break;

    case 'left':
        if(map.level[player.mapCoord.y][player.mapCoord.x+1]==9)
        {
            map.level[player.mapCoord.y][player.mapCoord.x+1]=99;
        }
        break;

    case 'right':
        if(map.level[player.mapCoord.y][player.mapCoord.x-1]==9)
        {
            map.level[player.mapCoord.y][player.mapCoord.x-1]=99;
        }
        break;
}
```

Slika 46: delovanje ovire razpoka

Vir: lasten

Cilj je blok, ki omogoča igralcu napredovanje v naslednjo stopnjo. V primeru da uporabnik izdela svojo stopnjo in jo igra se ob prihodo na cilj stopnja ponovno začne. V polju stopnje ima številko 10.



Slika 47: ovira - cilj

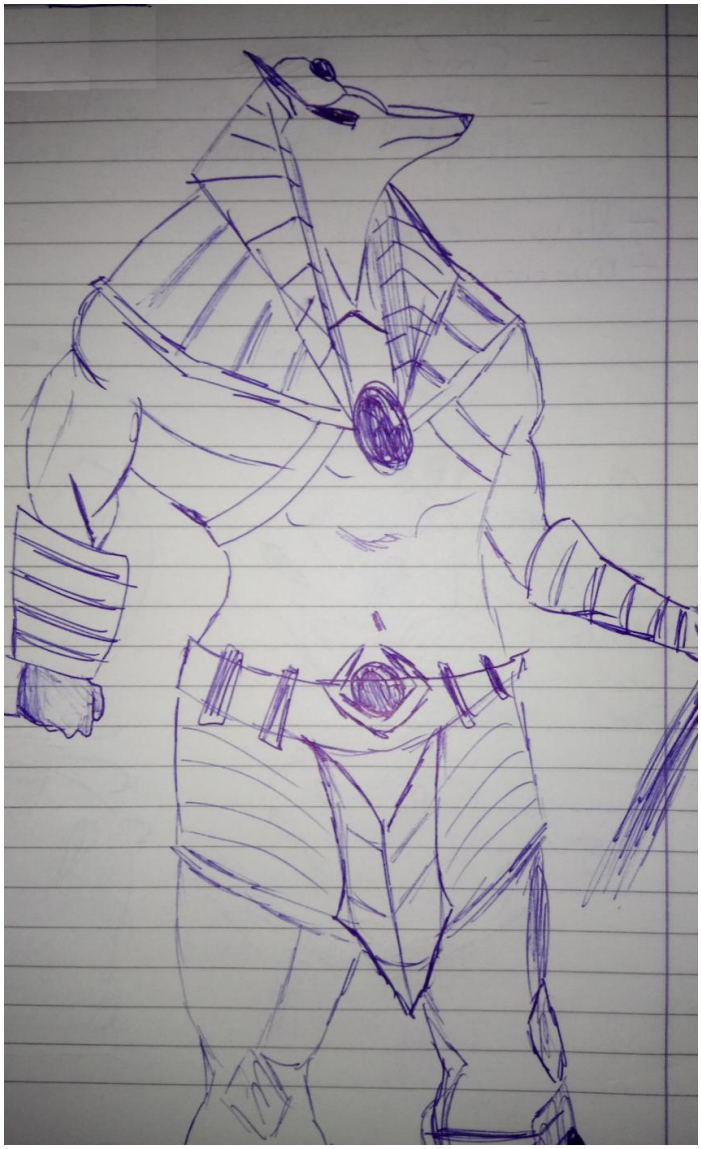
Vir: lasten

6 IZDELAVA GRAFIČNEGA GRADIVA ZA IGRO

Preden sem se lotil risanja kakršnihkoli tekstur sem moral določiti tematiko igre oziroma levelov le te. Odločil sem se, da se bo prvih deset levelov dogajalo v piramidi.

Nekatere teksture sem videl takoj, za druge sem navdih iskal kjerkoli sem ga lahko našel: video-igre iz mojega otroštva, narava, glasba, …

Sliko sem najprej narisal na papir, kjer sem jo večkrat tudi modificiral preden sem jo preko računalnika, s pomočjo programa Gimp pretvoril v pikselno obliko dimenzij 50x50, 25x25 ali poredkoma 150x150.



Slika 48: skica sovražnika

Vir: lasten

Skica

Slika 49: sovražnik (enemy04)

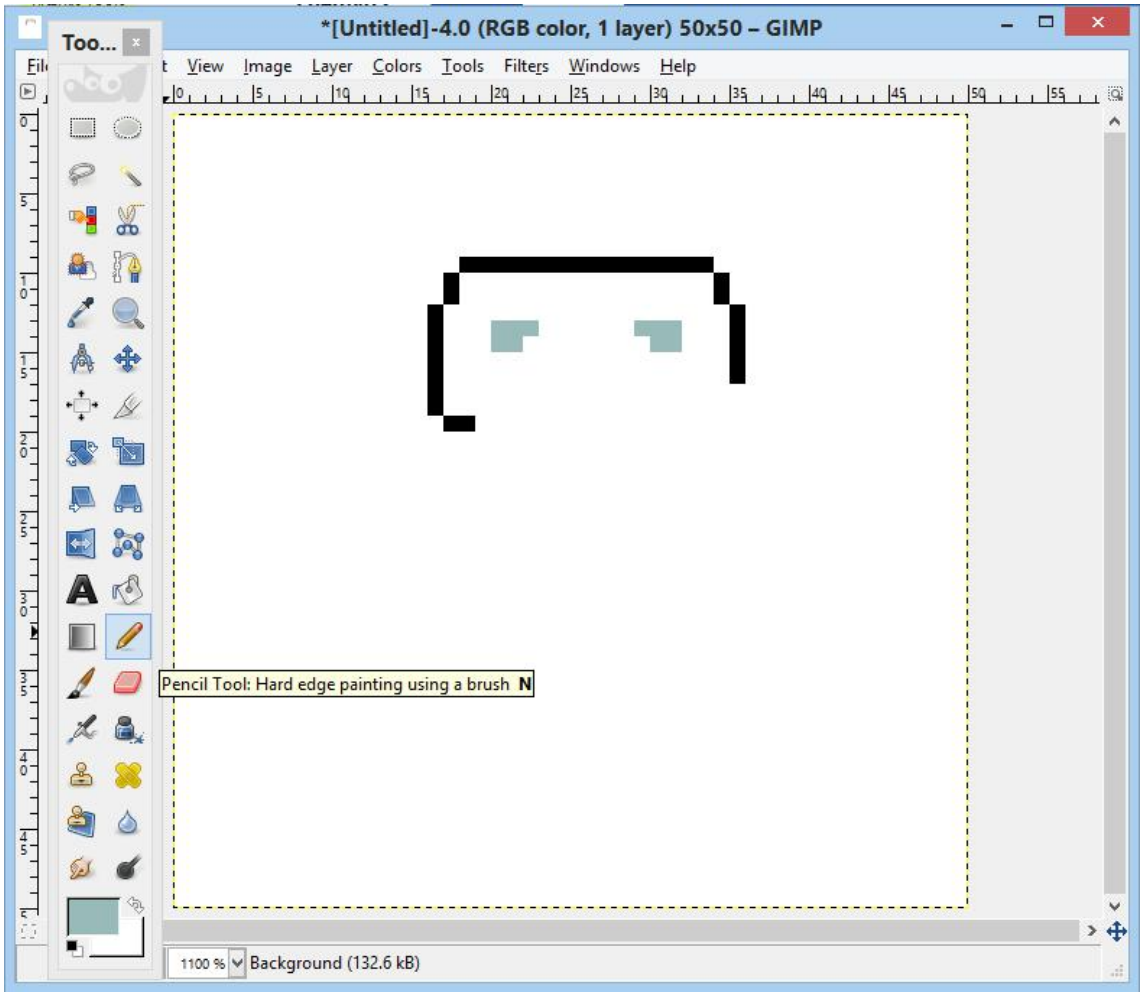
Vir: lasten

6.1 Animacije

Premikajoče-se objekte je bilo potrebno narisati iz vseh štirih strani, poleg tega pa jih je bilo treba tudi animirati, kar jih je naredilo zahtevnejše od ostalih, bolj preprostih tekstur.

6.2 Uporabljene funkcije

Za risanje animacij in tudi navadnih textur sem uporabil, že prej omenjen, program Gimp. Seznam orodij, ki sem jih uporabil je sledeč:

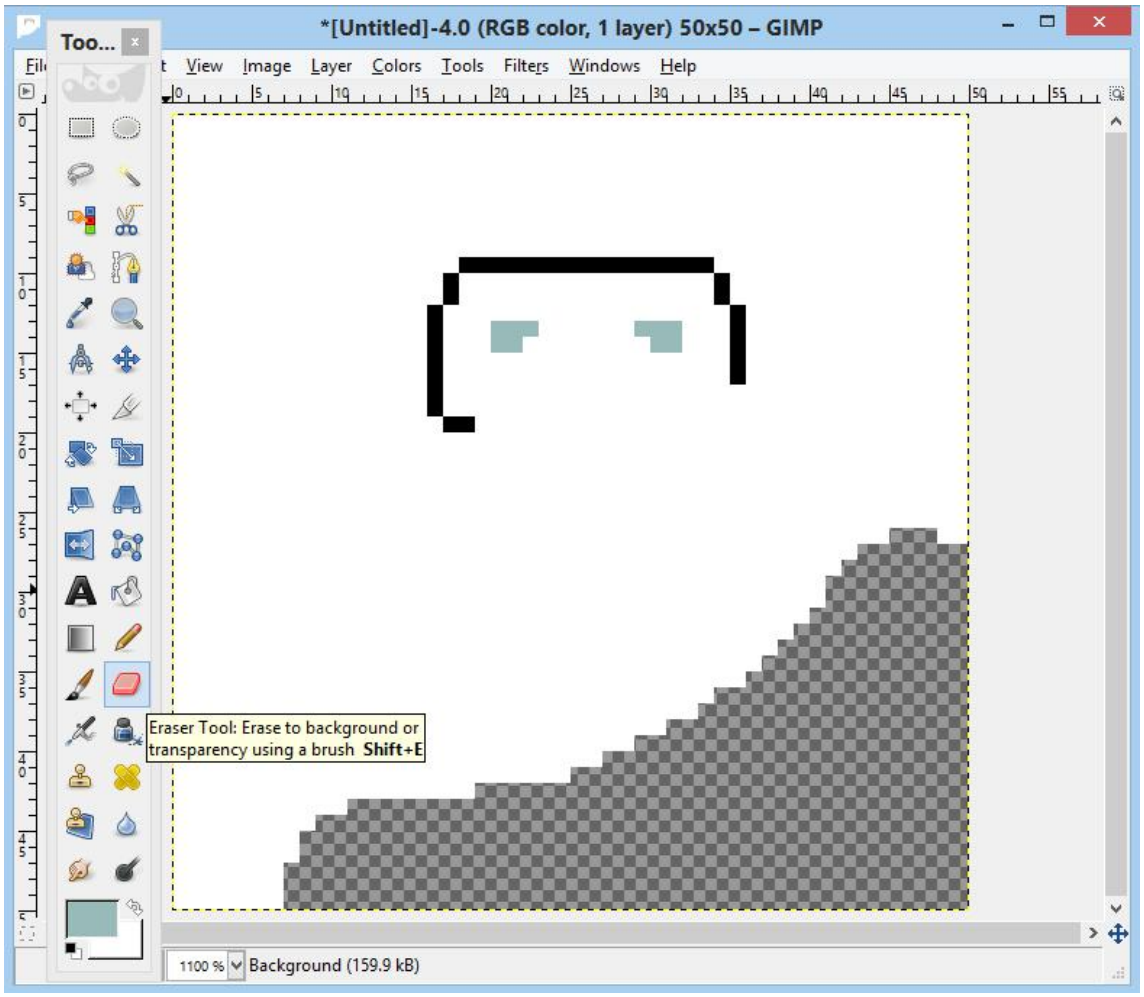


Slika 50: orodje svinčnik

Vir: lasten

Svinčnik (Pencil)

S tem orodjem sem ustvaril veliko večino tekstur, za nekatere nisem uporabil nič drugega.

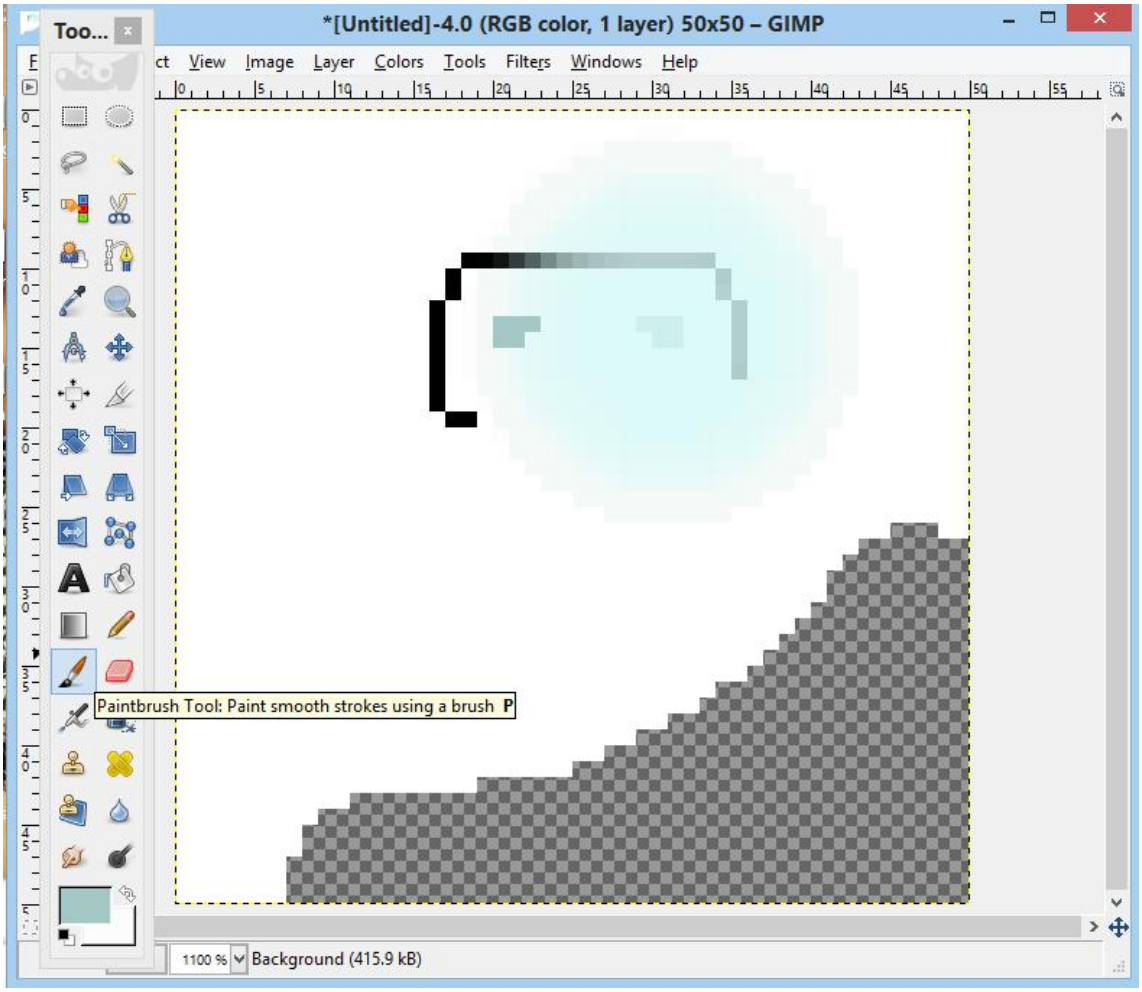


Slika 51: orodje radirka

Vir: lasten

Radirka (Eraser Tool)

To orodje sem uporabil, ko sem hotel označiti kje bo slika transparentna. Za radiranje napak sem uporabil prej opisano orodje, svinčnik (predele, ki sem jih hotel zradirat sem prebarval z belo barvo).



Slika 52: orodje čopič

Vir: lasten

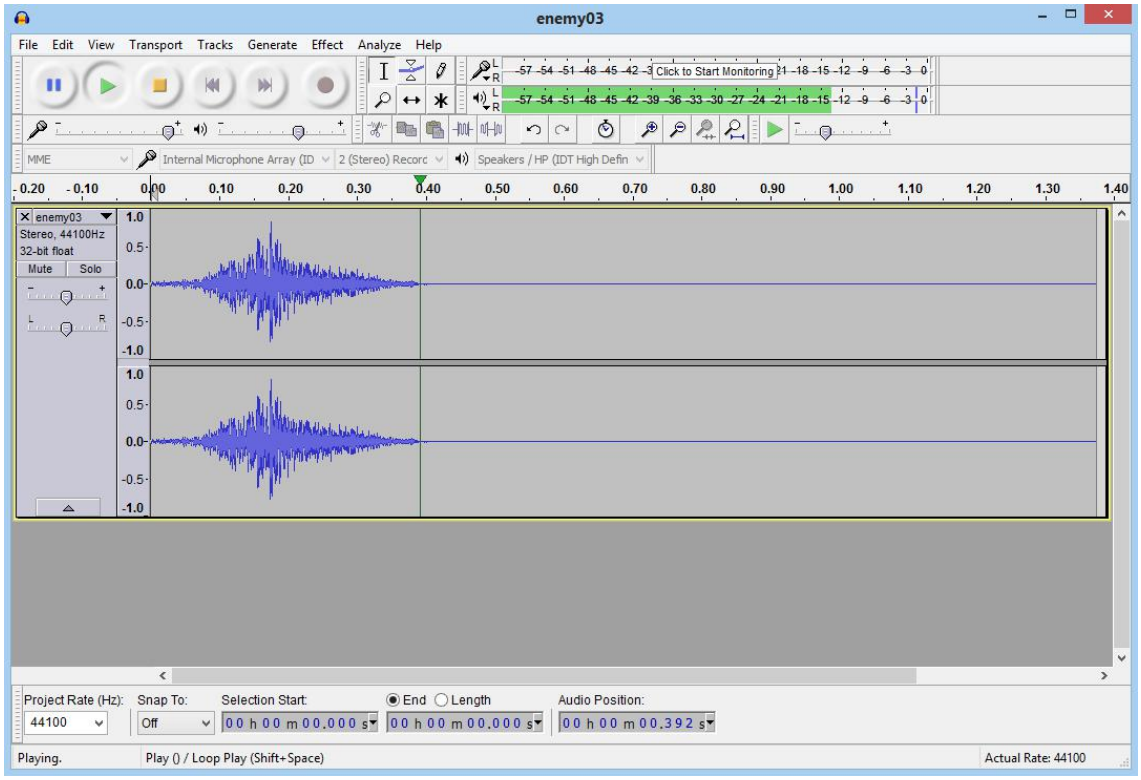
Čopič (Paintbrush)

To orodje sem sem uporabil za senčenje ali prikaz svetlobe. Uporabil sem ga tako da sem prekrivnost nastavil na <50%, velikost pa na >1px.

7 IZDELAVA ZVOČNEGA GRADIVA

Zvoke, ki sem jih potreboval sem iskal na internetu, večinoma na spletni strani https://www.freesound.org/. Zvoke, ki jih nisem našel sem posnel sam.

Ko sem imel vse potrebne zvočne datoteke v mapi, sem vsako posebej uvozil v program za urejanje le the, imenovan Audacity. Datoteke sem nato uredil tako, da je od njih ostal samo željen del npr.(Ko sem snemal zvoke za hojo, sem posnel 30-sekundno datoteko, ki je vsebovala zvoke večih korakov, izmed vseh sem jih izbral pet). Ti zvoki se bodo v zanki ponavljali preko funkcije random(), ki skrbi za to, da skoraj vsak korak zveni drugače od prejšnjega.

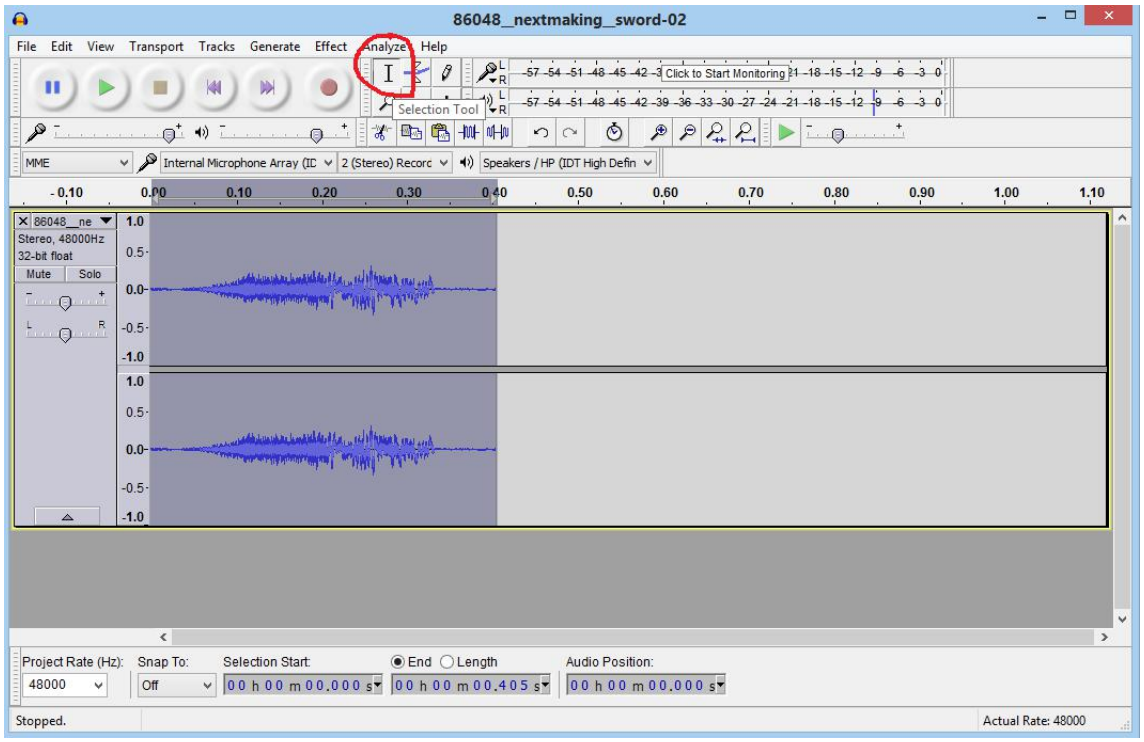


Slika 53: program Audicity

Vir: lasten

Ko sem imel vse željene zvočne datoteke v mapi sem začel ustvarjati zvoke, ki sem si jih zamislil. Za doseganje tega sem uporabil več funkcij, ki jih program ponuja.

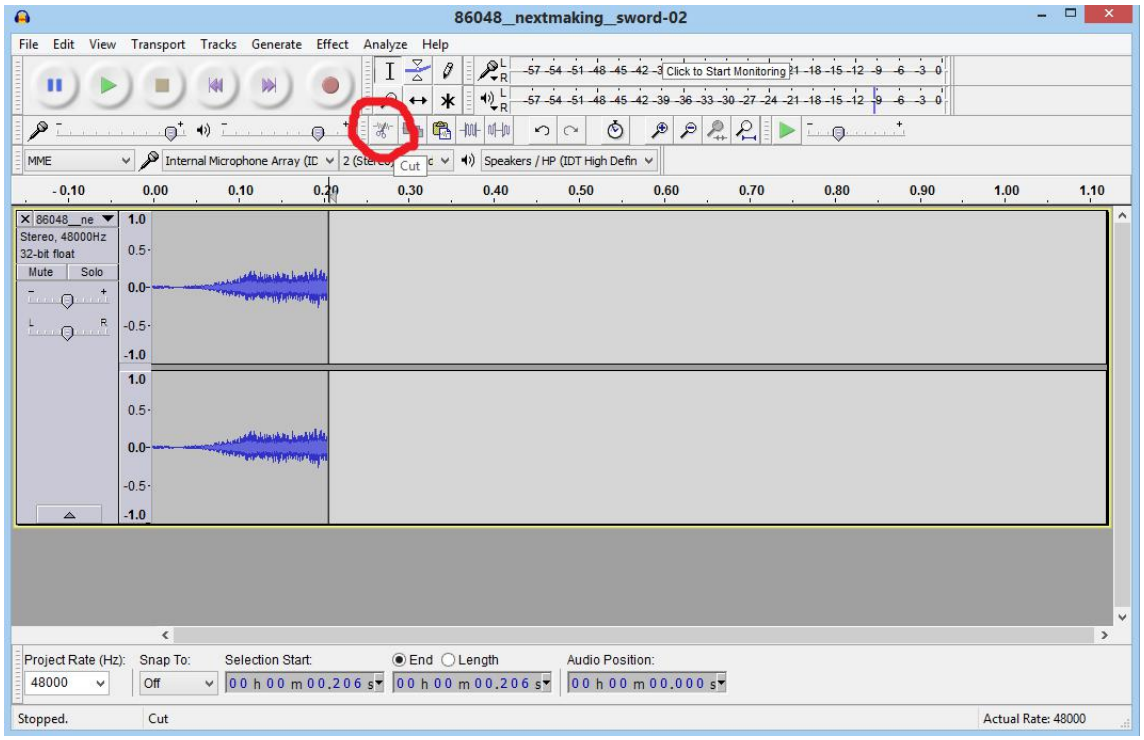
7.1 Uporabljene funkcije



Slika 54: orodje z izbiranje

Vir: lasten

Orodje za izbiranje (Selection Tool).



Slika 55: orodje za izrezovanje

Vir: lasten

Orodje za izrezovanje (Cut).

8 ZAKLJUČEK

Izdelana računalniška igra deluje v brskalnikih neodvisno od operacijskega sistema. Igralec ima možnost igrati že izdelane stopnje, lahko pa izdela svoje in jih shrani v bazo podatkov ali v datoteko. Izdelovanje izdelka je potekalo dokaj hitro, saj sem ob vsakem novem uspešno dodanem elementu igre dobil zagon za nadaljevanje razvoja igre. Probleme, ki so se pojavili, sem reševal s pomočjo spleta in metorja. Tudi če sem se z problemom ukvarjal več dni, je bil zame to izziv in sem lažje reševal nadaljne probleme ali napake. Igro je možno še izboljšati, trenutno vse funkcije igre delujejo brezhibno le v spletnem brskalniku Mozilla Firefox, v ostalih brskalnikih se lahko pojavijo problemi, saj brskalniki različno obravnavajo npr. nalaganje datotek in ostale podobne stvari. Zaenkrat se ob uporabi brskalnika, ki ni Firefox izpiše sporočilo, ki uporabnika obvesti o možnosti napak, v prihodnosti pa nameravam igro prilagoditi, da bo delovala brezhibno še v ostalih popularnih spletnih brskalnikih(Google Chrome, Microsoft Edge, Safari,...).

9 VIRI

Seznam virov:

- Rešitev problema z zakasnitvijo opisanega v povzetku : http://stackoverflow.com/questions/12111277/return-parent-function-from-child-function-on-click

- Rešitev problema z ustvarjanjem datoteke stopnje za prenos: https://jsfiddle.net/rce6nn3z/