Fuzzy Fuzzy Ping Pong

Seminarska naloga, Objektno programiranje 2011



**Nika Grabnar**

**Kaja Prah**

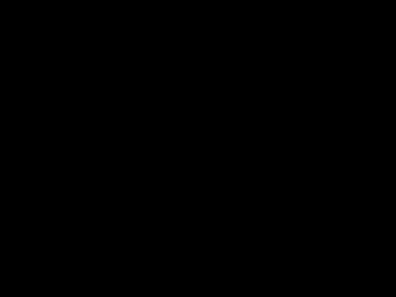
# Scenarij igre

Fuzzy fuzzy Ping-pong je igra, pri kateri mora igralec čim dlje odbijati poskakujočega žogico, da ne pade na ognjeno polje. Le ta pridobiva na hitrosti vsakič, ko ga odbijemo z loparjem. Uporabnik lahko z navigacijskimi tipkami na tipkovnici premika 'lopar' levo in desno. Vsak odboj z loparjem je vreden eno točko. Ko žogica pade na tla se mu hitrost zmanjša, igralec pa izgubi eno življenje ter točko. Vsak igralec ima na začetku igre na voljo tri življenja. Za vsakih 10 zbranih točk se igralcu dodeli novo življenje. Ko porabi vsa življenja, se igra zaključi.

- Fuzzy (žogica) :

fuzzy_fuzzy.png

- ozadje:



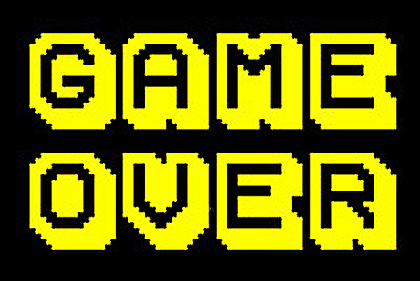
- lopar:

Paddle_blue.png

-ognjeno polje:



-konec:



-števec točk in življenj

# Programski predmeti

Opisali bomo posamezne predmete (razrede) v programu: kakšne so njihove lastnosti, kakšno je njihovo vedenje.

Opišite vaš svet :

* Ozadje je enobarvno, črno ter statično.
* V našem svetu je žogica, lopar ter ognjena past.
* Na začetku se lopar nahaja na sredini na dnu. Lopar igralec z navigacijskimi tipkami premika levo in desno.
* Žogica se premika po celotnem ozadju ter pridobiva na hitrosti vsakič, ko pride do dotika z
* število celic 600x400, in velikost celice 1x1 piksel.

Opišite posamezni predmet:

* Žogica preverja, kje se nahaja v prostoru, ter trke z loparjem in stenami. V kolikor se ne odbije od loparja ali stene pade v ognjeno polje. Premika se naključno glede na odboj od stene in loparja. Ob dotiku z loparjem se števec poveča za eno točko, v primeru da pade mimo loparja se odšteje točka in življenje.
* Lopar preverja, kje se nahaja v prostoru in če je prišlo do trka z žogico. V primeru, da je prišlo, do trka se žogici pospeši hitrost. Lopar upravlja igralec z navigacijskima tipkama levo in desno.
* Stene predstavljajo konec našega sveta, zato se žogica od njih odbija. Če se zaleti v steno (rob našega sveta), se od nje odbije (logaritemsko spremeni smer glede na kot trka). Razen spodnja stena, ki predstavlja ognjeno past.
* Ognjena past je spodnje del pod loparjem. To je past v kateri se znajde naša žogica v primeru, da je ne odbijemo. Ognjena past igralcu odbije eno od treh življenj ter žogico upočasni. V kolikor žogica trikrat pade v ognjeno past se igra zaključi.
* V zgornjem kotu sta nastavljena dva števca. Prvi nam prikazuje naša preostala življenja, drugi pa nam kaže naše točke. Ob vsakem dotiku žogice z loparjem se števec točk poveča za eno. Ob dotiku za ognjeno pastjo pa se števec življenj zmanjša za eno. Igralec lahko dobi eno življenje, tako da zbere deset točk oz. odbojev z loparjem.
* Igra se zaključi, ko igralec porabi vsa razpoložljiva življenja.

# Delovanje programa

Interakcije objektov so trki žogice z objekti (stene, lopar, ognjena past). Ob pritisku tipke se objekt lopar premika levo ali desno. Ostali predmeti niso odvisni od pritiskanja tipk. V vsakem novem trenutku igre objekt žogica preverja položaj ter, ali je prišlo do trka oz. ali je padla v ognjeno past. Ob trku z loparjem se števec poveča za eno točko. Ob padcu v ognjeno past, se odštejeta točka in življenje. Za vsakih 10 zbranih točk, igralec dobi novo življenje.

V kodi bounce žogica preveri ali je prišlo do trka in se odbije ter nadaljuje z enako hitrostjo. S pomočjo matematičnih funkcij funkcije random sva dosegli da se žogica odbije naključno.

V kodi ball smo določili, da ob pričetku igre se žogica začne premikati po našem svetu. V funkciji bounce se preverja ali se je žogica odbila od loparja (Wall) . Ob dotiku z loparjem poveča število točk za eno ter pospeši. V funkciji bounce2 pa se preveri ali se je odbila od ognjenega polja (backWall). Ob pritrdilnem odgovoru se hitrost žogice zmanjša, odšteje število točk in število življenj. Na koncu preveri ali je število življenj enako nič. Če je število življenj(lives) enako nič bo funkcija vrnila true, igra se konča.

Koda Wall preverja ali smo pritisnili gumb levo oz. desno. Ter lopar prestavi v ustrezno stran.

Koda ScoreCounter skrbi za izpis številk ter spreminjanje le teh.

Največ problemov nama je delala ognjena past, ki je bila na začetku pravokotnik, ki se je nahajal pod loparjem na sredini našega igralnega polja.

Najino girico sva nadgradili z odboji in dodatnimi življenji , ki jih igralec pridobi ko doseže 10 točk. Kako sva dosegli da žogica preverja kje se odbija in se pospeši:

public void bounce() //Odbije žogico od loparja(wall). Žogica pospeši, rezultat se poveča, če smo

//žogico odbili že 10 krat dobimo dodatno življenje.

{

setRotation(0-getRotation());

speed = speed + 0.3; //poveča hitrost, ki smo jo določili na začetku

move(speed); //naredi premik glede na novo hitrost

if(getY()>300){

count--; // prišlo je do odboja, ki je pod loparjem (žogica je padla na ognjeno polje), točke se zmanjšajo

}

else

{

count++; // prišlo je do odboja, ki je v višini loparja, točke se povečajo

extraLives++; //povečanje vrednosti spremenljivke extralives

if(extraLives==10){ //ko je vrednost vseh pridobljenih extralives točk enaka 10 pridobimo novo življenje

extraLives = 0;

lives++;

myWally.movingSpeed++; //hitrost se poveča

myWally.movingSpeed++;

}

}

myScoreCounter.setText("Lives: " + lives + " Score: " + count);

}

public void checkBounce2() //Če je žogica na zadnjem zidu(backWall) se bo od njega odbila.

//Rezultat in število življenj se zmanjšata, žogica upočasni.

{

if(getY()>398) { //če se je žogica dotaknica področja na kateremse nahaja ognjeno polje

count--; //rezultat se zmanjša

lives--; //Število življenj se zmanjša

speed = speed - 0.5; // hitrost premikanja žogice se zmanjša

move(speed); //premik glede na novo hitrost

myScoreCounter.setText("Lives: " + lives + " Score: " + count);

}

}