

PRIRODOSLOVNO-MATEMATIČKI FAKULTET
SVEUČILIŠTE U SPLITU

Kristijan Kelić

Flappy dot neuronske mreže i genetski algoritam

Dokumentacija uz projekt

Studij:	Preddiplomski studij
Studijska grupa:	Informatika
Predmet:	Umjetna inteligencija
Akadska godina:	2017/2018
Profesori:	Izv.prof.dr.sc.Saša Mladenović, dr. sc. Goran Zaharija

Sadržaj

1. Uvod.....	1
2. P5.js.....	2
3. Neuronske mreže	5
4. Synaptics.js	7
5. Genetički algoritam.....	9
6. Logika igre	12

1. Uvod

Ideja ovoga projekta je napraviti program kojim će ptica (točka) sama naučiti letjeti. Nadam se da su svi upoznati sa igrom flappy bird u kojoj igrač treba pritisnuti ekran kako bi ptica napravila mah i time prolazio između cijevi. Ja sam odlučio napraviti igru, grafički ne istu, ali ista logika. Postoji više točaka koje trebaju prolaziti između cijevi. Odlučio sam korsititi neuronske mreže i genetički algoritam kako bi iz generacije u generaciju točke naučile letjeti. Igru sam odlučio napraviti u JavaScript programskom jeziku pa sam za grafičko sučelje se poslužio gotovom zbirkom p5.js koja pruža brojne metode za crtanje grafičkih oblika i canvasa. U svrhu neuronskih mreža koristio sam gotovu zbirku synaptics.js, također zbirka JavaScript-a koja pruža metode i logiku funkcioniranja neuronskih mreža.



Slika 1 – prikaz igre

2. P5.js

Kao što sam spomenuo u uvodnom dijelu. Za grafički prikaz korištena je p5.js zbirka. Na sljedećim slikama prikazana je implementacija metoda za stvaranje podloge i likova.

```
JS birdjs x
1 //konstruktor za pticu
2 function Bird(index) {
3   this.boja = color(random(254), random(254), random(254));
4   this.index = index;
5
6   this.ziva = true;
7
8   this.brzina = 0;
9   this.y = random(50, 500);
10  this.x = 50;
11
12  this.trenutna_spremnost = 0;
13  this.trenutni_bodovi = 0;
14  this.prethodna_spremnost = 0;
15  this.prethodni_bodovi = 0;
16 }
17
18 //metoda koja regulira padanje
19 Bird.prototype.update = function() {
20   //povećavamo brzinu padanja ptice, i za taj iznos povećavamo y da ispada ko da ptica pada
21   this.brzina += 0.4;
22   this.y += this.brzina;
23 };
24
25 //metoda koja crta kuglicu(pticu)
26 Bird.prototype.draw = function() {
27   strokeWeight(1);
28   fill(this.boja);
29   ellipse(this.x, this.y, 25, 25);
30 };
31
32 //metoda za pokret ptice
33 Bird.prototype.clap = function(f) {
34   this.brzina = 0;
35   this.brzina += f;
36 };
37
38 //detekcija udara
39 Bird.prototype.sudar = function(pipe) {
40   if(this.y - 15 <= pipe.y - 65 || this.y + 15 >= pipe.y + 65){
41     if (this.x + 15 > pipe.x && this.x + 15 <= pipe.x + 60) return true;
42   }
43 }
```

Slika 2 – implementacija i crtanje ptice

Na slici iznad možemo vidjeti metode za stvaranje ptice. Metoda `Bird` je konstruktor, metoda koja stvara objekt ptice i svojstva svake ptice. Svaka ptica ima brzinu, točku x i y na kojoj se stvara i varijable za spremanje podataka o bodovima i spremnosti. `Update` je metoda iz zbirke p5.js koja se izvršava konstantno i ona simulira pad ptice. U toj metodi se vrijednost y od ptice smanjuje za iznos brzine. `Draw` metoda služi za prikaz ptice (točke). Također p5.js metoda koja crta grafički prikaz, svaka ptica je prikazana kao elipsa ispunjena radnom bojom. `Clap` metoda je metoda koja obavlja let, odnosno zamah krilima kako ptica nebi konstantno padala. `Sudar` je metoda koja regulira sudar ptice i cijevi.

```

//konstruktor za cijev
function Pipe(x, y, color) {
    this.x = width;
    this.y = y;
    this.color = color;
}

Pipe.prototype.update = function() {
    this.x -= 3.5;
};

//metoda za crtanje cijevi
Pipe.prototype.draw = function() {
    fill(this.color);
    strokeWeight(0);
    //rect(x-cor, y-cor, sirina, visina)
    //cijevi gornje
    rect(this.x, 0, 60, this.y - 65);
    //cijevi donje
    rect(this.x, this.y + 65, 60, height - this.y - 65);
};

```

Slika 3 – implementacija i crtanje cijevi

Za implementiranje cijevi vrijede ista pravila kao i za pticu, koriste se metode update i draw, te konstruktor cijevi. Svaka cijev je predstavljena kao 2 četverokuta, jedan se crta na gornjoj granici svijeta, a drugi na donjoj i između njih je ostavljena praznina kao prostor kuda točka treba proći.

```

JS flappyDot.js x
1  //globalne varijable
2  var birds = [];
3  var deadBirds = [];
4  var pipes = [];
5  var GA = new GeneticAlgorithm(20, 8);
6
7  var bodovi = 0;
8  var udaljenost = 0;
9  var put = 0;
10 var najboljiPut = 0;
11
12 //metoda koja postavlja okolinu i likove
13 function setup() {
14     createCanvas(window.innerWidth-150, window.innerHeight - 200);
15
16     for (var i = 0; i < GA.broj_ptica; i++) {
17         birds.push(new Bird(i));
18     }
19     pipes.push(new Pipe(width, random(height * 0.3, height * 0.7), color('green')));
20     cijevIspred = pipes[0];
21     GA.reset();
22     GA.stvoriPopulaciju();
23 }
24
25 //metoda koja crta okolinu i likove
26 function draw() {
27     background('nightsky');
28
29
30     if (frameCount % 100 === 0) {
31         pipes.push(new Pipe(width, random(height * 0.3, height * 0.7), color('green')));
32     }
33
34     for (var i = 0; i < pipes.length; i++) {
35         pipes[i].update();
36         pipes[i].draw();
37         if (pipes[i].x < -40) {
38             pipes.shift();
39             bodovi += 1;
40             cijevIspred = pipes[0];
41         }
42     }

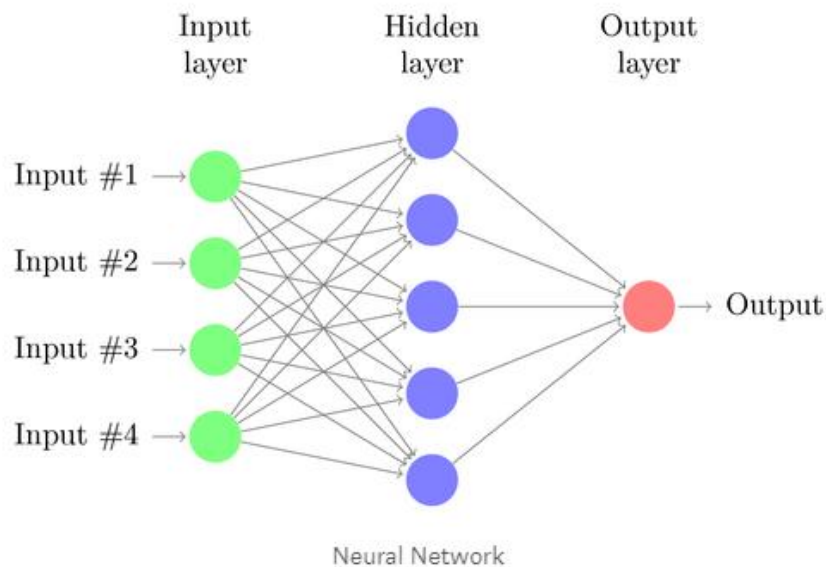
```

Slika 4 – kod za crtanje skice i postavljanje svijeta

Na slici iznad nalazi se glavna metoda `setup` koja stvara svijet, skicu i puni niz ptica sa novim pticama, i stvara cijevi.

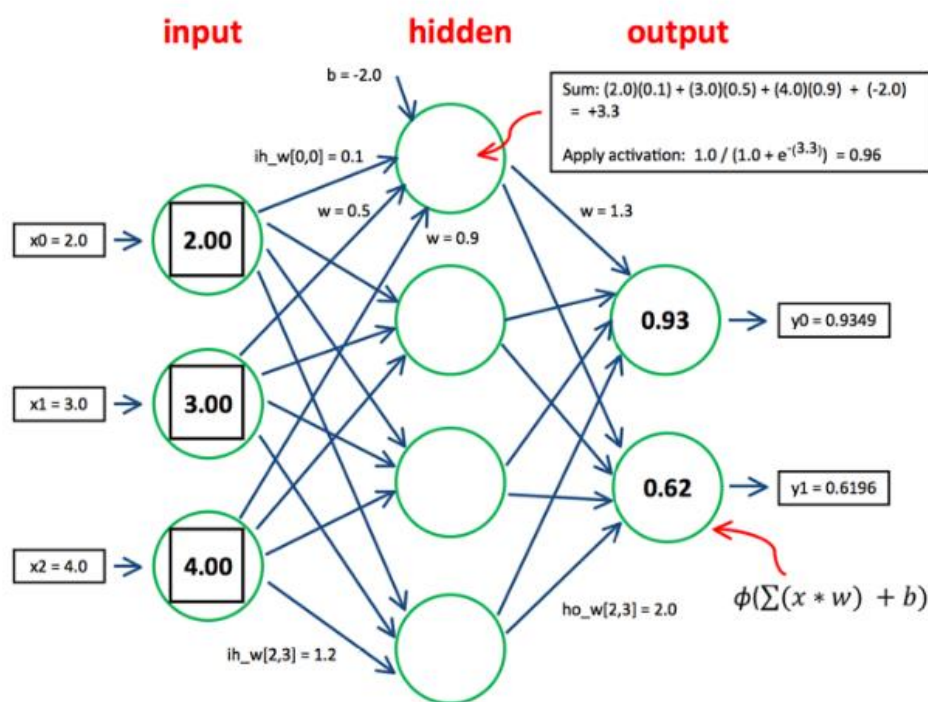
3. Neuronske mreže

Prije svega, želim razgovarati o onoj neuronskoj mreži. U programiranju, umjetna neuronska mreža je računalni model / algoritam za strojno učenje koji je inspiriran strukturom i funkcionalnim aspektima bioloških neuronskih mreža.



Slika 5 – prikaz neuronske mreže

Svaka neuronska mreža ima jedan ulazni sloj, jedan izlazni sloj i jedan ili više skrivenih slojeva. Svaki krug predstavlja neuron, a neuron ima vezu sa svakim neuronom u sljedećem sloju. Svaka veza ima vrijednost težine, a svaki neuron ima vrijednost pristranosti. Na primjer, ispod dijagrama prikazuje što se događa u neuronskoj mreži.



Slika 6 – demonstracija izračuna izlaza

Kada neuronska mreža izračunava izlaz, tu ima puno matematike s težinom i pristranošću. Jednostavno možete očekivati da ako izmijenim jednu od vrijednosti težine ili pristranošću, konačni se izlaz također mijenja. Drugim riječima, neuronska mreža za obuku znači pronalaženje i podešavanje vrijednosti težine i pristranošću koja daje najbolje rezultate koje želimo.

4. Synaptics.js

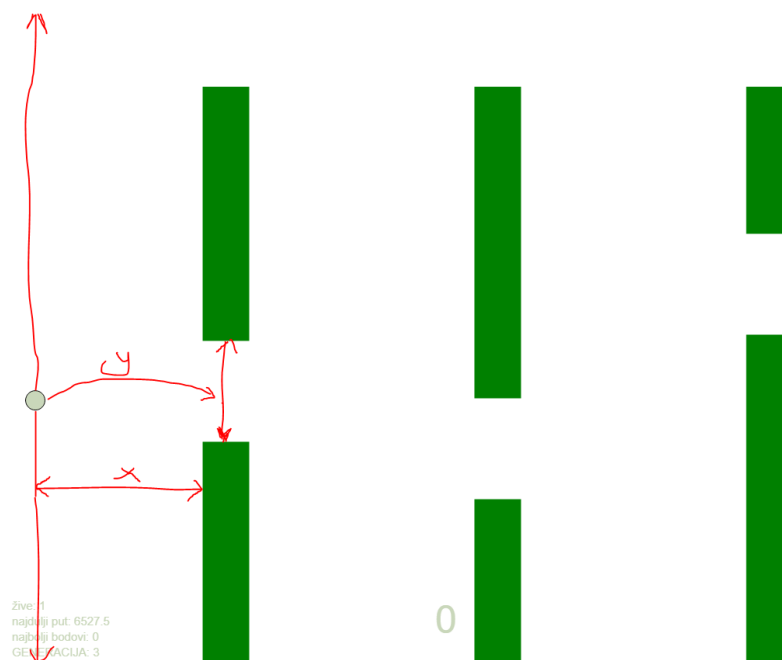
Synaptics.js je gotova zbirka koja sadrži metode i logiku funkcioniranja neuronskih mreža u JavaScript programskom jeziku.

```
GeneticAlgorithm.prototype.stvoriPopulaciju = function() {  
  // očisti neku postojeću populaciju  
  this.populacija.splice(0, this.populacija.length);  
  
  for (var i=0; i<this.broj_ptica; i++) {  
    // stvori novu pticu generiranjem ranomd Synaptic neuronske mreže  
    // sa 2 neurona u ulaznom, 6 neurona u skrivenom te 1 neuronu u izlaznom layeru  
    var novaJedinica = new synaptic.Architect.Perceptron(2, 6, 1);  
  
    // dodatni parametri za novu jedinicu  
    novaJedinica.index = i;  
    novaJedinica.spremnost = 0;  
    novaJedinica.bodovi = 0;  
    novaJedinica.jePobjednik = false;  
  
    // dodaj novu jedinicu u populaciju  
    this.populacija.push(novaJedinica);  
  }  
};
```

Slika 7 - metoda koja stvara neuronsku mrežu

Na slici iznad generira se populacija neuronski mreža. Svaka ptica posjeduje svoju neuronsku mrežu koja se sastoji od 2 neurona u ulaznom sloju, 6 neurona u skrivenom i 1 neuronu u izlaznom sloju.

Za ulazne parametre koristio sam udaljenost ptice od cijevi ispred i položaj ptice u odnosu na prolaz između cijevi.



Slika 8 – vizualni prikaz ulaza

Svaki put kada se pozove `draw` metoda neuronska mreža za svaku pticu dobije ulazne vrijednosti i aktivira ih te da izlaz. Ukoliko je izlaz veći od 0.5 ptica će skočiti.

```
41 GeneticAlgorithm.prototype.reagiraj = function(ptica, pipe) {
42     // ulaz 1: horizontalna udaljenost između ptice i pipe-a
43     var horizontaloX = pipe.x - 50;
44     // ulaz 2: razlika u visini između ptice i sredine između pipe-ova
45     var razlikaVisina = pipe.y - ptica.y;
46
47     // niz svih ulaza
48     var ulazi = [horizontaloX, razlikaVisina];
49
50     // izračunaj izlaz aktivirajući synaptic neural network za pticu
51     var izlaz = this.populacija[ptica.index].activate(ulazi);
52
53     // napravi clap ako je izlaz veći od 0.5
54     if(izlaz[0] > 0.5) ptica.clap(-8);
55 };
56
```

Slika 9 – metoda za aktiviranje skoka

5. Genetički algoritam

Za treniranje neurona i učenje nisam mogao koristiti gotovu propagate metodu iz synaptics.js-a jer ona zahtjeva neke testne podatke koje ja nemam. Za to smo koristili genetički algoritam da trenira neuronsku mrežu. U svakoj generaciji rodi se 20 ptica i svaka ptica ima svoju neuronsku mrežu sa 2 ulaza, 6 skrivenih i 1 izlaznim neuronom. Nakon svake generacije odabire se 8 najboljih ptica i nad njima se vrši crossover i mutacija. I proces se svaki put ponavlja dok ne dobijemo dobre rezultate i jedna ili više ptica počne letjeti bez problema.



```
56
57 // metoda za selekciju
58 GeneticAlgorithm.prototype.selekcija = function () {
59     // sortirati jedinice trenutne populacije u silazećem poretku po spremnosti
60     var sortiranaPopulacija = this.populacija.sort(
61         function (unitA, unitB) {
62             return unitB.spremnost - unitA.spremnost;
63         }
64     );
65
66     for (var i = 0; i < this.najbolje_ptice; i++) this.populacija[i].jePobjednik = true;
67
68     return sortiranaPopulacija.slice(0, this.najbolje_ptice);
69 };
70
71
72 GeneticAlgorithm.prototype.crossOver = function (roditeljA, roditeljB) {
73     var cutPoint = round(random(0, roditeljA.neurons.length - 1));
74     // zamijeni informaciju između dva roditelja
75     // 1. lijeva strana je kopirana od jednog roditelja
76     // 2. desna strana poslije crossovera je kopirana od drugog roditelja
77     for (var i = cutPoint; i < roditeljA.neurons.length; i++) {
78         var biasRoditeljA = roditeljA.neurons[i]['bias'];
79         roditeljA.neurons[i]['bias'] = roditeljB.neurons[i]['bias'];
80         roditeljB.neurons[i]['bias'] = biasRoditeljA;
81     }
82
83     return random(0, 1) == 1 ? roditeljA : roditeljB;
84 };
85
86 GeneticAlgorithm.prototype.mutacija = function (potomak) {
87     // mutiraj bias informacije potomkovih neurona
88
89     for (var i = 0; i < potomak.neurons.length; i++) {
90         potomak.neurons[i].bias = this.mutiraj(potomak.neurons[i].bias);
91     }
92
93     for (var i = 0; i < potomak.connections.length; i++) {
94         potomak.connections[i].weight = this.mutiraj(potomak.connections[i].weight);
95     }
96     return potomak;
97 };
98
```

Slika 10 – metode za selekciju, crossover i mutaciju

Metoda **selekcija** sortira populaciju neuronskih mreža po najboljoj spremnosti. Ptica koja najviše dogura ima najveću spremnost. I prvih 8 ptica postavlja vrijednost svojstva jePobjednik na true.

Metoda **crossOver** odabire random mjesto na kojem će vršiti se crossOver unutar niza neurona za mrežu roditelja. Nakon što se odabere mjesto križanja zamjene se vrijednosti bias-a svakog neurona kod dva roditelja pobjednika i vrati se ili roditeljA ili roditeljB.

Metoda **mutacija** mijenja vrijednosti bias-a i weight-a svakog neurona uz pomoć metode mutiraj koja je prikazana na slici ispod.

```
GeneticAlgorithm.prototype.mutiraj = function (gen) {  
  if (Math.random() <= this.mutacijaRating) {  
    var mutacijaFaktor = 1 + ((Math.random() - 0.5) * 3 + (Math.random() - 0.5));  
    gen *= mutacijaFaktor;  
  }  
  
  return gen;  
};
```

Slika 11 – metoda mutiraj

Metoda prima brojčanu vrijednost weight-a i bias-a i ukoliko je neki nasumično generiran broj uz pomoć Math.random() js metode manji ili jednak varijabli mutacijaRating koja je na početku 1, a ukoliko neka ptica prođe prvu cijev se smanji na 0.2, vrši se mutacija tog primljenog iznosa za gore navedeni iznos.

```

115 // evolucija nad populacijom vrši se po selekciji, križanju i mutaciju jedinica
116 GeneticAlgorithm.prototype.evolutija = function() {
117     // odaberi top jedinice trenutne populacije
118     // bit će kopirane u iduću populaciju
119     var pobjednici = this.selekcija();
120
121     if(this.mutacijaRating == 1 && pobjednici[0].bodovi == 0) {
122         this.stvoriPopulaciju();
123     }
124     else{
125         this.mutacijaRating = 0.4;
126     }
127
128     for(var i= this.najbolje_ptice; i<this.broj_ptica; i++){
129         var roditeljA, roditeljB, potomak;
130
131         if(i == this.najbolje_ptice){
132             roditeljA = pobjednici[0].toJSON();
133             roditeljB = pobjednici[1].toJSON();
134             potomak = this.crossOver(roditeljA, roditeljB);
135         }
136         else if(i < this.broj_ptica-2){
137             roditeljA = this.dohvatiNasumicno(pobjednici).toJSON();
138             roditeljB = this.dohvatiNasumicno(pobjednici).toJSON();
139             potomak = this.crossOver(roditeljA, roditeljB);
140         }
141         else{
142             potomak = this.dohvatiNasumicno(pobjednici).toJSON();
143         }
144
145         potomak = this.mutacija(potomak);
146
147         var novaJedinica = synaptic.Network.fromJSON(potomak);
148         novaJedinica.index = this.populacija[i].index;
149         novaJedinica.spremnost = 0;
150         novaJedinica.bodovi = 0;
151         novaJedinica.jePobjednik = false;
152
153         this.populacija[i] = novaJedinica;
154     }
155 }

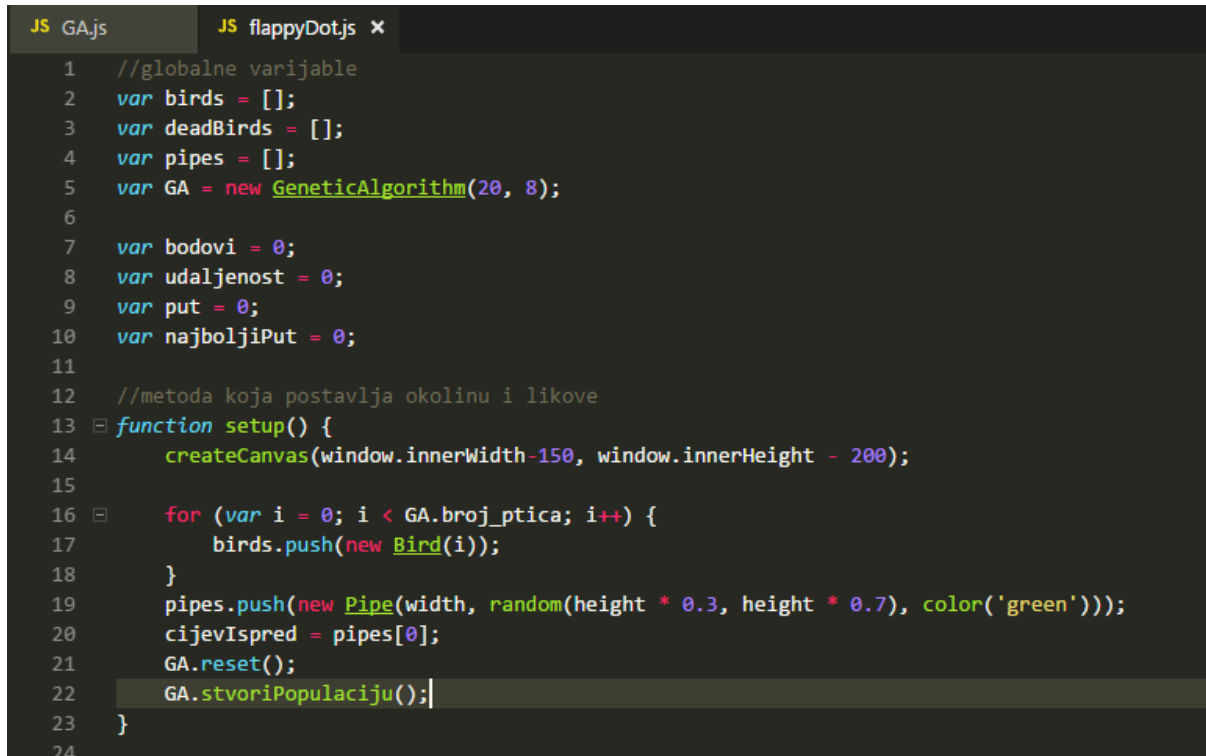
```

Slika 12 – glavna metoda evolucija

Evolucija je glavna metoda koja se izvršava kod svake izmjene generacije i u njoj se pozivaju sve gore navedene metode i vrše se „genetske promjene“ nad roditeljima i stvaraju se novi potomci.

6. Logika igre

Sva logika igre je implementirana u glavnoj `draw` metodi koja se izvršava konstantno.



```
JS GA.js JS flappyDot.js x
1 //globalne varijable
2 var birds = [];
3 var deadBirds = [];
4 var pipes = [];
5 var GA = new GeneticAlgorithm(20, 8);
6
7 var bodovi = 0;
8 var udaljenost = 0;
9 var put = 0;
10 var najboljiPut = 0;
11
12 //metoda koja postavlja okolinu i likove
13 function setup() {
14     createCanvas(window.innerWidth-150, window.innerHeight - 200);
15
16     for (var i = 0; i < GA.broj_ptica; i++) {
17         birds.push(new Bird(i));
18     }
19     pipes.push(new Pipe(width, random(height * 0.3, height * 0.7), color('green')));
20     cijevIspred = pipes[0];
21     GA.reset();
22     GA.stvoriPopulaciju();
23 }
24
```

Slika 13 – setup metoda i stvaranje ptica

Na slici iznad je uvodni kod za stvaranje igre. Navedene su potrebne globalne varijable te u metodi setup se stvara skica, ptice, cijevi i populacija neuronskih mreža za svaku pticu.

```

25 //stvara koja cija kreiraju i iskre
26 function draw() {
27     background('nightsky');
28
29
30     if (frameCount % 100 === 0) {
31         pipes.push(new Pipe(width, random(height * 0.3, height * 0.7), color('green')));
32     }
33
34     for (var i = 0; i < pipes.length; i++) {
35         pipes[i].update();
36         pipes[i].draw();
37         if (pipes[i].x < -40) {
38             pipes.shift();
39             bodovi += 1;
40             cijevIspred = pipes[0];
41         }
42     }
43
44     for (var b = 0; b < birds.length; b++) {
45         cijevIspred = pipes[0];
46         put += 3.5;
47         udaljenost += Math.abs(cijevIspred.x);
48         birds[b].update();
49         birds[b].draw();
50         GA.reagiraj(birds[b], cijevIspred);
51         birds[b].trenutna_spremnost += udaljenost - (cijevIspred.x - birds[b].x);
52         birds[b].trenutni_bodovi = bodovi;
53
54         if (birds[b].x > cijevIspred.x + 60) {
55             cijevIspred = pipes[1];
56         }
57         if (birds[b].y - 10 < 0 || birds[b].y + 10 > height || birds[b].sudar(cijevIspred)){
58             GA.populacija[birds[b].index].spremnost = birds[b].trenutna_spremnost;
59             GA.populacija[birds[b].index].bodovi = birds[b].trenutni_bodovi;
60             deadBirds.push(birds[b]);
61             birds.splice(b, 1);
62         }
63         if (birds.length === 0){
64             GA.evolutija();
65             GA.iteracija += 1;
66             if (put > najboljiPut) najboljiPut = put;
67             bodovi = 0;
68             put = 0;
69             udaljenost = 0;
70             pipes.splice(0, pipes.length - 1);
71             deadBirds.sort(function(a, b){
72                 return a.index - b.index;
73             });
74             for (var j = 0; j < deadBirds.length; j++){
75                 birds.push(new Bird(j));
76                 birds[j].prethodna_spremnost = deadBirds[j].trenutna_spremnost;
77                 birds[j].prethodna_bodovi = deadBirds[j].trenutni_bodovi;
78             }
79             deadBirds = [];
80         }
81     }
82     textSize(50);
83     text(bodovi, width/2, height - 50);
84     textSize(15);
85     text("žive: " + birds.length, 20, height - 80);
86     text("najdulji put: " + najboljiPut, 20, height - 60);
87     text("najbolji bodovi: " + GA.najbolji_bodovi, 20, height - 40);

```

Slika 14 – draw metoda i logika igre

Cijela logika igre nalazi se unutar draw metode koja se izvršava konstantno. Ukratko opisan kod počevši od vrha. Svaki 100 frame-ova stvaramo novu cijev i for petljom za sve cijevi unutar niza cijevi pozivamo metode draw i update za svaku cijev kako bi se cijevi prikazale i pomicale na skici. Zatim svaki put kada bi cijev izašla van lijeve granice svijeta izbacili bih je iz niza i kao sljedeću cijev bi postavili onu iza nje, tj iduću koja dolazi. Druga for petlja prolazi kroz sve ptice unutar niza ptica. Tu se računa prijedeni put, spremnost svake ptice i pozivaju se metode draw i update za pticu kako bi se nacrtale ptice i imali vizualni prikaz leta. Zatim se poziva za svaku neuronsku mrežu metoda reagiraj kako bi ptica izvršila clap. Ukoliko ptica se zabije u cijev ili izađe iz granica svijeta sprema se u niz

mrtvih ptica i izbacuje iz niza ptica. Kada sve ptice nestanu poziva se metoda evolucija u kojoj se vrši genetski algoritam i mutacija, crossOver i selekcija. Cijevi se prazne i ponovno sve kreće iz početka.