**PRIRODOSLOVNO-MATEMATIČKI FAKULTET   
SVEUČILIŠTE U SPLITU**

Alen Bašić i Luka Kovačušić

**Sezona Sniženja**

Dokumentacija

**Studij:** Preddiplomski studij

**Studijska grupa:** Informatika

**Predmet**: PROMA

**Ak. god.**: 2017/2018.

**Nastavnici**: Prof. dr. sc. Saša Mladenović, Prof. dr. sc. Goran Zaharija

Split, 2018.

**Sadržaj**

[1. Uvod 3](#_Toc507084965)

[2. Igra 4](#_Toc507084966)

[3. Razjašnjena logika neuronske mreže 4](#_Toc507084967)

[3.1 Kreiranje neuronske mreže 4](#_Toc507084968)

[3.2 Pružanje mogućnosti igranja 5](#_Toc507084969)

[3.3 Metoda koja omogućuje interakciju i učenje 6](#_Toc507084970)

[3.1.1 Slučaj kada neuronska mreža igra 6](#_Toc507084971)

[3.1.2 Slučaj kada neuronska mreža uči (poraz) 8](#_Toc507084972)

[Zaključak 10](#_Toc507084973)

# Uvod

U svrhu ovoga projekta, nakon neuspjelih pokušaja sa v-repom, pokušao sam obraditi sasvim mali dio neuronskih mreža i prikazati njegovu funkcionalnost na nekom jednostavnom i vizualnom primjeru.

Projekt „Self Learning Dino“ se zasniva sa višeslojnim neuronskim mrežama sa 3 ulaza (udaljenost od prepreke, veličina prepreke i brzina kojom se prepreke primiču T-Rexu).

Neuronska mreža također ima jedan skriveni sloj sa 9 skrivenih neurona uzetih na osnovu osobne procjene kvalitete igranja same igre, ali i na osnovu potreba same igre. Također imamo i jedan izlazni sloj sa jednim izlaznim neuronom koji nam uz pomoć sigmoidne aktivacijske funkcije vraća vrijednosti u rasponu [-1,1].

Kako bi omogućili neko učenje na greškama, koristili smo unazadnu propagaciju, tako da neuronska mreža uči sama, bez ikakve potrebe za intervencijom iz vana.

Još neke od bitnijih stvari za sami projekt su igra koja je modificirana na način da radi bolje učinkovitosti modificirana na način da su uklonjena prepreke koje se izbjegavaju saginjanjem ali i sama funkcija saginjanja.

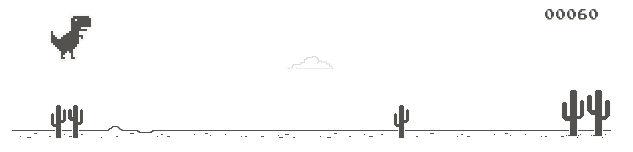
Za potrebe ovoga projekta je cijela logika smještena u okruženje koje omogućuje izgradnju projekta za korištenje na pametnim telefonima (Phonegap).

Projekt je rađen uz pomoć web tehnologija (HTML, CSS, JavaScript).

# Igra

Igra neslužbeno zvana „T-Rex runner“ sastavni je dio novijih verzija Google Chromea, realizirana pomoću standardnih web tehnologija, te uz pomoć service workera integrirana na „Unable to connect to internet“ stranicu.

Kompletna replika igre „T-Rex runner“ je djelo [Wayou Liu-a](https://github.com/wayou/t-rex-runner).



Realizacija ovoga dijela i nije bas srediste ovoga projekta pa se neće posebno razmatrati.

Bitno je samo spomenuti da je cijela logike ove igre smještena u datoteci „www/js/index.js“. Bitni unutar HTML-a „www/index.html“ su smješteni sami DOM-elementi koji se dohvaćaju u „www/js/index.js“, te unutar datoteke „www/css/indeks.css“ imamo sve stilove vezane za relevantnu igru.

# Razjašnjena logika neuronske mreže

Za potrebe ovoga projekta nije korištena vlastita logika za neuronsku mrežu već cijeli gotovi paket „synaptic.js“ koji sadrži veliki broj elemenata vezan za strojno učenje. U projektu su samo korišteni gotovi elementi na način da pokušavamo složiti dobru logiku igre, koja će koristeći već gotove elemente stvarno ponuditi mogućnost učenja na greškama.

## Kreiranje neuronske mreže

Prilikom igranja i pomnog promatranja, ali i istraživanja na internetu, bilo je očito da prilikom igranja „T-Rex runner“ igre, najviše trebamo obratiti pažnju na 3 bitne stvari:

* Udaljenost od sljedeće prepreke
* Veličina prepreke
* Brzina prilaženja prepreci

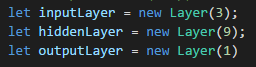
Na osnovu tih elemenata u stvarnom životu odlučujemo želimo li da nam T-Rex skoči ili ne.

Cijeli kod vezan za neuronske mreže se nalazi u 2 datoteke unutar direktorija www/js/ :

* Synaptic.js
* Brain.js

U predhodnim verzijama smo radili neuronsku mrežu sa 2 neurona u izlaznom sloju, no kako je uočeno da imamo igru možemo relizirati u samo 2 izlazna stanja, izlazni sloj je promjenjen na način da je realiziran sa samo jednim izlaznim neuronom kojem je izlazna vrijednost podjeljana na način da „T-Rex“ skače u slučaju kada mu je izlazna vrijednost veća ili jednaka 0.5, a u protivnom koristi akciju „DOWN“.

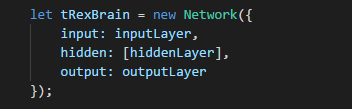
Što se tiče skrivenog sloja, realiziran je pomoću 9 skrivenih neurona do kojih se došlo metodom pokušaja-pogreške.



Nakon što smo kreirali sve potrebne slojeve sa neuronima, bilo je potrebno jos ih sve povezati. To je realizirano puten metode sloj1.project(sloj2).



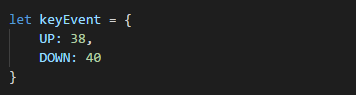
Nakon što smo ih povezali trebali smo još sve dodatno spakirati u jednu mrežu. U našem slučaju ta mreža se zove tRexBrain.



## Pružanje mogućnosti igranja

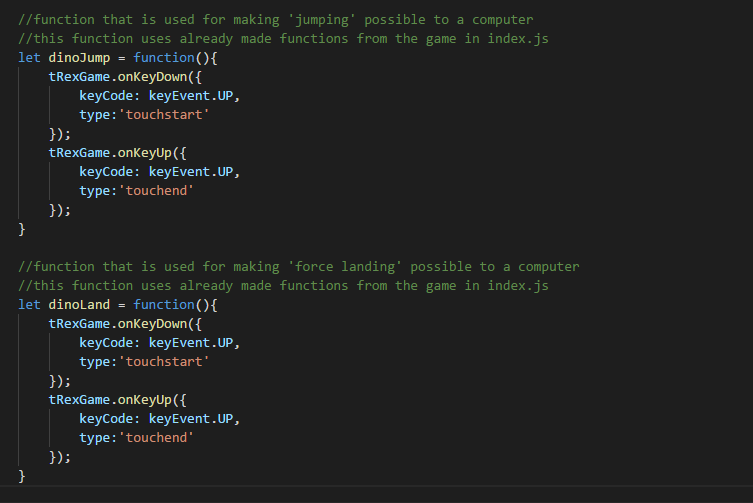
Kako bi omogućili neuronskoj mreži da koristi kontrole kao i igrač izvana, koristili smo neke već prethodno definirane metode u implementaciji same igre.

Prvo smo napravili objekt keyEvenet gdje smo definirali vrijednosti tipke gore i dolje.



Nakon što smo to napravili, bilo je potrebno definirati metode koje će „izvana “ koristiti metode definirane u indx.js.

Definirali smo metode dinoJump i dinoLand.



Time smo obuhvatili cijeli dio vezan za realizaciju metoda koje su posredne neuronskim mrežama, te samoj igri.

## Metoda koja omogućuje interakciju i učenje

Kako bi omogućili interakciju neuronske mreže i igre konačno smo kreirali i metodu koja to omogućuje. Metoda se zove neuralNetworkPlay.



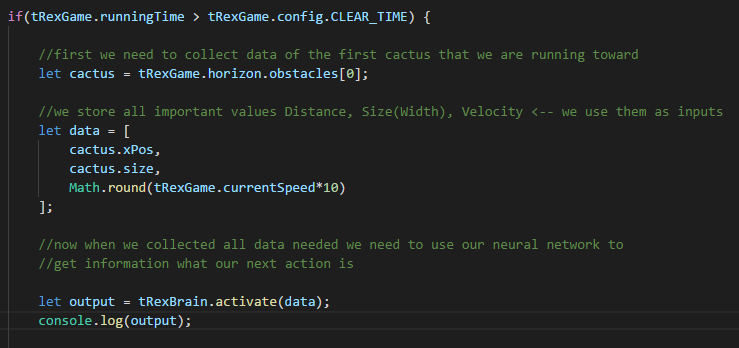
Na dalje smo u toj metodi razdvojili dva slučaja, slučaj kada neuronska mreža igra, i kada je izgubila i treba učini na greškama.

### Slučaj kada neuronska mreža igra

Kao provjeru jeli neuronska mreža i dalje igra ili je izgubila uspoređujemo dva svojstva već gotove igre, svojstvo igra.runningTime(vraća vrijeme igranja), te igra.config.CLEAR\_TIME(vraća početno vrijeme prije početka igre).

Nakon što smo ustanovili da je igra započela tražimo poziciju prve prepreke koja nam se primiče, nakon čega sve podatke spremamo u objekt dana kako bi kasnije sa tim podacima mogli zahtjevati od neuronske mreže da nam vrati nekakav realan rezultat kako bi kasnije mogli odlučivati o kasnijim akcijama.

Uz pomoć metode neuralnaMreza.activate(podaci) dobivamo rezultat u rasponu [-1,1].



Dalje odlučujemo što želimo raditi na osnovu izlaza.

Ukoliko imamo vrijednosti veće ili jednake 0.5, želimo da dinosaur skoči(ako već nije u skoku), dakle pozivamo metodu dinoJump() i spremamo parametre posljednje akcije.

U protivnom želimo da ostane pri zemlji ili odabere metodu dinoLand() koja također omogućuje brže padanje(samo ako se nalazi u skoku). Tu parametre također spremamo.

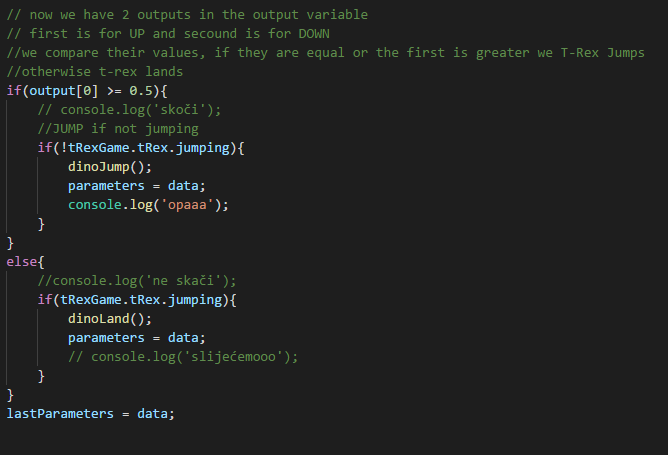
I na kraju želimo također spremiti negdje drugo posljednje parametre ukoliko nije nije aktivirana niti jedna od prethodnih metoda.

Paramatre spremamo u 2 niza:

* parameters(samo za parametre prilikom neke od akcija)
* LastParameters(parametri posljednjeg framea)



Dva niza su dodani izvan ove metode kako bi bili stalni.



### Slučaj kada neuronska mreža uči (poraz)

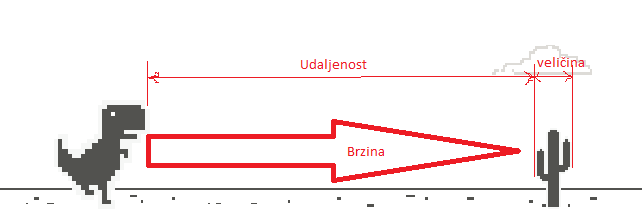
Kako bi provjerili jeli neuronska mreža izgubila(kako bi je mogli poučavati) koristimo već gotovo svojstvo same igre. Igra.crashed.



Prva bitna stvar sada je ta što reći neuronskoj mreži, jeli trebala ostati još malo na tlu pa onda tek natjerati dinosaura da skoči ili skočiti malo ranije.

To ćemo zaključiti na osnovu načina na koji je izgubila igru(pri skoku, ili frontalnim sudarom).

Trebamo odlučiti koliko želimo pomjeriti putanju dinosaura ukoliko izgubi… Prikažimo grafički.



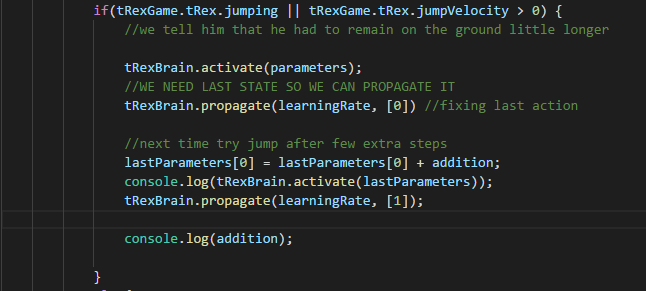
Iz ove slike zaključujemo da se proporcionalno sa brzinom udaljenost za skok povećava. Pošto T-Rex treba preskočiti udaljenost + veličinu onda nam je najbolje u slučaju greške uzeti neki mali dio toga puta i pridodati ga nekom od parametara, a potom aktivirati i propagirati promjene.

Za ovaj rad smo spremili (veličina/brzina)\*10 koje smo kasnije pridodavali parametru udaljenosti.

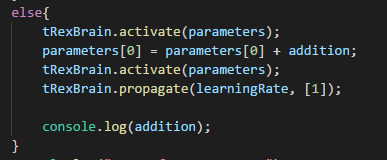


Dalje provjeravamo jeli T-Rex bio u skoku ili jeli padao dok je skočio(čisto radi osiguranja).

Ukoliko to je istina onda želimo prvo reći da je na tim koordinatama gdje je skočio trebao stajati, a nakon toga želimo uzeti posljednji i kažemo da skoči nakon nekoliko koraka dodajući „addition“ varijablu, te potom aktivirajući i propagirajući promjene.



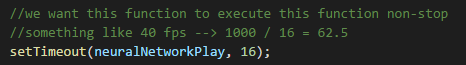
U svakom sljedećem slučaju ga pitamo što bi učinio i nakon toga mu kažemo da bi u tim slučajevima trebao skočit uz pomoć funkcije propagate.



Nakon svega toga želimo resertirati igru.



Također metodu koja omogućuje interakciju neuronske mreže i igre želimo ponavljati ~60 puta u sekundi kako bi omogućili brže i kvalitetnije učenje

.

# Zaključak

Igra i dalje nije optimalna što se tiče kvalitete učenja, te bi se mogla dosta poboljšati daljnjim dodavanjem skrivenih slojeva i neurona, ali i modificiranjem u kodu. Za potrebe ovoga projekta smatram da je uspješno realizirana mala neuronska mreža koja je nakon nekog vremena u stanju napraviti vrhunski rezultat.