



Eötvös Loránd Tudományegyetem

Informatika Kar

Programozáselméleti és Szoftvertechnológiai Tanszék

MI kliens 2 dimenziós ügyességi játékhoz

Témavezetők:

Bordás Henrik
Szoftverfejlesztő
CAE-Engineering kft.

Szerző:

Vecsernyés Márk
nappali tagozat
programtervező informatikus szak

Budapest, 2019

TARTALOMJEGYZÉK

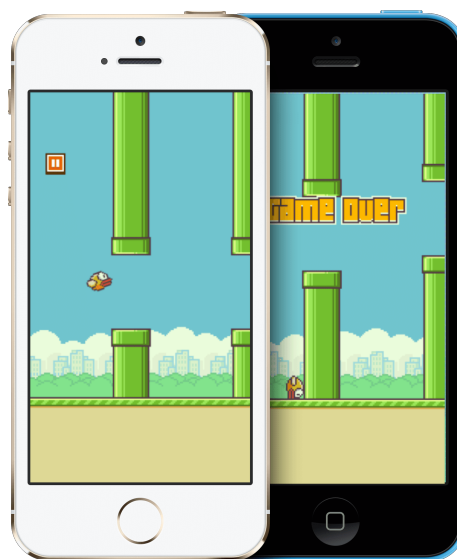
1. Bevezetés.....	3
2. Bevezetés az evolúciós algoritmusba	4
3. Felhasználói dokumentáció.....	9
3.1. A program által megoldott feladat	9
3.2. Célközönség.....	10
3.3. Minimális rendszerkövetelmények	10
3.3.1. Backend komponenseinek minimális rendszerkövetelményei	10
3.3.2. Frontend minimális rendszerkövetelményei.....	11
3.3.3. Hardveres minimális rendszerkövetelmények.....	11
3.4. Első üzembehelyezés	11
3.5. A program futtatása.....	12
3.5.1. Kezdőoldal.....	12
3.5.2. Egyszemélyes mód	14
3.5.3. Mesterséges intelligenciát használó mód	15
3.6. A mesterséges intelligencia mód által generált eredmények eredmények értelmezése	20
3.7. Hibaüzenetek	20
3.8. Általános információk	20
4. Fejlesztői dokumentáció.....	21
4.1. Probléma részletes specifikációja.....	21
4.2. Felhasznált módszerek részletes leírása	21
4.3. A program logikai és fizikai szerkezetének leírása	21
4.4. Tesztelés	21

5. Összefoglalás.....	22
6. Irodalomjegyzék	23

1. BEVEZETÉS

A világ legnagyobb videómegosztóján számos példa van arra, amikor a gép megtanul játszani egy játékkal anélkül, hogy ismerné annak szabályait. Komplexebb többszemélyes kompetitív játékokban a mesterséges intelligencia képes legyőzni a világ legjobb játékosait¹ is. Mindig is érdekelt, hogy ez hogyan lehetséges, ezért szakdolgozatom témájaként a gépi tanulást választottam egy ismert játékon evolúciós algoritmussal.

A tanítást a Flappy Bird² (1. ábra³) átiratán végeztem. Az eredeti 2013. Május 24-én jelent meg okostelefonos platformra. *Dong Nguyen* fejlesztette a *dotGears* cégnél, 2014 elején napi 50.000 dollárt hozott reklámbevételekből. Két dimenziós, retro játékokra hasonlít, az irányítandó karakter egy sárga madár, aki folyamatosan halad előre balról jobbra, közben a magassága a gravitáció hatására csökken. Egyetlen parancsot



1. ábra

lehet neki adni, ez az ugrás, így kell minél messzebb manővereznie az akadályok (oszlopok) között. 2014. Február 10-én *Nguyen* személyes okokra hivatkozva törölte a játékot. Ezután az egyik legtöbbször másolt játék lett az *Apple Storeban*, megközelítőleg napi 60 másolatot töltöttek fel, de PC-re is számos átirat megjelent.

Dolgozatom célja egy létező algoritmus implementálása egy olyan grafikus környezetben, amiben a felhasználó láthatja a működését a gyakorlatban.

Ezúton szeretnék köszönetet mondani a *családomnak* tanulmányaim támogatásáért, illetve *Bordás Henriknek*, aki konzulensként tudásával és tanácsaival segített felépíteni dolgozatomat.¶

2. BEVEZETÉS AZ EVOLÚCIÓS ALGORITMUSBA

Procedure EA

populáció := kezdeti populáció

while terminálási feltétel nem igaz ***loop***

szülők := szelekció(*populáció*)

utódok := rekombináció^{3. ábra,}(szülők)

utódok := mutáció(*utódok*)

populáció := visszahelyezés(*populáció*, *utódok*)

endloop

A feladat *célja* egy működő program létrehozása. Jelen esetben az elméleti háttér minimális ismerete nélkül nem értelmezhető kellő mélységben a program által megoldott feladat, ezért szerentném röviden, a teljesség igénye nélkül ismertetni az evolúciós algoritmust működését a programomban. Az evolúciós algoritmus egy *heurisztikákat* nem használó egy speciális *keresőrendszer*. Heurisztikus, vagy informált keresőrendszernek nevezzük azokat, amely a probléma definícióján túlmenően problémaszpecifikus tudást is felhasznál – hogyan képes hatékonyabban megtalálni a megoldást⁴. A *keresőrendszer*⁵ egy

Procedure KR

1. **ADAT** := kezdeti érték

2. while ¬ terminálási feltétel(**ADAT**) loop

3. **SELECT SZ FROM** alkalmazható szabályok

4. **ADAT** := **SZ**(**ADAT**)

5. endloop

end

algoritmus (2. ábra), ami a problémátérben keresi egy helyes választ. Általános alakja a következő:

A mesterséges intelligenciában használt keresőrendszereknek ennek az algoritmusnak speciális esetei, ilyen például az evolúciós algoritmus is.

A evolúció alapalgoritmus⁶ (3. ábra) a következő:

Egy adott pillanatban a problémátérnek sosem egy egyedét (lehetséges választ), hanem egy halmazát tároljuk. Ezt nevezzük *populációnak*. A populációt egyedeit valamivel kódolnunk kell, erre *neurális hálót* fogok használni (ami mátrixokkal van reprezentálva, de jelen dolgozatnak nem célja a neurális háló bemutatása).

Arra törekszünk, hogy az egyedek minél jobbak legyenek, ennek megítélésére rátermettségi függvényt (későbbiekben fitnessz) fogunk használni. Egy madár fitnessz értéke annál magasabb, minél messzebb jut a pályán (minden időpillanatban egyet nő). Minden pálya véletlenszerűen van generálva, sosem egy adott pályára tanulnak rá az egyedek. A tanulás mindig a generáció végén történik.

Legfontosabb fogalmak rövid áttekintése:

- *Egyed*: problémátér egy lehetséges válasza.
- *Kódolás*: egyed reprezentációja.

- *Populáció*: problémátér egy halmaza, az egyedeket tartalmazza.
- *Fitnessz*: az egyed rátermettségi függvénye.
- *Generáció*: az algoritmus egy ciklusa.
- *Evolúciós operátorok*: Szelekció, rekombináció, mutáció, visszahelyezés.
- *Terminálási feltétel*: az a feltétel, amikor az algoritmus véget ér (a fitnessz eléri a megadott küszöbértéket).

Az algoritmus futása vázlatosan a programomban:

1. Az algoritmus elején a populáció méretével megegyező számú egyedet (madarat) hozunk létre, tetszőleges neurális hálóval. Ez az első generáció inicializálása.
2. Elkezdünk egy játékot, amiben a madarak a neurális hálójuk alapján ugrálnak az oszlopok között (itt még természetesen nem tanulnak). A neurális hálónak 3 input paramétere van: madár magassága (*y koordinátája*), az következő oszloppár *x koordinátája*, illetve a következő oszlopok közül a felső tetejének *y koordinátája*. Minden időpillanatban minden madár lefuttatja a neurális hálóját a jelenlegi pozíciójához tartozó paraméterekre, majd ez alapján kétféle döntés hozhat: ugrik vagy nem ugrik. Ha az egyik madár fitnessz értéke elér egy küszöbindexet, akkor az algoritmus terminál. Amennyiben nem éri el egyik sem, és minden madár meghal, a 3. pontra lépünk.
3. Az első evolúciós operátorunk a szelekció. Itt kiválasztunk néhány (feltehetőleg rátermett) egyedet szülőnek. A cél, hogy a kevésbé rátermett egyedek is esélyt kapjanak (persze kisebb valószínűséggel). Erre több módszer létezik, én a *versengéses* módszert választottam, ahol tetszőleges csoportok (itt párok) legjobbját választom ki, pontosan $\left\lfloor \frac{s}{100} * p \right\rfloor$ darabot,

ahol $s \in \{50, 51, \dots, 100\}$ szelekciós együttható, illetve $p \in \{20, 25, \dots, 100\}$ a populáció mérete.

4. A második evolúciós operátor a rekombináció. Feladata, hogy a szülőkből olyan utódokat készítsünk, amik a szülők tulajdonságait öröklik. Erre a következő módszert használjuk: egyszerű jelcsoportokat cserélünk két véletlenszerűen választott szülőben. A szülők neurális hálójával vannak kódolva, ami 2 mátrixból, és 2 vektorból áll (lásd bővebben Fejlesztői dokumentáció megfelelő fejezetében). A mátrixokat ugyanabban a tetszőleges pontban kettévágom és összeolvasztom, a vektoroknál a kettő közül az egyiket választom ki, és azt adom az utódnak. Pontosan $\left\lfloor \frac{c}{100} * p \right\rfloor$ (alsó egészrész) darab utód jön létre, ahol $c \in \{40, 45, \dots, 90\}$ rekombinációs együttható, illetve $p \in \{20, 25, \dots, 100\}$ a populáció mérete.
5. A harmadik evolúciós operátor a mutáció. Feladata egy utód kisméretű változtatása. Meghatározásához 2 együtthatót kell definiálnunk, $M \in \{5, 10, \dots, 30\}$ és $m \in \{5, 10, \dots, 30\}$. $\left\lfloor \frac{M}{100} * p \right\rfloor$ számú utódot választunk ki, mindegyiknek $\left\lfloor \frac{m}{100} * p \right\rfloor$ darab komponense megváltozik (a neurális háló minden eleme $(-1, 1)$ intervallumba esik, így ezen az intervallumon belül változhat a kiválasztott elem).
6. A negyedik evolúciós operátor a visszahelyezés. Feladata a populáció utódokkal való frissítése. Kiválasztja a lecserélendő egyedet, majd azok helyére az újakat teszi. A populáció mérete legyen $p \in \{20, 25, \dots, 100\}$ és a törlési együttható $d \in \{1, 2, \dots, 10\}$. Az utolsó (legrosszabb fitnessz értékkel rendelkező) $\left\lfloor \frac{d}{100} * p \right\rfloor$ darab egyed helyett újat generálunk, majd a sorban következő (növekvő sorrendben) egyedet lecseréljük az összes utódra. A

maradék (legjobb fitnessz értékű) egyed marad. A generáció számát növeljük eggyel, majd 2. pontra lép az algoritmus.

Megjegyzés: az együtthatók intervallumait tapasztalati úton hoztam létre

Az evolúciós algoritmus egy nem-módosítható stratégia, a populáción végzett változtatások visszavonhatatlanok. Nem determinisztikus, a populáció módosítása mindig ugyanazzal a stratégiával, de véletlenszerű választásokkal történik. Ugyanarra a kezdőpopulációra más eredményeket ad általában. A program kipróbálásakor ezért tapasztalhatunk eltérő tanulási időket, például valamikor egy 3. generációs egyed eléri a megadott küszöbértéket, valamikor pedig a 200. generációra sem képes egyetlen egyed sem eljutni a 3. oszlopig.¶

3. FELHASZNÁLÓI DOKUMENTÁCIÓ

A következőkben ismertetjük *a program által megoldott feladatot, célközönségét*, majd a felhasználó szempontjait szem előtt tartva áttekintjük az

Név	Verzió
Google Chrome	70.0
Mozilla Firefox	63.0
Opera	53
Safari	12.0

első üzembehelyezést, végül részletesen bemutatjuk a *program használtát*, paraméterezését és a program által generált eredményeket, és értelmezését. Jelen dokumentum az olvasási szokások miatt (az egyes részeknek önmagukban is teljeseknek kell lennie) redundáns információkat tartalmazhat.

3.1. A program által megoldott feladat

A program a gépi tanulás egy módszerének, az *evolúciós algoritmusnak* a futását, nem pedig magát az algoritmust akarja részletesen megmutatni a felhasználónak. A szoftver korlátozottan, de paraméterezhető.

Bármilyen módszer működése egy adott probléma megoldására akkor szemléltethető a legjobban, ha a modell kellően bonyolult, viszont közérthető. A bonyolultságra azért van szükség, mert az esetleges egyszerűsítések következtében ha túl sok tulajdonságot elhagyunk, akkor a módszer hátrányait nem minden esetben láthatjuk. Közérthetőség nélkül pedig a probléma sokak számára értelmezhetetlenné válik. A bevezetőben ismertetett modellt fogom úgy átalakítani, hogy a gép képes legyen hozzáférni a megfelelő interfészekhez ("lássa, amit a felhasználó láthat", illetve "parancsot tudjon neki adni, hogy a felhasználó", de semmi többet).

A *kiindulási modell* a játék egy változata melynek forráskódja⁷ ingyenesen letölthető az internetről. Itt a felhasználó manuálisan, egy gomb lenyomásával tudja irányítani a karaktert. Ezt az input interfészt le kell cserélnünk, hogy a számítógép irányítsa madarat, illetve a populációnak méretének megfelelő számú madarat kell elhelyezni a játékban. A minden madarat eltérő mesterséges neurális hálóval kell kódolni, futása végén (egy futás játék indításától a madár haláláig tart) pedig fitnessz értékét feljegyezni. Az algoritmus a ezen értékeket ismeretében tudja elvégezni a kiértékelést. *A program által megoldott feladat a feljebb ismertetett modellen történő optimális egyed keresés evolúciós algoritmussal.*

3.2. Célközönség

A program célközönsége azok az emberek, akik már megismerkedtek az evolúciós algoritmussal, és szeretnék látni működés közben. A módszer előnyei, illetve hátrányai közül számos előjön futtatások során. Például egy alkalmazási terület amikor a tanár elmagyarázza az algoritmust, majd elindítja ezt a programot, és a diákok láthatják működés közben.

3.3. Minimális rendszerkövetelmények

A program jelenleg csak asztali számítógépen futtatható, operációs rendszertől függetlenül, amennyiben a szükséges programcsomagok telepítve vannak. Két fő komponense van: *backend* (szerverszolgáltatás) illetve a *frontend* (kliens rész).

3.3.1. Backend komponenseinek minimális rendszerkövetelményei

Név	Verzió	Feladat	Telepítés
Python	3.7.0	Interpretálás	Binárisból
Pip3	18.0	Python3 csomagok telepítése	Binárisból
PostgreSQL	10.5	Adatbázis-szolgáltatás	Binárisból

Flask	1.0.2	HTML szerver	Pip3-mal
Psycpg2	2.7.5	Adatbázis utasítások kezelése	Pip3-mal
Numpy	1.15.2	Mátrixok kezelése	Pip3-mal
Torch	0.4.1	Neurális háló kezelése	Pip3-mal

3.3.2. Frontend minimális rendszerkövetelményei

Elég egy böngésző következők közül tetszőlegesen a teljesség igénye nélkül (html, javascript, illetve ajax támogatás szükséges):

3.3.3. Hardveres minimális rendszerkövetelmények

Amennyiben a 3.3.1-es, és 3.3.2-es pontban leírt komponensek futnak az adott személyi számítógépen, akkor a program is futni fog. Pontos minimális hardveres követelmények a csomagok hivatalos honlapjain elérhetőek.

3.4. Első üzembehelyezés

Ha a minimális szoftveres és hardveres követelmények a rendelkezésünkre állnak, akkor a forráskód birtokában el fogjuk tudni indítani az alkalmazást, külön telepítést nem igényel, mert interpretert használ. Indítás előtt győződjünk meg a következőkről:

- Python3 interpretert el tudjuk indítani konzolos felületről
- PostgreSQL szolgáltatás fut, illetve rendelkezik 'postgres' role-lal

Első lépésben a backend szolgáltatást fogjuk elindítani: nyissunk meg egy konzolt, navigáljunk a forráskód gyökérkönyvtárába, majd a "python3 app.py"

```
markvecsernyes@szonor fbai (master) $ python3 app.py
Connecting to the PostgreSQL database...
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
Connecting to the PostgreSQL database...
```

parancsot kiadva a program ^{4. ábra} backend része elindul. Ekkor az alkalmazás a standard outputra kiír pár log üzenetet (4. ábra).

Ha sikerült csatlakoznia az adatbázishoz, és az üzenetek között nincs hiba, akkor a backend szolgáltatás fut, több teendőnk ezzel nincs (nem szükséges a továbbiakban leállítani, illetve újraindítani ezt).

3.5.A program futtatása

A böngészőbe <http://127.0.0.1:5000/> címet beírva az URL sávba, megkapjuk a program grafikus felhasználói felületét, amit GUI-nak (graphical user interface) nevezünk. Összesen 3 különböző oldal található meg a programban (Kezdőoldal (3.5.1), egyjátékos mód (3.5.2), mesterséges intelligenciát használó mód (3.5.3)), ezeket a következőkben ismertetem.

3.5.1.Kezdőoldal

Leírás: egy üdvözlőképernyő, ahonnan a felhasználó a többi oldalra tud navigálni

Elérés: A 3.5-ös pontban ismertetett URL a program kezdőlapjára navigálja a felhasználót (5. ábra).

Felület: A GUI két jól elkülíthető részre van osztva:

- Gombokra, ahol a "Single player mode" az egyjátékos módhoz (3.5.2), illetve "AI player mode" ami a mesterséges intelligenciát használó (3.5.3) verzióhoz vezet.
- Rajzvásznonra 576 pixel * 512 pixel-es mérettel, melynek világoskék a színe. Nyelve angol, a felirat jelentése pedig: "Üdvözljük! Kérem válassza ki a játék módját felül, jó szórakozást!"

Használat: A gombok valamelyikére kell kattintani, hogy el tudjuk kezdeni a játékot valamelyik verzióban.



6. Ábra

3.5.2. Egyszemélyes mód

Leírás: a játékot a felhasználó kipróbálhatja, megismerheti a játék menetét, illetve nehézségét.

Elérés: Erre az oldalra (7. ábra) a kezdőoldaltól (3.5.1), illetve a *mesterséges intelligenciát használó módból* navigálhatunk (3.5.3).



7. ábra

Felület: A GUI két jól elkülöníthető részre van osztva:

- Gombokra, ahol a "<Home" a kezdőoldalra (3.5.1), illetve "AI player mode" ami a mesterséges intelligenciát használó (3.5.3) verzióhoz vezet.

- Rajzvászónra 576 pixel * 512 pixel-es mérettel, melynek világoskék a színe. Középen egy vonallal van elválasztva bal oldalán a játék fut (automatikusan indul), jobb oldalon pedig az aktuális pontszám látható, minden időpillanatban frissítve. A madár meghal, ha oszloppal ütközik, ekkor a pontszám 0-ra csökken, a játék pedig újraindul. A szöveg fordítása: "Egyjátékos mód! Nyomja le bármelyik gombot, hogy a madarat vezérelje. A cél elkerülni az oszlopokat."

Használat: A mód használatához bármelyik gombot lenyomhatjuk a billentyűzeten, ekkor ugrik a madár. Az oldalt elhagyni valamelyik fent említett gombra kattintva lehet.

3.5.3. Mesterséges intelligenciát használó mód

Leírás: mesterséges intelligenciát használva (evolúciós algoritmust) a számítógép megpróbálja megtanulni a 3.5.2-ben futtatható játékot minél jobban játszani.

Elérés: Erre az oldalra a *kezdőoldaltól* (3.5.1), illetve az *egyszemélyes módból* navigálhatunk (3.5.3).

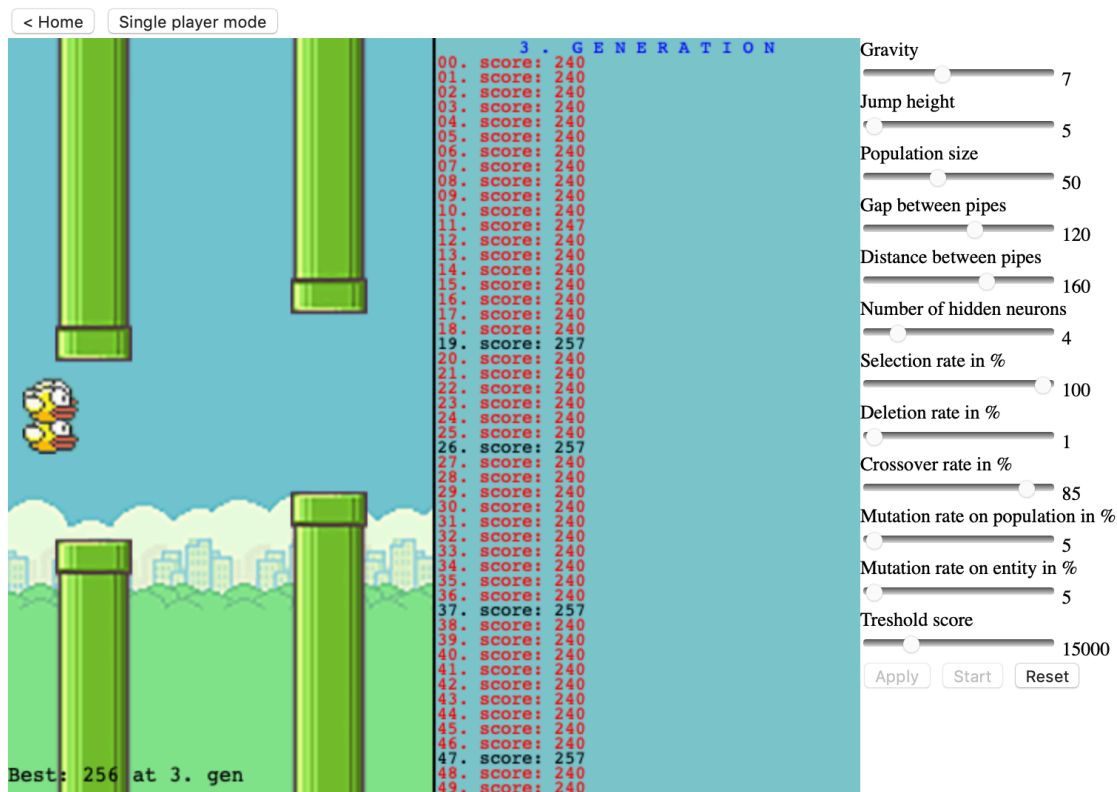
Felület: A GUI három jól elkülöníthető részre van osztva:

- Gombokra, ahol a "<Home" a kezdőoldalra (3.5.1), illetve "Single player mode" ami az egyszemélyes verzióhoz (3.5.2) vezet.
- Rajzvászónra 576 pixel * 512 pixel-es mérettel, melynek világoskék a színe. Két állapota van:
 - Indítás előtt (8. ábra): Utasítások vannak kiírva középre. A szöveg fordítása: "Mesterséges intelligenciát használó mód! 1. Állítsa be a futási paramétereket, 2. kattintson az Apply-ra, 3. kattintson a Start-ra."



8. ábra

- Indítás után (9. ábra): egy vonallal van a rajzvászon kettéosztva. Bal oldalt a játék fut, minden időpillanatban frissítve. A populáció méretével megegyező számú madár van kirajzolva, folyamatosan haladnak balról jobbra, a magasságuk a gravitáció hatására csökken. Az ugrási parancsok mesterséges intelligenciával vannak vezérelve. Ha egy madár oszloppal ütközik, akkor meghal, és eltűnik a képernyőről. Jobb oldalt felül az aktuális generáció száma, alatta az egyes madarak fitnessz értékei láthatóak sorszámozva feketével. A madár halálával a fitnessz értékét tartalmazó szöveg színe *piros* lesz. Ha minden madár meghal, akkor lefut a tanító algoritmus, majd a generáció száma eggyel nő, és a játék újraindul, de a madarak már máshogy fognak ugrálni.



9. ábra

- Input paraméterek, és a játék indítása. A bemeneti paramétereket csúszkával lehet állítani, majd a gombok segítségével lehet véglegesíteni, majd indítani a játékot. Alaphelyzetben a *csúszkák*, és az *Apply*, illetve a *Reset* gomb aktív, a *Start* pedig inaktív. A felület ezen része részletesen a következő elemeket tartalmazza:

- Gravity (csúszka): a gravitáció értéke, megadható vele, hogy milyen gyorsan csökkenjen a madár magassága. Állítható: 5-10 között, 1-es lépésekkel.
- Jump height (csúszka): ugrás magassága, megadható vele, hogy egy ugrással mennyit nőjön a madár magassága. Állítható: 5-10 között, 1-es lépésekkel.
- Population size (csúszka): populáció mérete, azt állítja, hogy hány madár (egyed) legyen a populációban. Ezek az egyedeket vezérli a

neurális háló, illetve rajtuk fut a tanítási algoritmus. állítható: 20-100 között, 5-ös lépésekkel.

- Gap between pipes (csúszka): egy oszloppár közti függőleges távolság. Állítható: 90-140 között, 5-ös lépésekkel.
- Distance between pipes (csúszka): két egymást követő oszloppár közti vízszintes távolság. Állítható: 120-180 között, 10-es lépésekkel.
- Number of hidden neurons (csúszka): a neurális háló rejtett neuronjainak száma. Állítható: 3-10 között, 1-es lépésekkel.
- Selection rate in % (csúszka): szelekciós ráta, az evolúciós algoritmus pontosan $\left\lfloor \frac{s}{100} * p \right\rfloor$ darab pár közül választja ki a legjobbat szülőnek (s a szelekciós ráta, p a populáció mérete). Állítható 50-100 között 5-ös lépésekkel.
- Deletion rate in % (csúszka): törlési ráta, az evolúciós algoritmus pontosan $\left\lfloor \frac{d}{100} * p \right\rfloor$ darab egyed helyett teljesen újat hoz létre (d a törlési ráta, p a populáció mérete). Állítható 1-10 között 1-es lépésekkel.
- Crossover rate in % (csúszka): rekombinációs ráta, az evolúciós algoritmus a kiválasztott szülők közül pontosan $\left\lfloor \frac{c}{100} * p \right\rfloor$ darab egyedpár kódjait keresztezi. Állítható 40-90 között 5-ös lépésekkel.
- Mutation rate on population % (csúszka): mutációs ráta a populáción, az evolúciós algoritmus pontosan $\left\lfloor \frac{M}{100} * p \right\rfloor$ darab utódot választ ki, ezek fognak mutálódni. Állítható 5-30 között 5-ös lépésekkel.

- Mutation rate on entity % (csúszka): mutációs ráta az egyeden, az evolúciós algoritmus a mutációra kiválasztott egyedek pontosan $\left\lfloor \frac{m}{100} * p \right\rfloor$ darab kódját lecseréli egy véletlenszerű értékre a (0,1) intervallumban. Állítható 5-30 között 5-ös lépésekkel.
- Threshold score (csúszka): a terminálási feltétel pontszáma, azt állítja, hogy az evolúciós algoritmus mekkora fitnessz értéknél álljon le. Állítható 5000-50000 között 5000-es lépésekkel.
- Apply (gomb): nyugtázás, az input paraméterek véglegesítése. Start gomb előfeltétele.
- Start (gomb): játék indítása, a gomb az Apply gomb megnyomása után válik elérhetővé.
- Reset (gomb): az oldal újratöltése, minden beállítás elveszik, és alapértelmezettre áll vissza.

Használat: A mód használatához először tetszőlegesen módosítsuk az input paramétereket a csúszkával (hagyhatjuk az alapértelmezett beállításokon is). Majd az Apply gombbal véglegesítsük (ekkor az Apply, illetve az input paraméterek inaktívvá válnak, a start gomb pedig aktívvá), majd a Start gomb megnyomásával elindíthatjuk a játék futását, Innentől semmilyen felhasználói input nem szükséges. A Reset gombbal bármikor újratölthetjük a lapot, ekkor minden eddigi beállítást elveszik. Az oldalt elhagyni valamelyik felső gombra kattintva lehet.

3.6. A mesterséges intelligencia mód által generált eredmények
eredmények értelmezése

3.7. Hibaüzenetek

3.8. Általános információk¶

4. FEJLESZTŐI DOKUMENTÁCIÓ

4.1.Probléma részletes specifikációja

4.2.Felhasznált módszerek részletes leírása

4.3.A program logikai és fizikai szerkezetének leírása

4.4.Tesztelés

5. ÖSSZEFOGLALÁS

6. IRODALOMJEGYZÉK

¹ <https://www.engadget.com/2017/08/12/ai-beats-top-dota-2-players/?guccounter=1> 2018.11.11.

² https://en.wikipedia.org/wiki/Flappy_Bird: 2018.10.03.

³ https://www.imore.com/sites/imore.com/files/topic_images/2014/topic_flappy_bird.png 2018. 11.13.

⁴ http://project.mit.bme.hu/mi_almanach/books/aima/ch04s01 2018.11.15.

⁵ Gregorics Tibor: Mesterséges intelligencia fóliasora 2017/18, tavaszi félév

⁶ Gregorics Tibor: Mesterséges intelligencia fóliasora 2017/18, tavaszi félév

⁷ <https://github.com/CodeExplainedRepo/FlappyBird-JavaScript> 2018.11.13