Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Jeszenszky Péternek a számomra nyújtott sok segítségért.

Nagyon köszönöm a párom és a kisfiam türelmét és támogatását.

Hálás vagyok barátainknak, Nyisztor Juditnak, Nyisztor Józsefnek és Bencze Dórának, hogy néhány órára szárnyuk alá vették a kisfiamat, amikor nagyon elmélyülten dolgoztam.

1. Bevezetés

A diplomamunkám célkitűzése az volt, hogy áttekintsem a mesterséges intelligencia egy speciális területének, az evolúciós számításoknak az elméleti hátterét, majd az evolúciós számítások egyik technikája, a genetikus algoritmus alapján egy általános, sokoldalúan felhasználható evolúciós keretrendszert valósítsak meg, amelyet aztán konkrét, saját művészeti vonatkozású alkalmazások alapjaként tudok használni. E keretrendszer használatának bemutatására két saját minta alkalmazást készítettem.

A 2. fejezet rövid betekintést nyújt az evolúciós számítások elméleti hátterébe, és bemutat néhányat azok művészi célú alkalmazásaiból.

A 3. fejezetben ismertetem az általam kifejlesztett általános keretrendszert.

Arra a célra, hogy szemléltessem, a keretrendszerem hogyan használatható evolúciós művészeti alkotások fejlesztésére, két egyszerű minta alkalmazást készítettem. Az első egy webes alkalmazás, amelynek segítségével olyan összetett, kreatív filtereket lehet fejleszteni, amelyeket tetszőleges (BMP, PNG, JPEG formátumú) kép átalakítására tudunk használni. A kreatív filterek alkalmazást a 4. fejezetben ismertetem. A második minta alkalmazásom úgynevezett „hangya-festményeket” fejleszt, erről az alkalmazásról bővebben az 5. fejezetben írok.

2. Az evolúciós algoritmusokról és művészeti alkalmazásaikról

2.1. Evolúciós számítások

Az evolúciós számítások a *mesterséges intelligencia* egyik részterülete. Olyan módszereket (algoritmusokat) foglal magában, amelyek a biológiai evolúció mechanizmusán alapulnak. Elsősorban *keresési* és kombinatorikus optimalizációs problémák megoldására használhatóak. Gyakran alkalmazhatóak olyan esetekben is, amikor a hagyományos módszerek csődöt mondanak, mert a probléma lehetséges megoldásainak halmaza kezelhetetlen méretű, és az optimális megoldás felkutatása reménytelen – ilyen például az utazóügynök-probléma.

2.2. Keresés a mesterséges intelligenciában

Ha a mesterséges intelligenciában keresésről beszélünk, bármilyen keresési algoritmust alkalmazunk is, a keresés során egy speciális térben, a lehetséges megoldások terében kutatunk. Ezt a teret, amelynek minden pontja a probléma egy-egy lehetséges megoldása, *keresési tér*nek nevezzük. A kereséskor a célunk gyakran az *optimalizáció*, azaz a valamilyen értelemben véve „legjobb” megoldás felkutatása. A keresési tér legnagyobb részét azonban rossz megoldások töltik ki és gyakran nehéz eljutni a jó megoldások környezetébe. A keresés folyamán ezért megpróbáljuk segítségül hívni a már „bejárt” megoldásokat arra, hogy eldöntsük, a térben merre célszerűbb haladnunk, merre valószínűbb, hogy jó megoldásokat találunk.

Nagyon sokféle keresési algoritmus létezik. A mesterséges intelligenciából ismert hagyományos módszerek úgy keresnek, hogy egyszerre egy megoldást megpróbálnak lépésről-lépésre úgy felépíteni, hogy az minél inkább megfeleljen a lefektetett céloknak, feltételeknek.

A gyakran alkalmazott heurisztikus keresés során például minden lépésben úgy választjuk ki az elérhető megoldások közül a következő lépést jelentőt, hogy valamilyen speciális ismeretet (heurisztikát) használunk fel, amellyel lényegében le tudjuk szűkíteni a keresési teret. Ha „nem jó” a heurisztikánk, elzárhat minket jó megoldásoktól, jelentősen behatárolhatja a lehetőségeket, a kreativitásnak nem ad szabad utat.

2.3. Az evolúciós keresés

Az evolúción alapuló algoritmusok a legrugalmasabb, leghatékonyabb, legrobosztusabb keresési algoritmusok közé tartoznak az összes ismert keresési algoritmus közül (Goldberg).

A biológiai evolúció fontos tulajdonsága, hogy az evolúciónak magának nincs célja. Az evolúciós folyamat alapján nem volt törvényszerű, hogy pontosan az az élővilág fejlődjön ki bolygónkon, amit ma látunk – egyszerűen a feltételekhez való szakadatlan alkalmazkodás szülte meg a denevéreket, kengurukat, mélytengeri halakat, embereket; a szárnyakat, szemeket, denevér-radarokat, az emberi agyat.

Az evolúciós algoritmusoknak sem mondjuk és nem is mondhatjuk meg explicit módon, hogy konkrétan mit fejlesszenek ki. Megpróbáljuk formálisan leírni, milyen alkotóelemekből áll az a valami, amit fejleszteni szeretnénk, és felállítunk egy feltételrendszert, amelynek a jó megoldásoknak lehetőleg minél jobban meg kell felelnie. Veszünk egy csokorral a formális leírásnak megfelelő megoldásokból, majd elindítjuk az evolúciós folyamatot, amely során a feltételeinknek egyre inkább megfelelő megoldásokat állítunk elő.

Szemben sok más (fent említett) keresési módszerrel, amelyek egyszerre egy megoldást építgetnek lépésenként, vagyis a keresési térben egyszerre egy „ösvényen” haladnak, az evolúciós keresés során megoldások egy halmaza van a kezünkben. Ezeket „versenyeztetjük”, és közülük a jó megoldások alapján a következő lépésben szintén megoldások egy halmazát állítjuk elő. Az evolúciós keresés fontos tulajdonsága tehát, hogy *párhuzamos* abban az értelemben, hogy egy lépésben mindig több irányba halad a keresési térben.

A megoldásokat az evolúciós számítások terminológiájában *egyed*eknek, a megoldások halmazát, amellyel az algoritmusok egy lépésben dolgoznak, *populáció*nak nevezzük.

Az evolúció folyamatán azt értjük, hogy sorra állítjuk elő az egymásra következő populációkat, vagyis az újabb és újabb *generáció*kat, az elhíresült *szelekciós elv*, „a rátermettebb túlélése” alapján. *Rátermett* egyedek alatt a jó megoldásokat értjük. Minden egyedhez egy értéket rendelünk, amely azt fogja mutatni, hogy mennyire felel meg az adott egyed az elvárásainknak, mennyire van közel a célunkhoz, mennyire rátermett. Az új generációt jelentő *utód- vagy gyermekpopuláció* egyedei úgy állnak elő, hogy a *szülők populációjá*nak egyedei szaporodnak, méghozzá úgy, hogy az egyedek esélye a szaporodásra egyenes arányban áll a rátermettségükkel. Azaz a rátermettebbeknek valószínűleg több utóduk lesz a következő generációban, mint a kevésbé rátermetteknek. Az egymásra következő populációk egyre több rátermett egyedből állnak, azaz a megoldások a keresési térben

koncentráltabban kerülnek ki a jó megoldások területeiről. Az idővel a populációkban a rátermettség maga is nő.

2.4. Az evolúciós számítások története

Egészen korán, az 1950-es években vetődött fel a gondolat, hogy a darwini elveket automatizált problémamegoldásban hasznosítsák.

A híres statisztikus, George E. P. Box kidolgozott a vegyipar számára egy „evolúciós operációnak" nevezett módszert, aminek segítségével növelni lehetett a vegyszereket előállító berendezések termelékenységét. A berendezések működési paramétereit, például a hőmérsékletet és a reagensek koncentrációját próbálták úgy beállítani a módszer alkalmazásával, hogy a legkisebb mértékű szennyeződés keletkezzen az előállított vegyszerekben. Box „evolúciós operációja" még nem volt teljesen automatizált, emberi beavatkozást igényelt a döntés során, amikor a következő generáció működési paramétereit kellett meghatározni. Box 1957-ben publikálta módszerét.

1956-ban George Friedman leírta egy „szelektív visszacsatolású számítógép" tervét, amely a természetes szelekció elvét felhasználva áramköröket fejleszt. Ezt a „robotot" sohasem építették meg, Friedman mégis megmutatta általa, milyen lehetőségek rejlenek az evolúció alapú keresésben.

1958-ban Richard M. Friedberg *Tanuló gép* című könyvében bemutatta, hogyan fejlesztett evolúciós módszerrel egyszerű aritmetikai műveleteket elvégző gépi kódot.

A 60-as években három különböző helyen három különböző interpretációját fejlesztették ki a számítógépes evolúció elképzelésének. Az USA-ban Lawrence J. Fogel vezette be az **evolúciós programozás**t, John Henry Holland pedig a **genetikus algoritmus**t, míg Németországban Ingo Rechenberg és Hans-Paul Schwefel kidolgozták az **evolúciós stratégiák**at. A három területet sokáig egymástól függetlenül fejlesztették, és csak a 90-es évek elejétől tekintenek erre a három módszerre úgy, mint egyazon elv, az **evolúciós számítások** különböző megvalósításaira [12].

1985-ben Nichael L. Cramer vetette el az evolúciós számítások negyedik változatának magjait, ám az csak a 90-es évek elején, John R. Koza munkája nyomán fejlődött **genetikus programozás** néven a negyedik irányzattá.

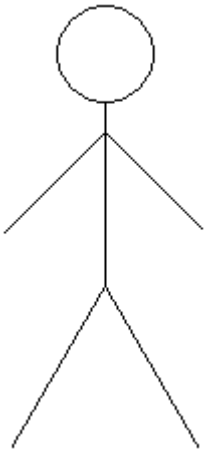
2.5. Az evolúciós számítások irányzatai

2.5.1. Genetikus algoritmusok

Ha általánosságban beszélünk az evolúció elvein alapuló algoritmusokról, röviden csak *evolúciós algoritmusok*at emlegetünk. A genetikus algoritmusok olyannyira a legismertebbek és legelterjedtebbek ezek közül, hogy gyakran a „genetikus algoritmus” kifejezés az „evolúciós algoritmus” szinonimájává lép elő – ám tudnunk kell, hogy valójában a négy irányzat egyikét jelöli.

A négy evolúciós algoritmusfajta közül a genetikus algoritmusokkal kapcsolatban végezték a legtöbb kutatást. Megállapítást nyert, hogy egy adott, speciális probléma megoldásában nem feltétlenül a genetikus algoritmusok a legeredményesebbek, de ezek az algoritmusok még rosszul implementálva, szegényesen megvalósítva is gyakran szolgáltatnak elfogadható eredményeket. Ezek közelítik meg legjobban a természetes evolúciót, és egyes vélemények szerint ezek állnak legközelebb az emberi innováció folyamatához is.

A genetikus algoritmusokban, ahogy a nevük mutatja, a genetika alapfogalmai köszönnek vissza. Az egyedek a biológiai analógiának megfelelően rendelkeznek *genotípus*sal és *fenotípus*sal. Vegyünk egy példát! Emberi arányokkal rendelkező pálcikaembert szeretnénk kifejleszteni. Egy konkrét pálcikaembert egy számsorozat fog leírni, a fej átmérője, a nyak, a karok, a törzs és a lábak hossza. (Az egyszerűség kedvéért a példánkban fix szögeket zárnak be a karok és a törzs, valamint a lábak és a törzs.) Az (50, 15, 50, 80, 80) számsor tehát egy konkrét pálcikaemberünk kódolt, formális leírása, *genotípusa*. Ezek a számok határozzák meg, hogy fog kinézni az egyedünk, ezek lesznek a pálcikaembert felépítő, megrajzoló eljárás *paraméter*ei. A pálcikaember rajzoló eljárás végére előáll az 1. ábrán látható kép, ez a kép lesz a pálcikaember megjelenése, *fenotípusa*.



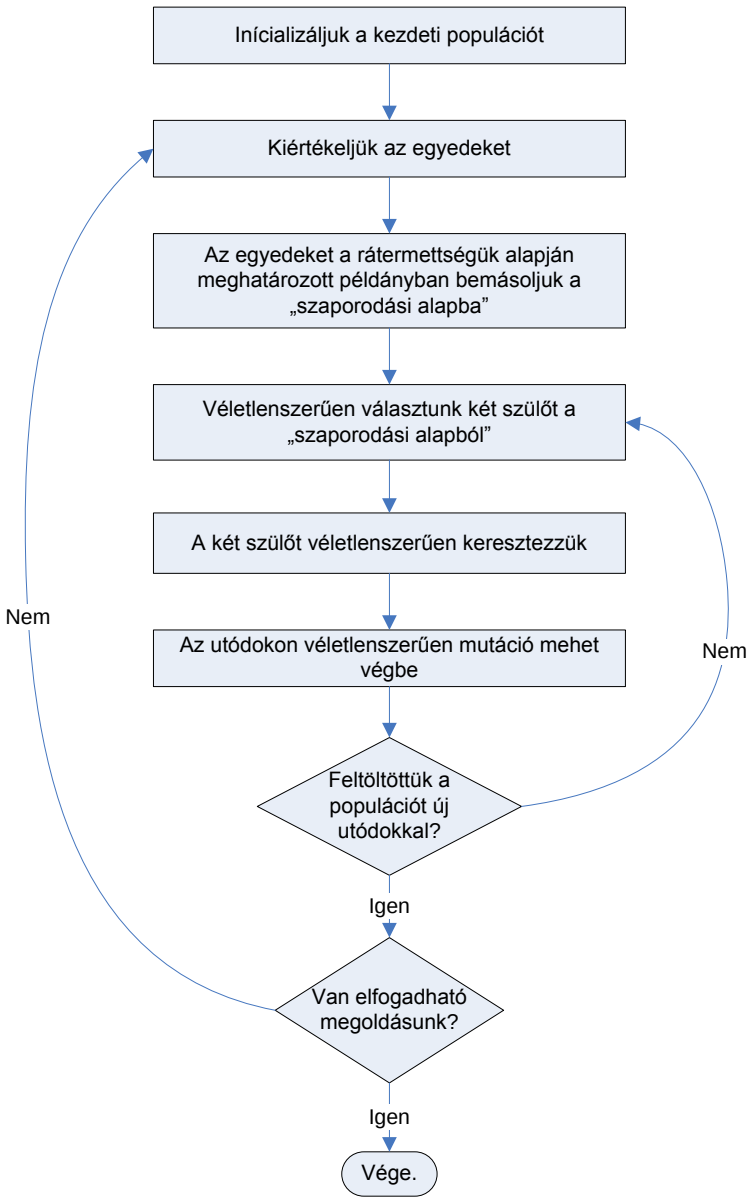
1. ábra: az (50, 15, 50, 80, 80) genotípusú pálcikaember fenotípusa

A genotípus tehát kódolt paraméterekből áll, amelyeket *gén*eknek nevezünk. A gének konkrét értékeit – akárcsak a biológiában – *allél*oknak nevezzük. A genotípus általában leírható sztringként, ezt a sztringet *kromoszóma*nak hívjuk.

A genetikus algoritmusok ezek szerint valójában két térrel dolgoznak. Az egyik a már korábban bevezetett *keresési tér*, ami a probléma kódolt megoldásait (genotípusait) tartalmazza, a másik az aktuális *megoldások* (fenotípusok) *tere*.

Rátermett egyedek alatt a jó megoldásokat, azaz a jó fenotípusú egyedeket értjük. Ezért a genotípusokat még az egyedek kiértékelése előtt le kell képeznünk a fenotípusok terére. Az evolúciós algoritmusokban az egyedek kiértékelését egy úgynevezett *fitness(rátermettség)-függvény* segítségével valósítjuk meg, amely egy konkrét egyedhez egy értéket rendel – tipikusan pozitív egész számot.

A legegyszerűbb, kanonikus genetikus algoritmus a 2. ábrán látható folyamatábrával írható le.



2. ábra: A genetikus algoritmus

A kezdeti populáció inicializálása úgy történik, hogy a populációt olyan egyedekkel töltjük fel, amelyek genotípusát véletlen allélekből építjük fel.

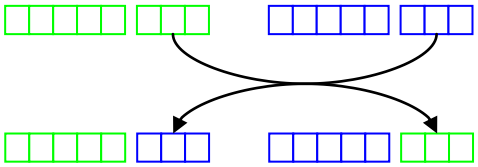
Ezután elkezdődik az algoritmus fő ciklusa. A populáció egyedeit kiértékeljük, azaz minden egyedhez a fenotípusa alapján a fitness-függvénnyel rendelünk egy értéket, amely az egyed rátermettségét mutatja.

A rátermettség alapján minden egyedből meghatározott számú példányt másolunk be a szaporodási alapba – minél rátermettebb az egyed, annál többet.

Ezek után véletlenszerűen választunk két szülőt a szaporodási alapból.

A genetikus algoritmus ezután ***két genetikus operátor***ral dolgozik: a keresztezés és a mutáció operátorával.

A *keresztezés* kétoperandusú művelet, két egyeden értelmezzük. A két szülő kromoszómájának alléljait *véletlenszerűen* elosztjuk, két új kromoszómát hozunk létre, amelyek az utódok genotípusai lesznek.



3. ábra: Keresztezés

Gyakran egészen egyszerűen úgy implementáljuk a keresztezést, hogy mindkét szülő kromoszómáját elvágjuk egy adott, azonos ponton, és a kromoszóma-felek új kombinációi lesznek az utódok kromoszómái, ahogy azt a 3. ábra mutatja.

A *mutáció* egy operandusú művelet, egy egyeden értelmezzük. Azt takarja, hogy az egyed kromoszómájának egy vagy több allélját véletlenszerűen megváltoztatjuk. Nem minden új egyedre alkalmazzuk a mutációt – azt, hogy milyen valószínűséggel jön létre mutáció, a *mutációs ráta* rögzíti.

Az így létrehozott két új egyedet bemásoljuk a szaporodási alapba. Addig folytatjuk a szülők kiválasztását és az új egyedek létrehozását, amíg fel nem töltjük az új generációt. Ha az új generációban találunk megfelelő megoldást, a keresésnek vége, ha nem, újra kiértékeljük az egyedeket, és létrehozzuk a következő generációt.

A genetikus algoritmusban több helyen megjelenik a véletlen. Fontos látnunk, hogy maga a keresési folyamat a véletlen elemek ellenére irányított, hiszen a potenciális szülők kiválasztása „a rátermettebb túlélése”-elv alapján, és nem véletlenszerűen történik, az így létrejövő *szelekciós nyomás* irányítja a keresést a keresési térben a jó megoldások területei felé.

A genetikus algoritmusoknak több módozata, fajtája van, ezek közül az evolúciós művészetben az *interaktív vagy kollaboratív genetikus algoritmusok*at alkalmazzák gyakran, ahol a felhasználói interakció részben vagy teljes egészében átveszi a fitness-függvény szerepét.

2.5.2. Genetikus programozás

A genetikus programozás alapalgoritmusa tulajdonképpen megegyezik a genetikus algoritmussal. A különbség a kettő között az, hogy a genetikus programozásban speciális kromoszómákkal dolgozunk, és ebből adódóan a genetikus operátorokat is másképp valósítjuk meg.

A genetikus algoritmus kromoszómája egy fix hosszúságú sztring volt, a genetikus programozás különleges kromoszómafajtája pedig a program. A programok rendszerint változó méretű, hierarchikus faszerkezettel reprezentálhatóak. Ezeket a kromoszómafákat nyilván másképpen tudjuk csak kezelni, de az alapelvek nem változnak.

A keresztezést úgy értelmezzük, hogy a két szülő kromoszómafáját véletlenszerűen elvágjuk egy-egy ponton, és a vágási pontok alól eredő egész részfákat cseréljük ki, így jön létre a két utód kromoszómája.

A mutációt is hasonlóképpen értelmezzük, vagyis az egyed mutációja úgy valósul meg, hogy a kromoszómafájában egy véletlenszerűen kijelölt pontból eredő részfát kicserélünk egy új (véletlenszerűen előállított) részfára.

Ha valóban programokat reprezentálnak a kromoszómafák, az egyed rátermettségét úgy állapíthatjuk meg, hogy konkrét inputsorozatokat rögzítünk előre, és azt is rögzítjük, hogy az adott inputértékekre milyen outputot várunk, majd a minden programegyedünket ezekkel a megadott inputokkal lefuttatjuk, és megvizsgáljuk, hogy mennyire van közel a program végeredménye az elvárthoz.

Programok esetén azt is célszerű megvizsgálni, hogy az új egyedek tartalmaznak-e fölösleges kódrészleteket, ugyanis a programkódok a haszontalan részletektől nagyon felduzzadhatnak.

A genetikus programozást eleinte valóban számítógépes programok fejlesztésére használták, majd alternatív területeken is bevezették. Kreatív evolúciós alkalmazások magjaként is eredményesen használják. Képzőművészeti alkotásokat először Karl Sims készített genetikus programozással a 90-es évek elején.

2.5.3. Evolúciós stratégiák

Az evolúciós stratégiák úttörői a német Bienert, Rechenberg és Schwefel voltak. Ők ezt a technikát aerodinamikai fejlesztésekhez alkották meg. Repülőgép alkatrészeket fejlesztettek, az alkatrészek megfelelőségét szélcsatornában végzett tesztek alapján állapították meg.

A legegyszerűbb evolúciós stratégia során egyetlen egyedet tartalmazó populációval dolgozunk, és nem alkalmazunk keresztezés operátort. Kezdetben véletlenszerűen létrehozunk és kiértékelünk egyetlen megoldást, ez lesz a szülő. Majd a szülőt véletlenszerű mutációnak vetjük alá, így jön létre a gyermek. Ezután a gyermeket kiértékeljük, majd amennyiben a gyermek rátermettebbnek bizonyul a szülőnél, úgy a szülőt kiselejtezzük, és a gyermek lép a helyébe. Amennyiben a gyermek a kevésbé rátermett, őt selejtezzük ki. Majd a szülőt újra mutációnak vetjük alá, és így tovább. Ez az úgynevezett „egy szülő, egy gyermek" típusú, vagy (1+1) evolúciós stratégia.

A korai genetikus algoritmusok bináris kromoszómákkal dolgoztak, vagyis egy megoldás paramétereinek listáját nullák és egyesek sorozataként ábrázolták, és a mutációt magát is a bitek szintjén valósították meg. Ezzel szemben már a korai evolúciós stratégiák kromoszómái is maguknak a paramétereknek a listái voltak. A paraméterek az evolúciós stratégiákban általában valós számértékek, a mutációt pedig az evolúciós stratégiák úgy értelmezik egy paraméteren, mint az eredeti értéktől való eltérést. Vagyis kis mértékben módosítják a mutáció által érintett gén értékét.

A megengedett kis eltéréseket génenként értelmezzük, és ezeket az evolúciós stratégia *stratégia-paramétere*inek nevezzük. A kromoszóma minden egyes génre a hozzá tartozó stratégia-paramétert is tartalmazza. A mutáció nem csak a paramétereket, hanem a stratégia-paramétereket is módosítja.

Az evolúciós stratégiák mutációja nem véletlenszerű, mint a genetikus algoritmusoké, hanem normális eloszlást követ. Minden egyes génhez adott egy stratégia paraméter, amely egy egyedi, nulla várható értékű normális eloszlás szórását határozza meg [2].

Lássuk, hogyan működik az evolúciós stratégia populációkon!

A kezdeti populáció létrehozása kétféleképpen történhet az evolúciós stratégiában. Vagy véletlenszerűen állítjuk elő a kezdőpopuláció megoldásait, vagy egyetlen, a felhasználó által megadott megoldás mutációiként állítjuk őket elő.

A kezdeti populáció létrehozása után indul az algoritmus fő ciklusa. Véletlenszerűen választunk két szülőt a szülőpopulációból, és több rekombinációs operátort alkalmazunk rájuk. A rekombinációk során ténylegesen keveredhetnek a szülők génjei, és a genetikus algoritmus keresztezéséhez hasonló eredményt kaphatunk, de az is előfordulhat, hogy a szülők génjei nem keverednek. A rekombinációkkal előállított utódokon a saját stratégia-paramétereik alapján mutáció megy végbe. Ennek során maguk a stratégia-paraméterek is

módosulnak. Ezután az utódokat a gyermekpopulációba helyezzük. Mindaddig folytatjuk az utódok előállítását az új szülők kiválasztásával, rekombinációkkal és mutációval, amíg a gyermekpopulációt fel nem töltöttük. Ha a gyermekpopulációt feltöltöttük, kiértékeljük a gyermekeket, meghatározva a rátermettségüket.

Ekkor minden véletlenszerűséget nélkülözve, a legrátermettebb gyermekeket helyezzük a szülőpopulációba, így téve lehetővé számukra, hogy szülőkké váljanak.

Az evolúciós stratégia algoritmusa akkor ér véget, ha elérünk egy előre rögzített generációszámot, vagy elfogadható megoldás kerül a szülőpopulációba. Ha egyik feltétel sem teljesül, új gyermekpopulációt állítunk elő.

Az 1980-as éves elején fejlesztették ki a (μ+λ) és a (μ, λ) evolúciós stratégiákat. Ebben a jelölésben a μ a szülőpopuláció, a λ a gyermekpopuláció létszámát jelenti. Míg a (μ, λ) evolúciós stratégia esetében a legjobb μ egyed csak a gyermekek közül kerül kiválasztásra, addig a (μ+λ) evolúciós stratégiában a legrátermettebb μ egyed (leendő szülő) nem csak a gyermekek közül, hanem a szülők közül is kikerülhet, azaz *a legrátermettebb szülők* tovább élhetnek*, tovább alkalmazkodhatnak* a feltételekhez a következő generációban. Ahhoz, hogy ilyenkor a szelekciós nyomás létrejöhessen, a szülők számának kisebbnek kell lennie a gyermekek számánál. Az ajánlott μ:λ arány 1:7 [2].

2.5.4. Evolúciós programozás

Az evolúciós programozást az 1960-as évek elején eredetileg véges állapotú automaták állapotátmenet-tábláinak evolúciójára alkotta meg Lawrence Fogel.

A mesterséges intelligencia megszületésének idején az evolúciós programozást, mint eljárást, arra használták, hogy segítségével intelligensen viselkedő gépet hozzanak létre. A véges állapotú automaták alkalmasnak látszottak arra, hogy intelligens viselkedést valósítsanak meg, azaz reagáljanak a környezetükre, megfelelő válaszokat adva, úgy, hogy közben megvalósítanak egy adott feladatot.

Az evolúciós programozás ezekben az időkben egyetlen populációval dolgozott, amely egy véges állapotú automata állapotátmenet-tábláiból állt, és az utódok létrehozására egyedül a mutáció operátora szolgált.

Az 1980-as évek végén David Fogel általánosabbá tette ezt az alapelgondolást, ennek köszönhetően figyeltek fel szélesebb körben az evolúciós programozás technikájára.

Az evolúciós stratégiákhoz hasonlóan az evolúciós programozásban is tulajdonképpen azonos a keresési tér és a megoldások tere, hiszen a kromoszómák közvetlenül az optimalizálandó megoldás paramétereit tartalmazzák. A két technika közötti további hasonlóság az, hogy az evolúciós programozás egyes válfajaiban is bekerülhetnek a legrátermettebb szülők is a rátermett gyermekeken kívül az új szülőpopulációba. Az is közös vonása a két technikának, hogy meghatározó szerepet biztosítanak a mutációnak – az evolúciós programozás ezen kívül nem is használ más operátort.

Az evolúciós programozás alapalgoritmusa úgy indul, hogy véletlenszerűen előállítunk egy kezdeti populációt.

Majd az egyedeket kiértékeljük, meghatározva azok rátermettségét. Ezután választunk *egy* szülőt a véletlenszerű *versengéses szelekció (tournament selection)* alapján. Ezt a speciális kiválasztási módot úgy valósítjuk meg, hogy előre rögzítjük azon egyedek számát, akiket versenyeztetni fogunk a szaporodás lehetőségéért (t), majd a szülőválasztáskor véletlenszerűen kiemelünk t egyedet a populációból, és ezek közül választjuk ki a legrátermettebbet, ő lesz az, akinek *egy* utódja keletkezik a saját másolatának mutációjával. Az utódot ezután kiértékeljük, és beillesztjük a populációba. Mindaddig folytatjuk az új szülő kiválasztását és az új utód létrehozását, amíg a populáció mérete meg nem duplázódik. Ekkor addig töröljük a legkevésbé rátermett egyedeket, amíg a populáció ismét eredeti méretére nem fogyatkozik.

Ha elértünk egy előre meghatározott generációszámot, vagy van elfogadható megoldásunk a populációban, az algoritmus megáll, ha nem teljesülnek a leállás feltételei, elkezdjük a következő generáció előállítását.

Ha annyit módosítunk ezen az alapalgoritmuson, hogy az utód létrehozása és kiértékelése után ő maga egyből a legkevésbé rátermett egyed helyére kerül, annak törlése mellett, az evolúciós programozás *folytonos* lesz.

1997-ben Kumar Chellapilla kiterjesztette az evolúciós programozást, alkalmassá téve azt szintaktikai elemzőfák fejlesztésére.

2.6. Evolúciós művészet

Az evolúciós művészet evolúciós számítások segítségével állít elő esztétikus alkotásokat. Bár evolúciós művészet alatt elsősorban képzőművészeti alkotásokat, leginkább képeket szokás érteni, és én szintén az evolúciós művészet ezen ágát járom körül alaposabban

dolgozatomban, nem szabad elfelednünk, hogy a művészet ezerarcú, számtalan megnyilvánulása van, és ezek közül nem a képzőművészet az egyetlen, amelynek keretei között az evolúciós elveken alapuló keresőrendszereket sikerrel alkalmazhatjuk újszerű alkotások létrehozására. Az evolúció alkalmasan munkára bírva alkothat zenét [2], akár költeményeket [16] is, de iparművésznek sem utolsó, hiszen alkalmazzák textilek nyomott mintáinak fejlesztése során [15, 17], és formatervezésre is [2].

A keresés célja és a folyamatot irányító szelekciós elv ezen esztétikus alkotások esetében speciális. Hiszen nem konkrét megoldásokat keresünk, hanem felfedező utakat teszünk a keresési térben, a kreativitás ösvényeit kutatjuk, amelyet az újszerűség megnyilvánulásai szegélyeznek. Az irányító erő pedig, a kreativitás és az esztétikum felismerése és mennyiségi kifejezése roppant nehezen megfogható, ha egyáltalán lehetséges leírni valahogyan. Ezért a legtöbb esetben ezen művészi célú, vagy kreatív evolúciós rendszerekben meg sem kísérlik formalizálni a szelekciót, hanem az irányítást a rendszert működtető felhasználó, az ember kezébe adják. Természetesen nagy érdeklődés övezi azt a témakört, hogy mi módon lehetne megfogalmazni e bonyolult és sok tekintetben szubjektív szelekciós elvet, mi módon lehetne a számítógépeket megtanítani az ember esztétikai preferenciáira, ám eddig még nem sikerült ezen a téren átütő erejű eredményeket produkálnia senkinek.

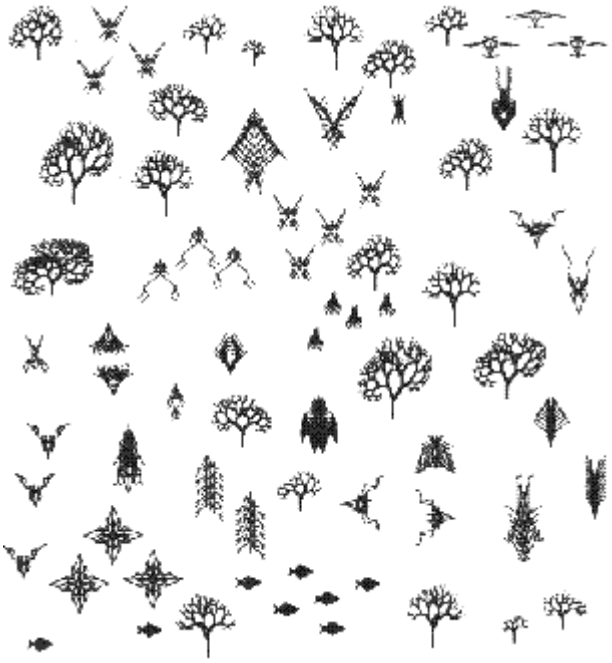
Ismerkedjünk meg néhány kiemelkedő evolúciós művésszel, néhány érdekes technikával!

2.6.1. Richard Dawkins

Richard Dawkins 1986-ban megjelent könyvében, *A vak órásmester*ben első ízben ismertetett egy olyan evolúciós rendszert, amelyben a szelekció esztétikai szempontok alapján megy végbe. Dawkins híres etológus és evolúcióbiológus, közismerten vallásellenes és aktív ateista, valamint a tudományos ismeretterjesztés elkötelezettje (1995 óta az Oxfordi Egyetem Charles Simonyi által alapított és szponzorált Tudomány-népszerűsítő Tanszékének vezetője)[13]. Oly hevesen és elhivatottan védelmezi a darwini evolúcióelmélet helytállóságát, hogy szokás „Darwin rottweilereként” emlegetni[1]. A vak órásmesterben ismertetett számítógépes programot azzal a céllal írta, hogy segítségével az evolúciós folyamatot szimulálja, és

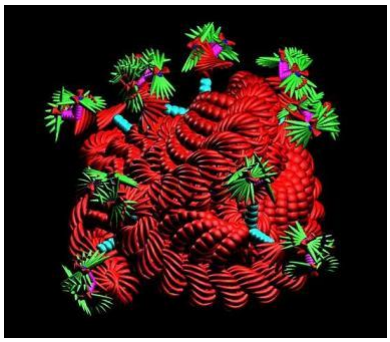
[1] A „Darwin bulldogja” jelző mintájára, amelyet Thomas Huxley kapott, aki Darwin egyik első követője volt, és aki a legtöbbet tette azért, hogy a darwini elméletet elfogadja a tudós és a laikus társadalom is. 1860-ban legendás győzelmet aratott Samuel Wilberforce oxfordi érsek elleni evolúcióról folytatott vitájukban.

demonstrálja, hogy a lépésenként bekövetkező apró változások az összegződő szelekció hatására egy kiinduló őstől jelentősen eltérő utódokat eredményeznek. Evolúciójának alanyai egyszerű, faszerkezetű alakzatok. Genotípusuk egész számokból áll, amelyek meghatározzák a fenotípust felépítő rekurziós lépések számát, az elágazások szögét, az ágak hosszát, vastagságát, színét. A szelekciót ő maga, az ember valósítja meg – a neki valamilyen szempontból tetsző egyedeket szaporítja tovább. Az eredmények őt magát is meghökkentették, fák véget nem érő sora helyett bonyolultabb élőlényekre, rovarokra, denevérekre, repülőgépekre hasonlító, a 4. ábrán látható alakok születtek, sőt „egy emlékezetes (bár megismételhetetlen) alkalommal a wykehami logikaprofesszor elfogadható karikatúrájával”[4] találta szembe magát. Dawkins ezeket az alakzatokat biomorfoknak nevezte. A „biomorfok országa” pedig programozók és művészek sorát inspirálta organikus alakzatok kifejlesztésére.



4. ábra
http://www.simonyi.ox.ac.hu/dawkins/WorldOfDawkins-archive/Dawkins/Work/Books/safari.jpg

2.6.2. William Latham



5. ábra
http://www.doc.gold.ac.uk/~mas01whl/imgs/themes_l/w_Page_022.jpg

Az eredetileg képzőművészeti képzettségű Latham 1987-től 2003-ig az IBM brit kutatócentrumának munkatársaként komputergrafikával foglalkozott [18]. Itt kollégáival, elsősorban a programozó Stephen Toddal olyan evolúciós rendszert fejlesztett ki, amellyel háromdimenziós organikus alakzatokat állított elő [11]. Ezeket egyszerű térbeli idomokból, „bordákból” (ribs), és ezen egyszerű idomokból rekurzió és iteráció sorozatával előállított összetett „szarvakból” (horns) építették fel [1][3]. Egy ilyen háromdimenziós organikus alakzat látható az 5. ábrán.

Számtalan kiállítást rendeztek képeikből és video-animációikból, és a kritikusok érdeklődését is magukra vonták, Latham mégis elégedetlen volt jelenésével a művészet színpadán, így 1994-ben megalapította saját cégét, a Computer Artworks-öt, és inkább számítógépes játékok fejlesztésével kezdett foglalkozni [11].

2.6.3. Karl Sims

Karl Simst sokan a legsokoldalúbb és leginspirálóbb evolúciós művésznek tartják. Számos figyelemre méltó evolúciós rendszert alkotott.

Nagy hatást gyakoroltak rá Richard Dawkins biomorfjai, ezek inspirációja nyomán mesterséges háromdimenziós növények egész erdeit fejlesztette ki, amelyeket 1990-es, Panspermia című filmjében csodálhatunk meg [22]. A film egy képkockáját láthatjuk a 6. ábrán.

6. ábra: http://www.genarts.com/karl/jungle200.jpg

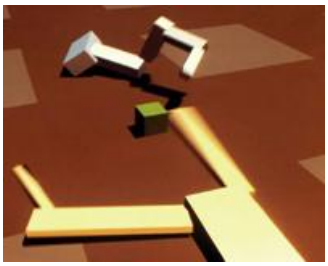
Ő volt az első, aki genetikus programozást alkalmazott kétdimenziós képek fejlesztésére 1991-ben. Ezeket a képeket egyetlen Lisp kifejezés írja le, amely meghatározza a pixel x és y koordinátája alapján az adott pixel színét. A Lisp kifejezés alapján felépíthető egy fa, ezekkel a fákkal dolgozik az algoritmus [9]. 1993-ban Genetic Images című interaktív kiállítási installációja nagy sikert aratott [20]. Két genetikus képe látható a 7. és a 8. ábrán.

7. ábra:
http://www.genarts.com/karl/extinct110.jpg

8. ábra:
http://www.genarts.com/karl/evo-jf110.jpg

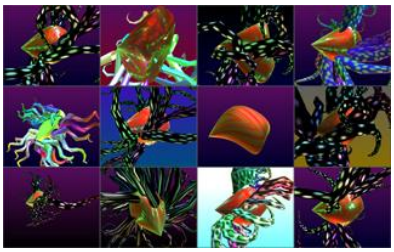
1994-ben Evolved Virtual Creatures című filmjében olyan virtuális, „kockákból” felépülő, viselkedéssel rendelkező lényeket mutatott be, amelyeket szimulált evolúcióval

hozott létre előre meghatározott célfeladatokra, mint például az ugrás, úszás, követés, vagy a 9. ábrán látható versengés [21].



9. ábra: http://www.genarts.com/karl/sweeper-vs-arm200.jpg

Latham-hez hasonlóan Sims is megalkotott egy keretrendszert a 10. ábrán látható háromdimenziós organikus alakzatok evolúciójára, amelyből Galápagos címmel ugyancsak interaktív kiállítási installációt készített 1997-ben [19].



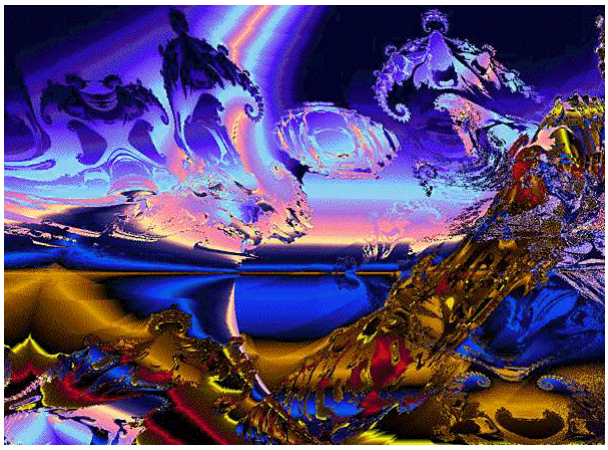
10. ábra: http://www.genarts.com/galapagos/grid3-320.jpg

2.6.4. Steven Rooke

Az eredetileg a biológia és a földrajztudományok iránt érdeklődő Rooke 1993-ban kezdett Karl Sims hatására genetikus képeket előállítani. Sims-hez hasonlóan, matematikai kifejezésekkel írja le képeit, de hosszabbakkal, és összetettebbekkel, mint mestere [11]. Talán leghíresebb képe a 11. ábrán megcsodálható afrikai álarc, az „Afman”, amelyet az alábbi hosszú Lisp kifejezés ír le: cos(pmult(pdiv(pdiv(M_PI,pdist (pmult(0.703097,pdist(pdiv (0.777147,sin(pdiv(y,pminus (-2.19658,cos(x))))),cos(cos(sin(y))))),sin(y))), cos(pplus(cos(pplus(atan(sin(cos(x))), pmult(x,x))), pdistlspiral(y,0.418353))))),0.494688))[3].

11. ábra: http://evonet.lri.fr/evoweb/images/news/th_evonews11_6-1.jpg

1997-től kezdett fraktálokat is beilleszteni képeibe, amelyekkel még hatásosabb képeket alkotott [1]. Rooke különösen kedveli a tájképekre emlékeztető képeket, mint amilyen a 12. ábrán látható.



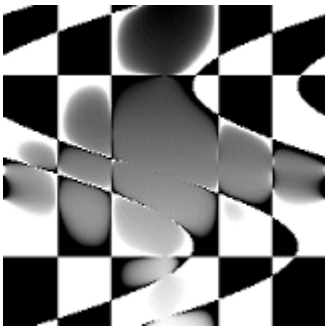
12. ábra: http://classes.yale.edu/fractals/panorama/Art/FracAsArt/GAArt/Rooke2.gif

2.6.5. Penousal Machado

Machado egy olyan projekten dolgozik munkatársaival a Coimbra-i Egyetemen, amelynek végső célja egy teljesen autonóm evolúciós művész kifejlesztése. A NEvAr [23] projekt keretein belül azt kutatják, hogyan lehet automatizálni az evolúció alatt álló képek esztétikai rátermettségének megállapítását. Kezdetben úgy gondolták, hogy neurális hálókat fognak betanítani, majd ezeket alkalmazzák a feladatra. Jelenleg a teljes automatizálást még nem tudták megvalósítani, a neurális hálókat egyéb technikák mellett csupán arra használják, hogy eltávolítsák a nem kívánatos, legkevésbé rátermett kép-egyedeket.

Jelenleg az automatikus fitness-függvény a kép fényereje alapján selejtezi ki azokat az egyedeket, amelyek egyáltalán nem esztétikusak. Ez a fajta fitness-függvény azonban csak

szürkeárnyalatos képeken működik. A 13. ábrán láthatunk egy olyan képet, amelyet a NEvAr rendszer a beépített fitness-függvény segítségével, automatikusan fejlesztett.



13. ábra: http://eden.dei.uc.pt/~machado/NEvAr/site-art/nevar-auto2.gif

2.6.6. Laurence Ashmore és Julian Miller

Laurence Ashmore Julian Miller irányítása mellett Miller speciális kromoszómafajtájára implementálta a genetikus programozás technikáját [1]. Az egyedeket indexelt csomópontokkal rendelkező irányított gráf írja le. Minden csomópontnak több inputja lehet, és van egy hozzárendelt függvénye, amely alapján a beérkező inputokra az outputot kiszámítja. Ezen, irányított gráffal leírt egyedek genotípusa egész számok listája, amely a gráf kapcsoltságát és funkcionalitását határozza meg. Ezt a technikát Julian Miller Descartes-i Genetikus Programozásnak nevezi. Ennek alkalmazásával Ashmore számos igazán esztétikus képet fejlesztett ki. Ezen képek egyike a 14. ábrán látható.

14. ábra: http://www.emoware.org/graphics/evoart5.jpg?noskin=1

3. Egy saját evolúciós művészeti keretrendszer

Amikor elkezdtem megvalósítani a saját rendszeremet, az első feladatom az volt, hogy létrehozzak egy teljesen általános evolúciós művészeti keretrendszert, amelynek magja a genetikus algoritmus, és amely bármilyen konkrét egyed-implementációval tud dolgozni, azaz bármilyen típusú egyedek populációin tudja szimulálni az evolúciót. Végül olyan keretrendszert építettem föl, amelyet nem csak művészi célra lehet felhasználni, hanem gyakorlatilag bármilyen keresési probléma genetikus algoritmussal való megoldására.

3.1. A fejlesztési platformról röviden

Magát az evolúciós keretrendszert, és a minta alkalmazásokat is a Microsoft .NET keretrendszerére készítettem, C# nyelven, a Microsoft Visual Studio 2005 intelligens fejlesztőeszköz segítségével.

A Microsoft .NET Framework egy objektum-orientált alkalmazásfejlesztési keretrendszer, amely az alkalmazások széles körének fejlesztését teszi lehetővé. A keretrendszer legfontosabb komponensei az egységes nyelvi futtatató környezet (CLR – Common Language Runtime), amely tartalmaz egy egységes típusrendszert (CTS – Common Type System), és a .NET osztálykönyvtárak. A CLR többek között egységes kivételkezeléssel, automatikus szemétgyűjtési mechanizmussal, biztonságos memóriakezelési modellel, szerepkör és kód alapú biztonsági modellel rendelkezik. A .NET keretrendszer nyelvfüggetlen, a különböző programozási nyelveken írt kódokat a fordítók egy közös nyelvre (CIL – Common Intermediate Language) fordítják. Az alkalmazás futtatási platformjának megfelelő kódot az egyes osztályok betöltése során egy futásidejű fordító (JIT – Just In-Time Compiler) állítja elő. A .NET keretrendszer támogatja az internet és intranet alkalmazások valamint a webszolgáltatások fejlesztését.

A .NET keretrendszert támogató programozási nyelvek közül kiemelkedik a C#, amely a .NET „natív” nyelvének tekinthető, mivel a keretrendszer nagy része is ebben készült. A C# egy C-hez, Java-hoz hasonló szintaxisú, objektum-orientált nyelv, amely a .NET keretrendszer minden funkcióját támogatja. A C# egyik legfontosabb jellemzője az objektum-orientáltság következetes, egyszerű, átlátható eszközökkel történő megvalósítása, amely igen hatékonnyá teszi a nyelvet.

A .NET alkalmazások fejlesztésére a Microsoft a Visual Studio 2005 nevű fejlesztőeszközt kínálja. A Visual Studio gyors alkalmazásfejlesztést (RAD – Rapid Application Development) lehetővé tévő, több programozási nyelvet és számos alkalmazástípust támogató eszköz. Fejlett kódszerkesztési, tesztelési, hibakeresési és vizuális eszközökkel rendelkezik.

3.2. A genetikus algoritmus alaposztályainak meghatározása

A genetikus algoritmus nagyobb vonalakban populációkkal dolgozik. Az algoritmus három fő lépése a kezdeti populáció létrehozása, egy adott populáció kiértékelése, majd ebből az adott populációból kiindulva a következő populáció létrehozása. Az elképzelésem az volt, hogy a programozónak, aki ezt a fajta evolúciós folyamatot vagy keresőrendszert kívánja használni, ezt a három lépést kell megfelelően a programjába illesztenie. Az egyik alapvető fontosságú osztály tehát a **Population** a rendszeremben, amely a populációkat reprezentálja.

A másik alapvető fontosságú osztályom az egyed, hiszen a genetikus algoritmusban egyedként jelennek meg a megoldások, amiket keresünk. A populáció maga is egyedekből áll. Ahhoz, hogy sokféle típusú konkrét egyedet tudjunk majd kezelni, egy általános egyedet, egy egyed-interfészt (**IIndividual**) hozok létre. Minden konkrét alkalmazásban ezt az egyed-interfészt kell majd implementálni ahhoz, hogy populációba csatolva a konkrét egyedeket, elindíthassuk az evolúciót.

A harmadik osztályom „belső” használatú, vagyis a programozónak, aki az evolúciós rendszeremet implementálja, nem kell vele dolgoznia. Azért hoztam létre, mert hűen szerettem volna implementálni a kanonikus genetikus algoritmust, amely az új populáció létrehozása során egy köztes „konténert”, egy úgynevezett „szaporodási alapot” használ arra, hogy a természetes szelekciót szimulálja. Ennek megfelelően létrehoztam még egy osztályt, a szaporodási alapot (**MatingPool**), amellyel csak a populáció objektumok dolgoznak a gyermekpopuláció létrehozásakor.

Ahhoz, hogy egy általános egyed-intrefészt tudjak létrehozni, még két segédosztályra volt szükségem, amelyekből a konkrét egyedtípus konkrét genotípusa és fenotípusa fog származni. Erre a célra két absztrakt osztályt hoztam létre, a genotípus-alapot (**GenotypeBase**) és a fenotípus-alapot (**PhenotypeBase**). A programozónak az egyed-interfészen kívül ezt a két absztrakt osztályt is implementálnia kell a saját, konkrét alkalmazásában.

3.3. A keretrendszer megvalósításáról

A keretrendszer megvalósításához az fentiekben bevezetett alaposztályokon kívül még szükség volt egy úgynevezett delegált, vagy függvénymutató létrehozására. Ahhoz ugyanis, hogy a teljesen általános populációt, amely egyed-interfészekkel dolgozik, az alkalmazásokban konkrét egyedtípusú véletlenszerűen előállított egyedekkel tudjuk feltölteni, a konkrét egyedtípusnak implementálnia kell a véletlenszerű egyedlétrehozás metódusát, azaz az egyed-interfész **GenerateRandomly** metódusát.

Ez a keretrendszer olyannyira általános lett, hogy azt a lehetőséget is biztosítja, hogy ne csak véletlenszerűen generált kezdeti populációval indíthassuk el az evolúciós folyamatot, hanem a felhasználó által megadott konkrét egyedekkel is.

Az én genetikus algoritmus implementációmmal kapcsolatban meg kell jegyezni, hogy ebben az esetben a 0 rátermettségi (fitness) érték jelenti azt, hogy az adott egyed egyáltalán nem rátermett, vagyis szaporodási esélye 0. Minél nagyobb a rátermettségi értéke az egyednek, annál nagyobb az esélye a szaporodásra. Ha a populáció minden egyedének 0 a rátermettsége, a populáció nem tud tovább szaporodni, kihal.

3.4. Az alaposztályok használata

A programozónak, aki ezekkel az alaposztályokkal kíván dolgozni, és a genetikus algoritmust szeretné használni, a következőket kell tennie.

A saját alkalmazásának megfelelően implementálnia kell a **GenotypeBase** és a **PhenotypeBase** absztrakt osztályokat, majd az **IIndividual** interfészt. Az interfész implementációban meg kell adni a saját genotípus típusát és a saját fenotípus típusát, majd a saját genotípus típusának megfelelően kell implementálni a **Crossover** és a **Mutate** operátorokat. A programozónak be kell illesztenie a **GenerateRandomly** metódust statikus metódusként az interfész implementációjában, és implementálnia kell a saját egyedtípusának megfelelő objektum véletlenszerű létrehozását, ha azt szeretné, hogy a **Population** objektumok véletlenszerűen tudják inicializálni magukat a saját egyedtípusuknak megfelelő objektumokkal.

A keretrendszert implementáló programozónak az alkalmazása megfelelő helyén létre kell hoznia egy **Population** objektumot, megadva a konstruktor számára a populáció egyedeinek számát és az egyedek véletlenszerű létrehozását szolgáló metódus nevét. Ha véletlen egyedekkel kívánja feltölteni ezt a kezdőpopulációt, meg hívnia a **Population**

objektum **Initialize** nevű metódusát. Ezután meg kell valósítania az egyedek kiértékelését, interaktív vagy automatizált módon. Miután a populáció egyedeinek kiértékelése megtörtént, a populáció objektumon meg kell hívni a **Breed** metódust, ez valósítja meg a gyermekpopuláció létrehozását.

4. Kreatív filterek fejlesztése

A *Kreatív filterek (CreativeFilters)* egy olyan egyszerű webes alkalmazás, amelynek segítségével a képfeldolgozásból ismert filtereket variálva újszerű, különleges, kreatív filtereket fejleszthetünk ki, amelyeket aztán tetszőleges képre alkalmazhatunk. Ez az alkalmazás saját ötleten alapul.

Egy kreatív-filter egyed genotípusa tulajdonképpen az alap filterek egy variációja, ahol minden filtert egyedileg paraméterezünk föl.

Egy adott kreatív-filter fenotípusa egy képen értelmezhető. Az alkalmazásban még az evolúciós folyamat kezdete előtt rögzítünk egy képet, amelyen majd az egyes kreatív-filter egyedek fenotípusát értelmezni fogjuk. Az egyedek fenotípusát úgy építjük föl, hogy a filtereket egymás után alkalmazzuk a rögzített képre. A végeredményül kapott filterezett kép lesz a kreatív filterünk fenotípusa, ezt a képet jelenítjük meg az alkalmazás weblapjain.

A Kreatív filterek egy kollaboratív evolúciós művészeti alkalmazás, amelyben a szelekció nem automatizált, a filterek rátermettségét felhasználó állapítja meg esztétikai szempontok alapján.

4.1. A felhasznált grafikus filterek

Az általam használt egyszerű filtereket Christian Graus implementálta, aki a „The Code Project” nevű .NET fejlesztői portál munkatársa. Ezen a portálon publikált egy cikksorozatot [6], amely bevezetést nyújt a képfeldolgozásba, és a cikkekhez kötődően egyszerű filter implementációkat tett közzé. Ezeket az implementációkat használtam fel az alkalmazásomban, pontos fellelhetőségeiket a dolgozatom végén, az irodalomjegyzékben tüntettem fel.

Minden filter BMP formátumú képet vár paraméterként, csak ilyen formátumú képeken dolgozik. A BMP formátum egyfajta bitmap formátum, amelyben a kép pixeltömbként kerül tárolásra. A pixelek színét az RGBA színmodellnek megfelelően négy komponensben tároljuk, a pixel színe e négy komponens alapján áll elő. Ezek a vörös, zöld, kék, és az alfa komponens. Ez utóbbi a szín áttetszőségét jellemzi. Minden komponenst egy bájton ábrázolunk, vagyis 0-től 255-ig vehet fel értéket.

Alkalmazásomban az alábbi alap filtereket használtam fel.

Az *Invert* nevű filter invertálja a színeket, azaz minden pixel mind a négy komponensének értékét kivonja 255-ből. Ha nullánál kisebb értéket kapna, 0 lesz a pixel adott színkomponensének értéke. Ennek a filternek nincs paramétere. Hatása a 16. ábrán látható.

A *Smooth* homályosító, „párásító" hatású filter. Egymás után többször alkalmazva egyre homályosabb képet kapunk eredményül. Az alkalmazásomban a Smooth filter paraméterének tekintem azt az egész számot, amely azt mutatja, hogy hányszor alkalmazzuk egymás után a képre. Hatása a 17. ábrán látható.

Az *EdgeDetectHomogenity* nevű filter egy él-detektáló filter, a képen kiemeli az alakzatok körvonalait. Ennek a filternek nincs paramétere. Hatása a 18. ábrán látható.

Az *Embossing* térbeli hatást keltő filter, a kép elemeit „kidomborítja". Ennek a filternek sincs paramétere. Hatása a 19. ábrán látható.

15. ábra:
az eredeti kép

16. ábra:
Invert

17. ábra:
Smooth(5)

18. ábra:
EdgeDetectHomogenity

19. ábra:
Embossing

A *Brightness* a kép fényerejét módosító filter. A kép pixeleinek komponenseihez hozzáadja a paraméterként kapott, -255 és 255 közötti egész értéket (ha a kapott érték kisebb nullánál, az értéket nullára állítjuk, ha nagyobb 255-nél, akkor 255-re), így világosítja vagy sötétíti a képet. Hatása a 20. ábrán látható.

A *Color* színfilter. Három paramétere van, a kép pixeleinek megfelelő (sorrendben: vörös, zöld, kék) színkomponenseihez hozzáadja a megfelelő paraméter értékét. A paraméterek -255 és 255 közötti egész értékek. Ha a komponens és a paraméter értékeinek összeadásával keletkező eredmény kisebb nullánál, az értéket nullára állítjuk, ha nagyobb 255-nél, akkor 255-re. Hatása a 21. ábrán látható.

A *Flip* tükrözést hajt végre, a vertikális vagy a horizontális tengelyre, vagy mindkét tengelyre egyszerre, attól függően, hogy a két logikai értékű paraméterét hogyan állítjuk be. Hatása a 22. ábrán látható.

A *RandomJitter* olyan hatást kelt, mintha a pixelek vibrálnának, rezegnének, és egy pillanatfelvétel készülne erről a rezgő képről. Egy egész értékű paramétere van, amellyel a vibrálás mértékét szabályozhatjuk. Hatása a 23. ábrán látható.

A *Swirl* olyan hatást kelt, mintha a kép örvénylene. Két paramétere van, az egyik az örvénylés mértékét szabályzó valós értékű, a második egy logikai érték, amely azt jelenti, alkalmazunk-e a képre homályosítást is, vagy sem. Hatása a 24. ábrán látható.

20. ábra: Brightness(-50)

21. ábra: Color(200, 0, 0)

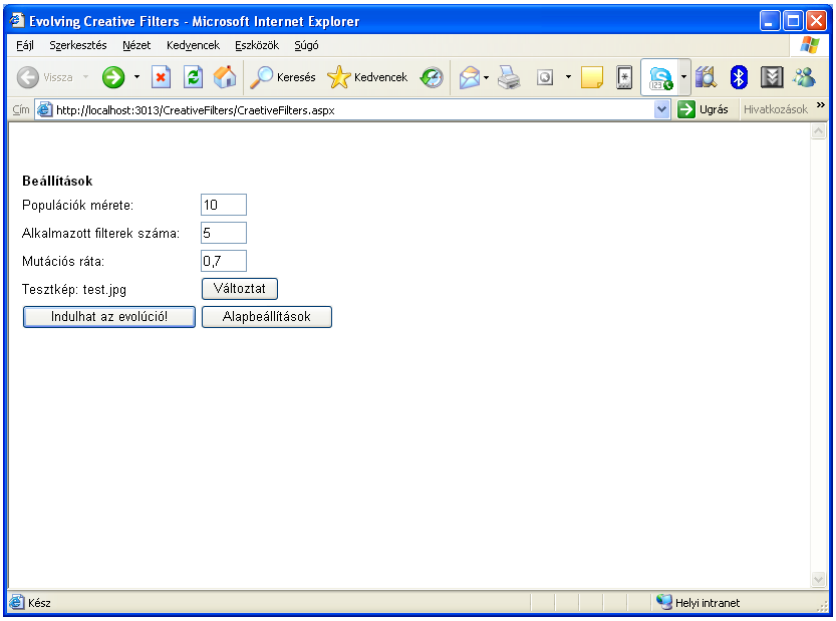
22. ábra: Flip(false, true)

23. ábra: RandomJitter(5)

24. ábra: Swirl(0.045, false)

4.2. A Kreatív filterek alkalmazás működése

A Kreatív filterek webes alkalmazás indításakor az 25. ábrán látható oldal jelenik meg. Ezen az oldalon először az alapbeállításokat módosíthatjuk.



25. ábra

Ha nem módosítjuk az adatokat, a képen látható alapértelmezett beállítások érvényesülnek.

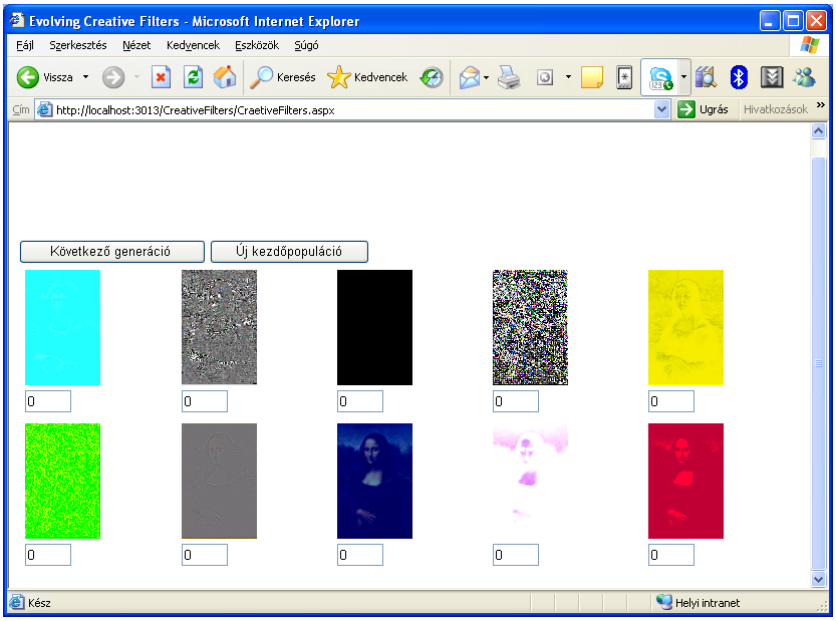
A populációk mérete az egyes populációk egyedeinek száma, vagyis az egyazon generációba tartozó filterek száma, praktikusan a képek száma. Ha türelmetlenek vagyunk, és nem szeretnénk sokat várni az új képek megjelenésére, nem érdemes túl nagyra állítani a populációméretet.

Az alkalmazott filterek száma azt takarja, hogy mennyi filter alkalmazásával álljon elő a kreatív filter. Ezt az értéket sem célszerű nagyra állítani, mert a műveletek lassulásán kívül minél több filtert alkalmazunk egymás után, annál nagyobb valószínűséggel keletkeznek „használhatatlan”, vagyis monokróm (egyszínű), szürke, fekete filterek. Persze kísérletezni szabad, és érdemes!

A mutációs ráta azt mutatja, hogy az utódok genotípusa milyen valószínűséggel módosul, azaz mutálódik. Ha a mutációs ráta értéke 1, minden egyeden mutáció megy végbe. Ha azt szeretnénk, hogy az evolúció folyamán ne homogenizálódjanak a generációink, érdemes a mutációs rátát viszonylag nagy értékűre, az 1 közelére állítani.

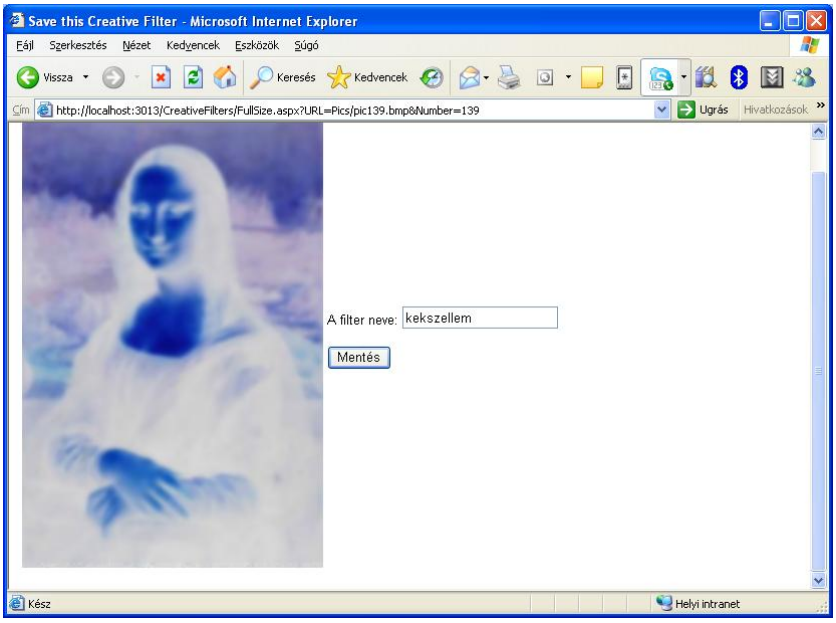
Az alkalmazás egy alapértelmezett képen dolgozik, de ezt meg lehet változtatni (ehhez a tesztkép neve melletti „Változtat” gombra kell kattintanunk). A megadható kép BMP, PNG és JPEG formátumú lehet. Mivel a filterek csak BMP képeken dolgoznak, az alkalmazás szükség esetén erre a formátumra alakítja a képet.

Az „Alapbeállítások” gombra kattintva visszaállnak az alapértelmezett értékek.
Ha az „Indulhat az evolúció!” gombra kattintunk, a lapról eltűnnek a beállításokat tartalmazó mezők, és megjelenik az első generáció, ahogy azt a 26. ábrán látjuk.



26. ábra

A populáció egyedeit a lap kicsinyítve jeleníti meg, hogy könnyen összehasonlíthatóak legyenek a képek. Ám érdemes kinagyítani az egyes képeket, bármelyikre kattintva, mert érdekes részleteket fedezhetünk fel rajtuk, amik kicsinyítéskor nem láthatóak. Ha a megtekintett filter annyira tetszik nekünk, hogy szívesen alkalmaznánk a saját képeinkre is, elmenthetjük azt, tetszőleges (állománynévnek megfelelő) néven.



27. ábra

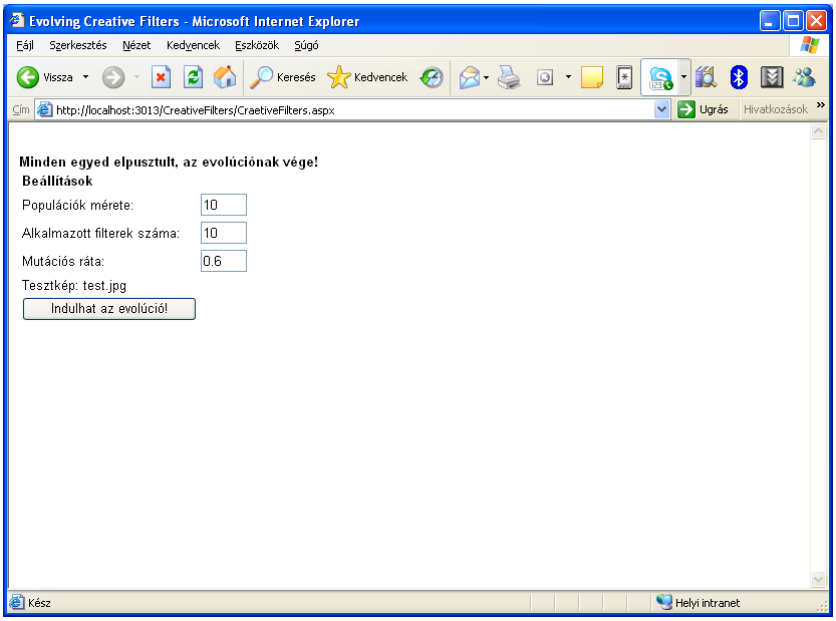
Az alkalmazás a kreatív filter egyedeket XML állományokban tárolja. A filterek egy rögzített könyvtárban kerülnek mentésre. Később ezeket a filtereket ténylegesen alkalmazhatjuk a saját képeikre.

Mivel sok olyan kreatív filter jön létre a véletlenszerű inicializálás során, amely nem nyújt jó kiindulási alapot az esztétikus filterek fejlesztéséhez, lehetőségünk van arra, hogy új kezdeti populációt generáljunk, ha az adott kezdőpopuláció nem nyerte el a tetszésünket. Ha az „Új kezdőpopuláció” gombra kattintunk, a lapon egy új kezdeti populáció egyedei jelennek meg.

Ha találunk több jó kiindulási alapot jelentő filtert a kezdőpopulációban, az általunk megítélt rátermettségi értékeket beírjuk a képek alatt található mezőkbe.

Ha minden nekünk tetsző filtert kiértékeltünk, a „Következő generáció” gombra kattintva a lapon kis várakozás után megjelenik a következő generáció.

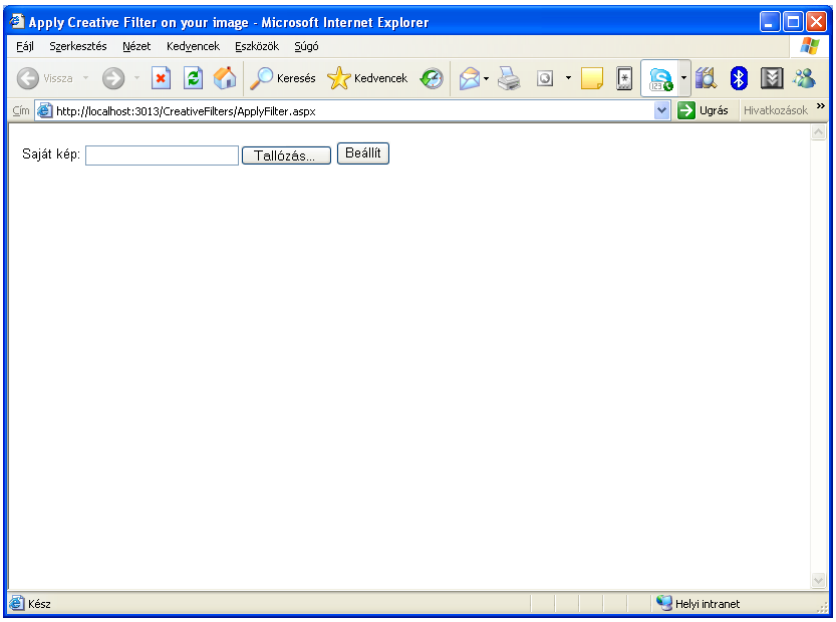
Ha a felhasználó véletlenül, vagy szándékosan nem ad nullától különböző rátermettségi értéket egyetlen filternek sem, az evolúció leáll, mert egyetlen egyed sem képes tovább szaporodni. Ekkor a 28. ábrán látható üzenetet kapjuk.



28. ábra

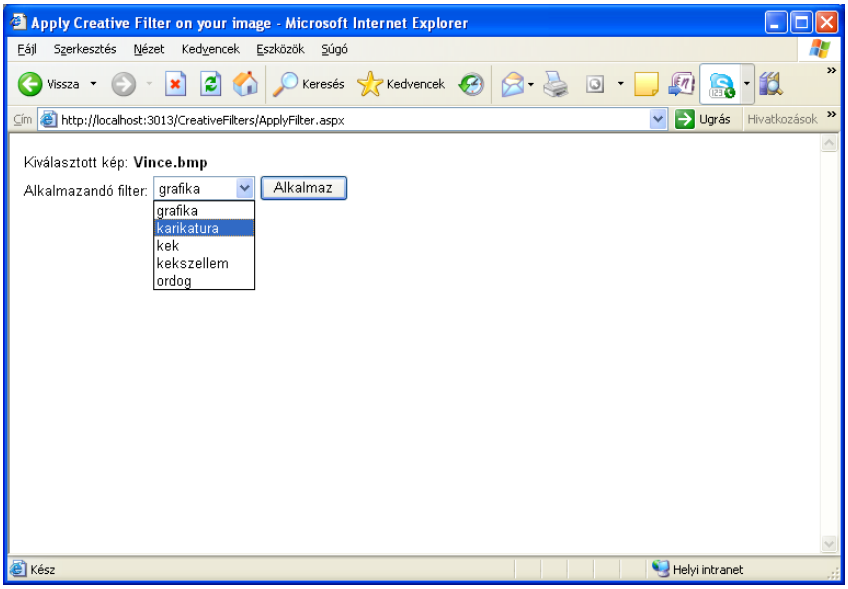
Ilyenkor egy új evolúciós folyamatot indíthatunk el, vagy a böngésző ”Vissza” gombjára kattintva visszaléphetünk a legutóbbi generációhoz, és mégis rátermettnek nyilváníthatunk néhány egyedet a tovább szaporodásra.

Az elmentett filtereket tetszőleges képünkre alkalmazhatjuk a 29. ábrán látható lapon.



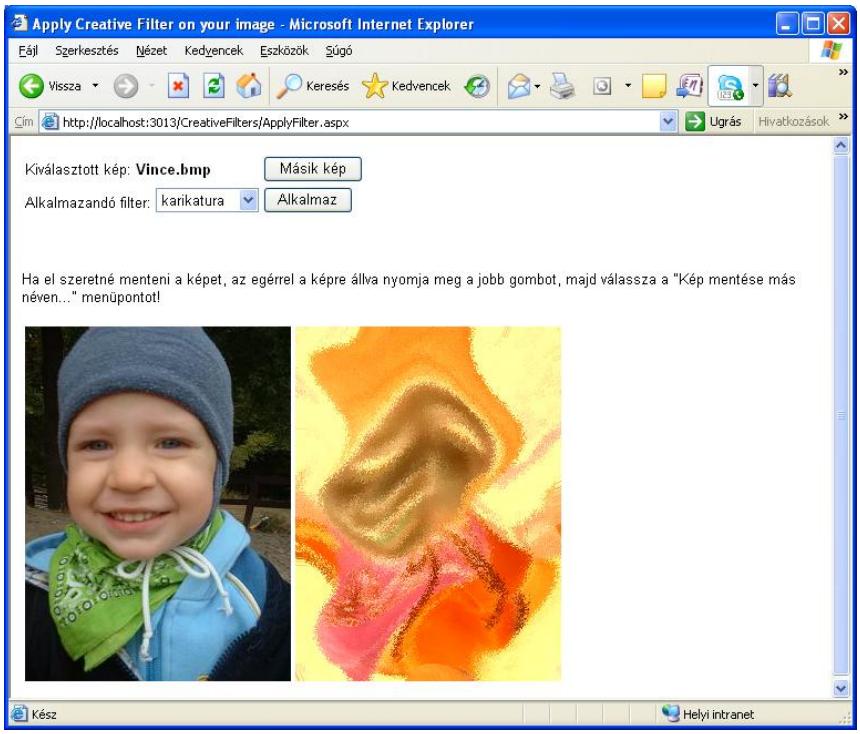
29. ábra

Feltölthetjük bármely BMP, PNG, JPEG formátumú képünket, majd ha a „Beállít” gombra kattintunk, megjelenik az elérhető filtereket mutató legördülő lista, a 30. ábrán látható módon.



30. ábra

A legördülő listából kiválaszthatjuk a kívánt filtert, majd az „Alkalmaz” gombra kattintva megtekinthetjük az eredményt az eredeti képpel összehasonlítva (31. ábra).



31. ábra

4.3. Az alkalmazás megvalósításáról

4.3.1. A genotípus megvalósítása

A kreatív filter egyedek genotípusa egy rögzített hosszúságú, összetett lista, amelynek egy eleme, génje vagy egy egész szám, vagy egy újabb lista. Az a gén, amely egyetlen egész számot tartalmaz, az adott azonosítójú, paraméter nélküli filtert kódolja. Ha a gén maga is egy lista, akkor ez a gén egy paraméterezett filtert kódol, és úgy kell értelmezni, hogy a lista első, egész típusú eleme a megfelelő filter azonosítója, a lista további elemei pedig az illető filter paraméterei.

4.3.2. A fenotípus megvalósításáról

A kreatív filterek fenotípusát a korábban leírt módon, egy rögzített tesztképen alkalmazva tudjuk értelmezni. A tesztképet az evolúciós folyamat kezdetén, az egész folyamat idejére rögzítem, minden egyed fenotípusa ez alapján az egy kép alapján fog előállni.

4.3.3. Az egyedfejlődés megvalósításáról

Az egyedek kiértékelése előtt, vagyis az egyedek képernyőn történő megjelenítése előtt a populáció minden egyedének fenotípusát legenerálom, azaz a rögzített tesztképre sorban alkalmazom a genotípusban meghatározott filtereket. Az így előállított BMP formátumú kép lesz a kreatív filter egyed fenotípusa.

4.3.4. A keresztezés operátorának megvalósításáról

A keresztezést a genetikus algoritmusok felfogása szerint valósítom meg. Véletlenszerűen választok egy vágási pontot a szülők kromoszómáján (genotípusán). Mindkét szülő kromoszómáját elvágom ezen a ponton, és a következőképpen állítom elő az újonnan létrehozott gyermekek kromoszómáit:

Az első gyermek első kromoszómafele az első szülő első kromoszómafelével fog megegyezni, a második kromoszómafele pedig a második szülő második kromoszómafelével. A második gyermek első kromoszómafele a második szülő első kromoszómafelével fog megegyezni, második kromoszómafele pedig az első szülő második kromoszómafelével.

4.3.5. A mutáció operátorának megvalósításáról

A mutációt úgy valósítom meg egy egyeden, hogy véletlenszerűen kiválasztom egy génjét, amit kicserélek egy másikra. Ha a módosítandó gén egész szám volt, vagyis paraméter nélküli filter, a helyére véletlenszerűen generálok egy új gént, ami lehet szintén paraméter nélküli filtert azonosító egész szám, vagy egy paraméterezett filtert tartalmazó lista. Ha azonban a módosítandó gén egy lista, vagyis egy paraméterezett filter, a mutációt úgy értelmezem, hogy az új gén a módosítandó gén alapján áll elő, úgy, hogy annak csak valamely paraméterét változtatom meg véletlenszerűen.

5. Hangya-festmények

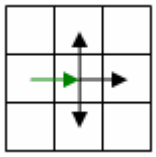
A *Hangya-festmények (Ant paintings)* egy olyan egyszerű Windows alkalmazás, amely egy, a 2003-as Evolúciós Számítások Kongresszuson (Congress on Evolutionary Computation) ismertetett interaktív evolúciós rendszer alapötletének egy megvalósítása.

Az alapötlet Nicolas Monmarche-tól és társaitól [7] származik, és részben már a mesterséges intelligencia egy új területe, a rajintelligencia (swarm intelligence) felé mutat. A rajintelligencia viszonylag fiatal tudományterület – magát a kifejezést először Gerardo Beni és Jing Wang használta 1989-ben [14]. A rajintelligencia a kollektív viselkedést vizsgálja decentralizált, önszerveződő rendszerekben. Ezekben a rendszerekben olyan ügynökök egy populációjának viselkedését vizsgálják, amelyek nem állnak központi irányítás alatt, így egyéni viselkedésüket csak a környezetük, és az ügynökök egymás közötti interakciói befolyásolják. Az alapötlet megvalósításához nincs szükség arra, hogy ennél mélyebben betekintsünk a rajintelligenciába. Elég annyit megjegyeznünk, hogy a képeinket ilyen, a rajintelligenciában ismeretes ügynökök fogják „megfesteni", akiket a továbbiakban hangyáknak nevezünk.

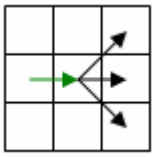
Ezek a hangyák a természetben előforduló hangyák egyszerű analógiái. A természetben az egy fajhoz tartozó hangyák olyan feromonokat (szaganyagokat) hagynak hátra maguk után az ösvényen, amelyen végighaladnak, amelyet fajtársaik felismernek, és követhetnek. A feromonok idővel elillannak, de azon ösvények mentén, amelyek bő élelemforrásokhoz vezetnek, a szaganyagok újra és újra elhelyeződnek. Azon ösvényekről pedig, amelyek már nem vezetnek élelemhez, fokozatosan eltűnik a fajtársakat oda irányító feromon-jelzés.

A mi mesterséges hangyáink egy kezdetben üres képen indulnak el, és ezen mozognak. Akárcsak természetes „társaik", bizonyos jelzőanyagot hagynak hátra az ösvényen, amelyen járnak – ez esetünkben egy, az adott hangyára jellemző szín lesz.

Mesterséges hangyáink saját, egyéni viselkedéssel bírnak. Ez azt jelenti, hogy minden hangya másképpen halad, és másképpen reagál a többi hangya által hátrahagyott jelzésre (színre). A hangyák kétféle módon lépnek tovább egy adott pontból. Vagy derékszögben, vagy 45°-os szögben keresik a következő lépés lehetőségét. Egy adott érkezési irány (zöld nyíl) esetén a lehetséges következő lépések (fekete nyilak) a következőek a két lépésmód esetén:



32. ábra: derékszögű lépésmód



33. ábra: 45°-os szögű lépésmód

Egy hangya egyéni viselkedését a lépések irányán kívül az is befolyásolja, hogy nem egyforma valószínséggel lép a három lehetséges irányba. Egy adott hangya P_b valószínűséggel lép balra, P_e valószínűséggel lép egyenesen és P_j valószínűséggel lép jobbra, ahol három valószínűség összege 1.

A hangya döntését egy lépés megtételekor még egy tényező befolyásolja. Minden hangyára értelmezünk egy, az adott hangya által keresett színt is. Ha valamelyik lehetséges lépés helyén a hangya megtalálja a keresett színt, P_k valószínűséggel követi azt. Azaz P_k valószínűséggel választja a keresett színt tartalmazó pontot következő lépés gyanánt.

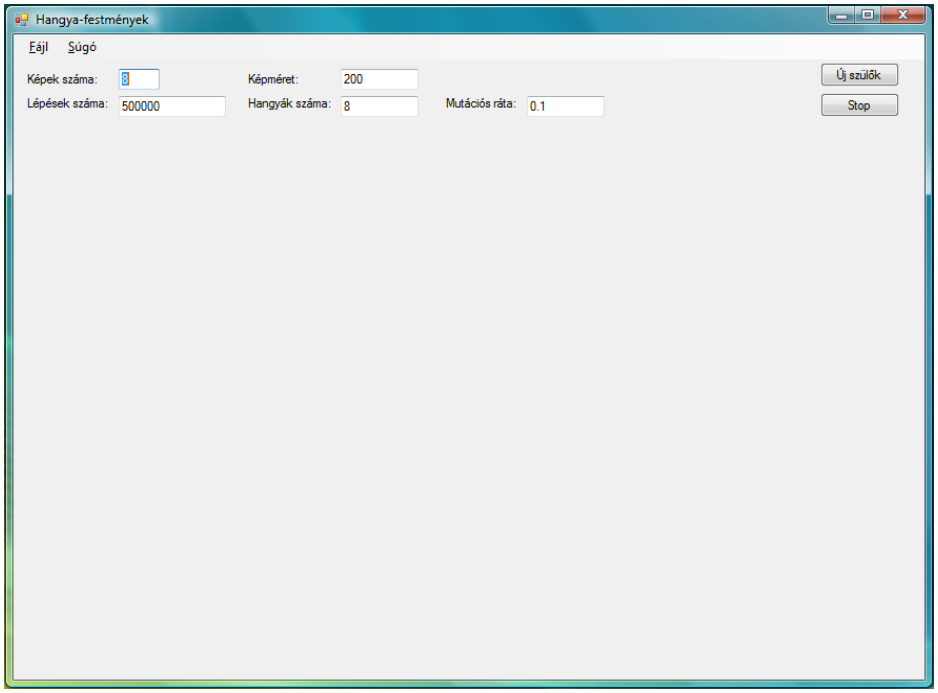
Amikor egy hangya a következő lépéséről dönt, először is megvizsgálja, hogy a lehetséges lépésirányok valamelyike tartalmazza-e az általa keresett színt. Ha igen, P_k valószínűséggel lép abba az irányba, ahol a keresett szín található. Ha úgy dönt, hogy nem követi a keresett színt, akkor a P_b, P_e, P_j lépéspreferenciák alapján hozza meg a döntését arról, hogy balra, egyenesen, vagy jobbra haladjon tovább.

Egy hangya-festmény úgy jön létre, hogy egy üres képen véletlenszerűen elindítunk kettő, vagy több hangyát, akik, miközben a képen haladnak, festéknyomot hagynak maguk után. Elegendően sok lépés megtétele után a képet festékpacák és rajzolatok fogják kitölteni.

Ezt a fentiekben leírt ötletet használtam fel egy újabb saját kollaboratív evolúciós alkalmazás létrehozásához. Alkalmazásomban olyan képeket fejlesztek, amelyeket kettő vagy több hangya „fest” meg. A hangyák egy kezdetben üres képen fognak mozogni. Azokon az ösvényeken, amelyeken járnak, hátrahagyják a saját színüket. Egyéni viselkedésük alapján különböző megjelenésű nyomokat hagynak. Vannak, akik szálakat húznak maguk után, vannak, akik csipkés mintát rajzolnak, vannak, akik festékpacákat hoznak létre. A hangyákat mozgását egy pillanatban megállítjuk, és így rögzítjük a képet. Az alkalmazással ilyen, hangyák által festett képeket esztétikus képeket lehet keresni.

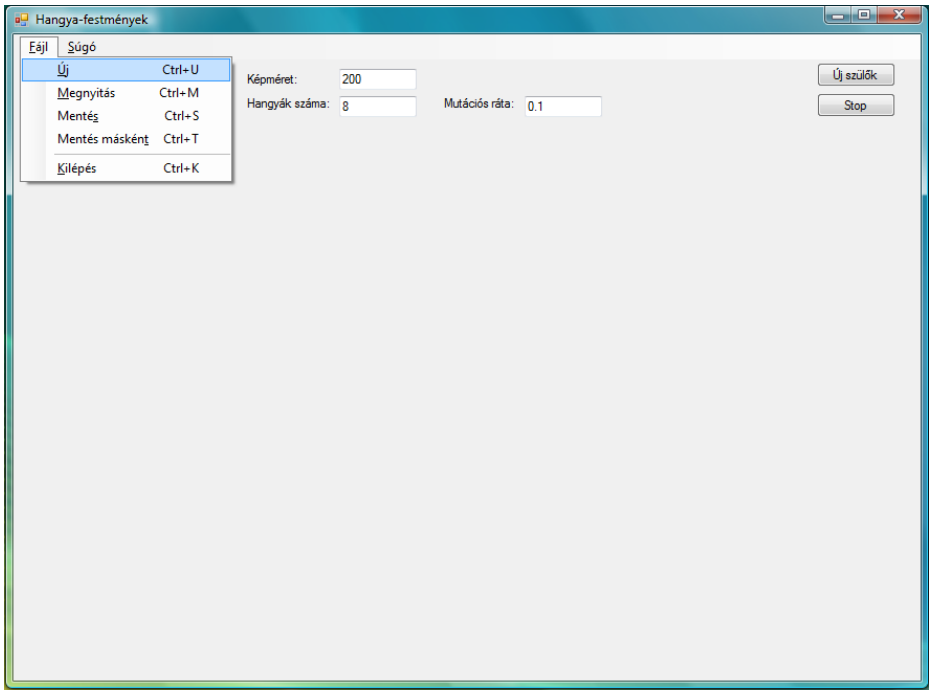
5.1. A Hangya-festmények alkalmazás működése

Az alkalmazás indításakor megjelenik az alkalmazás ablaka (34. ábra). A Hangya-festmények alkalmazás valamennyi funkciója ezen a felületen keresztül érhető el.



34. ábra

Az űrlap felső részén módosíthatjuk az alapbeállításokat. Beállíthatjuk a fejlesztendő képek számát, vagyis a populációk méretét, az egy képen található hangyák maximális számát, a képek méretét, a mutációs rátát, és azt, hogy a hangya-festmények hány lépés után véglegesítődjenek, azaz a hangya-festmény fenotípusa mennyi lépés után kerüljön rögzítésre. Egy 200×200-as képet általában 1.000.000 lépés után járnak be a hangyák. Hangsúlyozom, hogy általában, ugyanis vannak olyan hangyák, amelyek nagyon „introvertáltak”, azaz úgy viselkednek, hogy csak nagyon kis területet képesek bejárni akár 1.000.000 lépés alatt is. A 34. ábrán látható, alapértelmezett beállítások érvényesülnek, ha másképp nem rendelkezünk.



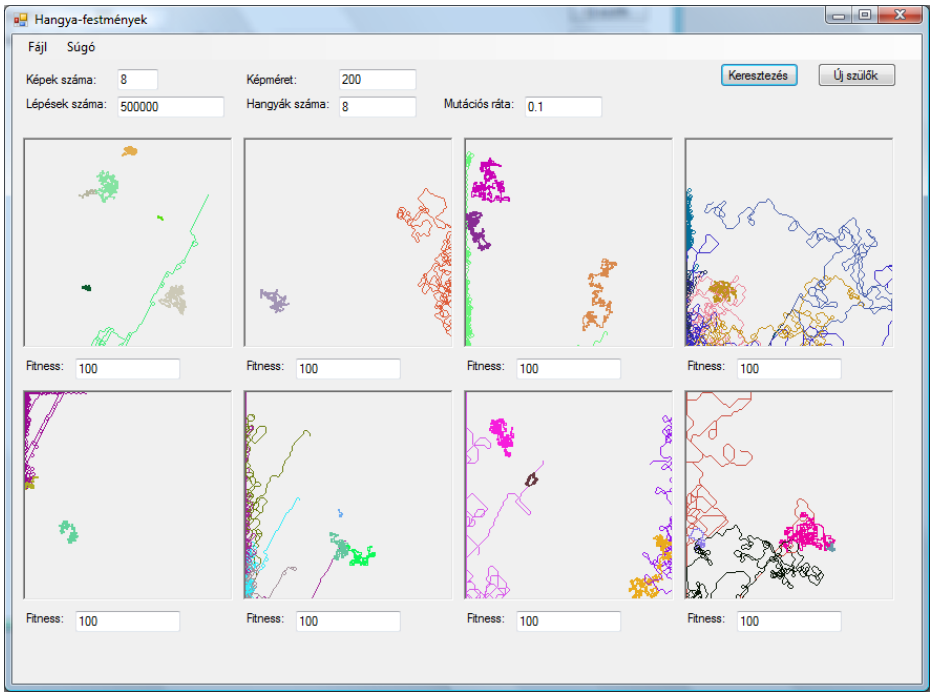
35. ábra

Ha a Fájl menü „Új” parancsát választjuk (35. ábra), egy új evolúciós folyamatot indítunk el. Ezen evolúciós folyamat során automatikusan mentést készít az alkalmazás az egymás után keletkező populációkról. Az alkalmazás működésének bármely pontján választhatjuk az új evolúciós folyamat indításának lehetőségét, ekkor törlődnek az előző evolúciós folyamat automatikus mentései.

Ahhoz, hogy előálljon a kezdeti populáció, az „Új szülők” gombra kell kattintanunk.

Az alkalmazás a kezdetben üres képeket egyidejűleg, egymással párhuzamosan kezdi el létrehozni. A képhez tartozó hangyákat a kép véletlenszerűen kiszámított pontjából, véletlenszerűen generált kezdő lépésiránnyal indítjuk el. A képekek 100 lépésenként frissítjük, 100 lépésenként jelenítjük meg rajtuk a változásokat. A 36. ábrán egy pillanatképet láthatunk az új kezdőpopuláció egyedfejlődésének kezdetéről.

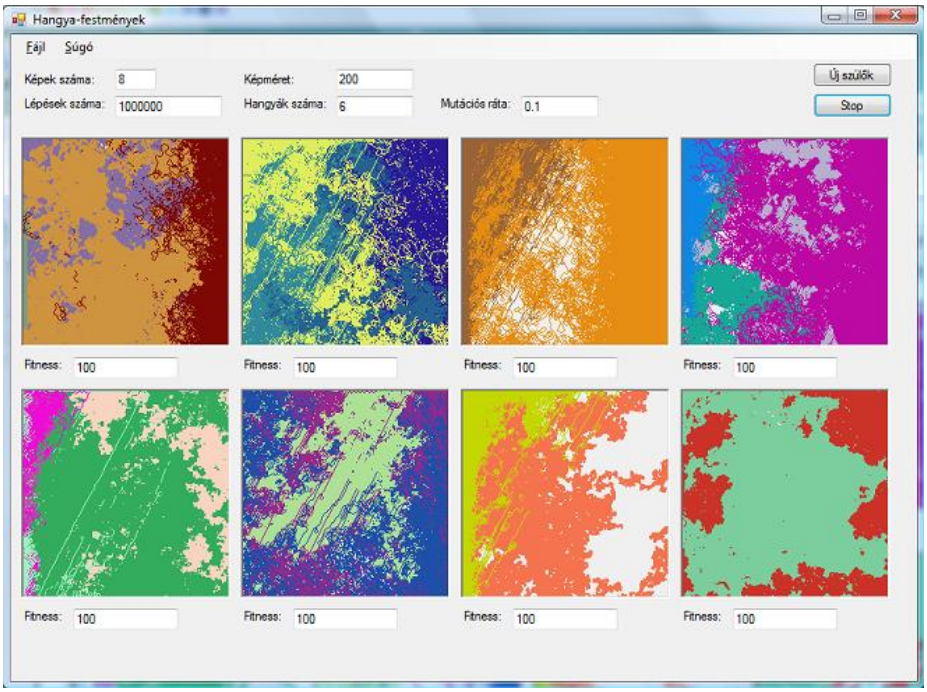
Ha a felhasználó a „Stop” gombra kattint, megállítja a képek generálását. Ha a felhasználó nem avatkozik be a képrajzolás folyamatába, akkor a fent megadott maximális lépésszám után az magától leáll.



36. ábra

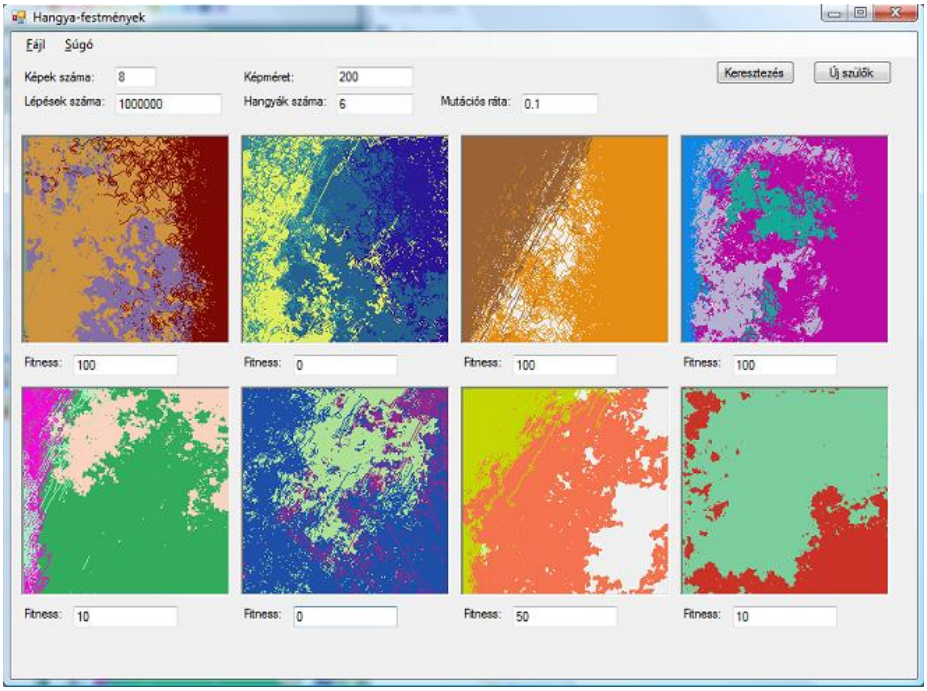
Ha nem vagyunk elégedettek a kezdőpopulációval, az „Új szülők” gombra kattintva ismét egy új kezdőpopulációt generálhatunk.

A 37. ábrán az új kezdőpopuláció egyedfejlődésének egy pillanatképét láthatjuk.



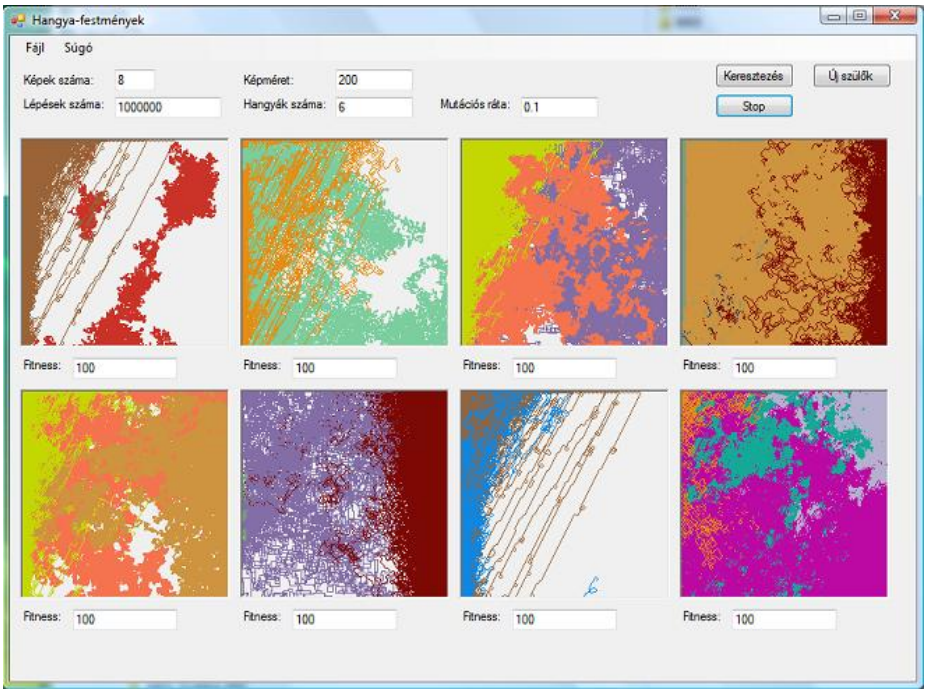
37. ábra

Miután megvártuk az egyedfejlődés végét, vagy leállítottuk azt, kiértékeljük a populációt. Azokat az egyedeket, amelyeket ki szeretnénk zárni a további szaporodásból, 0 fitness értékkel kell ellátnunk. Minél inkább tetszik nekünk egy kép, annál nagyobb rátermettségi értéket állítunk be rá. Az ábrán a populáció egy kiértékelése látható. Ha elő kívánjuk állítani a következő populációt, a „Keresztezés” gombra kell kattintanunk.



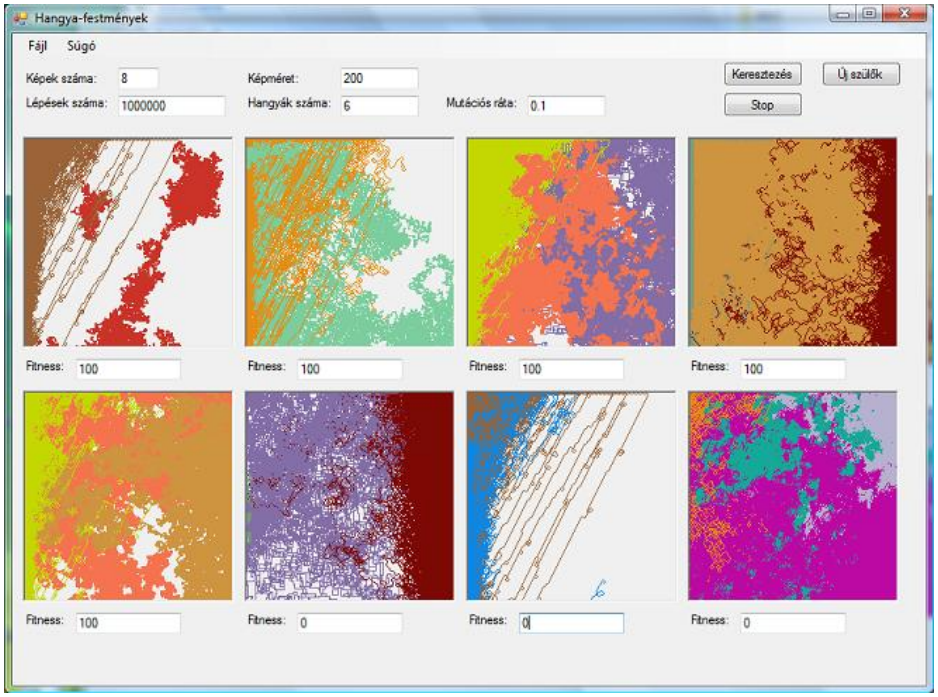
38. ábra

A 39. ábrán az előbbi kezdőpopuláció alapján a 38. ábrán látható kiértékelésünknek megfelelően létrejött második generáció egyedeit láthatjuk.



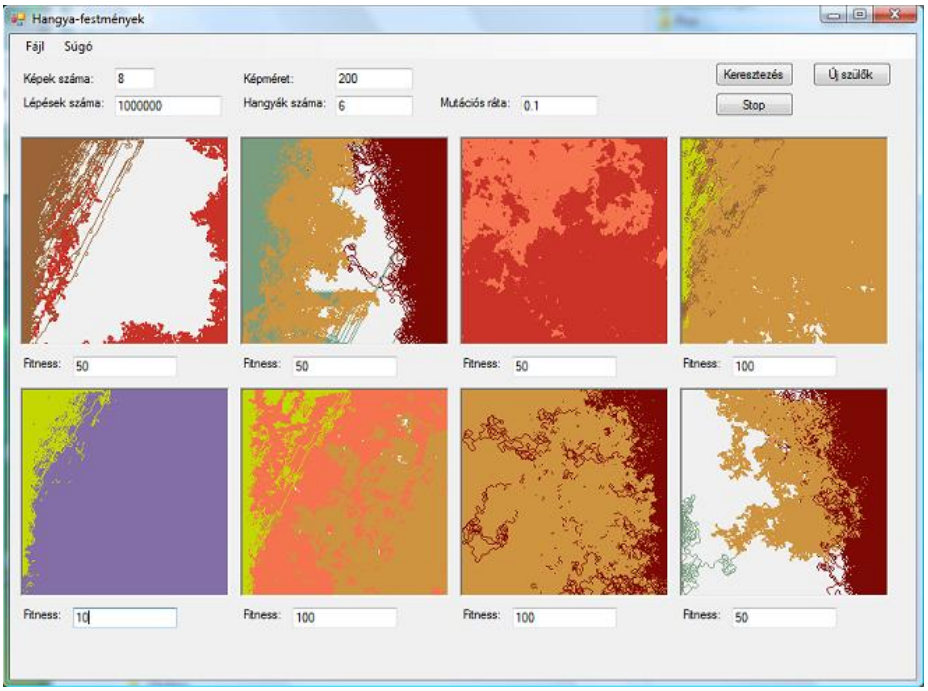
39. ábra

Lássuk, milyen képek keletkeznek néhány lépés után! A második generációt a 40. ábrán látható módon értékeltük ki, majd a „Keresztezés” gombra kattintottunk.



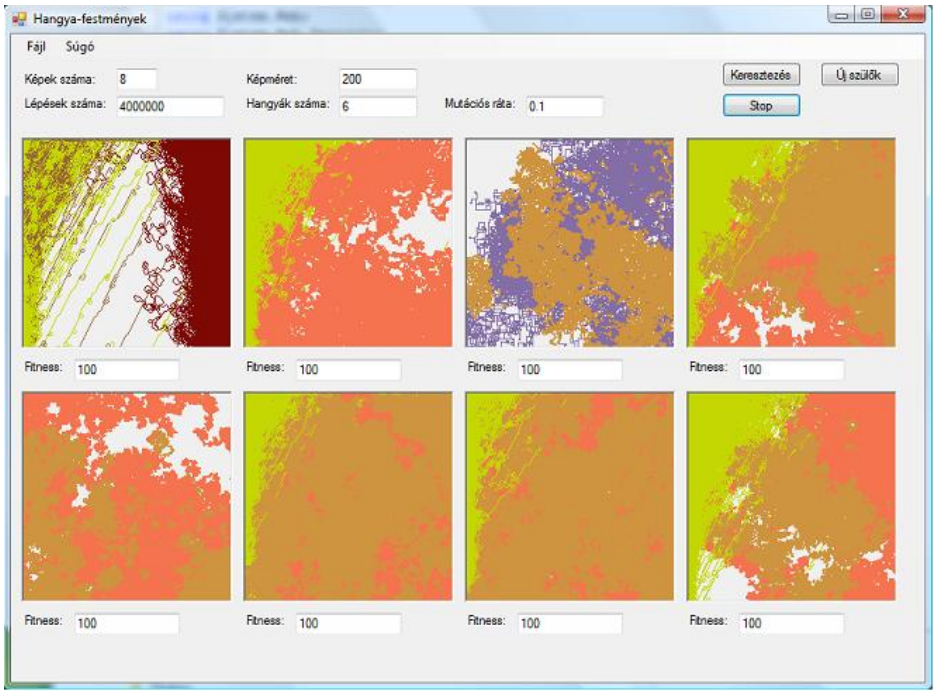
40. ábra

Így létrejött a harmadik generáció. A harmadik generációt a 41. ábrán látható módon kiértékelve, és a „Keresztezés” gombra kattintva életre hívjuk a negyedik generációt.



41. ábra

A negyedik generáción (42. ábra) már szépen látszik, hogy hogyan terelgethetjük a képek evolúcióját a nekünk tetsző irányba.



5.2. Az alkalmazás megvalósításáról

5.2.1. A hangyák megvalósításáról

A hangyák megvalósítására létrehoztam az **Ant** osztályt, amelynek objektumai a következő tulajdonságokkal rendelkeznek: a hangya saját színe, a hangya számára követendő szín, a hangya lépéstípusa (derékszögű-e vagy sem), a hangya iránypreferenciái (a lehetséges lépésirányok választásának valószínűségei), a színkövetés valószínűsége, valamint a hangya előző és aktuális pozíciója. Ez utóbbi két tulajdonságra azért volt szükségem, hogy a hangya tisztában legyen a saját helyzetével a térben, ahol mozog (esetünkben a kétdimenziós képen), és valóban külső irányítás és felügyelet nélkül, önállóan mozoghasson.

5.2.2. A genotípus megvalósításáról

Egy hangya-festmény genotípusa nem más, mint a képen mozgó hangyák listája.

5.2.3. A fenotípus megvalósításáról

A hangya-festmények fenotípusa egyszerűen megfogható, egy (BMP formátumú) kép, amely úgy jön létre, hogy nyomot hagynak rajta a hangyák, akik a saját, egyéni viselkedésük szerint haladnak, miközben egymás nyomaira, mozgására is reagálnak.

5.2.4. Az egyedfejlődés megvalósításáról

A hangya-festmények esetében az az érdekes kérdés, hogyan valósítsuk meg az alkalmazásban az egyedfejlődést, azaz a fenotípus felépítését. Én a képek egyedfejlődését lépésenként értelmeztem, vagyis úgy implementáltam, hogy kezdetben minden hangya meghatározza a rá vonatkozó véletlenszerű kezdőlépést a képen, majd időben egymással párhuzamosan haladnak lépésről-lépésre. A fenotípus így, lépésenként épül fel. Hogy mennyi lépés után mondjuk azt, hogy a fenotípus létrejött, az lehet a programozó vagy a felhasználó döntése. Rögzíthetünk egy maximális lépésszámot, vagy ha interaktív felületet biztosítunk az egyedfejlődéshez, addig folytathatjuk azt, amíg a felhasználó meg nem állítja a folyamatot.

A hangya-festmények egyedfejlődése nem determinisztikus, hiszen valahányszor elindítjuk ugyanazt a hangyacsapatot a képen, mindannyiszor más és más eredményt kapunk, csak a kép színvilága és egy-egy nagyon karakteres mozgású hangya festékpacáinak vagy csíkjainak jellege lesz többé-kevésbé változatlan.

5.2.5. A keresztezés operátorának megvalósításáról

Két hangya-festmény keresztezését úgy értelmezem, hogy a szülők hangyáinak másolatait egy listába fűzöm, és véletlenszerűen elosztom őket a két hangya-festmény utód között (úgy hogy mindkét utód legalább két hangyát kapjon). Miután elosztottam a hangyákat, el kell érnem, hogy legyenek olyan hangyák a képen, akik követnek egy másik, a képen található hangyát. Ezt úgy érem el, hogy felfűzöm egy listába a képen található hangyák saját színét, majd végigvizsgálom a kép összes hangyáját. Minden hangya 90%-os valószínűséggel fog követni egy másik, a képen található, véletlenszerűen kiválasztott színt a listából.

5.2.6. A mutáció operátorának megvalósításáról

A mutációt egy hangya-festményen úgy értelmezem, hogy vagy hozzáadok egy véletlenszerűen generált hangyát a genotípushoz, vagy kiveszek belőle egy véletlenszerűen kiválasztott hangyát, vagy módosítok egy hangyát (annak színét, lépéstípusát, iránypreferenciáit, színkövetésének valószínűségét – szintén véletlenszerűen).

6. Összefoglalás

A diplomamunkám eredeti célkitűzéseit sikerült megvalósítanom. Áttekintést adtam az evolúciós számítások elméleti hátteréről, a kiemelkedő evolúciós művészekről, megvalósítottam a genetikus algoritmus alapján egy általános, sokoldalúan felhasználható evolúciós keretrendszert, és két konkrét, saját, művészeti vonatkozású alkalmazást készítettem az általános keretrendszer használatának és az interaktív evolúciós művészeti rendszerek működésének bemutatására.

Az alkalmazások fejlesztése és a témában való tájékozódás során több olyan ötletem támadt, amelyek alapján az alkalmazásaimat továbbfejleszthetném.

Érdemes lenne kipróbálni, hogyan működik a Kreatív filterek alkalmazás, ha változó hosszúságú genotípust alkalmazunk a jelenlegi, rögzített méretű (rögzített filterszámú) genotípus helyett.

Ha a filterek belső működésének egyes paramétereit is a kromoszómába ágyaznám, magukat az alapfiltereket is tudnám fejleszteni, ezzel tovább lehetne általánosítani a kreatív filterek előállítását.

A Hangya-festmények alkalmazásban például megpróbálhatnám finomítani a hangyák viselkedését.

A hangya-festmények esetében továbbá érdemes lenne beépített rátermettségi függvényt létrehozni. Egy kép rátermettségét annak függvényében állapíthatnám meg, hogy mennyire hasonlít egy előre rögzített cél-képhez. A cél-kép tartalmazhatna színeket, formákat, vonalakat, pacákat. A rátermettségi függvény tulajdonképpen egy adott hangya-festmény távolságát mérné a rögzített célképtől [5, 8, 10].

7. Kiemelt irodalom

Bár sok helyütt nem hivatkoztam a szövegben konkrétan Peter J. Bentley és David W. Corne Creative Evolutionary Systems [2] című könyvére, mégis szeretném jelezni, hogy a 2. fejezetben az evolúciós számítások tárgyalása során számtalan gondolatot ebből a könyvből merítettem.

8. Irodalomjegyzék

[1] Laurence Ashmore: *An Investigation into Cartesian Genetic Programming Within the Field of Evolutionary Art*. 2004.
http://www.emoware.org/work/evoartdis.zip

[2] Peter J. Bentley – David W. Corne: *Creative Evolutionary Systems*. London, 2002, Academic Press.

[3] Alastair Channon: *Nature-inspired design*.
http://www.cs.bham.ac.uk/~adc/nidMaterial/lecture4.pdf

[4] Richard Dawkins: *A vak órásmester*. Budapest, 1994, Akadémiai Kiadó – Mezőgazda Kiadó.

[5] Steve DiPaola: *Evolving Creative Portrait Painter Programs using Darwinian Techniques with an Automatic Fitness Function*. 2005.
http://ivizlab.sfu.ca/media/eva05l.pdf

[6] Christian Graus: *Image Processing for Dummies with C# and GDI+*. 2002.
http://www.codeproject.com/cs/media/csharpgraphicfilters11.asp
http://www.codeproject.com/cs/media/csharpfilters.asp
http://www.codeproject.com/cs/media/edge_detection.asp
http://www.codeproject.com/cs/media/imageprocessing4.asp
http://www.codeproject.com/cs/media/displacementfilters.asp

[7] Nicolas Monmarche et al: *Interactive Evolution on Ant Paintings*. 2003.
http://www.i3s.unice.fr/tea/TEACHING/progEvo-M2/articles/AupBorMonSliVen03a.cec.pdf

[8] David Oranchak: *Evolutionary synthesis of photographic artwork using human fitness function derived from web-based social networks*. 2007.
http://oranchak.com/photosome/results/

[9] Karl Sims: *Artificial Evolution for Computer Graphics*. 1991.
http://www.genarts.com/karl/papers/siggraph91.html

[10] Jeffrey J. Ventrella: *Evolving the Mandelbrot Set to Imitate Figurative Art*. 2007.
http://www.ventrella.com/Tweaks/Portraits/EvolvingMandelbrot.pdf

[11] Mitchell Whitelaw: *Breeding Aesthetic Objects: Art and Artificial Evolution*. 1999.
http://citeseer.ist.psu.edu/cache/papers/cs/18964/http:zSzzSzwww.spin.net.auzSz~mitchellwzSzbreeding.pdf/whitelaw99breeding.pdf

[12] Wikipedia Enciklopédia, *Evolutionary computation* címszó.
http://en.wikipedia.org/wiki/Evolutionary_computation

[13] Wikipedia Enciklopédia, *Richard Dawkins* címszó.
http://en.wikipedia.org/wiki/Richard_Dawkins

[14] Wikipedia Enciklopédia, *Swarm intelligence* címszó.
http://en.wikipedia.org/wiki/Swarm_intelligence

[15] Stephen Wolfram: *A New Kind of Science*. 2002.
http://www.wolframscience.com/nksonline/toc.html

[16] Darwini költészet.
http://www.codeasart.com/poetry/darwin.html

[17] Igor Bakshee honlapja.
http://www.artlandia.com/products/artlandia/

[18] William Latham honlapja.
http://www.doc.gold.ac.uk/~mas01whl/

[19] Karl Sims, Galapagos.
http://www.genarts.com/galapagos/index.html

[20] Karl Sims, genetikus képek.
http://www.genarts.com/karl/genetic-images.html

[21] Karl Sims, virtuális teremtmények.
http://www.genarts.com/karl/evolved-virtual-creatures.html

[22] Karl Sims, Panspermia.
http://www.genarts.com/karl/panspermia.html

[23] NEvAr project.
http://eden.dei.uc.pt/~machado/NEvAr/Site.htm