

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

Evoluție fizică prin Algoritmi Genetici

propusă de

Mihaila Andrei

Sesiunea: Iulie, 2019

Coordonator științific

Asist. dr. Eugen Croitoru

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Evoluție fizică prin Algoritmi Genetici

Mihaila Andrei

Sesiunea: Iulie, 2019

Coordonator științific

Asist. dr. Eugen Croitoru

Avizat,
Îndrumător lucrare de licență,
Asist. dr. Eugen Croitoru.

Data: Semnătura:

DECLARAȚIE PRIVIND ORIGINALITATE ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licență cu titlul **Evoluție fizica prin Algoritmi Genetici** este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Data:

Semnătura:

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Evoluție fizica prin Algoritmi Genetici**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Mihaila Andrei**

Data:

Semnătura:

Cuprins

Motivație	2
Introducere	3
1 Related Work	4
1.1 Locomotie prin algoritmi evolutivi	4
1.1.1 Cell Evolution	4
1.1.2 Muskoskeletal Evolution	5
1.1.3 Joint Evolution	5
1.1.4 Concluzii	6
1.2 Algoritmi evolutivi in cercetare	7
1.2.1 Evolved Antenna	7
2 Tehnologii Folosite	8
2.1 C++	8
2.2 OpenGL	8
2.3 OpenGL Mathematics (GLM)	8
2.4 GLFW	9
2.5 Concluzii	9
3 Mediul Fizic	11
3.1 Integrarea ecuatiilor miscarii dupa dt	11
3.1.1 Moduri de integrare	12
3.2 Rigid Bodies	13
3.3 Coliziuni	13
3.3.1 Sphere Bounding	14

4	Optimizare	15
4.1	Cromozom	15
4.2	Fitness	15
4.3	Selectie	15
4.4	Mutatie	15
5	Rezultate	16
	Concluzii	17
	Bibliografie	18

Motivație

motivatie

Introducere

Prin crearea unui motor fizic 3D, avem ca scop simularea aproximata a unor actiuni fizice, ce au loc in lumea reala.

Capitolul 1

Related Work

In prezent, exista mai multi algoritmi genetici ce sunt devoltati pentru a perfectiona o actiune fizica sau pentru a optimiza probleme prin selectie naturala.

1.1 Locomotie prin algoritmi evolutivi

1.1.1 Cell Evolution

O incercare de a simula evolutia locomotiei de la un nivel celular apare in articolul **Fine grained artificial development for body-controller coevolution of soft-bodied animals**^{1 2}.

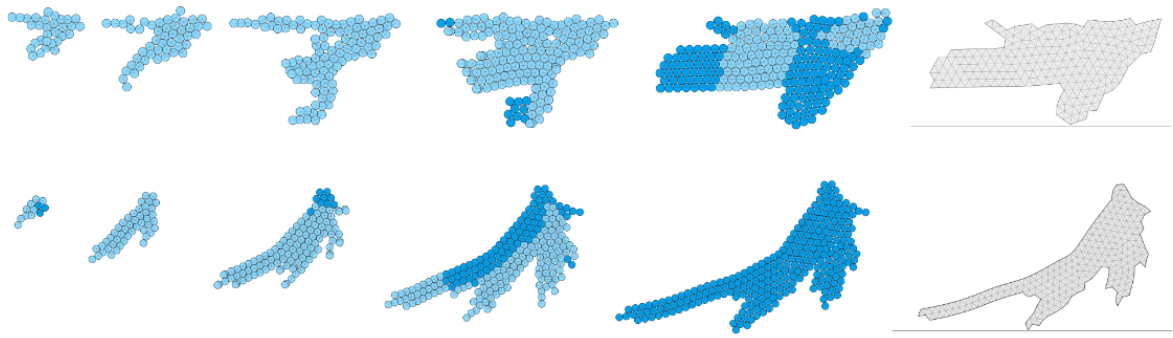
Procesul evolutionar incepe de la o singura celula ce sufera divizie celulara. O celula este reprezentata de un disc ce sufera coliziuni elastice simulate prin arcuri cu alte celule. O celula are o pozitie, viteza, si o orientare ce reprezinta directia diviziunii celulare. O celula se poate "contracta" iar compartamentul acesteia este controlat de o retea neuronală si poate fi afectata de celulele din jur.

Pentru procesul de evolutie a fost folosit **NEAT**³, *neuroevolution of augmenting topologies* Stanley and Miikkulainen, 2002. Populatia aleasa a fost $\lambda = 300$ cu genomi de lungimi arbitrare si 2000 de generatii.

¹<https://www.mitpressjournals.org/doi/pdf/10.1162/978-0-262-32621-6-ch040>

²<https://www.youtube.com/watch?v=CXTZHHQ7ZiQ>

³<http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf>



source: <https://www.mitpressjournals.org/doi/pdf/10.1162/978-0-262-32621-6-ch040>

video example:

<https://www.youtube.com/watch?v=Q6XXohgPP4A&feature=youtu.be>

Evolutia unui embrion rezultat. Fiecare celula luminoasa reprezinta o celula ce este pregatita de divizie celulara.

1.1.2 Muskoskeletal Evolution

Articolul **Flexible Muscle-Based Locomotion for Bipedal Creatures**⁴ ce a simulat locomotia prezinta un algoritm evolutiv ce foloseste CMA-ES⁵ (*covariance matrix adaptation evolution strategy*) cu o populatie $\lambda = 20$ si step-size $\sigma = 1$. Modul in care au optimizat problema a fost de a minimiza eroarea $E(K)$:

$$E(K) = E_{speed} + E_{headori} + E_{headvel} + E_{slide} + E_{effort}$$

Acestia reusesc sa optimizeze un set de muschi si incheieturi 3D pentru a obtine indivizi finali, care pot ajunge la o viteza dorita, fac fata unui teren inegal si a evenimentelor externe.

1.1.3 Joint Evolution

Un alt algoritm genetic, numit **Genetic Walkers**⁶ incearca sa simuleze o miscare in 2D. Spre deosebire de abordarea anterioara acest algoritm optimizeaza distanta dintre *head* fata de *feet*, distanta pe care fiecare individ reuseste sa o parcurga. De asemenea

⁴<https://www.cs.ubc.ca/~van/papers/2013-TOG-MuscleBasedBipeds/2013-TOG-MuscleBasedBipeds.pdf>

⁵<https://en.wikipedia.org/wiki/CMA-ES>

⁶https://rednuht.org/genetic_walkers/

fiecare individ primește 100 de puncte în fitness pentru fiecare pas corect realizat. Spre deosebire de primul articol, selecția se realizează prin o funcție f , $f(k) = \frac{1}{k+2}$, unde k este indexul unui cromozom, ordonați în ordinea fitnessului. Primii doi cromozomi aleși vor realiza un copil în generația următoare. Optimizarea se face asupra unghiului unui joint și a duratei de timp.

1.1.4 Concluzii

Ambii algoritmi lucrează cu un corp 3D fixat de la început, dar evoluează mișcarea în moduri diferite, unul lucrează low-level, cu un schelet muscular iar altul strict cu rotația unui joint. Algoritmul low level produce rezultate⁷ mult mai fidele în timp ce al doilea algoritm reușește să facă 3 pași după 500 de generații.

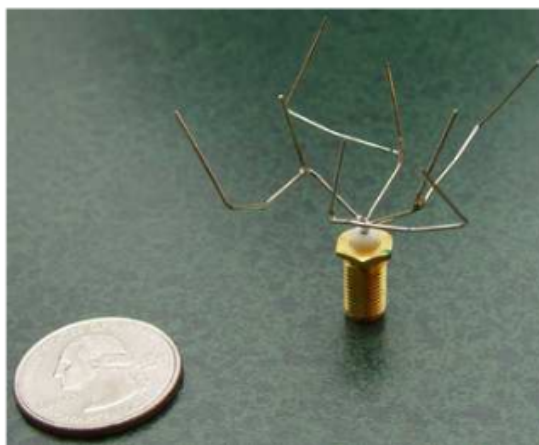
⁷<https://www.youtube.com/watch?v=pgaEE27nsQw>

1.2 Algoritmi evolutivi in cercetare

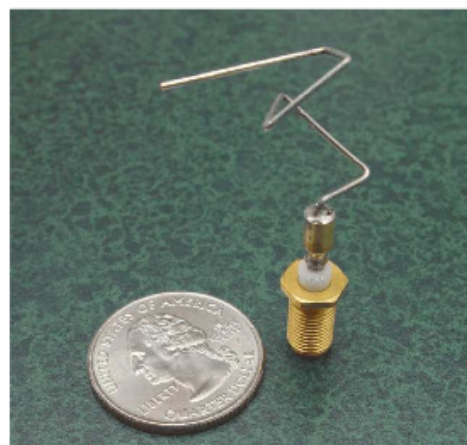
1.2.1 Evolved Antenna

In radio comunicatii, o **Antena Evoluata**⁸ reprezinta o antena proiectata complet sau partial de un algoritm genetic.

Pentru a gasi solutii pentru probleme noi, cum ar fi detectarea anumitor radiatii, au fost incercati algoritmi genetici, iar in 2006, in cadrul misiunii *Space Technology 5*⁹, a fost dezvoltata prima antena folosita in spatiu. Urmatoarele doua antene au fost proiectate de catre un algoritm genetic:



(a)



(b)

source: [https://ti.arc.nasa.gov/m/pub-archive/1244h/1244%20\(Hornby\).pdf](https://ti.arc.nasa.gov/m/pub-archive/1244h/1244%20(Hornby).pdf)

⁸https://en.wikipedia.org/wiki/Evolved_antenna

⁹[https://ti.arc.nasa.gov/m/pub-archive/1244h/1244%20\(Hornby\).pdf](https://ti.arc.nasa.gov/m/pub-archive/1244h/1244%20(Hornby).pdf)

Capitolul 2

Tehnologii Folosite

Pentru implementarea propriu-zisa a licentei, a fost utilizat un anumit set de tehnologii, care sa ajute la implementarea motorului fizic si a algoritmului genetic de la un nivel cat mai primitiv, de unde sa poate fie modelat orice tip de comportament dorit.

2.1 C++

Mediul de lucru este C++¹ sub Visual Studio 2019. C++ a fost ales in principal datorita performantei si a experientei anterioare. Prin intermediul OOP au fost create structuri pentru o implementare cat mai naturala a motorului fizic si a algoritmului genetic.

2.2 OpenGL

Reprezentarea grafica se realizeaza cu ajutorul libreriei OpenGL 3.3². Motivatia libreriei OpenGL a fost atat introducerea ei in Anul 3 cat si permitarea implementarii cat mai primitiva a motorului fizic.

2.3 OpenGL Mathematics (GLM)

Fiind un motor fizic, este necesar ca toate calculele matematice sa fie realizate cat mai corect si rapid, astfel libraria GLM³ a fost aleasa pentru viteza si compatibilitatea

¹<http://www.cplusplus.com/>

²<https://www.opengl.org/>

³<https://glm.g-truc.net/0.9.9/index.html/>

ei cu OpenGL.

Principalele structuri folosite din GLM au fost:

- `glm::vec2/glm::vec3/glm::vec4`
- `glm::quat`, (glm quaternions). Motivatia quaternionilor in favoarea matricelor de rotatie a fost performanta si evitarea problemelor unghiurilor euler, cum ar fi Gimbal Lock ⁴
- `glm::mat3/glm::mat4`

Librarii alternative au fost: Eigen ⁵ si CML ⁶. Fiecare librerie aduce un anumit punct forte, GLM a fost ales pentru compatibilitatea cu OpenGL. Rezultatele exacte sunt urmatoarele:

	laptop	laptop (SSE)	armeabi	armeabi-v7a	armeabi-v7a with neon
Eigen additions	8065	30	9944	2181	2145
Eigen multiplications	22404	86	59460	5143	5113
GLM additions	2375	76	10256	1506	1407
GLM multiplications	7337	400	59008	2189	3108
CML additions	12336	96	9587	2885	2996
CML multiplications	21603	551	58399	5306	5280

source: <https://github.com/mfoo/Math-Library-Test> ⁷

2.4 GLFW

Pentru initializarea ferestrei OpenGL a fost folosit GLFW ⁸. Aceasta librerie ofera in mod simplistic un context OpenGL si mai multe optiuni cum ar fi VSync sau DoubleBuffering. De asemenea, keyboard si mouse events sunt capturate prin GLFW.

Alternative au fost : GLUT ⁹. GLFW a fost ales in favoarea GLUT deoarece pe masina unde a fost dezvoltata aplicatia, GLUT nu reusea sa instanstieze OpenGL 3.3+.

2.5 Concluzii

C++ impreuna cu OpenGL si GLM ofera un mediu de lucru cat mai aproape de procesor si in acelasi timp un set de structuri si unelte pentru dezvoltarea naturala a

⁴https://en.wikipedia.org/wiki/Gimbal_lock/

⁵http://eigen.tuxfamily.org/index.php?title=Main_Page

⁶<https://github.com/demianmnave/CML/wiki/The-Configurable-Math-Library>

⁷<https://github.com/mfoo/Math-Library-Test>

⁸<https://www.glfw.org/>

⁹<http://freeglut.sourceforge.net/>

Motorului Fizic. Pentru algoritmului genetic, C++ contine deja tot de ce este nevoie si permite implementarea unui algoritm genetic rapid, "from scratch".

Capitolul 3

Mediul Fizic

Algoritmul genetic isi va optimiza problema in functie de mediul fizic oferit. Asta inseamna ca putem simula orice conditii fizice atat timp cat le putem reproduce. In cazul curent a fost simulat un mediu terestru cu $g = 9.8m/s$. Mediul fizic incearca sa aproximeze fenomene fizice naturale folosind formule matematice.

3.1 Integrarea ecuatiilor miscarii dupa dt

Stiind ca $F = ma$ atunci putem scoate acceleratia unui obiect in functie de masa acestuia: $a = F/m$. Daca integram dupa timp atunci $dv/dt = a$ dar nu uitam ca $a = F/m$ deci putem spune ca viteza unui obiect va fi rata de schimbare a pozitiei acestuia in functie de timp $dx/dt = v$.

Punand tot cap la cap obtinem:

$$a = F/m$$

$$dv = a * dt$$

$$dx = dv * dt$$

Deci daca vrem sa obtinem pozitia unui obiect dupa un anumit dt vom calcula in urmatorul mod:

$$a = F/m$$

$$v = v + a * dt$$

$$pos = pos + v * dt$$

Acesta este modul de integrare Semi-Implicit Euler.

3.1.1 Moduri de integrare

Exista mai multe moduri de integrare, cel prezentat mai sus este Semi-Implicit Euler.

Explicit euler integreaza pozitia inaintea vitezei si anume:

$$dx = dv * dt$$

$$a = F/m$$

$$dv = a * dt$$

Astfel apare o eroare in integrarea vitezei, iar aceasta depinde de timp. De exemplu aplicam o forta $F = 10N$ asupra unui corp de masa m , $m = 1kg$ timp de 10 secunde ($t = 10s$). Folosind formula miscarii uniforme $x = x_0 + v_0t + \frac{at^2}{2}$ atunci $x = 500$, deci conform formulei, corpul se va misca 500 de unitati. Rezultatele integratorilor difera si anume:

Eroarea integratorului Explicit Euler creste in functie de dt ales. Pentru $dt = 1.0$:

```
t = 1, position = 0
t = 2, position = 10
t = 3, position = 30
t = 4, position = 60
t = 5, position = 100
t = 6, position = 150
t = 7, position = 210
t = 8, position = 280
t = 9, position = 360
t = 10, position = 450
```

La fel si pentru Semi-Implicit Euler

```
t = 1, position = 10
t = 2, position = 30
t = 3, position = 60
t = 4, position = 100
t = 5, position = 150
t = 6, position = 210
t = 7, position = 280
t = 8, position = 360
t = 9, position = 450
t = 10, position = 550
```

Se observa ca ambele integratoarea au acelasi grad de eroare, dar Semi-Implicit euler are o eroare in fata, comparat cu rezultatul cautat $x = 500$ in timp ce Explicit-Euler este in urma. Eroare ambelor integratoare scade in functie de dt , si anume pentru $dt = 0.5$ si $dt = 0.016$ (timpul unui frame pentru un monitor cu o rata de reimprospatare de 60Hz):

Explicit Euler, $dt=0.5$ & $dt = 0.016$

t = 5.5, position = 137.5	t = 9.856000000000007, position = 484.9151999999955
t = 6, position = 165	t = 9.872000000000007, position = 486.49215999999547
t = 6.5, position = 195	t = 9.888000000000007, position = 488.07167999999547
t = 7, position = 227.5	t = 9.904000000000007, position = 489.65375999999543
t = 7.5, position = 262.5	t = 9.920000000000007, position = 491.2383999999954
t = 8, position = 300	t = 9.936000000000007, position = 492.8255999999954
t = 8.5, position = 340	t = 9.952000000000007, position = 494.41535999999536
t = 9, position = 382.5	t = 9.968000000000007, position = 496.00767999999533
t = 9.5, position = 427.5	t = 9.984000000000007, position = 497.6025599999953
t = 10, position = 475	t = 10.000000000000007, position = 499.19999999999527

Semi-Implicit Euler, $dt=0.5$ & $dt = 0.016$

t = 5.5, position = 165	t = 9.856000000000007, position = 486.49215999999547
t = 6, position = 195	t = 9.872000000000007, position = 488.07167999999547
t = 6.5, position = 227.5	t = 9.888000000000007, position = 489.65375999999543
t = 7, position = 262.5	t = 9.904000000000007, position = 491.2383999999954
t = 7.5, position = 300	t = 9.920000000000007, position = 492.8255999999954
t = 8, position = 340	t = 9.936000000000007, position = 494.41535999999536
t = 8.5, position = 382.5	t = 9.952000000000007, position = 496.00767999999533
t = 9, position = 427.5	t = 9.968000000000007, position = 497.6025599999953
t = 9.5, position = 475	t = 9.984000000000007, position = 499.19999999999527
t = 10, position = 525	t = 10.000000000000007, position = 500.79999999999524

Pentru $dt = 0.016$ integratorul Semi-Implicit Euler obtine o eroare $\epsilon = 0.79$ iar integratorul Explicit Euler $\epsilon = 0.81$, deci Semi-Implicit Euler este mai bun cu $\epsilon = 0.02$ fara nici un cost suplimentar din punct de vedere al timpului de calcul sau a memoriei folosite.

3.2 Rigid Bodies

Un corp rigid este un corp ce nu sufera deformatii in niciul fel. In cadrul algoritmului genetic si a mediului fizic, orice obiect este considerat un corp rigid cu un anumit factor de elasticitate $\sigma \in [0, 0.7]$. Avand in considerare scopul curent al algoritmului genetic, au fost alese cuburi cu $l = 1$.

3.3 Coliziuni

Din moment ce calcularea fitnessului are loc in mediul fizic, este nevoie ca simularea sa se efectueze cat mai repede, astfel se va recurge la verificarea coliziunii in 2 faze.

3.3.1 Sphere Bounding

Primul pas in detectarea unei coliziuni va fi compararea intre 2 sfere S_1, S_2 ce se realizeaza in mod rapid:

if $d(C_{S_1}, C_{S_2}) > \max(r_{S_1}, r_{S_2})$ further checks... else false

Fiecare obiect va fi invelit intr-o sfera de raza $r = l$, unde $l = d_{\max}((0, 0), v), v \in \text{Vertexes}$.

Capitolul 4

Optimizare

intro

4.1 Cromozom

4.2 Fitness

section

4.3 Selectie

section

4.4 Mutatie

section

Capitolul 5

Rezultate

Concluzii

conclusions

Bibliografie

- Kenneth O. Stanley, Risto Miikkulainen. **Evolving Neural Networks through Augmenting Topologies**, <http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf>
- Michał Joachimczak, Reiji Suzuki, Takaya Arita. **Fine grained artificial development for body-controller coevolution of soft-bodied animats**, <https://www.mitpressjournals.org/doi/pdf/10.1162/978-0-262-32621-6-ch040>
- Thomas Geijtenbeek, Michiel van de Panne, A. Frank van der Stappen. **Flexible Muscle-Based Locomotion for Bipedal Creatures** <https://www.cs.ubc.ca/~van/papers/2013-TOG-MuscleBasedBipeds/2013-TOG-MuscleBasedBipeds.pdf>
- Rafael Matsunaga. **Genetic Walkers**, https://rednuht.org/genetic_walkers/
- **Evolved Antenna** https://en.wikipedia.org/wiki/Evolved_antenna
- Gregory S. Hornby and Al Globus, Derek S. Linden, Jason D. Lohn. **Automated Antenna Design with Evolutionary Algorithms** [https://ti.arc.nasa.gov/m/pub-archive/1244h/1244%20\(Hornby\).pdf](https://ti.arc.nasa.gov/m/pub-archive/1244h/1244%20(Hornby).pdf)
- Martin Foot. **Math-Library-Test**, <https://github.com/mfoo/Math-Library-Test>
- Glenn Fiedler. **Integration Basics** https://web.archive.org/web/20190405191806/https://gafferongames.com/post/integration_basics/
- Glenn Fiedler. **Physics in 3D** https://web.archive.org/web/20181107181511/https://gafferongames.com/post/physics_in_3d/
- Glenn Fiedler. **Fix Your Timestep!** https://web.archive.org/web/20190403012130/https://gafferongames.com/post/fix_your_timestep/