Proceedings of the International Multiconference on Computer Science and Information Technology pp. 111–121
ISSN 1896-7094
© 2006 PIPS

EMOT – Evolutionary Approach to 3D Computer Animation

Halina Kwasnicka, Piotr Wozniak[1]

Institute of Applied Informatics, Wroclaw University of Technology

Abstract. Key-framing and Inverse Kinematics are popular animation methods, but new approaches are still developed. We propose a new evolutionary method of animation creation – system EMOT (Evolutionary MOTion). It allows for automation of motion of animated characters. It uses new evolutionary approach – Gene Expression Programming (GEP). Characters are controlled by computer programs, an animator has to provide the way of motion evaluation. GEP works with a randomly selected initial population, uses directed but random selection. Experiments show that the proposed method can develop robust controllers.

1 Introduction

The paper concerns a problem of automation of computer animation of characters. Animation is understood as specification of motion in such a way that a given entity makes some actions and expresses thoughts and emotion, important for the related story. In the paper we focus on entities (characters) consisting of a set of stiff blocks with a number of joints.

Recently, majority of animation is created using Key Framing technique. This technique requires from animators high qualifications: an animator has to take care about general nature of an animated character, and simultaneously, about all visualization details and physical credibility. It causes that at least partially automation of this process is very desirable. We observe intensive study in this area and a number of different approaches have been used for the automation.

Usually, an evolutionary paradigm is used rather for developing geometrical and physical models than animation itself (e.g. Karl Sims study [1]). Our approach shows that the new evolutionary technique – Gene Expression Programming (GEP) [2] could be useful for automation of generation of moves that meets animator's expectations.

Proposed solution – system EMOT (Evolutionary MOTion) – allows to automate motion of an animated character. Characters are controlled by computer programs (controllers); an animator has to provide the way of motion evaluation (fitness function) but GEP searches acceptable controllers. GEP, as all evolutionary methods, works with a randomly selected initial population, uses directed but random selection. Naturally, it does not guarantee the proper solutions but experiments show that the proposed method can develop robust controllers. The method is rather general and it can be applied to different animation tasks.

The paper is structured as follows. Next section presents problems connected with animation, some useful approaches and techniques are shortly described. Section 3 introduces into GEP, presents the coding schema and genetic operators. Section 4 is devoted developed computer system EMOT (Evolution of MOTion). The results of experiments with two animated characters are described in section fifth. Short Summary ends the paper.

2 3D animation – a short overview

An animation is not only simulation of physics, not only simulation of an animal behavior, but it is also a kind of art [3]. Often real biological and physical features are violated when animators tray to tell a story – it is caused by personality of a character and narration. Using evolutionary approach for evolution of behavior of an animated character allows to optimize its behavior taking into account the physical features, but – what about expressivity of characters? How we can incorporate the expressivity in fitness function?

A number of different objects can move in animation, but characters are only these one, which express thoughts, emotion and action (important for the related story). Animators, designing motion, must care for individuality and emotion of an animated character. Not only living organisms can be used as characters in animation, e.g., shoal of fish is alive but it is perceived more as mechanical nature force than as a character.

Special effects, as rain, water waves, flying plane, etc. are other category of animation. There are a number of techniques achieving commercial success – such effects can be reached by physical simulation. In the paper we focus on specific kind of character animation: animation of a set of stiff blocks with a number of joints. Additional assumption is that our character (blocks with joints) forms an acyclic graph.

2.1 Basic rules of animation

Most of rules of 'perfect' creation of animation were developed on the beginning of twenties century, especially in Disney Studios. Frank Thomas and Ollie Johnston, in their book entitled *The Illusion of Life: Disney Animation* proposed twelve basic principles of animations:

Timing – a tempo of moving object gives a meaning of the object and prompts why this object is moving, e.g., a quick wink denotes that the object is worried or stimulated but slow – that our character is tired and asleep. Additionally, velocity of character creates a subconsciously imaginary about mass and strength of a character, engaged in a given scene.

Slow in and out – moving between some extremely positions cannot starts and stops violently, e.g., a limb achieving some position slows down, or gradually accelerate starting moving.

Arcs – in real world almost each move of character (or its part) goes along an arc. Usually biological joints are rotary.

Anticipation – action of an animated character usually goes in three phases: a preparation phase, movement, and a finish phase. Anticipation concerns the preparation. A jumper usually sags his knees before jumping what is connected with physics. Looking for some object usually comes before reaching it.

Exaggeration – exaggeration of moving, emotion or even constitution of the character body causes that animation focuses attention of spectators. Often such an exaggeration helps in story understanding.

Squash and stretch (deformations) – this rule describes a way in which animated characters react to movement or when it is subjected to some forces. Falling down ball is flatting when it touches playing field (squash) but after rebound this ball lengthens in direction of movement (stretch). These deformations allow spectators for imaginaries about physical features of the animated object.

Secondary action – living organisms have natural tendency to make simultaneous a number of actions. In general, a basic activity occurs (required for basic task, e.g., our character goes), but less important, secondary activities go with accompany the former (e.g., the moving character looks around). Thanks to this, the character becomes more natural and interesting.

Follow through and overlapping action – sequence rule (follow through) acts as anticipation but it concerns the third phase of movement – finish. Usually it lies on moving something and next – comeback to the neutral placement. Overlapping action means that one action comes from other, previous one.

Straight-ahead action and pose-to-pose action – two alternative ways of animation creations. Forward animation is made by creation of starting frame and next, adding to it the subsequent frames up to the final one (when the action is stopped). The second way is developing frames in a number of key moments and next, building the rest, intermediate frames.

Staging – clear and easy to understand presentation of the idea of animation. Action should be presented in the manner easy to understand.

Appeal – it means attractiveness, all what we like to see. It covers character creation as well as a scenario of story.

Solid drawing – it means ability of an animator for creation of realistic, three dimensional objects (including characters).

Presented above basic rules of animation concern different aspects of animation (see Table 1). A part of them concern artistry – we do not try to automate artistic elements. We are interested only the rules connected with physical aspects of animation.

2.2 Methods overview

Continuous simplification of animators' work is a main feature of newly developed animation methods. Successive techniques apart from 'technical' details of animation allowing animators to focus on behavior and nature of animated characters instead of solid geometry [4].

Rules of animation	Aspect
Squash and stretch (deforming) Follow through and overlapping action Slow in and out Arcs	Physical
Secondary action Anticipation Timing	Physical and artistic
Straight ahead action and pose-to-pose action Staging Exaggeration Appeal	Artistic
Solid drawing	Not concerns computer animation

Table 1. Rules of animation and their aspects

Key framing. Key framing is the oldest animation method – the main animator creates figures of characters in some, characteristic positions, and the remainder frames are drawn by less skillful animators [5]. They make a kind of 'manual interpolation' [5]. Later, the 2D animation was made using computer for interpolation basic, characteristic positions [6]. Further development allows to transform key framing into 3D animation [7, 8].

IK – Inverse Kinematics. Inverse Kinematics [9] is an other well-known technique. It allows for finding end positions of an animated element, e.g., hand, foot, using some coordinate system as a referred set. Forces and moments that cause movement are omitted [9]. A number of methods were developed to transform position and velocity from the character set (joints angles) to the Cartesian system (e.g., the end of arm) [10, 11].

Physical simulation: dynamics. Systems based only on kinematics are intuitive but resulting animation often is unrealistic due to gravitation or inertia. Naturalism of moves can be achieved by taking into account forces and momentum. Physical simulation methods can be divided into two groups: without and with constraints. The first approach is more popular; mainly it lies on parameters adjusting. The second approach requires presentation of features in the form of constraints [12–15].

Behavioral techniques. Behavioral systems can be seen as particle systems in which a number of moving objects are controlled by the same set of relatively simple rules revealing a complex and complicated move ([16–18]).

Optimization methods. Recently, the optimization methods of computer animation are developed. They can be divided into the three groups:
- energy minimization [20],
- spacetime constraints [21, 22],
- evolutionary algorithms, as Genetic Programming [19, 1, 5], evolution of computer programs able to control motion).

3 Evolutionary Computation and GEP

Evolutionary **C**omputation (EC) covers a number of methods, e.g., **G**enetic **A**lgorithms, **E**volutionary **S**trategies, **G**enetic **P**rogramming, that have good searching skill. They are nature inspired and often used as optimization methods – searching through the solution space [3, 23, 24].

The most popular evolutionary approaches are Genetic Algorithms (GA) [23] and Genetic Programming (GP) [24, 25]. In GA individuals (potential solutions) are coded as linear chains, usually but not always, as a bit strings with constant length. GP uses the same paradigm as GA but it evolves different structures – instead of linear chains (of bits or real numbers) potential solutions are coded into trees, each individual has different size. It requires special genetic operators (mutation and crossover).

Gene Expression Programming (GEP) [2] is a kind of GP but it uses linear structures as individuals: parse trees are coded into linear strings with constant length. An individual – a genotype represented as a linear structure – is easy decoded into a phenotype, called Expression Tree (ET). It allows for using simple, traditional genetic operators. GEP offers similar functionality as GP but it is easier for evolution. In GEP the process goes according the following pseudocode:

```
Pseudocode 1: Evolution in GEP

Generate of an initial population  (random selected chromosomes)
Express of chromosomes   (decoding genotypes into phenotypes)
Repeat for each individual of population
  Fitness calculation  (it requires calculation of expressions or running the programs)
  Checking the stop conditions
  If stop condition occurs do
     Return the best solution
     STOP
  else do
     Store the best solution
     Reproduction process  (creation of the next generation of population)
          Select individuals according assumed selection method
          Perform mutation and crossover operators
  end if
end repeat
```

Evolved individuals are mutated and crossed according to assumed probabilities. Genetic operators have to assure a proper structure of each gene (chromosome) [2]. *Mutation* can occur in any place in the chromosome, but in head, each symbol can be changed into any symbol, and in tail – only into terminal symbol. GEP offers three kinds of *crossover*: *1-point crossover*, *2-points crossover* and *Gene crossover.* In GEP, besides mutation and crossover, three kinds of transposition are defined: *Insertion sequence* (IS), *Root insertion sequence* (RIS) and *Transposition of genes* (GT) [2].

4 System EMOT – Evolutionary MOTion

In spite of researchers effort, the majority of professional animators still use key-framing. The technique proposed in this paper allows for automation of characters motion, giving a user possibilities of specification what motion should be generated. The technique can be used for different animation tasks. The base rule of our technique is: the system generates some motion and a user (man) assesses the quality of the motion. Therefore we applied in the developed system a relatively new evolutionary technique known as Gene Expression Programming (GEP). In our approach:

- Characters are controlled by control programs, GEP is searching proper controllers for the assumed motion of character.
- An animator does not create controllers; he provides the evaluation of a motion generated by an evaluated controller. The evaluation seems to be easier than creation of animation, especially for amateurs.

Because of stochastic nature of GEP we are not sure that the developed method finds the solution of our problem, but we can run the program many times with different initial parameters till we obtain satisfying solution.

4.1 EMOT – an overview of the system

Figure 1 presents a general scheme of EMOT. From an animator point of view, subsystems GEP, scene service, physical simulation, and simulator itself can be treated as black boxes.

The system always has to know geometry of a scene and dynamic of each character involved in the simulation – it is a precondition of EMOT. The model of character contains geometry, mass, and inertias of particular joints, description of character's possibilities – i.e., ranges of joints' angles, forces and momentum of muscles, location of sensors of strength pressure. These values are constant during the animation process. Each sequence of movement has to be evaluated. It is done using a fitness function defined by an animator. One, the best control program (i.e., the best individual) is the output of the system.

The system of physical simulation takes care about natural appearance of animation. In the proposed system we use software package Open Dynamics Engine (ODE) developed by R. Smith. A character is represented as a set of rigid blocks connected by joints, the rest of scene, e.g., walls, floors, are treated as geometrical elements with infinity mass (they are stationary but they interact with animated characters). Joints moves are limited, each joint is propelled by an angular motor that gives demanded relative angular velocities all elements joined with the considered joint. The velocity is achieved by using a force (each angular motor has defined power).

A controller of character is calculated (developed) in each step of physical simulation and it is a list of demanded angular velocities for all joints. From the physics point of view the resulting move looks quite natural because of ODE detects collisions and reactive forces. The pseudocode of simulation of moves generated by character's controller is following:

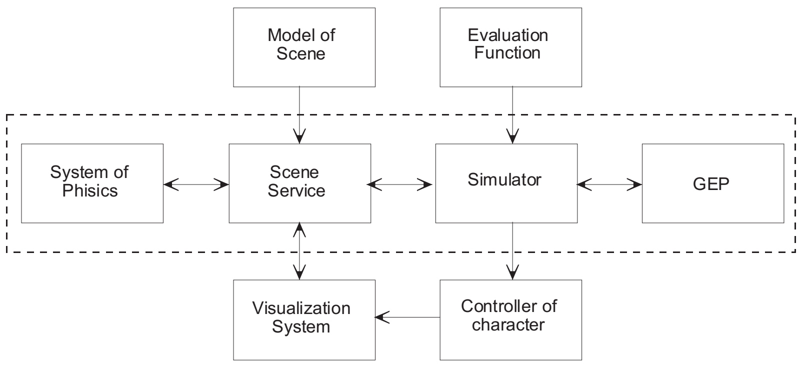

Fig. 1. A diagram of EMOT system

```
Pseudocode 2:  Simulation of moves

t=0
while t < time limit do
   make controller-program (calculate demanded angular velocities)
   simulate dynamics with moving ahead delta_t
   t <- t+delat_t
end while
```

4.2 GEP in designing of control engines

System EMOT uses standard schema of GEP [2]. Each degree of freedom of each joint has to be controlled, so we must generate a control program for each of them. Therefore a number of genes has to be equal a number of degrees of freedom of all joints of the animated character. We assume that each gene codes a control program for one degree of freedom of one character's joint; it means that each gene is an independent program (see Figure 2).

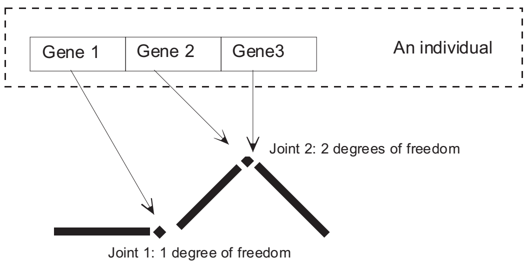

Fig. 2. A coding schema

A set of functions and a set of terminal symbols are presumed. Initial experiments allow to choose adequate sets of functions and terminals. A set of functions consists of the basic mathematical operators and *IfNeg* predefined function: $\{+, -, *, /, IfNeg\}$, where $'/'$ means 'safety division', and the *IfNeg* function (with three arguments) returns:
– a value of the second argument (subtree) if the value of the first argument is negative,
– a value of the third argument (subtree) if the value of the first argument is positive or *zero*.

A set of terminals consists of internal variables (position, velocity, angles of joints, external forces, time, etc.) – these values are taken from the simulation system of dynamics when the program-controller is running, and from random real values.

Each of mini-programs is calculated (ran) separately, but usually they use state variables of the whole character model, therefore subprograms have tendency to generate mutually dependent moves always when it is more efficient than a series of separated moves. For each programs a fitness value is calculated on the basis of statistics returned by the dynamic simulation system. The fitness value can be divided into two parts: satisfying the main task and a reward for style. For example, if we try to evolve a character that should move to a given point X, a distance between the character and the point X at the end of simulation is a measure of the task fulfillment. EMOT is able to produce different ways of moving because some characters can e.g., crawl instead of jump, so we should take into account a style of moving by enlarging or diminishing the fitness value as a reward for style. A number of categories can be distinguished in the style evaluation, namely:

– safety – a penalty for collisions with other objects
– time – a penalty for inefficient behavior (e.g., selection of longer road or quick moving after a long inactivity)
– attaining equilibrium – a reward for staying in the neutral position at the end of movement (e.g., character should stand, not lie)
– other – depending on the kind of generated movements.

It is worth mentioning that the initial, *raw fitness* (the initial one) should be transformed to some *normalized fitness*. In some cases the raw and the *standardized fitness* can be the same. *Standardized fitness* means that better individual has lower fitness value (we try to minimize it). The ideal individual should have standardized fitness equal to zero. *Adjusted fitness* (f_{adj}) is given by following equation:

$$f_{adj} = \frac{1}{1 + f_{std}} \tag{1}$$

where f_{std} is a standardized fitness.

Normalized fitness is an adjusted fitness divided by the adjusted fitness of a whole population:

$$f_{norm} = \frac{f_{adj}}{\sum_{i=1}^{M} f_{adj_i}} \tag{2}$$

where M denotes a size of the population.

Normalized fitness is used for the selection process. EMOT uses *n-tournament selection* [23, 25].

5 Examples and experiments

The two tasks have been chosen for experiments. The first is called *Jumper* – it is used for study the suitability and efficiency of the proposed in EMOT approach. The second example – a kind of spider (we call it *Madzia*) is a bit more complicated task. Obtained results are very interesting.

5.1 Jumper

Example *Jumper* is taken due to its clarity. The task of EMOT was to learn *Jumper* relocates into a defined place. A model of *Jumper* is fully three-dimensional: *Jumper* consists of five elements (a head, a two-parts leg, a bearing, and a base) joined by four joints (see Figure 3). Each joint has one degree of freedom and is controlled by developed program. The *Jumper*'s head moves in right and left, both parts of the leg move in vertical, and the bearing can turn in the plane of base. Such *Jumper* can move in any direction. We defined a task for *Jumper* as: moving to the defined point and standing in such a way that a center of base is exactly in the assumed point.

A number of fitness functions were tested. The best found fitness function consists of the following constraints:
1. The main aim: a distance between a center of *Jumper*'s base and desired (assumed) position at the end of simulation
2. Style: a weighted sum of values ascribed for particular subtasks:
a) a penalty for each collision of a head with any scene element
b) a penalty for too long idleness
c) a reward (exactly – a lack of penalty) for the neutral position at the end of simulation.
The used fitness function is presented in the form of Pseudocode 3.

Some values are normalized into a range $[0; 1]$. We do not want to punish strongly individuals, who perform two subtasks relatively good, but the third one is evaluated as vary badly, therefore we cut fitness on the defined level. Violating the safety constraint (stroking the head) causes the worst evaluation. Safety moves were evaluated taking into account a tempo, the final position, and staying without moving after achieving the goal.

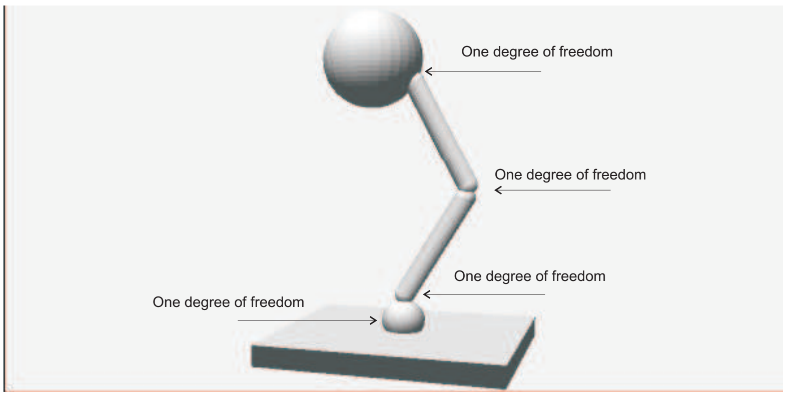

Fig. 3. Jumper

```
Pseudocode 3:The pseudocode and a fitness function used for Jumper evaluation

normalized distance = distance / initial distance
phase = sqrt(recent generation / max number of generation)
style = 0
if safety rules are violated then
  style = style +1
else
  style = style + timing             (do not be inactive)
style = style +sum of angles         (initially sum of angles is equal to zero)
style = style +sum of velocities     (stop after achieving the goal)
evaluation = normalized distance + phase * style
GEPFLOAT CEvolution:: GEPFitness(int generation)
{
GEPFLOAT dist, phase, style;
Dist = (lpPhysics -> GetDistance(ID_1,ID_2))/dOriginalDistance;
Phase = sqrt((double)generation/MAX_GENERATIONS)
Style = 0.0;
if (lpPhysics->iGetCollisions(lpPhysics->BodyByID(5), lpPhysics->GeomByID(10000)) ||
 lpPhysics->iGetCollisions(lpPhysics->BodyByID(5), lpPhysics->GeomByID(10)))
 style += 1.0;
else
{
 style += 0.33*clamp(0.0, 1.0, lpPhysics->GetTiming());
 style += 0.33*clamp(0.0, 1.0, lpPhysics->GetAngleSum());
 style += 0.33*clamp(0.0, 1.0, lpPhysics->GetVelocitySum());
}
return dist+style*phase;
}
```

Taking into account that evolution with simulation consumes a lot of time, in the presented below experiment we ran EMOT ten times for 50 generations with the same parameters. All parameters used in the experiments, as well as sets of functions and terminals are collected in Table 2.

All ten runs ended with success, it means, EMOT found adequate controllers, but different controllers were produced in particular runs. An exemplar animation of the best individual is shown in Figure 4. Some statistics (as an averages of ten runs) are shown in Figure 5. The common feature of all found solutions (controllers) is physical correctness, simplicity and surprisingly natural moves of *Jumper*.

A length of a head of genes influences the size of Expression Trees. Experiments were made for the length changing from three to 50. The best individuals were produced for the length equal to 10 and 15 (0.96 and 0.95 average fitness, respectively) while for a head length equal to three and 50 the average fitness is equal to 0.32 and 0.34. Too short head makes impossible the evolution of proper controllers. On the other hand, longer head increases a size of a search space, and the task becomes more difficult.

Working with GEP we found a population size as an important issue. We tested populations of 10, 30, 50, 70, 100, 200, 300, 500, 1000 i 2000 individuals. Populations of 50 or less individuals are not able to evolve acceptable solutions. Meaningful change was observed for a population size enlarged up to 70 individuals. Evolution of 100 or more individuals does not generate essentially better controllers. It seems that 100 individuals are a good balance between quality of solutions and efficiency of GEP.

5.2 The Spider

In this experiment we used a simple model of spider. Our spider, named *Madzia* (in Polish it is a girl name), consists of trunk, four legs, each leg contains two segments, joined by 12 joints, each joint has one

Table 2. Parameter values used in experiments

Aim	Controller that relocates the Jumper from one to the other, defined point
A set of terminal symbols	*t* (time) *k0, k1, k2, k3* (angles of joints) *Oxxx.px, Oxxx.py, Oxxx.pz* (coordinates of object *xxx*) Vxxx.x, Vxxx.y, Vxxx.z (velocity of object xxx) Random constants
A set of functions	$+, -, *, /, IfNeg$
Parameters of GEP	Size of population 100 Number of generations 50 Length of a head of gene 10 Tournament size 6 Mutation 0.01 Transposition IS 0.1 Length of IS 1, 2, 3, 4 Transposition RIS 0.1 Length of RIS 1, 2, 3, 4 Gene transposition 0.05 1-point crossover 0.2 2-points crossover 0.5 Gene crossover 0.1 Mutation of constant 0.05 Tuning of constant 0.9 A range of constants -6.0 - 6.0
Other parameters	Maximal time of simulation 20.0 Simulation step 0.01 A number of slots of animation 2000

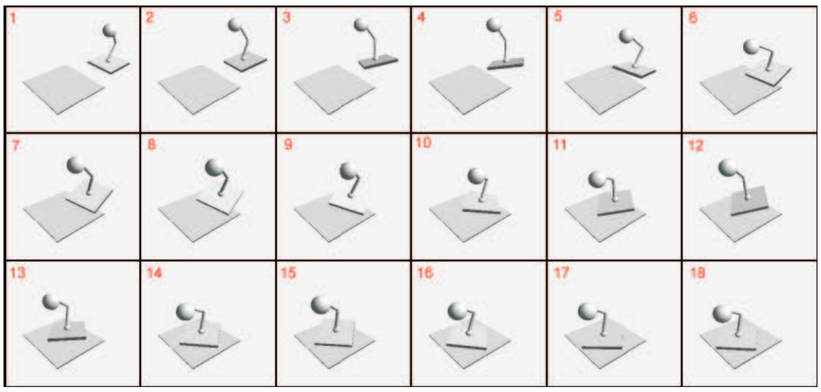

Fig. 4. The best jumper, 50th generation of third run. Jumper access the desired point precisely after three jumps and remains stable.

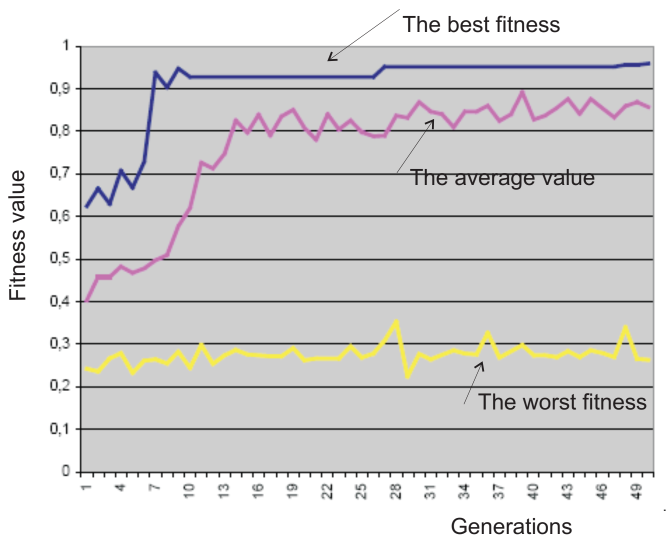

Fig. 5. Fitness values in generations (average from 10 runs); fitness value is normalized, the best possible one is equal to zero, the worst – 1.

degree of freedom. The main task of *Madzia* was to go strictly over the box. Used fitness function consists of two parts: distance measure and safety rule (*Madzia*'s trunk cannot to touch the floor). *Madzia*'s model contains more degrees of freedom than *Jumper*'s model, what causes that a search space is more complex and the evolution process (together with simulation) consumes more time. First experiments shown that *Madzia* tried to achieve a box jumping not walking. To force walk we have set a *Madzia*'s mass and strength of her muscles very carefully to make *Madzia* not able to jump.

En exemplary (interesting) result is shown in Figure 6. *Madzia* amazed us, she attends equilibrium relatively good, but instead of moving directly over the box she approaches the box enough to reach it by one of her front legs. Next, she pushed the box directly under her body (trunk). Because she shifted the box a bit further, she moved back a little and achieved the aim. This behavior is completely compatible with the assumed aim and used fitness function, but unexpected for us. We can say that it was caused by our mistake – we set a mass of box very small comparing to mass of *Madzia*, and she discovered it.

In our opinion it is a very natural result generated by artificial evolution. Artificial evolution surprises us discovering solution that people can find with troubles.

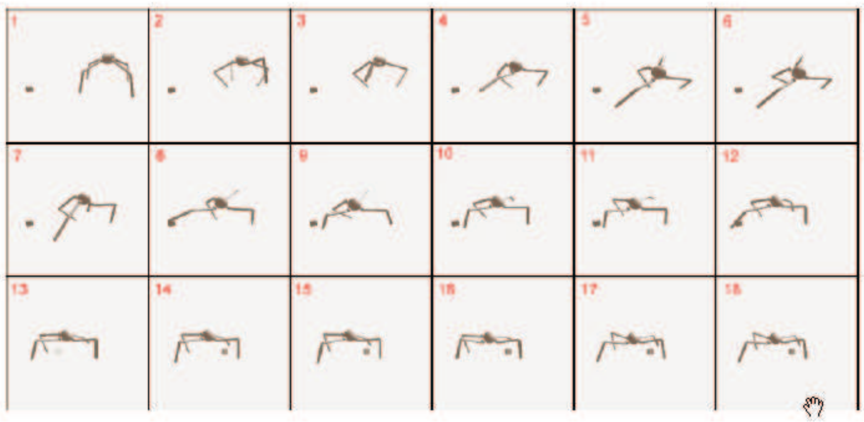

Fig. 6. Madzia tries to stand up over the box

6 Summary

Recently a number of techniques of animation automation are developed, they support animators work. Using dynamic simulation for proper presentation of physical properties of objects and characters gives good results but it is very difficult. Majority of approaches require manually written parts of control programs.

In the paper we propose to use controllers that control characters' moves by assuring proper values of angles velocities of joints. Characters are built from rigid blocks connected by joints. Programs-controllers are developed not manually but by mean of Evolutionary Computation, namely – Gene Expression Programming (GEP). Characters controlled by programs evolved using GEP are simulated in dynamic environment. The controllers-programs are evaluated on the basis of simulation results, using manually written fitness function.

Our method generates credible moves from the physical point of view, allows for specification of goals of animation, and resulting animation is quite natural. The technique is not connected with only one type of animation, it depends on imagination and skillful of an author of fitness function. The results are pleasant for observers, moves are fluent, natural and they are effects of artificial evolution.

As it is seen in *Madzia* example, the result can surprise us. In our opinion, we have shown that GEP can be useful approach in the supporting of automation of animation. Presented examples have relatively small number of degrees of freedom. Possibilities of scaling the method up to high number of degrees of freedom seems to be a good point for starting future work.

The EMOT system reviles also disadvantages. Generated controllers are not flexible, a program generated for one move is sensible for the initial condition and cannot be used for other moves. Evolution of controllers stressing their skill not a defined task seems to be a proper direction of future work. The second direction is enlarging the method about 'animation news', for example, objects deformation.

References

1. Sims K., *Evolving Virtual Creatures,* SIGGRAPH'94 Proceedings.
2. Ferreira C., *Gene Expression Programming: A New Adaptive Algorithm for Solving Problems,* Complex Systems vol. **13,** 2001.
3. Ventrella J. J., *Disney Meets Darwin. The Evolution of Funny Animated Figures,* Computer Animation '95 Proceedings, IEEE, 1995. `http://www.ventrella.com/Alife/Disney/disney_0.html`
4. Terzopoulos D., *Artificial Life for Computer Graphics,* Communications of the ACM vol. **42.**
5. Gritz L. , Hahn J. K., *Genetic Programming for Articulated Figure Motion,* Journal of Visualization and Computer Animation vol. **6**.
6. Burtnyk N., Wein M., *Computer Generated Key Frame Animation,* Journal of the SMPTE, 1980.
7. Leffler S. J., Reeves W. T., Ostby E. F., *The MENV Modelling and Animation Environment,* Journal of Visualization and Computer Animation vol. **1.**
8. Hesler A., *Genetic Algorithms and Character Animation,* Comp 291 Final Paper, 1998.
9. Thalmann N. M., Thalmann D., *Computer Animation,* MIRALab 1997.
10. Maciejewski A. A., *Dealing with the Ill-Conditioned Equations of Motion for Articulated Figures,* IEEE Computer Graphics and Applications vol. **10.**
11. Baerlocher P., *Inverse Kinematics Techniques for the Interactive Posture Control of Atriculated Figures,* PhD. thesis, Ecole Polytechnique Federale de Lausanne, 2001.
12. Barzel R., Barr A. H., *A modeling system based on dynamic constrains,* SIGGRAPH'88 Proceedings vol. **22.**
13. Baraff D., *Non-penetrating Rigid Body Simulation,* Eurographics'93 State of the Art Reports.
14. Baraff D., *Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies,* SIGGRAPH'89 Proceedings vol. **23.**
15. Hahn J. K., *Realistic Animation of Rigid Bodies,* Computer Graphics vol. **22.**
16. Reynolds C., *Computer Animation with Scripts and Actors,* vol. **16,** 1982.
17. Reynolds C., *Flocks, herds and schools: A distributed behavioral model,* SIGGRAPH'97 Proceedings vol. 21.
18. Reeves W. T., *Particle Systems: A Technique for Modeling a Class of Fuzzy Objects,* ACM Trans. Graphics vol. 2.
19. Reynolds C. W., *An evolved, vision-based behavioral model of coordinated group motion, From Animals to Animats (conf. proceedings),* MIT Press.
20. Witkins A., Fleischer K., Barr A., *Energy constraints on parametrized models,* SIGGRAPH'87 Proceedings vol. 21.
21. Witkins A., Kass M., *Spacetime Constraints,* SIGGRAPH'88 Proceedings vol. 22.
22. Ngo J. T., Marks J., *Spacetime Constraints Revisited*
23. Goldberg D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley 1989.

24. Koza J. R., *Genetic Programming: On the Programming of Computers By Means of Natural Selection,* MIT Press 1992.
25. Kwasnicka H., *Evolutionary Computation in artificial Intelligence (in Polish),* Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 1999.