9.1.1 Mathematics: Computer Graphics and Geometry
Department of Algebra, Geometry and Didactics of Mathematics
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Animation of Flocks and Herds

(Master Thesis)

Bc. Jana Dadová

Supervisor: doc. RNDr. Andrej Ferko, PhD.

Bratislava, 2010

I hereby declare I wrote this thesis by myself, only with the help of referenced literature, under the careful supervision of my thesis supervisor.

. . . . . .

I would like to thank my supervisor doc. RNDr. Andrej Ferko, PhD. for his great help, advices and supervising. I would like to thank also Mgr. Martin Ilčík and Norbert Ketterl with their help in crowd simulation and Mgr. David Běhal and Tomáš Dado with their help in particles.

Abstract

Nowadays, simulations of crowds are important to bring more natural look to the virtual world. Moreover, large crowds can be used not only to fill space, but to create visually interesting shapes with perceptive meaning. Positioning of people to create shapes is well know from live performances, or shows, where dancers are positioned to create formations. This principle is transferred from people to the animals or even things in various motion pictures as another channel of metaphor.

Our approach in distribution of individuals is different from regular ones, where participant are equally distributed. We use particle systems for representation of individuals and apply repulsive forces for distribution. This brings a semi-regular approach, where participants are more naturally, less equally distributed with shape-preserving. Moreover we bring more automation to the process, where chief animators, or designers define only shapes. Afterward we compute distribution of individuals and create movement between formations.

Shape-preserving is a feature which depends on human perception. Therefore we verified our results with the group of children that are an important target group of most motion picture movies and also with the group of adults. We tested meaning of the formations for humans in various ages and backgrounds.

KEYWORDS: Crowd simulation, Animation, Automatic distribution, Formation control.

Abstrakt

V dnešnej dobe sú simulácie davov dôležitou súčasťou prirodzene vyzerajúcich virtuálnych svetov. Okrem toho, veľké davy nachádzajú využitie nielen vo vypĺňaní priestoru, ale aj vo vytváraní zaujímavých tvarov. Polohovanie ľudí je známy problém v tanečných vystúpeniach, kde sú tanečníci ukladaní na pozície, čím vytvárajú formáciu. Tento princíp môžme preniesť z ľudí na zvieratá, prípadne dokonca na veci, čo bolo využité aj v rôznych animovaných filmoch ako ďalšia úroveň metafory.

Náš prístup v rozložení jednotlivcov je odlišný od pravidelných, kde sú účastníci rovnomerne rozložení. Používame časticové systémy na reprezentáciu davu a aplikujeme odpudivé sily pre správne rozloženie. Týmto postupom vytvoríme takmer pravidelnú metódu, kde sú účastníci prirodzenejšie rozložení so zachovaním tvaru. Okrem toho prinášame viac automatizácie pre animátorov a dizajnérov. Potom vypočítame rozloženie jednotlivcov a vytvoríme animácie medzi formáciami.

Zachovávanie tvaru je vlastnosť, ktorá závisí na ľudskom vnímaní. Preto sme výsledky našej práce overili s deťmi, ktoré sú hlavnou cieľovou skupinou animvaných filmov a taktiež so skupinou dospelých ľudí. Overovali sme význam formácií pre ľudí rôznych vekových kategórií ako aj rôzneho zázemia.

KĽÚČOVÉ SLOVÁ: Simulácia davu, Animácia, Automatické rozloženie, Kontrola formácie

Abbreviations

LOD - Level Of Detail

CDT - Constrained Delaunay Triangulation

FPS - Frames Per Second

HVS - Human Visual System

XML - eXtensive Markup Language

W3C - World Wide Web

HTML - HyberText Markup Language

IDE - Integrated development environment

OpenGL - Open Graphics Library

JOGL - Java Open GL

GUI - Graphic User Interface

COLLADA - COLLAborative Design Activity

SGML - Standard Generalized Markup Language

List of Figures

"Animation offers a medium of story telling and visual entertainment which can bring pleasure and information to people of all ages everywhere in the world."
Walt Disney

Contents

1

Introduction

In our approach we would like to consider either flocks or herds of various kinds of animals in more complex and general way. We define crowd of participants, where participants can be animals as well as human beings.

Both crowd behavior and crowd animation are quite newish research topics in computer science. Within the last three decades a wide range of scopes are focused on. Crowd animation is used in entertainment, commercials, and simulations as well. The primary objective is to achieve a high-level simulation of socially unconnected collectives and connected groups both in behavioral terms and in a visual way.

Dealing with crowd behavior focuses on much more topics than considering only a singular scientific discipline. Therefore the research field of crowds should be discussed in an interdisciplinary way. However most state-of-the-art computer models dealing with crowd animation cover either only graphical, physical or sociological foci. A plausible visual environment have to extend the level of pure geometrical models. Realistic simulation of living entities (e.g. plants, animal, human) does not only belong to the transformation of physical laws into virtual worlds but first and foremost on the interface between computer science, biology and psychology [KD09].

Goal of our work is to bring more automatic control over the group and create visually attractive shapes[1] with right distribution of participants. Positions of individuals in

[1] shapes - iconic representaion of objects

a group create a formation[2], which could be assigned automatically using repulsive forces. These forces can used also for animation between formations.

Iconic representaion and metaphorical meaning of shapes highly depend on personal perception of spectators. Therefore people with various backgrounds, education and ages should be tested to proof our hypothesis, that formations have some meaning. Moreover the meaning is similar to more people, when standard shapes are used.

In the next chapter 2 different methods for crowd simulation and animation are discussed and also state-of-the-art approaches are shown. Chapter 3 is more specific in animation and movement between formations, where other methods are shown. We also implemented our solution and specification to it is discussed in chapter 4, where implementation itself is shown in chapter 5. Results to the work are shown in 6, where are also renders included. In this chapter we discuss also answers of questionnaire and performance statistics and we successfully confirm our hypothesis. In the last chapter 7, conclusion we sum up everything.

[2]formation - exactly defined positions of individuals. When looked from further away usually create visually attractive shape or hidden meaning.

2

Related Work

There have been many papers published on Crowd animation and simulation topics. In this work we would like to mention only few, important ones from our perspective. The areas that discuss problems in crowd animation can be divided into following categories, that can overlap [TM07]:

- Heterogeneous crowd
- Animation of a crowd
- Behavior of a crowd
- Including a crowd in an environment
- Rendering of a crowd

The following subsections present a short overview of the special cases mentioned above.

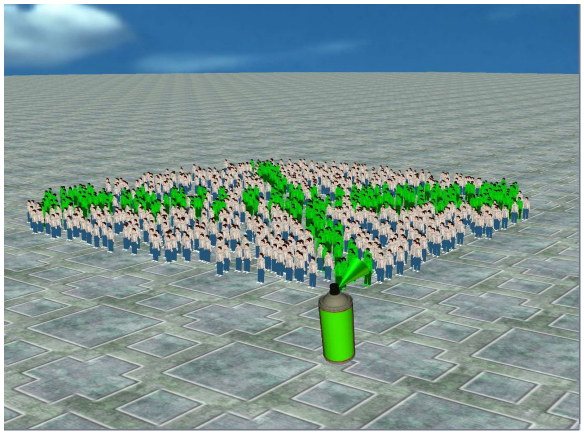

Figure 2.1: Color brush with uniform operator [UHT04].

2.1 Heterogeneous crowd

Firstly, when the crowd is going to be simulated it has to be created. This is not a trivial step, because crowd should consist of heterogeneous individuals[1]. A crowd, that is generated only from the copies of the same individual, is not looking natural especially considering crowd of humans. Usually individuals do not look, move, or behave the same way. It is costly to model and animate each individual separately, therefore some automatic tools need to be developed. One of the possibilities of creating scenes involving thousands of animated individuals is *Crowdbrush* [UHT04]. In this interactive tool authors implemented a brush option. With this brush user can add, remove or modify the individuals and easily create the crowd. The area to be modified can be selected as a point or area. Results from the Crowdbrush are shown in Fig. 2.1. When considering dancers, or flock of fish the more participants look similar, the better, because they should create impression of a consistent group. We would like to use technique with more similarities.

An interesting research area is crowd of human population. For creating such crowd, the individuals are defined as *somatotypes*[2]. To define such criteria, the anthropome-

[1]Heterogeneous crowd means group of characters in various colors, sizes, movements, poses that preserve common features of all individuals in a crowd. Examples could be flock of birds, where every participant is a bird, but each bird could be of another kind.

[2]Somatotype is term for classification of human body shape. There are three endomorphic,

try[3] is used. Considered factors are endomorphic[4], mesomorphic[5], and ectomorphic[6]. Endomorphic factor is calculated by measurement of skinfolds, the mesomorphic factor by body dimensions and ectomorphic by weight and height. This techniques for defining human body are automated, or semi-automated, but in both databases of models need to be used. These models are then automatically processed and final result is given. This automation reduces possibility of change by an artist, but parameters can be set to bring different result. When creating a crowd the final output models have to be suitable for animation [TM07], either with predefined skeletons or with ability to interact with obstacles and other participants.

When the distributed models are created, they need to be placed in a space. This step is called *Point Placement*. Positions of individuals can be defined as formations [TYK+09], as in dance performance where choreographer specifies positions of artists. Another examples are individuals placed in some object or on some object [Bez01], like mass scenes where members of a group are distributed on the terrain or in the hall.

2.2 Animation

This section emphasizes animation of a group eliminating discussions about animation of individuals such as skeletal animation, poses and movements to be more specific here.

After a crowd is generated and individuals are placed in positions, the next step is animation of this crowd. In this step we consider entities as particle systems only with physical forces, without any social forces involved. Particle systems model crowds as whole with a global view on movement [Ree83]. Therefore path planning and collision

mesomorphic, and ectomorphic. Each of these types can gain values in range 1-7 to define human body shape.

[3]Anthropometry is a theory where parts of human body are measured. It is part of Anthropology, theory of human fysiology.

[4]Endomorphic somatotype defines fat in the body.

[5]Mesomorphic somatotype defines medium bones.

[6]Ectomorphic one defines muscles in the body.

avoidance need to be considered. Collision detection should be computed not only with obstacles, but also with another individuals in a crowd.

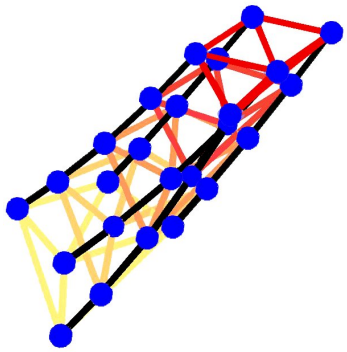

Figure 2.2: A group motion graph [LCF05].
A group motion graph constructed from the clip. Vertices (blue dots) are connected in two sets of edges, formation edges (colored edges in the figure) and motion edges (black edges).

Kwon et al. [KLLT08] proposed a solution for a *motion graph* in order to interactively manipulate motion of the crowd. The graph is created in a way, that each vertex represents the location of an individual at a sampled frame, edges represent moving trajectories and neighborhood formations as shown in Fig. 2.2. Moreover, once a graph is created a user can manipulate vertices and by them also the whole group. Therefore graphs can be stitched together and longer motion can be achieved. Furthermore, using this motion graphs also efficient collision detection can be calculated similarly to approach proposed in Sung et al. [SKG05]. The key positions of individuals and poses are defined and then the optimal path is found. At first, probabilistic roadmaps are created for navigation and path planner. Afterward the randomized search algorithm is created for refinement. Motion graph is good for deterministic methods for animation, but less for stochastic. Therefore it is not very suitable for our approach.

Path planning for groups is the main topic of a recent paper by Takahashi et al. [TYK+09]. Here the path is calculated for the individuals in a crowd from two given keyframe formations, such as dancers or artists. This work is explained in more detail in the next sections 3.

In many situations simple animation could be insufficient, considering the needs of film industry. A specified position introducing the crowd or specified path of one individual in the crowd involves *constrained animation*. It allows definition of constraints such as exact position of agents in the time, or shapes of group layout [AMC03].

Figure 2.3: Motion capture example from the movie The Lord Of The Rings. *Motion capturing of Gollum [Rem03].*

For creating set of movements, that are later associated to the participants, content creation tools like Maya[7] or Blender[8] are used. Defining motion manually for each individual is inefficient in the large crowd, therefore also here some automations are investigated. Other way of efficiently creating animation is Motion Capture[9], where movement can be later associated to various participants. Example of use of motion capture technique in the movie The Lord of the Rings for character named *Gollum* in Fig. 2.3 and recently also in the Cameron's [10] movie Avatar 2.4. This technique was used in The Lord of the The Rings movie [Rem03]. More automatic method is *procedural animation* widely used for the motion of particles. This motion is based

[7] `http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112\&id=13577897` - Official Webpage of Maya 3D modeling software by Autodesk, 2009 (accessed January 14th, 2010)

[8] `http://www.blender.org/` - Official Webpage of Blender modelling software by Blender Foundation, 2010 (accessed January 14th, 2010)

[9] Motion Capture is a technique, where motion is extracted from optical markers attached to the body of actor and then mapped to the virtual body

[10] `http://www.imdb.com/name/nm0000116/` (accessed 4.2. 2010) - James Cameron is Hollywood director and also a director of movie Avatar, that has premier in 2009 and helped to develop special technique for facial motion capture.

Figure 2.4: Motion capture example from movie Avatar.
Motion capturing of main characters in the movie Avatar, where not only movement of a body was captured but also facial expressions.
Source: `http://www.thinkhero.com/2010/01/07/james-cameron-confirms-avatar-sequel/`

on physical or mathematical description of the movement [Ree83] and better fit our needs.

2.3 Behavior

The problem with path planning can be solved by defining of participant's behavior. Individual behaviors include being confused (specially for panic situations) or being leader (for flocking behavior). Behavioral animation considers members of the crowd as agents with defined rules and this allows creating of complex behaviors [Rey87]. Depending on which behavioral model is chosen, the path can be planned and the efficiency of algorithms can be achieved.

Behavior is a result from social and psychological interaction between entities. These behavioral models depend on the specified task. In many cases the works are multi-

disciplinary, where social behavioral of animals is studied [CKFL05]. For this method the cooperation with ethologists was necessary. In some works the behavior of animal species is studied and the crowd simulation is created afterward. Another example are simulations of evacuation drills, where human behavior in stressful situations has to be considered [Nyg07]. In this approaches cooperation with psychologists is necessary.

The main focus within the crowd behavior covers the intelligence of the crowd behavior model. It is used in situations with low demand for graphical quality but strong emphasizes on sociological and safety aspects. Realism of behavior is widely used in simulations of real-life scenarios in virtual-reality applications. To simulate the behavior of large groups in emergency or panic situations it is less important having good looking characters than checking the safety and robustness of the architectural planning in case of fire or evacuation processes.

The most important work in this field is [Rey87]. Here members of the group are considered as individuals, autonomous agents with specified goal. The boids (bird-like agents) have three local rules of behavior:

- Flocking - boids tend to stay near each other, to stay in the flock.
- Collision avoidance - boids tend to have a distance to other boids to avoid collisions.
- Velocity matching - boids tend to match the velocity with other boids in the flock in order to stay together.

This model can be used not only for birds, but also for other animal species.

Furthermore, derived from this approach, not only this three features are considered, but the *change of leadership* is added [HB06]. Usually the flock is following the leader, not really considering the other members separately. In this approach a boid can separate from the crowd and violate the rule of flocking. Others follow to satisfy the rule of flocking in the smaller group that separates. Behavioral animation does not preserve shape and does not bring position control over the crowd. Therefore these rules does not fit our approach very well.

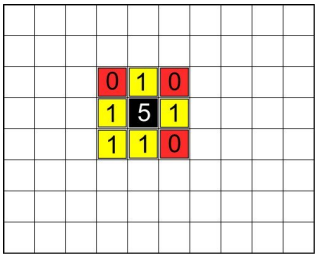

Figure 2.5: Example of discretized environment.
Environment for cellular automaton in Conway's Game of Life. 8 - neighborhood is used, where we are finding next state for the black cell. Yellow cells are occupied and each brings 1 to the sum. Behavior of black cell for number 5 is defined as death, because it is overcrowded around. Red cells brings 0 to the sum, because are not occupied, but are in the neighborhood.

Cellular automaton is another approach that derives from procedural modeling. In this approach instead of physical dynamics, discretized environments are used as is in Fig. 2.5. Usually there is a grid of cells, that define space and each cell has set of neighbors (four or eight for planar space), states and behavioral rules, that depend on states of neighbors. State changes according to these rules and initial state is set in an random way to the cells. Popular examples for cellular automaton are the Nagel-Schreckenberg model [NS92] for traffic simulation, simulation of behavior in panic situation by Lightfoot-Milne [LM03] or the *Game of Life* [Gar70] from John Conway. Cellular automation can create plausible distribution, but does not offer very good control over the animation, therefore it not very useful for our approach.

The behavior of animal and human crowds slightly differs. To simulate a flock of birds it is important to know which species should be simulated. In the case of simulating migratory birds there is important to recognize the V-formation of the birds within a large compound. This kind of formation reduces the energy every bird has to spend during their long-term flight [HB06]. Also the leader of the formation will be exchanged circular to gain an almost equivalent energy distribution within the whole flock. The realistic behavior of large human crowds remains a challenge. A human spectator can easily recognize slightly unnatural movements of simulated humanoid characters. Both verbal and non-verbal communication are key features of

the interaction with other humans.

On the other hand, Hodgins and Brogan proposed method where they consider not only the needs of an agent in the group but also problems of individual, such as stability, individual velocity and position in the group [HB94]. This approach can be used for human crowds as well as for animals in a flock. These systems are called *Systems with significant dynamics*.

2.4 Environment

There could be various problems with environment, especially obstacles, or paths that are suitable for crowd movement. Environments considered include buildings for emergency procedure, or buildings for special purposes, like museums [TM97]. Another examples are pedestrian simulations where cars usually stay at the road and people on the pavement [TCP06]. This kind of the environment is called *Populated Environment* [TM07] and should allow cooperation with the crowd to avoid collision with obstacles.

In other way we can consider environment as utilizing dimensions of space, where participants can move, 2D - everything is set in plain, 2,5D - everything lies on the ground, that could be with height, 3D - space in the sky or water in the sea. For environmental planning there are growing expectations for added realism. Especially for architectural planning and designs of cultural heritage there is a huge demand for getting some kind of l̈ife. Within urban environments like architectural pre-visualization or virtual museums it looks much more natural if crowd of humans are present, than looking on a deserted atmosphere. Especially in VR (Virtual Reality) applications there is an increasing need for realistic immersion. In case of VR rehabilitation projects (e.g. crowd phobia, performance anxiety) it is very important to achieve a plausible immersive environment where the patient feels some kind of reality. The movie industry also focuses on using high-quality graphics for producing large-scale crowds for reduced costs and flexibility both in animated films (e.g. Madagascar [Ker05]) and real films (e.g. The Lord of The Rings trilogy [Rem03]).

2.5 Rendering

Figure 2.6: Movie Madagascar.
In the figure a group of lemurs is shown, where they are positioned and rotated in a way, that they naturally look at their king Julian. Belongs to the DreamWorks Animation SKG [Ker05]

Both in computer games and movies the realism of good looking crowds is much more important than a physical plausible appearance. Rendering of a crowd is a challenge, especially for large groups of participants. Therefore various researchers specialize in this topic. In the real-time rendering the main problems are efficient computation time, data storing and fast access to the stored data. Thus, in state-of-the-art games detailed characters do not appear in large crowds, but mostly only few at a time, because with an increasing number the frame-rate in real-time systems drops fast. Moreover participants can be in such application represented via billboards. Unlike realtime applications, in the field of non-realtime applications animation of the whole crowd is scripted precisely in advantage. Offline rendering usually has to fulfill special needs of director and artists. Therefore, animation should be flexible and easy to manipulate.

Because of both - limited computational power and the complexity of the overall mapping from real life to their virtual abstraction - there is no all-in-one device for every purpose possible. Rachel McDonnel depicts [McD06] that the game industry is the

primary market for realtime crowd animation. Therefore there is a huge demand for simplification techniques both in motion and geometry. Any modern graphics card is limited on the number of polygons which can be processed per second. Therefore the modern approach is to pre-compute animation and save it to database, or even to texture, where coordinates can stand for the key-frames [TM07]. For better performance several levels-of-detail (LOD)[11] are chosen. This improvement lowers the costs for computation of skeletal animation.

Non-real-time applications have to show visually attractive scenes usually with a lot more members of the group. These members usually perform specified actions and they are in the scene at specified times. The main problems are with the motion itself, because these individuals in a crowd should perform similar action, but not look as copies. *High-level behavior* approach [Ker05] was introduced to deal with such animation, where behavior defines motion in more natural way with preserving another part of a group. This behavior is applied on the extra layer of animation on top of existing animated cycles. Example of the scene where lemurs should turn their heads to the King Julian can be seen in Fig. 2.6.

On the other hand, also human crowds are used in the film industry. The most famous from recent years are crowds in The Lord of the Rings movie. The mass scenes were here created with Massive software[12]. Motion capture was used here for the animation key frames. Then *action tree* was created by the choreographers. To each individual is then assigned set of rules depending on the fight strategy (like offensive or strike). Although in non-real-time rendering there are no such problems as memory, or computational costs that makes simulation harder for the real-time rendering, the visual part is important.

[11] Levels of detail (LOD) is technique for creating several meshes for same model with various count of triangles. When rendering one of those meshes is used depending on the distance between viewer and the object of interest. LOD technique is also used for various individual motions, where characters next to the viewer have detailed motion and further away skeleton is simplified [OCD+02], [OCV+02].

[12] `http://www.massivesoftware.com/` - Massive is the premier simulation and visualization solution system for generating and visualizing realistic crowd behaviors and autonomous agent driven animation.

3

Path Planning

In this section we would like to bring more detailed knowledge to the related work of path planning for group of living beings. Path planning will be further discussed in a question of usage for our research and proposed solution. Therefore we would like to firstly bring related work and then find solution for our problem in the chapter 4.

Path planning is a topic that includes generation of layout for further group movement, interpolation between keyframes and finding the path avoiding the obstacles. We refer in this paper to the formation as some shape created with the positioning of individuals of a group in some keyframes. Formations can be defined as a shape the particles create or by the rules they follow.

Moreover, we require formations to be in meaningful shapes such as artistic layouts in mass performances. In computer graphics designers often want to provide simulation or approximation of real processes. These processes can be divided into:

- **Shepherding**

  Motion of one group is here controlled by another group [LRMA05]. Shepherds are agents that influence the movement of a flock, where flock is a group of agents that move in coordinated manner and respond to the external factors as is shown in Fig. 3.1a. Goal of the shepherds is to steer the flock toward specified position. Formations and path planning here is affected by collisions in the flock and heading toward the position. No special visual formations are

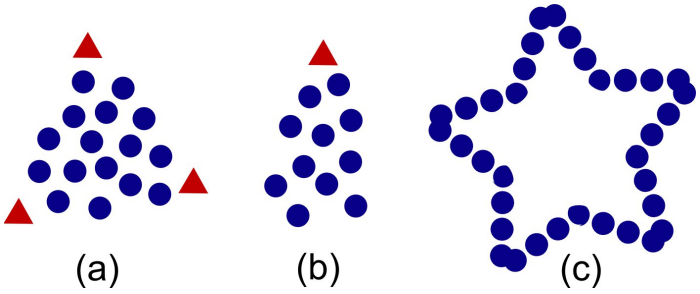

Figure 3.1: Difference in a motion.
Between the motion based on shepherding (a), following the leader (b) and creation of formation with defined positions.(c). (a) Group of shepherds is in red, group of other animals is in blue. (b) Leader is in red and followers are in blue. (c) Each participant is equal to other and together they create visually attractive formation.

created.

- **Following the leader**
  Motion of the whole group depends on a few individuals that are leaders as is shown in Fig. 3.1b. Leaders are informed agents and affect whole group [CKFL05]. Leadership can be changed during the movement. Animation with the leadership is based on the observation of the groups of animals. Some species were chosen and similarities were found. From our, similarly to the previous approach, group moves together to gather the defined goal. When considering visual formation, the V-formation is a well known example of this approach.

- **Creating a formation**
  Previous approaches provide solutions for group motion based on agents, but also can be emulated with particle systems, where the description of forces between individuals allows control of crowd movement. Each individual is considered as agent with defined behavior. Another method is considering a crowd as whole, where movement is based on exact mathematical description, or on physical forces. Positions of individual are considered with respect to a whole group. Particle systems approach is a method that allows such a definition. Takahashi et al. [TYK+09] bring control of movement over whole group and the particles can create visually attractive formations as is shown in Fig. 3.1c.

The difference between agent-based systems and particle systems is in a away to manipulate with a crowd. While in agent-based approach individual simulation is considered, in particle system the whole group is manipulated at once. Individual approach allows defining more specific behavior of individual, in particle systems the simulation is usually controlled with physical forces. Reynolds describes the *seek* and *pursuit* behaviors [Rey99]. Although these are behavioral methods the goal is the same as in a method proposed by Takahashi et al. [TYK+09]. The individual tries to reach defined static position (seek) or moving target (pursuit). Distinct from these behaviors, the *path following* sweeps object along path, usually defined by spline curve. This method uses less agent based approach and can be used with particles, where each particle has a pre-defined path.

3.1 Constrained animation

Constrained animation defines restrictions on the animation of a group. This can be either a specific layout with obstacles, specific position of the group or movement of individual. Obstacles are a problem in almost any realistic group animation or simulation because they are common in real world. On the other hand, positioning members of the group or special definition for individual movement are more common for artistic reasons in the film industry.

The simplest way of creating constrained animation is to manually modify the path if it is possible. This is not a trivial problem and in methods that do not deal with collisions directly it is left for the future work or is adjusted manually. Another methods [AMC03] deal with constrained animation directly. Key-frame based methods allow full control over the animation, that artists like to have, but it is time-consuming and the defined paths for some individuals do not allow them to cooperate with the rest of crowd. Cooperation or influence is a natural ability of humans and animals, therefore it is often required and some randomness in the movement looks less mechanical. More natural look brings also smoothness. One of the approaches that bring smoother movement are splines. In a crowd simulation they are used to calculate the smooth moving path of participants. To calculate splines, the keypoints as control

points are created. These keypoints are extracted from the keyframes. Then smooth interpolation is obtained with splines. Splines can be successfully used also with the constraints [RBB97].

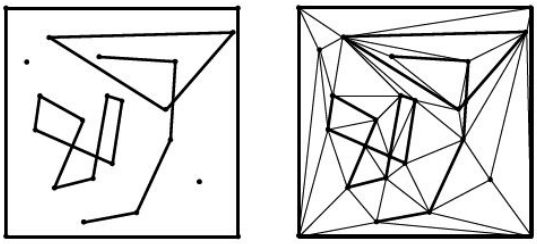

Figure 3.2: Constrained layout.
Constraints in layout (left) Delaunay triangulation with preserving the constraints (right) [KBT03]

The problem of dynamic constrained path planning is also discussed in [KBT03]. Here the problem is solved with creating a constrained Delaunay triangulation (CDT), which is similar to Delaunay triangulation, but respects constraints as is shown in Fig. 3.2. Once the CDT graph is created, the two points are defined (p_1, p_2) and the shortest path between them is found. These points are located in triangles of the triangulation. Then over the adjacency graph the path of triangles is created. Let P be a polygon defined with these path. Finally the shortest path from p_1 to p_2 is inside the polygon P. The shortest path is found that respects constraints. Because the CDT gets dynamically changed, this approach can be used also for moving constraints and also can help with shape preserving, but is rather complex.

3.2 Particle systems

Fluids, water, smoke and other physically well described objects in computer graphics are hard to model with polygonal surfaces or curved surfaces. Particle systems firstly introduced by Reeves [Ree83] are an easy way of describing such phenomena. Particle systems describe an object as a cloud of primitive particles that defines volume instead of surface. Particles are very simple, primitive objects, that interact with environment

and among each other. These simple objects can be changed in higher level for meshes with more complicated geometry (such as animals, or humans) or even with individual animation. A particle system has advantage in the computational time, because number of particles can vary, animation is physically based. With lowering the number of particles, level of detail is easily computed with lowering the number of particles. The movement and description of particles is usually mathematical, therefore it is widely used for physical processes, as motion of fluids [Ree83] or even for motion of a crowd [HLTC03]. Physically based description of movement in particle systems is well discussed by Andrew Witkin [Wit97] and workshop[1].

In the elementary idea of particle systems for fluid dynamics, the particles have a property of lifetime. For each frame the following is calculated [Ree83]:

- new particles are created and assigned individual features and initial settings
- any particles that have existed past their prescribed lifetime are extinguished
- other particles are moved and transformed according to their attributes
- image is rendered

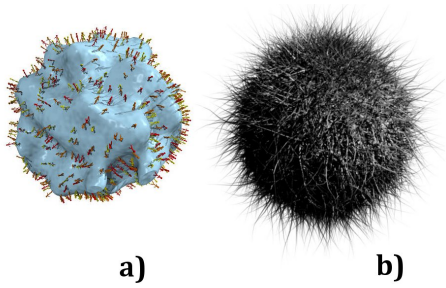

Figure 3.3: Distribution of particles.
Generation of the layout and distribution of particles for the crowd simulation (a) and for the furry object [Bez01] (b). In both cases distribution of particles is approximately uniform [Blender Association]

[1] `http://www.pixar.com/companyinfo/research/pbm2001/` (accessed 4.2. 2010) - Witkin's course notes from the Siggraph 2001. Where physically based particle systems are well explained

Particle systems used in crowd simulation and for fuzzy objects, differ in creating and destroying the particles. While in traditional approach the particles are created and destroyed [Ree83], in the crowd simulation the number of objects does not change [TYK+09]. Other features of particle systems can be used also when considering crowd simulation, such as movement, transformation of features and rendering. In fluid dynamics particles are usually partially transparent, simple and can blend together to create more realistic motion, but in crowd simulation particles are usually solid, more complex in geometry and moreover have to calculate collision instead of blending. Similar conditions of collision are applied for furry objects, where are particles also widely used [YSWW06]. There are some similarities in the distribution of particles in the generation step between crowd simulation and furry objects as it can be seen in Fig. 3.3. Another enhancement is particle hierarchy introduced Reeves [Ree83] the particles can be themselves particle systems. Transformations applied on the parent particle then are applied on the whole system in this particle. This way the movement of the whole group can be either divided into independent subgroups or the whole group can be easily transformed.

Before 2006 almost all simulations of crowds were agent-based. It has the advantage that also in real world each individual is independent. Therefore, these agents can respond to situations based on the rules. The disadvantage is that it is hard to develop realistic motion and manipulate the crowd as whole. Approach discussed in [TCP06] defines motion synthesis model for large crowds without agent-based dynamics. Here the global planning provides also avoiding obstacles and other people by minimizing the energy. Particle systems are a base of this approach.

In approach introduced by Bouvier et al. [BCN97] it is shown that particle systems are powerful and can be used in various situations. Particles can interact among themselves, with other systems and also with obstacles. Firstly, these systems were designed for animation of fluid flow, but with some enhancement they can be used also for crowd simulation. A human flow simulation is a complex problem, because human behavior needs to be provided for better realism. Therefore some sort of decision making and reaction to the environment need to be possible. These behaviors have to have mathematical description that can be implemented to the particle system.

In [BCN97] such description is provided for reflex reactions and decisions. Reflex reactions concern immediate change in movement considering avoiding obstacles and other particles. Decisions are a more complex problem, but a person with decision charge can be influenced in the same way as a particle with an electric charge. And that is a physical process that has mathematical description.

3.3 Motion Graphs

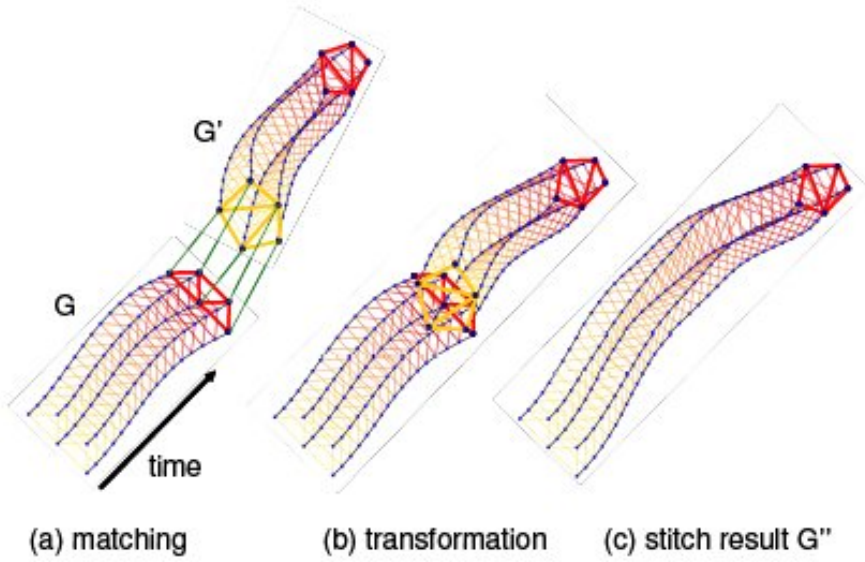

Figure 3.4: Group motion stitching steps.
(a) finding corresponding points (b) aligning of the graphs (c) smooth blending between the graphs [KLLT08].

As it was said in related work (chapter 2), for the group animation motion graph can be used. In a method by Kwon et al. [KLLT08] the motion graph is enhanced to provide multiple motion clips in the same timeline. Either larger groups can be divided into smaller or groups can be combined together and form larger formation. In graph the edges are divided into formation and motion edges. Formation edge represents neighborhood relationship between vertices and can be useful to preserve adjacent relationship. However, these relations are usually not well defined. Delaunay triangulation can produce reasonable connectivity between individuals in keyframes.

For mentioned paper this approach is sufficient with the optional user adjustment. With this triangulation local enhancement is automatically calculated to preserve the distortion of local features and shape. Motion graph and formation edges provide the advanced formation planning.

Formation edges define movement in the space and motion edges define movement in the time. For spatial features the whole group needs to be considered to compute movement, but for temporal features only moving paths of individuals are considered. Spatial movement is calculated for the keyframe. Both these edges together define spatiotemporal movement of a participant. The main problem is to stitch motion groups together. To do this, following conditions need to be satisfied [KLLT08]:

1. establishing one-to-one mapping between first group and second group Fig. 3.4a
2. alignment of the clips Fig. 3.4b
3. smooth morphing from one group to another Fig. 3.4c

First condition needs to be always satisfied when considering a simulation of a group of living beings, such as dancers. They need to have defined spatial positions in respect to a keyframe in a time. The movement from one position in a keyframe to a new position in the next frame can be automatically calculated.

Collision detection is solved in post processing because only when linear trajectory of a point is inside the obstacle, this trajectory is adjusted by moving the deepest point from collision to the boundary. This process is iterative and continues until trajectory does not penetrate obstacle. The results of this method shown in Fig. 3.5. However, motion graph is very powerfull solution for control over the positions in a crowd, it does not solve problem of assigning positions in a formations and we would like to solve it.

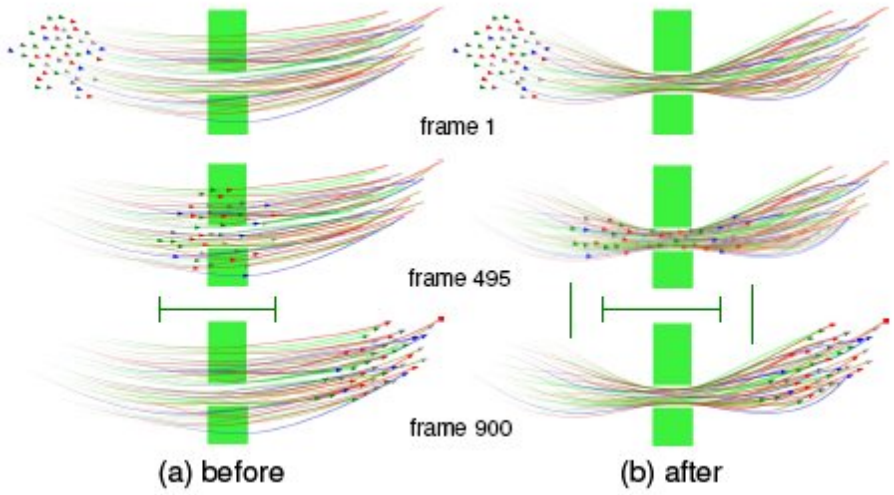

Figure 3.5: Avoiding obstacles with motion graph.
Forcing a circular group (a) through a narrow opening such that the group must elongate to pass. In this specific example, penetrating points inside obstacles are pulled toward the opening [KLLT08].

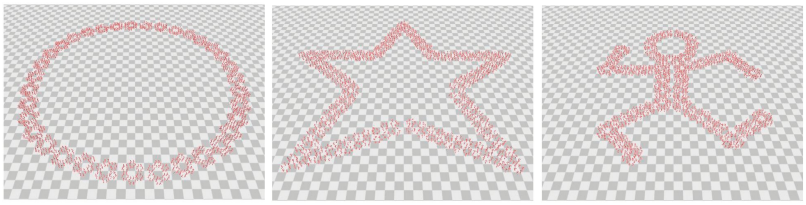

Figure 3.6: Formations created by positioning performers [TYK+09].

3.4 Spectral Based Group Formation Control

The article by Takahashi et al. [TYK+09] is the most recent in the manual path planning field. Group formations in the real world situations can be found in the mass performances or tactical sports such as soccer. This situations have inspired authors of the paper to provide a mathematical description of the motion. This model allows to compute the smooth and realistic movement of a group while respecting adjacency relationships. In mass performances relative positions between neighbors are usually kept to achieve visually pleasant movements.

Firstly, to create smooth transitions between formations, these shapes need to be defined 3.6. There are many ways how to define positions in formations, one of which is manual association of individuals with their spatiotemporal positions. The other way is to extract positions in the formation from a captured video as in [TYK+09]. Usually a scenario and spatiotemporal correspondence is provided beforehand by an artist or a choreographer by defining the shapes or even position of individuals. Afterward automatic calculation of simulation is possible, but artists would like to have a possibility of further control.

After the formations have been created, second step is to calculate a smooth motion. Pre-processing step is needed where the formation is analyzed and calculations are prepared for the next steps. Adjacency relationships are extracted from the formation with *Delaunay triangulation*. In the final graph vertices represent individuals and edges are defined by the triangulation. Edges have positive weights, that are defined as the inverse of the distance between vertices. In order to respect the spectral-based structure of a group formation the *Laplacian matrix* is analyzed. This matrix is symmetric, positive and definite. Therefore the eigenvalues are nonnegative and eigenvectors define *orthonormal basis* called *eigenbasis*. These structures allow spectral decomposition of the graph and with this decomposition, calculation of an interpolation respecting adjacency relationships is possible.

In the previous sections smooth motion between two formations is discussed, but a mass performances of group simulation usually consists of more than two formations. Therefore, also a solution for multiple formations interpolation is provided. Takahashi et al. [TYK+09] propose use of extended Catmull-Rom splines for the smooth transformation.

Unfortunately, avoiding obstacles and collision between members of the group is not a simple problem, when considering a group from a global point of view. Therefore also paper [TYK+09] considers environment better unconstrained and in the common mass performances or stage performances there are usually not many obstacles, even better the movement space can be considered as 2D. However, there is a potential of this approach, enhancements such as collision detection. This idea of distributing individuals in a group to create formations is very similar to our goal. Also we would

like to use particle systems, but solve also a problem of automatic distribution in a formations and use physical forces.

Some of the methods described above are very useful for our goal (particle systems, physical forces). Some have very good ideas, but does not fit whole our goal (spectral-based group formation control, motion graphs), because they do not solve problem of automatic distribution in a shape. Constrained animation can also help in control over the crowd, but forced based methods are rather complex when used together with constrained animation as discussed before in section 3.1.

4

Specification

Formations, that create visible shapes are well known methods used in the movie industry. The problem is that methods are kept but company secret and are not published. Distribution in a shape is not even included in the first specified book about crowds [TM07]. Therefore we decided to develop our own method of control over the crowd.

In this section we will describe my understanding of the topic *Animation of Flocks and Herds* as we will implement it. As could be seen from previous chapter 2 the topic is broad, but we have chosen only small part of it and intend to bring something new to that specific field. This specification is related to the software solution that is implemented and verified in the next chapters 5, 6.

4.1 Goal of Work

Goal of my work is to find solution for automatic position control of crowd that preserves shape.

Moreover, this work should help creators of crowd animation, when the goal of animation is to create certain shapes. As an example, one could imagine flock of fish. These fish with right positioning of individuals can create letters and that way tell

something to spectator. This effect was used in the movie Finding Nemo[1]. To create this visually pleasing effects two steps need to be done. One of those are formations[2] itself and the second one is movement between formations. In this work, both steps should be automatic. Only options for user are shapes for formations, number and geometry of participants. Our implementation will allow export to Collada[3] format and use it in modeling software.

4.2 Animation

Animation means change of features in computer graphics in time [SK96]. Usually this change is either in motion of objects, camera, change in geometry, color, etc. With change in this features observer has feeling of motion pictures compared to still pictures. Something changes.

From classical times of animation[4] it is possible to create feeling of animation, specially motion with series of still picture that vary in time. This principle was first analyzed in 1824 by Peter Mark Rodget, who called it *the persistence of vision.* This principle rest in the fact that our eyes temporarily retain the image of anything they have just seen. Without this principle neither movies nor animation would be possible and we would never get the illusion of an unbroken connection in a series of images [WS01]. For computer graphic purposes we consider images as rendered frames in certain resolution instead of photographs, or drawings as it was in classical times. Afterwards an interesting question arises: *What is the reasonable frame-rate*[5] *?*. Rea-

[1] http://www.imdb.com/title/tt0266543/ - Finding Nemo is an animated movie produced by Pixar Animation Studio's in 2003 and directed by Andrew Stanton.

[2] Positions of individuals, that create certain shape are meant by *formation* in this work.

[3] `http://collada.org` - File format for digital asset exchange of 3D interactive scene based on XML schema - `http://www.w3.org/XML/`

[4] by term *Classical Times* is meant time from 1600 BC, where Egyptian Pharaoh Ramses II built a temple to the goddess Isis which had 110 columns. Each column had a painted figure of Isis in a progressively changed position. As one passed this temple fast (on a horse), Isis appeared to move [WS01].

[5] Frame-rate is a frequency at which imaging device produces frames in sequence. Frames are images produced by devices in certain resolution used in sequence.

sonable frame-rate for motion pictures is a number in which the motion looks fluid without any visible aliases and flickers. It is not a same number for real-time rendering games, where frame-rate includes not only rendered frames, but also interaction. Therefore the number here includes not only smooth animation but also realistic interaction. To achieve realistic animation this frame-rate should be higher. So, at what frame-rate HVS[6] receives smooth motion? As is said in [10010] this number depends on a speed of movement, on brightness or sharpness of a scene and even on color of a scene. For our purpose it is interesting to investigate frame-rate in term of smoothness. We have to calculate positions of participants for every frame to achieve smooth motion. In the silent movies, frame-rate was usually 16 fps[7], which was not very smooth, but still it was in 1920s and it was fastest as could technology bring. Since then technology improved and nowadays frame-rate is at least 24 fps. Sensitive and well trained HVS can notice flicking, but for most of human population and certain period of time it is enough. To improve performance most of productions have done some anti-aliasing like motion blur. When the image is more blurry, less sharp the movement is smoother.

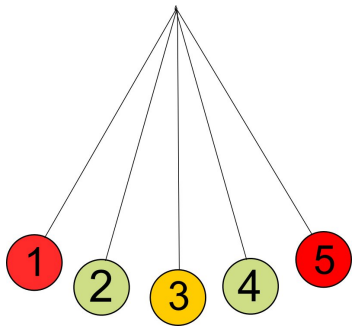

Figure 4.1: Inbetweens.

Positions for animation of a ball on a rope. Extremal positions are nr. 1, 5 in red. Passing position is nr. 3 in yellow. Without this position movement of a ball would not be natural, it would go directly from 1 to 5 and length of rope would change. Inbetweens are nr. 2 and 4, which are linear interpolation between 1, 3 and 3, 5 respectively.

[6]HVS - human visual system
[7]FPS - frames per second

Therefore, if we are going to render animation, we need to render at least 24 frames per second. This is consuming in terms of time and memory needs and it was consuming also in times of analog animation[8] where animator had to draw all frames. In 1920s process of creating drawings for animation has changed. Dick Huemer was the top New York animator and it consumed too much of his valuable time to do all drawings and he came with idea of assistant that would draw *inbetweens* [WS01]. In sequence of positions, there are so-called extreme positions, passing positions and inbetweens, see Fig. 4.1. Extremes are where the curve of movement has local extrema and inflex point, passing positions are where curve has local extrema or this position is important for reasonable movement and inbetweens are positions that are between those others. Inbetweens are in animation to make movement smooth, but from the descriptive point of view are not important, do not bring anything new to the movement. Inbetweens in analog animation were created by assistants, nowadays when computers are powerful, inbetweens can be automatically calculated. Moreover, sometimes this extreme positions are called *keyframes*. Keyframes are also sometimes called the frames, where something important from the story is provided, also known as the storytelling frames. For our work it is important, that we call keyframes the frames where participants of a crowd are positioned in exact pre-defined shape.

4.3 Requirements

For every software solution there are requirements that have to be satisfied. There was need for standardization of requirements, therefore ISO 9001 norm was created. In this norm requirements for software solutions are divided into categories and every solution should satisfy them. In this work we consider this requirements as *General Requirements* and categories are:

- Efficiency
- Economy
- Security

[8]analog animation - animation that is created with series of drawing set in time

- Exactness

This are described in next subsections 4.3.1, 4.3.2, 4.3.3, 4.3.4. For our work there are more requirements to be considered, we refer to those as *Special Requirements* which include:

- Target Groups
- Interaction
- Shape Preserving

More about requirements is specified in next subsections 4.3.5, 4.3.6.

4.3.1 Efficiency

Software solution proposed in this work should be efficient in terms of computation time and memory needs. Because we are dealing with a group of geometrically not simple objects, it is efficient to use pointers instead of copying them. On the other hand, creating heterogeneous crowd means creating various objects. This could be done by changing size and color of the same model. We will take simple fish model for prototype of software solution. Memory needs depend on geometry of objects, which are imported to the software, so we are not dealing with it. In the computation of positions and movement, objects are only particles, which are very simple and it lowers costs for memory and computation time. Also with efficient algorithm for movement, that depend on calculation of distances between particles and distance to the border can improve efficiency. Therefore we will use proper algorithms and data structures to handle distance.

4.3.2 Economy

In terms of price, not only software itself should be considered. To have proper and pleasing result also 3D models of participants and rendering of final output

need to be considered. Software itself is free for use and applet solution can be even reached by web browser. Models of participants can be either low-cost, just simple geometry created for free, or complex models created by well-skilled artists. Moreover, rendering of final output could be done in commercial software as Autodesk 3Ds Max[9] or free software as Blender. Neither models, nor rendering is something we could have influence on, therefore we are not dealing with it, but for final result are as important, as our application. However, output from application should be compatible with import either to Blender, or to 3Ds Max.

4.3.3 Security

Security is important question concerning digital devices. Even small mistake in a source code or in thoughts about possible problems can cause unbelievable results. We will create safer code with as less unhandled errors as possible. Expectation of inputs have to be verified and errors properly handled. In our solution one of the inputs we expect XML file format, either as Collada 1.3 file or as plain XML[10] and we will work with parts of this file. If something is wrong, proper error message will occur. Other inputs will be either slider, or x, y positions of points obtained from clicks by mouse.

4.3.4 Exactness

To be exact in animation and computer graphics output does not necessary mean that result have to be realistic. In our work what we want to achieve is the same as in animated motion pictures. We want to achieve believability instead of realism [WS01]. Flocks of fish or herds of horses are not naturally moving in a way, that they create shape, but we want to pursue spectator, that they could. On the other hand we want these animals or humans to create exact shape with right positioning of

[9] `http://usa.autodesk.com/adsk/servlet/pc/index?id=13567410&siteID=123112` - 3Ds Max modeling software created by Autodesk

[10] XML - eXtensive Markup Language, developed by W3C (World Wide Web) consortium as a successor of HTML (HyberText Markup Language)

participants. We take into account only polygonal shape, because there we could use the Inner Point Property[11], curved shapes are left for further discussion and possible future work. This shape will be only approximation, because we have limited number and size of participants and we want to position them in right place inside the shape. Right place for them is that they fill space inside the polygon and space between participants is as large as possible.

4.3.5 Target groups

Requirements could be different for different groups of users. It is important to identify these groups to deal with those requirements in more detail. In our case these groups are as follows:

- Choreographers
- Chief Animators
- Spectators

Choreographers are a group of people with vision how should final scene look and what shape should participants create. These people usually do not possess extraordinary computer skills and this application should help without special training and lot of work. For them is not really important visual result, but positions of individuals with respect to formations, that will create certain shape. For them interaction is only basic clicking and dragging with mouse.

Chief Animators of a whole animation are the most important target group for our application. They will use it for defining shapes, setting numbers and confirming results. Core of our idea here is to bring as simple as possible interaction with creating shapes and most of calculation will be automatic, therefore creators of a work can have their work faster and easier. Difference between chief animators and

[11]Inner Point Property returns true, if input point is inside the polygon and false if it is outside. Inside point has odd number of intersection between borders of polygon and polyline from point to one direction.

choreographers is that animators are interested in geometry and look of participants itself and choreographers are interested only in positions and orientation. Where chief animators create motion pictures, choreographers work with real human dancers and live performance. Therefore creators has additional option of loading pre-modeled objects as input to the application.

Spectators is the last target group. These people are not interested in creating and interaction, but will judge only final result, final animation. For those people the movement itself is important. Other aspects highly depend on input models and rendering which are both done in external application. To satisfy needs of these people, application will create positions for frames as smooth as possible, but the movement, can depend also on modelling software and interpolation defined in the software.

4.3.6 Shapes

Shape that will be used in the application will be simple polygon, either convex or concave and defined by vertices that are clicked into the planar canvas. This way we can define 2D shape. There is also possible extension to 3D, but there are more methods to be considered.

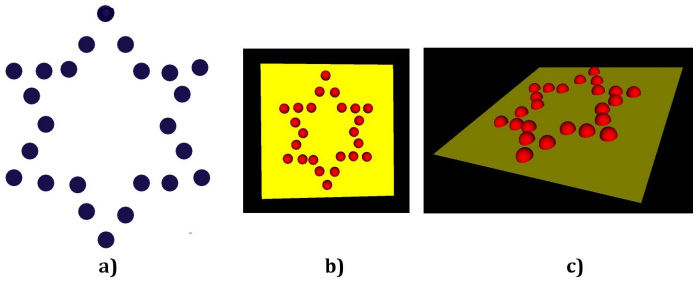

Figure 4.2: Difference between formation in 2D and planar formation in 3D. *(a) formation in 2D (b) formation in 3D, TopView (c) formation in 3D, rotated view*

Positions in formation lay in the same plane as is illustrated in Fig. 4.2. This plane can be rotated, so the final formation is set in a 3D space. The only difference to 2D

formation is that position are defined by 3 coordinates. For example, such formations are created by free-fall parachutists, that create formations that lay in a plane while flying in the 3D space. Viewing it from the right viewpoint (from above or below) can bring pleasant virtual shape.

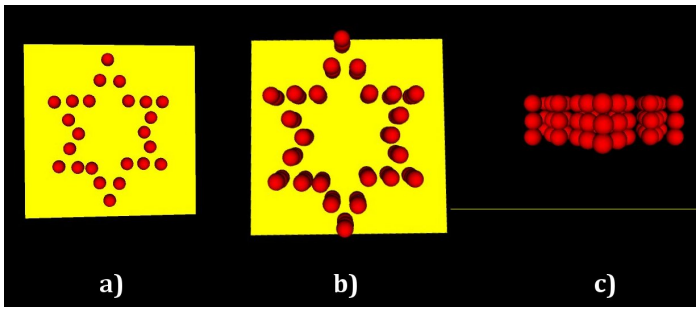

Figure 4.3: Difference between planar formation in 3D and layered formation in 3D. *(a) planar formation (b) layered formation, TopView (c) layered formation FrontView*

Another type is a formation that can be divided in subgroups, where all members of the subgroup lay in a plane as in Fig. 4.3. If all formations can be divided in similar layers, the interpolation can be obtained in the layers separately.

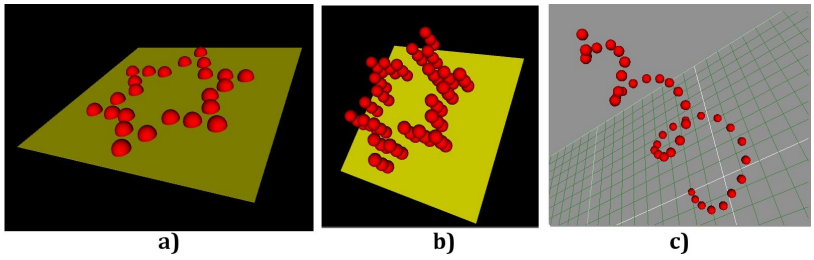

Figure 4.4: Difference between planar formation in 3D. *(a) layered formation (b) and full 3D formation (c)*

The last option is a full 3D formation where individuals are distributed in all three directions Fig. 4.4. For this option also movement needs to be considered in all three directions. Examples are flying birds, swimming fish or an exhibition of airplanes. These shapes are well defined with either polygon or set of polygons as boundary representation of 3D shape. For both Inner Point Property can be successfully used, where in 3D shape calculation of intersection is with polyline and faces.

4.4 Pipeline

In this section the pipeline of the software solution is be described.

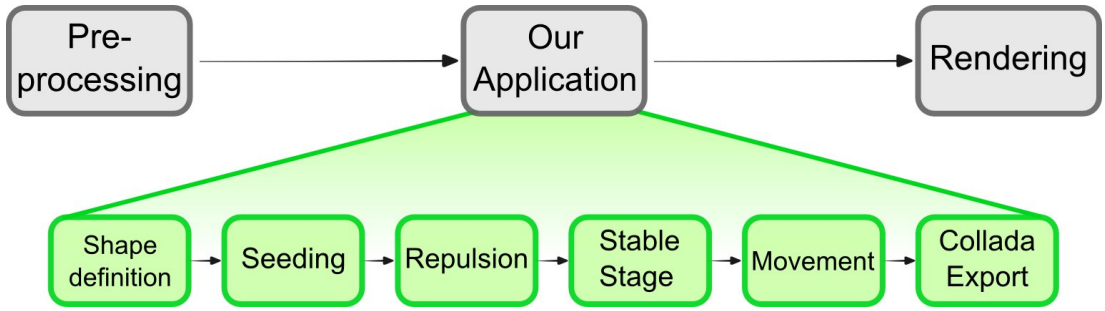

Figure 4.5: Pipeline of work with of our application.

What are inputs, outputs in various levels is shown also in the Fig. 4.5. There are two different ways, which are similar, but have slightly different pipeline. One approach is when we have shapes defined for 2D space, another is when we have shape defined for 3D space. Firstly, **pre-processing** step is needed. For 2D shapes only objects have to be created, for 3D shapes also shapes itself have to be created and written to the XML file. Afterwards, shape has to be set, this step is called **shape definition**. For 2D shapes, user firstly has to click vertices of our polygonal shape in graphical area and position them by dragging. When whole polygon is defined, we can start calculating positions. For 3D shape we omit this step, because we assume, that we chose positions from predefined database. This shapes are saved as XML files. A simple shape is cube, or pyramid and one of them need to be chosen by user. After we have shape and number of participants set, we can start **seeding** step. Seeding means randomly positioning of participants in a shape. After these random positions are set, another level **repulsion** is calculated. We apply repulsive forces, where repulsion between two particles is taking account only if they are in some distance and repulsion depend on inverse distance. After all forces are calculated for one particle, it is moved respect direction, if it is not after the move outside the shape. Forces are calculated, until particles move. The level, when they do not move is called **stable stage**. At this stage each particle remembers it's position. We achieved one formation. Afterwards interpolation of control points is calculated,

level of **movement**. Interpolation between one shape to another. Shapes are again manually moved for 2D shape to another formation, or are interpolated with selecting another shape from database. And same repulsive forces are still applied during the movement. This was we have force-driven calculation of movement. Finally, after all positions in all shapes are done, last stage is calculated. This stage is called **export**. We import pre-defined objects as Collada files, apply size change and color change, assign it to the particles and export all of it to new Collada file with position change, as it is calculated by our application. Finally render everything in some 3D rendering software supporting Collada import, but it is not really important which one. We will use Blender v2.49.

5

Implementation

The broad variety of algorithms were presented in the previous chapters either for crowd simulations and animations (chap. 2) or more specifically for shape preserving and path planning (chap. 3). For our work we decided the direction of force-based shape preserving algorithm and we present implementation of specialized approach that will be widely applicable, as is shown in section 5.8. Decisions in platform, file format and interface are discussed in sections 5.1, 5.2, 5.3. Step by step implementation is discussed in sections 5.4, 5.5, 5.7.

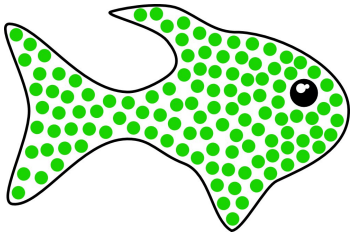

Figure 5.1: Logo of ShapeFish application.

The name of our application is **ShapeFish** and logo is shown in Fig. 5.1. From now on we refer to the application as to ShapeFish software or application. The main idea of enhancement and help that should our software bring to the computer graphics field is in automation. Some of our ideas have been already used in motion pictures industry, but have not been published or are done manually by an artists. We would like to help them with creating stunning group formations.

5.1 Platform

As platform from our point of view we consider operating system, development environment and software needs for users. Operating system for development is Microsoft Windows 7[1] but the application itself is cross-platform. Object-oriented Java programming language[2] developed by Sun Microsystems[3] was chosen, because it's independence on operating system and benefit of Java Applets[4] that can be embedded in web browser. Java Applets are simple extension of plain Java for Desktop application with some restrictions that will not cause problem in our work. For proper functionality in the browser JRE[5] plugin need to be installed, but it is a common addition in development of modern web pages. Use of application in the browsers or online is in modern world important, because this can bring ShapeFish to the wide public and gain more users. However the goal of our work is not primary to be online, but this could be a nice benefit.

When we were considering programing languages, widely used are also C++, C# or Delphi. C++ is widely used with OpenGL[6] for graphical application, but Java also has its own support for OpenGL and this combination is called JOGL[7]. Our main contribution is not in the visualization of 3D scene, but in creating importable files and rendering itself will be done externally, so we will not need OpenGL, but we think also about the future enhancements and possibilities.

As developing environment we have chosen Netbeans 6.1 IDE[8] because of Java Applets support, easy use and various templates and visual forms.

[1] `http://www.microsoft.com/windows/windows-7/` - Windows 7 is the newest operating system from Microsoft company - `http://www.microsoft.com`

[2] `http://java.sun.com/`

[3] `http://www.sun.com`

[4] `http://java.sun.com/applets/`

[5] JRE - Java Runtime Environment

[6] `http://www.opengl.org` - OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications.

[7] `https://jogl.dev.java.net/` - JOGL combination of Java with OpenGL.

[8] `http://netbeans.org/` - Integrated development environment supporting various programming languages.

5.2 XML

XML (eXtensible Markup Language) is a simple file format that is derived from SGML (ISO 8879). This format was originally designed for creating extensible and customized web pages, but it is widely used for data exchange in various applications and also for customizing non-web application. This format is popular because of simple syntax and it is human readable, not like other simple formats. XML is also suitable for parsing in various programing languages, including Java, with their support in existing libraries.

Parsing XML is very easy by using predefined libraries. We use XERCES [9] library that parse between XML documents and tree structure, where nodes are same nodes as in XML. XML allows three node types:

- **document node** - this is root node and has to have this header where version can vary.

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  ```

  Next node usually includes url to the xml schema. This schema will tell the program what nodes include.

- **element node** - this is node that has to have children and can have attributes.

- **text node** - this node has no children or attributes. It is just a plain text. Usually used between tags, but for the consistent structure it is considered as node.

5.2.1 COLLADA

COLLAborative Design Activity is a technology that was finally accepted by Khronos Group in January 2006. Collada should support lossless exchange between applica-

[9] http://xerces.apache.org/xerces-j - XML parser for Java

tions [VD08]. It is a scene description language, where scene is organized in hierarchical structure and nodes with same semantic meaning are grouped together and others are separated. We decided to support version 1.4 that is not newest, but is supported in most applications that we consider as relevant modeling softwares. Collada does not impose any specific order of nodes in the same hierarchical level. List of important nodes that are children of `<collada>` from our perspective includes:

```
<asset></asset>
<library_animations></library_animations>
<library_geometries></library_geometries>
<library_effects></library_effects>
<library_images></library_images>
<library_materials></library_materials>
<library_nodes></library_nodes>
<library_visual_scenes></library_visual_scenes>
<scene></scene>
```

According to Collada specification [AB06] there are also another libraries supported, like physics and fluids. Since we are working with characters, not fluids, we won't need them and omit them. Other libraries such as cameras and lights are left for the custom setting in the modeling software. All these libraries should be supported by any application that imports collada. In our application we only copy to the output following nodes: `<library_effects>`, `<library_images>`, `<library_materials>`, `<scene>`. These libraries deal with materials, textures, geometry, current scene and we only process nodes and animations. Nodes include grouped geometry with transformations. Most exports and imports supports material, geometry, cameras, light and all static scene nodes. Most of modeling applications also supports export of animations, but there is an import of animations. For example standard Blender v2.49 export to Collada 1.4 allows also sample animation, but import does not. Therefore installation of a plugin[10] is needed [AB06]. Further process is discussed in sec. 5.5.6.

[10] `http://colladablender.illusoft.com/cms/content/blogcategory/25/29/` - **ColladaBlender** plugin for export and import Collada for Blender, there are other plugins for

5.3 GUI

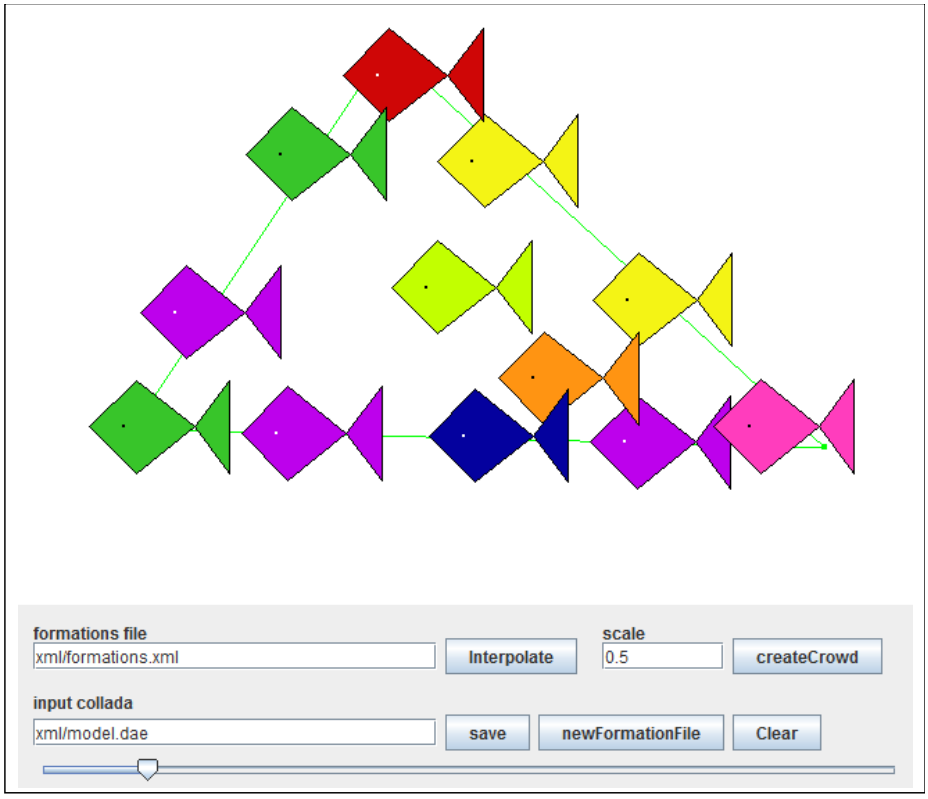

Figure 5.2: Graphic User Interface.
Buttons for manipulation of crowd and export, textfield for file paths and slider to determine the number of participants.

As the whole application is programmed in Java language, for graphical interface, we choose standard applet solution with one canvas, buttons, slider, labels and textfields. Library we decided to use for them is **Swing**[11], that brings modern design of buttons. Screenshot of interface is shown in fig. 5.2. To render 2D scene Graphics of JPanel bring real-time interaction with user clicks and fast reloading. In this list there are components and their functionality:

- **canvas** - canvas of jPanel to interact with user mouse click and for rendering.

Maya and Max3D.

[11] http://java.sun.com/docs/books/tutorial/uiswing/ - creating Java user interfaces with Swing library

When user clicks to the canvas a new point of polygon is created. When click is near existing point movement on mouseMove event follows, another click confirms new position. Everything is redrawn during movement. When right mouse button is used near already defined point, it is deleted.

- **slider** - changing slider position changes number of participants.
- **createCrowd** - crowd with size specified by slider is created and rendered in the canvas to fit in predefined space inside the polygon.
- **clear** - clears everything and redraw canvas with white rectangle.
- **formation file** - path to the file where shapes and formations are saved.
- **input Collada** - path to the file, where model exported to Collada is saved.
- **save** - current shape and positions of participants from canvas are saved in the intern node structure.
- **interpolate** - loads formation file, parse it to the intern structure, interpolates between formations and export everything to the `xml/collada.dae`.
- **newFormationFile** - creates formations file, or rewrites existing from intern structure, interpolates between formations and export everything to the `xml/collada.dae`.
- **scale** - scale of an imported model to the result.

5.4 Input and Output

As a main contribution of our application is to bring more automatic steps inputs and outputs from are minimized. Therefore as inputs we consider number of participants in a crowd, shape of formation and model of object. Number is set by the move of slider in the application. Model of the participant is imported as Collada file that should include only plain scene without camera, light and physics. Shapes are either included from pre-defined XML file, or vertices are simply clicked by the user to the

applet. Moreover, except number, model and shape there are another buttons and setting to customize output.

Output from our application is either applet itself where the methods are shown, but is used in only 2D scene or exported files. There are two types of exported files. One is XML file where vertices of polygon and positions of participants in a crowd are saved. This file can be used as output from the application but was designed to save pre-defined formations. Afterwards these formations are loaded back to the application and are further processed. Another Collada file is with animations and copies of imported objects. This file can be simply imported to the modeling software, where it could be rendered. For our purposes, we tried the solution in Blender v2.49 (and from this point when we refer to the Blender, we mean Blender v2.49).

5.5 Algorithm

Our algorithm is force-based. That means that we are setting random position of particles with only restriction that they are inside the shape. After this we set repulsive forces and particles repel from each other until they are out of other's range or hit the wall and cannot move elsewhere, because force push them against the wall.

5.5.1 Pre-Processing

As pre-processing step we consider everything that has to be prepared before user launches our application. This step includes creating instance of Collada model that will be copied. This model should be geometry, material and texture of participant. Optional can be animation including rotation, but user should omit animation including translation, because that will be rewritten. If Blender export is used, preferably export only selection (when only model is selected without camera and lights), triangles and baked matrices, optionally sample animation. Unselect physics, because it will not be included in the final result anyway. We created a simple fish model, that is composed of more geometries and material than one, shown in Fig. 5.3. Except

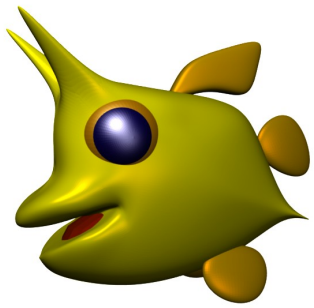

Figure 5.3: Render of sample fish.
Model that we have created to render sample animation.

model, optional is to create polygonal shapes and save them in XML file. Sample of this file is at the end of thesis in Appendix. Instead of predefined models, it is also possible to import any Collada file. For the best results, this model should be exported with the following settings: only selected objects (where whole model is selected), triangles (instead of polygons), without physics (we will not need it), unselect sample animation (if no rotation is included).

5.5.2 Shape Definition

Firstly, before creating a crowd, user has to create a shape that participant should create. We consider shapes as 2D polygonal structures either convex or concave, without any crossing edges. Nevertheless these shapes are defined with only 2D polygons, they are enough for creating some kind of meaning in 3D space as is discussed in section 4.3.6. The restriction here is that these formations need to be viewed by right angle, but still were used in various motion pictures such as Finding Nemo where fish created shapes to tell audience something. Right angle is that, where camera direction is parallel and opposite to normal of the plane where participants lay. Camera should look at the center of formations as is shown in 5.4. These polygonal shapes in our application are created with simple clicking and dragging to the canvas, where new vertex is added next to the last clicked. Other manipulation with vertices is described in 5.2. These shapes can be saved one by one with click on button **Save**.

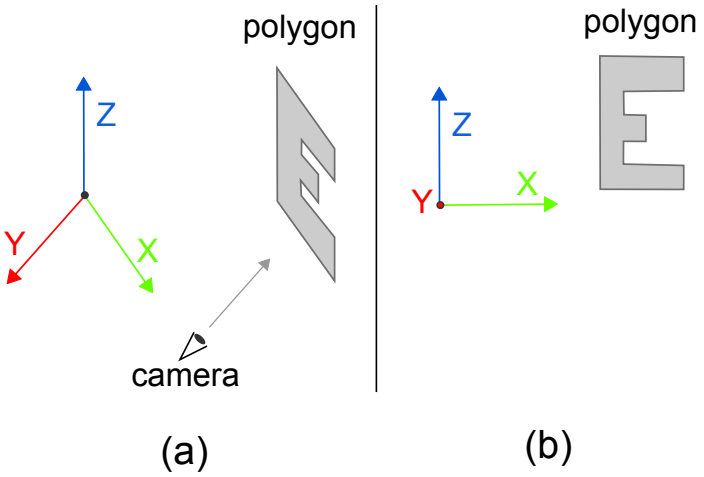

Figure 5.4: Camera position and direction for right view.
(a) Camera position from the side to show how it is set. (b) Camera position same as viewing to show what camera see.

Without saving these shapes, no export is possible, or shapes for interpolation will be loaded from saved XML file.

5.5.3 Seeding

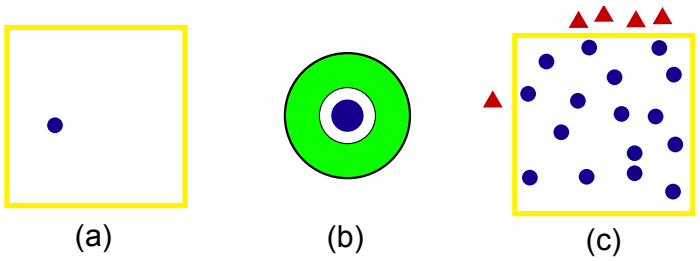

Figure 5.5: Finding initial position.
(a) Polygon with first position found. (b) Surrounding of point, where next position will be found. (c) Particles outside the polygon as red triangle and inside the polygon as blue disc.

After shapes are defined crowd has to be created, then we can either continue manipulating with crowd or directly creating interpolation and export. Because our approach is based on stochastic finding position there will not be same distribution

of crowd for same shape after two creations. Crowd is created when**CreateCrowd** button is pressed or when slider is changed, because for change in the number of participants new crowd has to be created.

When creating crowd the first step we call **Seeding**. In this step initial positions for participant are found, where position of participant is meant by center of the bounding sphere. After this stage forces are applied. There are two different algorithm to find initial position, both are included in the ShapeFish application, because from our perspective, they are suitable for different results. In the first one, we have to find position of the first particle in the center of polygon. We do this by simple calculating barycentric coordinates of a polygon:

$$C = \sum_{i=0}^{n} \frac{1}{n} * P_i, \tag{5.1}$$

where C are coordinates of center point, n is number of vertices in polygon, P_i are coordinates of a vertex from polygon. This works fine with the convex polygons and some concave. Center of the polygon is also drawn in the canvas, because it is very important point as is discussed later 5.5.4. Positions of other particles are calculated with finding random angle and length in some range. This range can vary and depend on the polygon and size of objects. With these two coordinates, position is found as polar coordinates in a system, where position of first particle is a zero point:

$$x = F_x + cos(\alpha) * l, \tag{5.2}$$

$$y = F_y + sin(\alpha) * l, \tag{5.3}$$

where (x, y) are coordinates of a new point, $F = (F_x, F_y)$ are coordinates of a center, α is angle and l is length as discussed before. This way particles are distributed around the randomly around the center point as is shown in the Fig. 5.5(b). In the second approach, instead of center point, we take vertices from polygon and calculate next positions around these vertices. Question which vertex take into account is answered by *modulo n* where n is number of vertices in the polygon. This will evenly distribute crowd around all vertices. Only thing that we check is if coordinates of a new position are inside the polygon with **Inner Point Property**[12], if not, we find another length

[12]Inner Point Property is easy calculation function that returns if point is inside the polygon, or

and angle. In Fig. 5.5(c) are shown points inside (blue) and outside the polygon (red).

5.5.4 Stable Stage

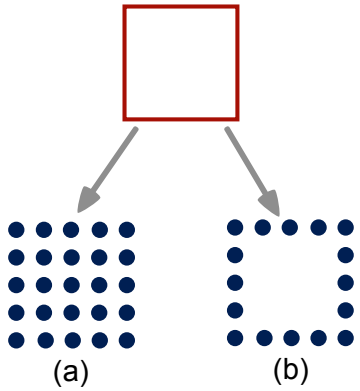

Figure 5.6: Distribution of particles.
(a) Distribution of particles on edge. (b) Distribution of particles inside the shape.

After initial positions of a crowd are found, we start to apply forces. We evaluate them in iterative steps, where in each step forces applied to one particle are calculated. We start with the last added, because in the second approach of seeding, we have last particles more in the center, so they have space to move, while those in the corners do not have space to move. One repulsive force between two particles is defined as:

$$\vec{f} = \frac{(B - A) * C}{d}, \tag{5.4}$$

where $\vec{f}$ is final force, B is point, on which are applied forces and we calculate them, A is a point that affects B, d is distance between A, B and C is constant. Distance we calculate as standard spherical Euclidean distance:

$$D = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2}, \tag{5.5}$$

outside and it works for both convex and concave polygons. Define a halfline through point with any direction and calculate sum of intersection with edges of polygon. If sum is odd, then point is inside, if it is even, point is outside.

where D is distance, $B = (B_x, B_Y)$, $A = (A_x, A_y)$ are points. Constant C depends on an area of polygon, number of particles and it means in which distance particles affects each other. It is calculated as follows:

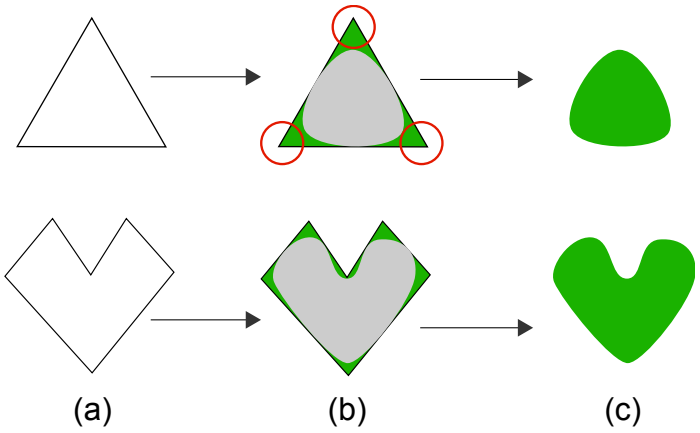

Figure 5.7: Shape preserving for centered initial step.
(a) Only shape. (b) Shape with bounding of crowd. (c) Bounding shape that crowd created.

$$C = \sqrt{\frac{a}{p * \Pi}}, \tag{5.6}$$

where C is constant, a is area of polygon and p is a number of particles. If this force is too large all particles stops at the edge of polygon as is shown in Fig. 5.6. If it is too small, particles do not repulse and stay at the place and do not create a right shape. After calculation of effects from all particles around the one, we scale force vector as follows:

$$\vec{f} = \frac{\vec{f} * C}{l}, \tag{5.7}$$

where $\vec{f}$ is force vector, C is same constant before, and l is length of old force vector.

There is a problem, when particles should move across the border, when after applying forces article will be outside. Then particle is moved toward the force, but only until intersection with edge is found. Another problem is when some particles are suddenly outside before application of forces. This can happen when user moves with the

vertices of polygon. We solve this case by finding first intersection with line defined by the edges of polygon and another line defined by point outside and center point. Because of this solution the center point is important. It is important that center point will be inside the polygon. Forces are applied in recursive order, where recursion stops, when all particles do not move, or when number of movements is exceeded. When recursion stops this is a stable stage. When seeding is done by the second approach, corners of a polygon are preserved. That is important for some shapes, such as triangle, for some, as heart it is better not to preserve corners and better results are with first seeding approach. This is shown in Fig. 5.7

5.5.5 Movement

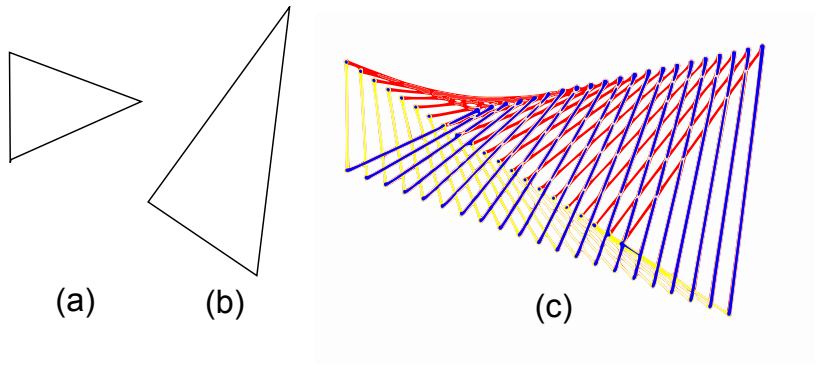

Figure 5.8: Linear interpolation.
(a) First polygon (b) Second polygon. (c) Interpolations between first and second frame.

In the previous sections 5.5.3, 5.5.4 movement between the formations is done manually by moving the vertices in of the polygon. This is suitable for visualization in the ShapeFish application, but not for export. Therefore we developed algorithm for interpolation between shapes. By this solution keyframes and middle frames can be saved for formations created by participants, where keyframes are defined by the shapes, that are saved in the XML file and middle frames are calculated by the interpolation. we use linear interpolation between shapes, where vertices from one shape are mapped to vertices to another with bijective mapping as is shown in Fig. 5.8.

Equation for linear interpolation is following:

$$X = K_1 * (1 - t) + K_2 * t;\ \mathrm{t} \in < 0, 1 >, \tag{5.8}$$

where X is a point between K_1 and K_2 defined by t If we have more vertices in one of the shapes, we add vertices to another so that bijection can be applied. For each keyframe and middle frame we calculate position as is discussed in 5.5.3, 5.5.4, where seeding step is done in for the first formation and for other is seeding omitted with using positions of previous formation. We save these new position of a particles in an array, so we can use them in export step.

5.5.6 Export

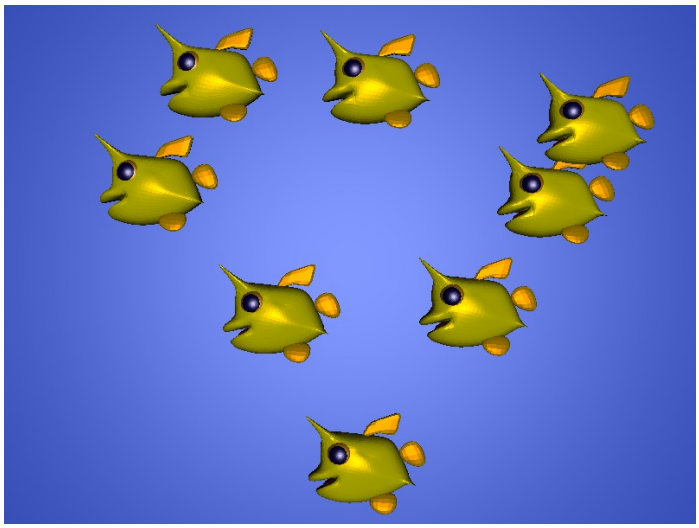

Figure 5.9: Render of scene imported to Blender.

From the previous step we have saved positions of particles in keyframes and middle frames, in this step we process imported model for the export. Here we work with the Collada specifications, then we tried to import our result to the Blender and render everything. In the Fig. 5.9 is shown one of the rendered image, but the final result as included on the CD as .avi video file.

As was said in the section 5.2.1 in the export file we are copying only some library nodes offered by Collada. Whole `library_geometries` is copied without the change,

because we are not creating geometry for every participant, we are using only instance of that geometry. That means, that we store vertices and triangles only once in the file and save memory. Therefore every participant looks alike. To instance this geometry we could either use `library_visual_scenes` and create nodes for every participant in an one visual scene. Using this option we would have nodes for every geometry and every participant. Which means $g * n$ nodes, where n is number of participants and g is number of geometry parts. Moreover, there could not be any translations and all parts of the model will be on the center of coordinate system and model will not look as it was modeled. Therefore we use rather `library_nodes` where we can apply transformations to the static model and use visual scene nodes to apply animation. This way we still have $g * n$ nodes twice (in library visual scenes and library nodes), but model itself will look same way as it was exported. According to the Collada specification [AB06] we could group all nodes into one and create only one animation node for one participant, but this is not supported by either Blender, or 3D Studio Max. Therefore we decided to use this version, which is more memory consuming, but work well in the Blender. It does not work in the 3D Studio, because it does not have support for `library_nodes` at all. From the library visual scenes in the original model Collada file we copy nodes to our Collada export file and library nodes for every participant same nodes, just with unique ID's. And we create nodes in the library visual scenes to have right reference from library animations.

With this instancing of geometry, we will have n participants represented by the model, but they will be all at the same place. As is said in section 5.5.5 we have prepared times and positions for keyframes and middle frames, so now we create $g * n$ animation nodes in the `library_animation` with right instancing to the nodes in a library visual scenes. We know ID's of the nodes, because we rename them. We put in the library animations same translation for every part of geometry. Times for the frames are set for all crowd same, and we set time to stay in the keyframes, where crowd created meaningful formation. Example of an animation node is below:

```
<animation id="translate">
        <source id="input">
```

```
            <float_array count="1" id="input-array">1</float_array>
            <technique_common>
                    <accessor count="1" source="#input-array" stride="1">
                            <param type="float" name="TIME"></param>
                    </accessor>
            </technique_common>
    </source>
    <source id="output">
            <float_array count="3" id="output-array">4 0 0</float_array>
            <technique_common>
                    <accessor count="1" source="#output-array" stride="3">
                            <param type="float" name="X"></param>
                            <param type="float" name="Y"></param>
                            <param type="float" name="Z"></param>
                    </accessor>
            </technique_common>
    </source>
    <source id="interpolation">
            <Name_array count="3" id="inter-array">BEZIER BEZIER
            BEZIER </Name_array>
            <technique_common>
                    <accessor count="1" source="#inter-array" stride="3">
                            <param type="Name" name="X"></param>
                            <param type="Name" name="Y"></param>
                            <param type="Name" name="Z"></param>
                    </accessor>
            </technique_common>
    </source>
    <sampler id="sampler">
            <input semantic="INPUT" source="#input"/>
            <input semantic="OUTPUT" source="#output"/>
            <input semantic="INTERPOLATION" source="#interpolation"/>
```

```
        </sampler>
        <channel source="#sampler" target="Cube/translate"/>
</animation>
```

---

where we are setting one keyframe, in the first `<source>` there is time for keyframe, in the second `<source>` there are positions of local coordinates, and in the third `<source>` type of interpolation is set. In the `<sampler>` node we link `<source>` nodes together and in `<channel>` there is reference to the nodes in library visual scenes and to the transformation node there. These tags are in every animation node, just their values are not the same. When time for the first keyframe is > 0.0 then position of the nodes is same as it is after applying transformations for static model. Example of the whole Collada file with one cube animated in 2 keyframes is in the additions.

5.6 Problems

The main problems were with export to the Collada format. The export itself is not a problem because of a very good specification, but we wanted export that can be easily imported to the modeling software. As our solution is brought to the public, we would like also a modeling software to be free, Blender is therefore a very good solution. It is free, allows import from Collada and very powerful. Another popular modeling software is 3D Studio Max. Therefore we wanted to export Collada file, that could be imported to the both.

As is discussed in 5.2.1, according to specification Collada supports child nodes in the nodes of *library_nodes*. Using this benefit geometry nodes from the imported model could be easily grouped, manipulated and copied. Grouping whole geometry of the model in one parent node will allow manipulation of a whole model with one *animation* node in *library_animations*. Unfortunately import to 3D Studio Max does not support neither *library_nodes* itself nor children nodes of nodes and Blender does not support children nodes of nodes. Therefore we need to set same animation not

only for every participant, but also for every geometry in it. Moreover, because of this constraint, only one animation of translation can be applied on the models.

5.7 Extensions

There are some extensions that are easily available. One of these is creating heterogeneous crowd with setting an material and textures for exported participants. It is done for 2D shape, where fish are drawn in more colors. In case of 3D model it is more complex, because ShapeFish software does not have more information about geometry. Geometry nodes have names, but do not have necessary semantic meaning. To create heterogeneous crowd application need to have information which geometry is what part of body. This way we can create plausible colorful models.

Our model with little enhancement is suitable for 3D shapes that are defined with B-rep [13] We need to calculate inner point property same way, but we calculate intersection with the face instead of edge. Moreover we have to change, force count from 2D to 3D with simple change of distance to

$$d = \sqrt{(x - f_x)^2 + (y - f_y)^2 + (z - f_z)^2}, \tag{5.9}$$

where $\vec{f} = (f_x, f_y, f_z)$ is force vector and (x, y, z) are coordinates of center of participant and change polar coordinates from seeding step to spherical.

Another improvement can be done with better solution for participants that are in the bounding box of others, that collide. Now it is done with simple movement in the third direction that is not a problem when camera is set right, but could be a problem when whole crowd is viewed from the angle. On the other hand when we look at the crowd from angle, shapes will not be seen at all. We leave these extensions for the future work.

[13] B-rep - boundary representation, where object is described by list of vertices, list of edges and list of faces.

5.8 Applicability

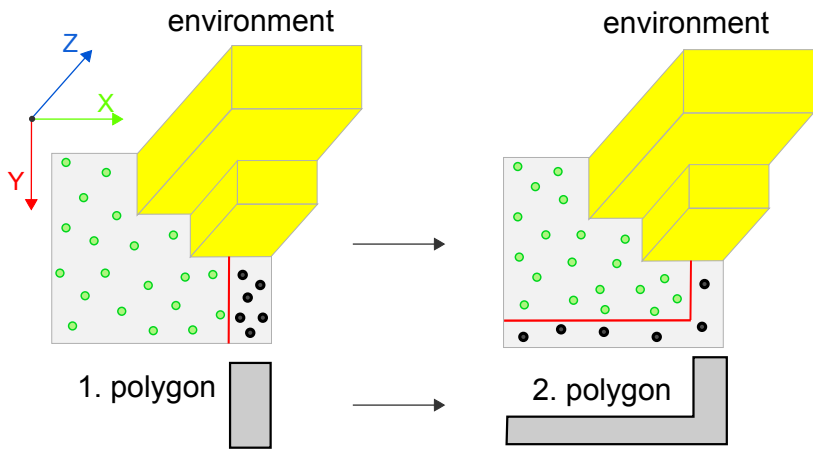

Figure 5.10: ShapeFish for urban environment.

Movement of a crowd in the city based on our application. (a) Initial step (b) Final step.

ShapeFish application was originally designed to create plausible animation that would preserve shape. This could be help for motion picture animators. Moreover there are another possibilities of use. One of these possibilities is shown in Fig. 5.10. Simple definition of shape in which crowd is distributed in the first frame, and then definition another shape in the right place for final frame, with right definition of middle frames can create animation of crowd. With another import to Blender, or with modeling environment afterward can bring plausible animation with crowd spread in the square. Advantages of this approach are that it is very easily defined and with right middle frame also collision detection with obstacles can be preserved. Drawbacks of this approach is that there is no intelligence behind the movements, therefore crowd moves by the rules of physics, not by the rules of behavior.

6

Results and evaluation

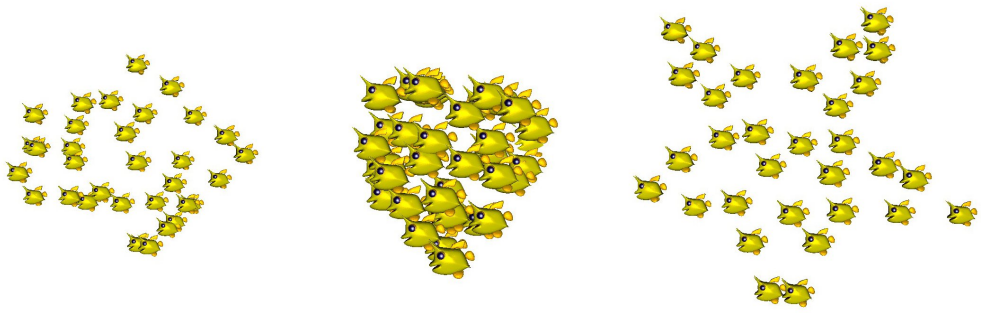

Figure 6.1: Results of the method.
Flock of fish in a three formations - arrow, heart, star.

We took flock of fish as our referencing crowd, where result of our approach is shown in the figure 6.1, where fish create a shapes *arrow, heart, star*. Here is shown that when there are participant with same x, y coordinates, fish had to be moved also in z-directions, by random setting z-coordinate in some range. This will solve problem of intersecting geometry. Moreover, because formations are rendered with the right setting of camera, it is not visible, that participants actually does not lay in the plane.

We have tested our approach in three different tests.

- **Performance**, section 6.1 where we run the application with various inputs,

- **Human Perception**, section 6.2 where we asked users about their personal perception of rendered images,
- **Fractal dimension**, section 6.3 where we tested distribution of participants.

Some of the tests confirmed our hypothesis, some brought questions that should be answered in a future work. What these test showed is discussed in sections below.

6.1 Performance

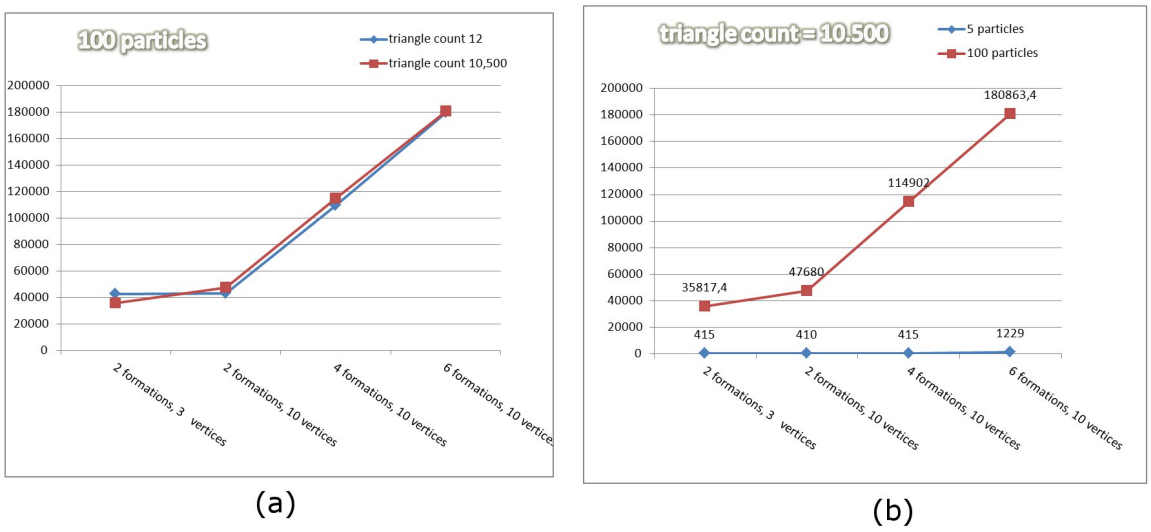

Figure 6.2: Graphs of performance.
4 different files with different formations. (a) 100 particles, 10 interpolation steps, curves for 2 models with different triangle counts (b) 10 interpolation steps, 10,500 triangles in model, curves for 2 different amounts of particles.

We have used HP Pavilion Tx1000 computer, with Microsoft Windows 7 operating system, AMD Turion 64x2 T-58 1.9 GHz processor, nVidia GeForce 6150 Go graphic card, 4GB RAM. Working environment was Netbeans IDE 6.1. We let computer ran 5 tests for same settings and we changed 4 formation files, various particle counts (5,100) and various input models (10,500 triangles and 12 triangles). We rendered

every step on the screen, where participants were represented by a circle to gather actual performance, not only export.

Assumption, that performance time is not depend on number of triangles in a model was successfully confirmed as is shown in fig. 6.2(a). In this figure are two results shown, one for triangle count 12 and one for triangle count 10,500 where curves are similar. Another fig. 6.2(b) successfully confirms assumption, that time highly depend on number of participants. This is because when there are more participants, repulsion forces cause more iterations. Of course, when there are more key frames with final formations, also time rises, simply because there are more positions to count.

These results show us that our assumptions were confirmed, that our approach is independent on geometry of input object, but convergence of participants in formation should be more analyzed and we lave this to the future work.

6.2 Human Perception

We tested our approach for 2D shapes in 2D space, where also objects are 2D fish and move only in two directions. Moreover we also tested our results in 3D space where we exported 3D objects with Collada. We rendered animations in Blender v2.49 and final .avi file is included on a CD. Static images of stable stages for keyframes where included in online questionnaire. We asked users to write what kind of shape formations create. Some of the users did not understand meaning of a question and instead of shape they answered *fish*, because in the picture was flock of fish. We gathered 46 answers for 15 questions and users were in age 20 - 32 years. Formations, that were as questions are included in additions at the end of work.

Moreover we showed our results also to 30 children of dancing class in an age 8 - 12 years and explained what should be observed. These children understand task, because it is commonly used in dancing. We took these answers personally therefore, there were less misunderstanding answers than in online test, as is shown in fig. 6.3(b). Most of the children understood meaning behind the formations, however

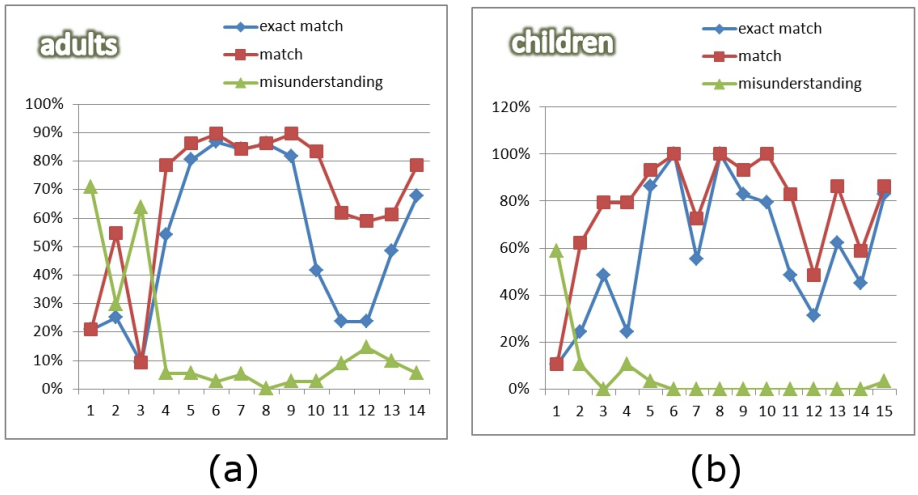

Figure 6.3: Results of a questionnaire.
(a) results from online questionnaire (b) results from children's answers

some of them have stunning imagination and slight spaces in a triangle can cause, that child see there a face.

In the results from online questionnaire, fig. 6.3(a) is significant, that there were more misunderstanding in the beginning (answers 1-4), where users did not understand task. Moreover, there were pictures that were successful, but there were also pictures, where formations failed from our point of view (question 2,3). On the other hand, in the first pictures fish created perfect triangle, but 71% adults and 59% of children misunderstood, therefore it means, that more participants is needed to bring visual meaning by the formation not by the participant. Question no. 2 from our opinion failed, because it should present triangle but it does not preserve shapes (first method used for initial step) and also answers showed it. Only 25% answers were exactly right (*triangle*), but 55% and 62% answers mean something reasonable. Therefore more participants helped. Question no. 3 also failed, because it has only 4 fish and does not preserve corner, but only 9% adult answers identified formation correctly as *nothing* and none of them as *triangle*. This shows that when human perception system is asked to describe what is shown, it will try to find something. Therefore when in the movie it is significant, that flock is telling spectator something, he/she will try to find it and finding the meaning is easier than in our isolated pictures.

Nevertheless there were chosen simple pictures, there were only metaphors of some objects and it could have equally right meaning to the various users with various backgrounds and educations, which is visible also in the fig. 6.3 where in questions no. 11, 12 metaphoric representation is not as common as others and only 24% answers were exactly correct compared to the intention of designer, but 60% answers were equally correct. Other questions were successfully answered with accuracy between 50%-90% for exact match and between 60% - 90% for reasonable match, which is from our point of view accurate.

We tested only static images of group in a stable stage, but when formations are shown in a row meaning can vary. These shapes were simple, therefore without previous knowledge some iconic representations can have more meanings. Generally, questionnaire shown, that every person percepts in the shape some kind of meaning, so it were not only group of participants distributed in a plane, but their formations meant something and this was our goal.

6.3 Fractal Dimension

We have tested our solution by extracting black and white picture, where participants are rendered as black filled circles. We loaded these pictures in a Fractalyse software [1] and analyze them by correlation setting set to quadratic and default. We achieve fractal dimension in a range 0.82 - 1.12. We have also tested full regular rectangle, where participants were presented as same black filled circles and were equally distributed along the edge with fractal dimension (0.77) and inside the rectangle with fractal dimension (0.81). Full black triangle has fractal dimension (1.9) with settings specified above. Moreover we also tested fractal dimension of same polygons as were for formations, but these were filled with black and results was in range 1.84 - 1.90. Same shapes with only thick edges, but white space inside have fractal dimension in range 1.0 - 1.16.

These results show us, that our approach is more different to a full shapes than it is

[1] `http://www.fractalyse.org/en-home.html` - software for fractal analysis

to edge lines. More significantly, it is closer to the regular distribution, but it could vary because of number of participants, their size, intersection and in the 3D case it has even wider range. Therefore this was just an experiment, that does not have very significant results.

7

Conclusion

In conclusion we would like sum up our contribution, which is in bringing not common control ever the crowd, automations as help to designers, dance choreographers of large crowds and even in flexible use by export to Collada. We would like to point out that we successfully used repulsive forces for control over the group, positioning and movement. These formations did not act by behavior rules, but created reasonable shapes. Crowd can be used by right positioning to bring more metaphoric layer to the spectator's experience and large groups of animals or humans are already used to bring such experience. Either in live performances of real dancers, or in motion pictures. Moreover our approach brings more automation to the control over the group where positions are important. If our approach creates meaningful shapes we have tested in questionnaire and confirmed assumptions, that right distribution of a group can result in iconic representation of another shape. Our approach needs more evaluation, because it is not common to use such a method.

Bibliography

[10010] 100fps.com. How many frames per second can human eye see. *published online* `http://www.100fps.com/how_many_frames_can_humans_see.htm`, accessed 8.2. 2010.

[AB06] Remi Arnaud and Mark C. Barnes. *Collada: Sailing the Gulf of 3D Digital Content Creation*. A.K.Peters, Ltd., 1 edition, 2006.

[AMC03] Matt Anderson, Eric McDaniel, and Stephen Chenney. Constrained animation of flocks. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 286–297. Eurographics Association, 2003.

[BCN97] Eric Bouvier, Eyal Cohen, and Laurent Najman. From crowd simulation to airbag deplyoment: particle systems, a new paradigm of simulation. *Journal of Electronic Imaging*, 6(1):94–107, January 1997.

[Bez01] Dusan Bezak. *Mass Scenes Rendering Framework*. PhD thesis, Faculty of mathematics, physics and informatics, Commenius University, Bratislava, Slovakia, 2001.

[CKFL05] Iain D. Couzin, Jens Krause, Nigel R. Franks, and Simon A. Levin. Effective leadership and decision making in animal groups on the move. *Nature*, 433:513–516, February 2005.

[Gar70] Martin Gardner. The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223:120–123, October 1970.

[HB94] Jessica K. Hodgins and David C. Brogan. Robot herds: Group behaviors for systems with significant dynamics. In *Proc. Artificial Life IV.*, pages 319–324. IEEE Computer Society, 1994.

[HB06] Christopher Hartman and Bedrich Benes. Autonomous boids. *Computer Animation and Virtual Worlds*, 17(3-4):199–206, July 2006.

[HLTC03] Laure Heigeas, Annie Luciani, Joëlle Thollot, and Nicolas Castagné. A physically-based particle model of emergent crowd behaviors. In *Graphicon*, 2003.

[KBT03] Marcelo Kallmann, Hanspeter Bieri, and Daniel Thalmann. Fully dynamic constrained delaunay triangulations. In *Geometric Modelling for Scientific Visualization*, pages 241–257. Springer-Verlag, Heidelberg, Germany, first edition, 2003.

[KD09] Norbert Ketterl and Jana Dadova. Crowd simulation. Research seminar, Vienna University of Technology, 2009.

[Ker05] Laurent Kermel. Crowds in madagascar. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*. ACM, 2005.

[KLLT08] Taesoo Kwon, Kang H. Lee, Jehee Lee, and Shigeo Takahashi. Group motion editing. In *SIGGRAPH '08 Proceedings*, pages 1–8. ACM, 2008.

[LCF05] Yu-Chi Lai, Stephen Chenney, and Shaohua Fan. Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 281–290. ACM Press, 2005.

[LM03] Timothy J. Lightfoot and George J. Milne. Modelling emergent crowd behaviour. In *Proceedings of the 1st Australian Conference on Artificial Life (ACAL 03)*, pages 159–169, 2003.

[LRMA05] Jyh-Ming Lien, Samuel Rodriguez, Jean-Phillipe Malric, and Nancy M. Amato. Shepherding behaviors with multiple shepherds. In *Proc. of the 2005 IEEE International Conference on Robotics and Automation*, pages 3402–3407. IEEE, 2005.

[McD06] Rachel McDonnell. *Realistic Crowd Animation: A Perceptual Approach*. PhD thesis, University of Dublin, Trinity College, 2006.

[NS92] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. In *Journal of Physics I France 2*, number 12 in 2, pages 2021–2229. JOP, 1992.

[Nyg07] Martin Nygren. *Simulation of Human Behavior in Stressful Crowd Situations*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2007.

[OCD+02] Carol O'Sullivan, Justine Cassell, Simon Dobbyn, Christopher Peters, William Leeson, Thanh Giang, and John Dingliana. Crowd and group simulation with levels of detail for geometry, motion and behaviour. In *Third Irish Workshop on Computer Graphics*, 2002.

[OCV+02] Carol O'Sullivan, Justine Cassell, Hannes Vilhjalmsson, John Dingaliana, Simon Dobbyn, Brian Mcnamee, Christopher Peters, and Thang Giang. Levels of detail for crowds and groups. *Computer Graphics Forum*, 21(4):733–741, 2002.

[RBB97] Ravi Ramamoorthi, Cindy Ball, and Alan H. Barr. Dynamic splines with constraints for animation. Technical report, California Institute of Technology, Pasadena, CA, USA, 1997.

[Ree83] William T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *SIGGRAPH '83, ACM Transactions on Graphics*, 2:359–376, 1983.

[Rem03] Scott Remington. Sparking life: notes on the performance capture sessions for the lord of the rings: the two towers. *SIGGRAPH Comput. Graph.*, 37(4):17–21, 2003.

[Rey87] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, July 1987.

[Rey99] Craig W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference 1999*, pages 763–782. Miller Freeman Game Group, San Francisco, California, 1999.

[SK96] Laszlo Szirmay-Kalos. *Theory of Three Dimensional Computer Graphics*. Akademiai Kiado, Budapest, 1996.

[SKG05] Mankyu Sung, Lucas Kovar, and Michael Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 291–300, New York, NY, USA, 2005. ACM.

[TCP06] Adrien Treuille, Seth Cooper, and Zoran Popovic. Continuum crowds. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1160–1168. ACM Press, 2006.

[TM97] Daniel Thalmann and Soraia R. Musse. A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Workshop Computer Animation and Simulation of Eurographics*, pages 39–52, 1997.

[TM07] Daniel Thalmann and Soraia R. Musse. *Crowd Simulation*. Springer, 1 edition, October 2007.

[TYK+09] Shigeo Takahashi, Kenichi Yoshida, Taesoo Kwon, Kang Hoon H. Lee, Jehee Lee, and Sung Yong Y. Shin. Spectral-based group formation control. *Computer Graphics Forum*, 28:639–648, 2009.

[UHT04] Branislav Ulicny, Pablo Heras, and Daniel Thalmann. Crowdbrush: interactive authoring of real-time crowd scenes. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 243–252. ACM Press, 2004.

[VD08] Martin Vataha and Jana Dadova. Skybox as info billboard. In *Student Scientific Conference 2008*. Faculty of mathematics, physics and informatics, Comenius University, 2008.

[Wit97] Andrew Witkin. An introduction to physically based modeling: Particle system dynamics. *ACM Siggraph Course Notes*, 1997.

[WS01] Richard Williams and Imogen Sutton. *The animator's survival kit*. Faber and Faber, 1 edition, November 2001.

[YSWW06] Gang Yang, Hanqiu Sun, Wencheng Wang, and Enhua Wu. Interactive fur modeling based on hierarchical texture layers. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 343–346, New York, NY, USA, 2006. ACM.

Appendix

- Formations.xml
- Cube.dae
- Questionnaire
- SCCG Poster
- included CD
  - Java - source files for ShapeFish Applet
  - Renders - rendered images
  - Thesis.pdf - pdf version of thesis
  - Video - video with rendered animations and screen capture