Chapter 6
Scalable Solutions for Simulating, Animating, and Rendering Real-Time Crowds of Diverse Virtual Humans

Daniel Thalmann, Helena Grillon, Jonathan Maïm, and Barbara Yersin

Abstract In this chapter, we describe how we can model crowds in real-time using dynamic meshes, static meshes and impostors.Techniques to introduce variety in crowds including colors, shapes, textures, individual animation, individualized path-planning, simple and complex accessories are explained. We also present a hybrid architecture to handle the path planning of thousands of pedestrians in real time, while ensuring dynamic collision avoidance. Several behavioral aspects are presented as gaze control, group behavior, as well as the specific technique of crowd patches. Several case-studies are shown in cultural heritage and social phobia.

6.1 Introduction

To simulate large crowds at high frame rates, it is necessary to use several levels of detail (LOD). Characters close to the camera are accurately rendered and animated with costly methods, while those farther away are represented with less detailed, faster representations. In this chapter, we describe how we can model crowds in real-time using dynamic meshes, static meshes and impostors. Techniques to introduce variety in crowds including individual animation, individualized path-planning, simple and complex accessories are explained.

D. Thalmann (✉)
Institute for Media Innovation, Nanyang Technological University, Singapore and EPFL, Switzerland
e-mail: danielthalmann@ntu.edu.sg

H. Grillon
Centrale de Compensation, Geneva, Switzerland
e-mail: helena.grillon@gmail.com

J. Maïm • B. Yersin
Minsh.net, Bengaluru, Karnataka, India
e-mail: Jonathan.Maim@gmail.com; Barbara@minsh.net

S. Ali et al. (eds.), *Modeling, Simulation and Visual Analysis of Crowds*, The International Series in Video Computing 11, DOI 10.1007/978-1-4614-8483-7_6,

We also present a hybrid architecture to handle the path planning of thousands of pedestrians in real time, while ensuring dynamic collision avoidance. The scalability of our approach allows to interactively create and distribute regions of varied interest, where motion planning is ruled by different algorithms. Practically, regions of high interest are governed by a long-term potential field-based approach, while other zones exploit a graph of the environment and short-term avoidance techniques. Our method also ensures pedestrian motion continuity when switching between motion planning algorithms. Tests and comparisons show that our architecture is able to realistically plan motion for many groups of characters, for a total of several thousands of people in real time, and in varied environments. We finally introduce a method for populating large-scale interactive virtual environments with walking and idle humans, as well as animated and static objects. The key idea of our solution is to build environments from a set of blocks, the crowd patches, that describe periodic motion for a small local population, as well as other environment details. Periodicity in time allows endless replay.

Several case-studies are shown in cultural heritage, emergency situations, and social phobia.

The rest of this chapter is organized into sections. Section 6.2 explains the principles of multiple levels of detail and how they are processed during the rendering of large crowd simulation. Section 6.3 is dedicated to the variety aspects. We explain how to create various shapes, colors, textures, animation styles. We also emphasize the concept of accessories like bags, glasses, mobile phones, hats. In Sect. 6.4, we introduce path planning with three approaches: navigation graphs, continuum crowds, and into more details an hybrid method. Section 6.5 describes two aspects related to behavior: gaze control and group behavior. The notion of crowd patches is also presented in this section. Finally, Sect. 6.6 briefly discusses a few applications like simulation of ancient cities or treatment of agoraphobia.

Since 1997, when Thalmann and Musse [1] started their pioneering work on crowds of Virtual Humans, a lot of researchers have investigated into this field. Today, there is almost no graphics or animation conference without papers on crowd simulation. We cannot easily introduce a state of the art in this chapter, but a few selected works will be cited along the chapter. The reader can find an exhaustive state of the art in the second edition of the book “Crowd Simulation” [2].

6.2 Representing Virtual Humans Using Multiple Levels of Detail

Animation and rendering of virtual humans relies on a set of geometric, kinematic and appearance models. When dealing with crowds made of thousands of virtual humans (see Fig. 6.1), it is not possible to dispose of a unique model for each of them; this would result in huge memory consumption and computation times. A key solution is to use templates defining each type of virtual human, which can be continuously derived in order to generate an infinite variety of instances of virtual humans.

Fig. 6.1 Large crowd of diverse virtual humans

The first step is to create a sufficient combination of templates with different genders and ages, i.e., adults, elderly, children, males and females. In a second step, creating different textures for each template allows additional variation in age and appearance. Finally, the colors of skin, hair, and clothes are varied per texture, as described in [3]. To further vary appearance for virtual humans in the vicinity of the camera, it is possible to add fine details like make-up, freckles and beard. Also, by controlling shading parameters per body part, as proposed in [4], allows to vary the materials used for the clothes, as for example to model shiny clothes. Thus, with a small number of templates, it is possible to synthesize a large number of virtual human instances that seem unique. Under some conditions recently studied in [5], it is possible to make the use of templates practically undetectable for a spectator. Further varying the shape of instantiated virtual humans can be achieved using accessories, i.e., simple items like hats or more complex accessories like cell phones or shopping bags that require upper body variations only.

The goal of the real-time crowd visualizer is to render a large number of entities according to the current simulation state, which provides the position, orientation, and velocity for each individual. System constraints include believability, real-time updates (25 frames per second) and the number of digital actors ranging in the tens of thousands. We make the population believable by varying the appearance (textures and colors) and animation of the individuals. Their graphical representation is derived from a template, which holds all the possible variations.

Thus, with only a limited set of such templates, we can achieve a varied crowd, leading to considerable time savings for designers.

A template is defined as:

- A set of three meshes with decreasing complexity (LODs),
- A set of textures in gray scale (except for the skin) identifying color modulation areas (pants, shirt, hair etc.),
- A skeleton (kinematic structure),
- A corresponding animation database as skeletal orientations (here 1,000 different walk cycles are generated using a motion blending-based loco-motion engine [6, 7]).

Each human in the visualization system is called an instance and is derived from a template. Individualization comes from assigning a specific gray scale texture and a color combination for each identifiable region. Instances have individualized walk velocities and are animated by blending the available walk animations.

The rendering pipeline advances consecutively in four steps. The first one consists of culling, that is, determining visibility, and choosing the rendering fidelity for each simulated human. By re-using the information stored in the navigation graph of the simulation system, this task is not done for each individual but at the vertex level, thereby determining fidelities for a whole subset of characters at once.

During this phase, humans are distributed in three different groups according to their fidelity level, which ensures efficient batched rendering. The next step of the pipeline is the rendering of dynamic meshes. This is the most detailed fidelity where the animation is obtained by interpolation of skeletal postures. According to the current instance state (linear and angular walk velocities and time), animations are retrieved from the database and interpolated, yielding a smooth animation, with continuous variations of velocities, and no foot-sliding. The resulting skeletal posture is sent to a hardware vertex shader and fragment shader deforming and rendering the human on the graphics card.

Static meshes (also called baked or predeformed) constitute the second rendering fidelity, which keeps a pre-transformed set of animations using the lowest resolution mesh of the deformed mesh in the previous step. Pre-computing deformations allows substantial gains in speed, but constrains the animation variety and smoothness.

The final rendering fidelity is the billboard model which, compared to previous approaches, uses a simplified scheme of sampling and lighting. World-aligned billboards are used, with the assumption that the camera will never hover directly above the crowd. Thus, only sampled images around the waist level of the character are needed. In our case, the templates are sampled at 20 different angles, for each of the 25 key frames composing a walk animation. When constructing the resulting texture, the bounding box of each sampled frame is detected to pack them tightly together. When rendering bill-boarded pedestrians, a specificity of our technique is to apply cylindrical lighting instead of using normal maps: each vertex normal is set to point in the positive Z direction, plus a small offset on the X axis, so that it points slightly outside the frame. We then interpolate the light intensity for each pixel in the fragment shader. Figure 6.2 shows an example of the distribution of the three fidelities.

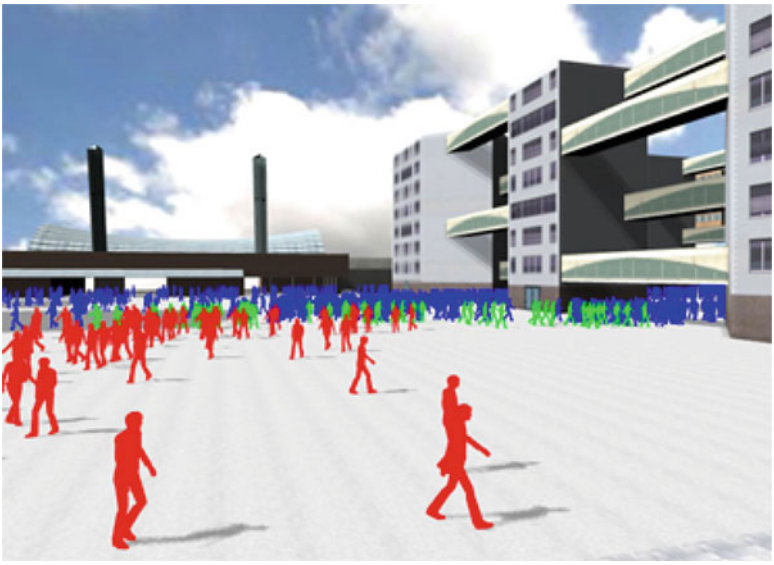

Fig. 6.2 The three different rendering levels of detail: deformed meshes in *front*, rigid meshes in the *middle*, and billboards *behind*

6.3 Adding Variety in Appearance and Animation of Virtual Humans

6.3.1 Introduction

Our main interest is focused on real-time applications where the visual uniqueness of the characters composing a crowd is paramount. On the one hand, it is required to display several thousands of virtual humans at high frame rates, using levels of detail. On the other hand, each character has to be different from all others, and its visual quality highly detailed. Instantiating many characters from a limited set of human templates lead to the presence of multiple similar characters everywhere in the scene. However, the creation of an individual mesh for each character is not feasible, for it would have too high requirements in terms of design and memory. Thus, methods have to be introduced to modify each instance, so that it is visually different from all the others. Such methods also need to be scalable for all LOD used in crowd simulations to avoid inconsistencies in the individual appearances. Our main contribution is the introduction of techniques to improve the variety of crowds in three domains: visual appearance, shape, and animation. More details may be found in [11].

6.3.2 Appearance Variety

6.3.2.1 Color Variety

Previous work on color variety is based on the idea of dividing a human template into several body parts, identified by specific intensities in the alpha channel of the template texture. At runtime, each body part of each character is assigned a color in order to modulate the texture. Tecchia et al. [8] used several passes to render each impostor body part. Dobbyn et al. [3] extended the method to 3*D* meshes and avoided multi-pass rendering with programmable graphics hardware. Although these methods offer nice results from a reasonable distance, they produce sharp transitions between body parts. Based on the same idea, Gosselin et al. [9] showed how to vary characters with the same texture by changing their tinting. They also presented a method to selectively add decals to the characters' uniforms. However, their approach is only applied to armies of similar characters, and the introduced differences are not sufficient when working with crowds of civilians.

Using a single alpha layer to segment body parts has several drawbacks. No bi-linear filtering can be used on the texture, because incorrect interpolated values would be fetched in the alpha channel at body part borders. Moreover, for individuals close to the camera, the method tends to produce too sharp transitions between body parts, e.g., between skin and hair, due to the impossibility of associating a texel to several body parts at the same time. Also, character close-ups bring the need for a new method capable of handling detailed color variety. Subtle make-up, or detailed patterns on clothes greatly increase the variety of a single human template. Furthermore, changing illumination parameters of materials, e.g., their specularity, provides more realistic results. Previous methods would require costly fragment shader branching to achieve such effects. We apply a versatile solution based on segmentation maps to overcome previous method drawbacks.

For each texture of a human template, we create a series of segmentation maps. Each of them is an RGBA image, delimiting four body parts, i.e., one per channel, and sharing the same parameterization as the human template texture. This method allows for each texel to partially belong to several body parts at the same time through its channel intensities. As a result, it is possible to design much smoother transitions between body parts than in previous approaches. Figure 6.3 shows the principles of color variety.

6.3.2.2 Height and Shape Variety

Magnenat-Thalmann et al. [10] classified the methodologies for modeling virtual people into three major categories: creative, reconstructive, and interpolated. Geometric models created by artists such as anatomically based models fall into the former approach. The second major category built 3D virtual human's geometry by capturing existing shape from 3D scanners, images and even video sequences.

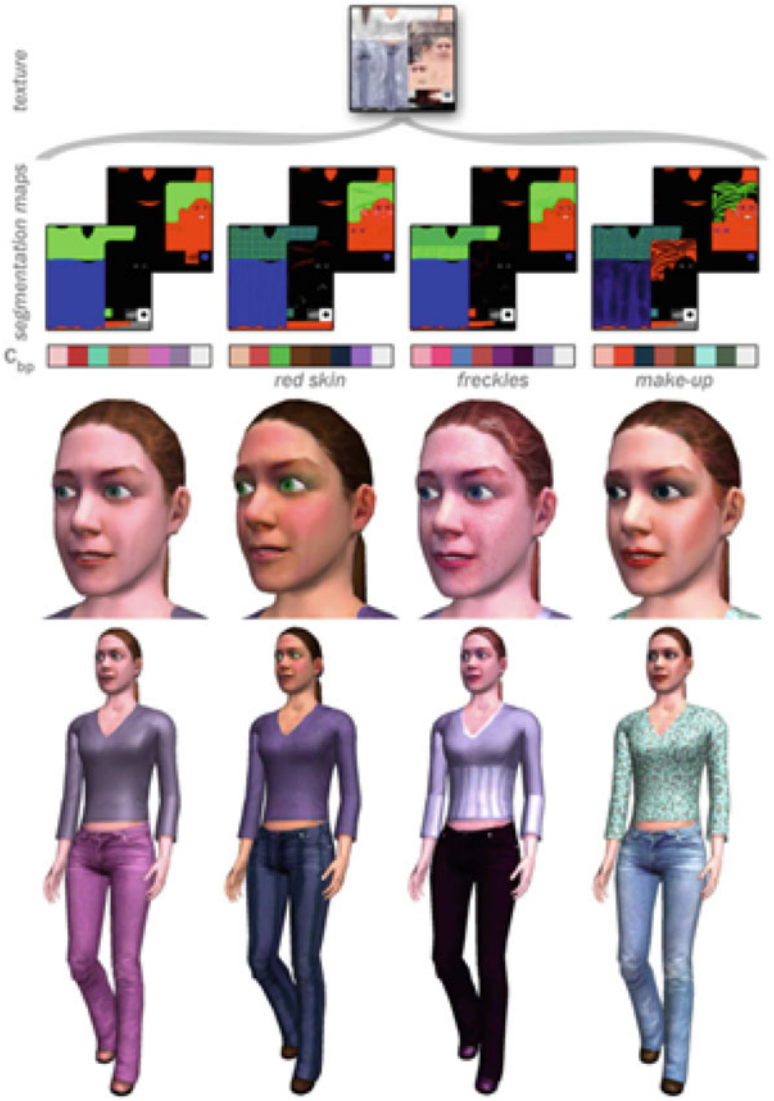

Fig. 6.3 Principles of color variety using segmentation maps

The interpolated modeling uses sets of example models with an interpolation scheme to reconstruct new geometric models. For crowds, the first approach is too expensive in terms of manual work. The second way is also prohibitive for large crowds and also presents a lack of flexibility. The last approach is the most convenient. For large crowds, another approach consists of modifying separately the height of the human body and its shape.

Fig. 6.4 Variation of height

We can modify the height of a human template, by scaling its skeleton. To help the designer in this task, we provide additional functionalities to the design tool presented above: for a given human template skeleton, the global space of height scaling can be defined. Fine-grained local tuning for each joint can also be specified, i.e., minimal and maximal scale parameters on the x, y, and z world axes. These data allow several different skeletons to be generated from a single template, which we call the meta-skeleton. Figure 6.4 shows an example. For each new skeleton, a global scale factor is randomly chosen within the given range. Then, the associated new scale for each of its bones is deduced. Short/tall skeletons mixed with broad/narrow shoulders are thus created. The skin of the various skeletons also needs adaptation. Each vertex of the original template is displaced by each joint that influences it. For the shape, the human mesh is modified using three steps:

1. Using a commercial 3*D* package like 3DSMax, it is possible for a designer to paint a FatMap for a given template, as seen in Fig. 6.5. The FatMap is an extra gray-scale UV texture that is used to emphasize body areas that store fat. Darker areas represent regions where the skin will be most deformed, e.g., on the belly, and lighter areas are much less deformed, like the head. When the creation of the FatMap is complete, the gray scale values at each texel are used to automatically infer one value for each vertex of the template's mesh. Each of these values, called a fatWeight, is attached to the vertex as an additional attribute.

Fig. 6.5 FatMaps designed in 3DSMax. *Dark areas* represent regions more influenced by fat or muscles modification, while *lighter* parts are less affected

2. The next step is to compute in which direction the vertices are moved when scaled. For this, we compute the scaling direction of each vertex as the weighted normal of the bones influencing it.
3. Once the direction of the body scaling is computed for each vertex, the actual scaling can take place. The extent to which we scale the body is defined by a fatScale, randomly chosen within a pre-defined range.

Figure 6.6 shows the same character with various heights and shapes.

6.3.2.3 Accessories

Accessorizing crowds [21] offers a simple and efficient alternative to costly human template modeling. Accessories are small meshes representing elements that can easily be added to the human template original mesh. Their range is considerable, from subtle details, like watches, jewelry, or glasses, to larger items, such as hats, wigs, or backpacks, as illustrated in Fig. 6.7. Distributing accessories to a large crowd of a few human templates varies the shape of each instance, and thus makes it unique. Similar to deformable meshes, accessories are attached to a skeleton and follow its animation when moving.

Fig. 6.6 Various body shapes and heights

Fig. 6.7 Population with accessories: bags, hats, glasses

The first group of accessories does not necessitate any particular modification of the animation clips played. They simply need to be correctly placed on a virtual human. Each accessory can be represented as a simple mesh, independent from any virtual human. First, let us outline the problem for a single character. The issue is to render the accessory at the correct position and orientation, accordingly to the movement of the character. To achieve this, we can attach the accessory to a specific

joint of the virtual human. Let us take a real example to illustrate our idea: imagine a walking person wearing a hat. Suppose that the hat has the correct size and does not slide, it basically has the same movement as the head of the person as he walks.

The second group of accessories we have identified is the one that requires slight modifications to the played animation sequences, e.g., the hand close to the ear to make a phone call, or a hindered arm sway due to carrying a heavy bag. Concerning the rendering of the accessory, we still keep the idea of attaching it to a specific joint of the virtual human. The additional difficulty is the modification of the animation clips to make the action realistic. We only focus on locomotion animation sequences. There are two options to modify an animation related to an accessory:

- If we want a virtual human to carry a bag for instance, the animation modifications are limited to the arm sway, and maybe a slight bend of the spine to counterweight the bag. The motion is restricted, in this case, we clamp the joints defining the limits of the joint angle (minimum angle, maximum angle).
- If it is a cell phone accessory that we want to add, we need to keep the hand of the character close to its ear and avoid any collision over the whole locomotion cycle. The motion is blocked and the angle is freezed to a certain value (Freezing Angle).

At runtime, the animation is updated as usual, the frozen joints are overwritten, and we use exponential maps to clamp joints.

6.3.2.4 How to Vary the Animation of Characters?

Animation is an another important factor that determines crowd heterogeneity. If they all perform the same animation, the results are not realistic enough [3]. We have implemented three techniques to vary the animation of characters, while remaining in the domain of navigating crowds, i.e., working with locomotion animations:

We introduce variety in the animation by generating a large amount of locomotion cycles (walking and running), and idle cycles (like standing, talking, sitting, etc.), that we morphologically adapt for each template. We take care to make these animations cyclic, and categorize them in the database, according to their type: sitting or standing, talking or listening, etc. For locomotion clips, walk and run cycles are generated from a locomotion engine based on motion capture data (see Sect. 6.3.2.5). We compute such locomotion clips for a set of speeds. Thus, during real-time animation, it is possible to directly obtain an adequate animation for a virtual human, given its current locomotion velocity, and its morphological parameters. Then, we designed the concept of motion kit, a data structure, that efficiently handles animations at all levels of detail (LOD).

We use a second technique of animation variety, i.e., how pre-computed animation cycles can be augmented with upper-body variations, like having a hand on the hip, or in a pocket.

Fig. 6.8 Locomotion generated using PCAs

Finally, we introduced procedural modifications applied at runtime on locomotion animations to allow crowds to wear complex accessories as mentioned in the previous section.

6.3.2.5 Locomotion Animation Clips

To generate our original set of walk and run cycles, we use the locomotion engine developed by Glardon et al. [6]; it is an integrated walking and running engine able to extrapolate data beyond the space described by the PCA basis. In this approach, the Principal Component Analysis (PCA) method is used to represent the motion capture data in a new, smaller space. As the first Principal Components contain the most variance of the data, an original methodology is used to extract essential parameters of a motion. This method decomposes the PCA in a hierarchical structure of sub-PCA spaces. At each level of the hierarchy, an important parameter of a motion is extracted and a related function is elaborated, allowing not only motion interpolation but also extrapolation. Figure 6.8 shows an example.

There are mainly three high-level parameters which allow to modulate these motions:

- Personification weights: several people, different in height and gait have been captured while walking and running. This variable allows the user to choose how he wishes to parametrize these different styles.

- Speed: the subjects have been captured at many different speeds. This parameter allows to choose at which velocity the walk/run cycle should be generated.
- Locomotion weights: this parameter defines whether the cycle is a walk or a run animation. Thus, the engine is able to generate a whole range of varied locomotion cycles for a given character. Each human template is also assigned a particular personification weight so that it has its own gait. With such a high number of animations, we are already able to per-ceive a sense of variety in the way the crowd is moving. Virtual humans walking together with different locomotion styles and speeds add to the realism of the simulation.

6.4 Motion Planning for Large-Scale Crowds

6.4.1 Introduction

Realistic real-time motion planning for crowds has become a fundamental research field in the Computer Graphics community. The simulation of urban scenes, epic battles, or other environments that show thousands of people in real time require fast and realistic crowd motion. Domains of application are vast: video games, psychological studies, and Architecture to name a few. In this section, we first focus on the Navigation Graph approach. A Navigation Graph is a simple structure that represents an environment topology by distinguishing navigable are-as from impassable obstacles. We then briefly discuss the continuum crowd concept. We finally present our motion planning architecture, offering a hybrid and scalable solution for real-time motion planning of thousands of characters in complex environments.

6.4.2 Navigation Graphs

Real-time crowd motion planning requires fast, realistic methods for path planning as well as obstacle avoidance. The difficulty to find a satisfying trade-off between efficiency and believability is particularly challenging, and prior techniques tend to focus on a single approach [12, 13]. We have presented [14, 15] a novel approach to automatically extract a topology from a scene geometry and handle path planning using a navigation graph. Figure 6.9 shows a crowd moving using navigation graphs. The main advantage of this technique is that it handles uneven and multi-layered terrains. Nevertheless, it does not treat inter-pedestrian collision avoidance. Given an environment geometry, a navigation graph can be computed [14, 21]: the vertices of the graph represent circular zones where a pedestrian can walk freely without the risk of colliding with any static object in the environment. Graph edges represent connections between vertices. In the environment, they are viewed as intersections

Fig. 6.9 Crowd moving

(or gates) between two circular zones (vertices). From a navigation graph, path requests can be issued from one vertex to another. Using an algorithm based on Dijkstra's, we are able to devise many different paths that join one point of the environment to another one. It is possible to provide the navigation graph with a second model of the environment, which usually has a much more simple geometry, annotated with information. This second model is automatically analyzed, its meta-information retrieved, and associated to the corresponding vertices. Figure 6.9 shows an example.

6.4.3 Scalable Motion Planning

More recently, Treuille et al. [16] proposed efficient motion planning for crowds. Their method produces a potential field that provides, for each pedestrian, the next suitable position in space (a waypoint) to avoid all obstacles. Compared to agent-based approaches, these techniques allow simulating thousands of pedestrians in real time, and are also able to show emergent behaviors. However, they produced less believable results, because they require assumptions that prevent treating each pedestrian with individual characteristics. For instance, only a limited number of goals can be defined and assigned to groups of pedestrians. The resulting performance depends on the size of the grid cells and the number of groups.

6.4.4 An Hybrid Architecture Based on Regions Of Interest (ROI)

We proposed a hybrid architecture [17] to handle the path planning of thousands of pedestrians in real time, while ensuring dynamic collision avoidance. The scalability of our approach allows to interactively create and distribute regions of varied interest, where motion planning is ruled by different algorithms. Practically, regions of high interest are governed by a long-term potential field-based approach, while other zones exploit a graph of the environment and short-term avoidance techniques. Our method also ensures pedestrian motion continuity when switching between motion planning algorithms. Tests and comparisons show that our architecture is able to realistically plan motion for many groups of characters, for a total of several thousands of people in real time, and in varied environments.

The goal of our architecture is to handle thousands of pedestrians in real time. We exploit the vertex structure described in Sect. 6.4.1 to divide the environment into regions ruled by different motion planning techniques. Regions of interest (ROI) can be defined in any number and anywhere in the walkable space with high-level parameters, modifiable at runtime.

By defining three different ROI, we obtain a simple and flexible architecture for realistic results: ROI 0 is composed of vertices of high interest, ROI 1 regroups vertices of low interest, and ROI 2 contains all other vertices, of no interest. For regions of no interest (ROI 2), path planning is ruled by the navigation graph. Pedestrians are linearly steered to the list of waypoints on their path edges. To use the minimal computation resources, obstacle avoidance is not handled. Path planning in regions of low interest (ROI 1) is also ruled by the navigation graph. To steer pedestrians to their waypoints, an approach similar to Reynolds' [18] is used, and obstacles are avoided with an agent-based short-term algorithm. Although agent-based, this algorithm works at low level, and thus stays simple and efficient. In the regions of high interest (ROI 0), path planning and obstacle avoidance are both ruled by a potential field-based algorithm, similar to Treuille et al. [16]. Figure 6.10 shows a crowd moving using the hybrid path planning algorithm.

6.5 Controlling Individual and Group Behavior in Crowds

6.5.1 Introduction

To make believable a crowd simulation is not just to generate many various individual characters. Modelling the behavior of the individuals, the groups, and even the crowd itself is very important. An important aspect is to give the feeling that people are aware of the environment and the other people. For this objective, the role of gaze control is essential. Group behavior is also a key issue. For example, in

Fig. 6.10 Crowd using hybrid path planning

real cities, many pedestrians are part of a group, whether they are sitting, standing, or walking toward their shared goal. They behave differently than if they were alone: they adapt their pace to the other members, wait for each other, may get separated in crowded places to avoid collisions, but regroup afterwards. Finally, we explain in this section a method to generate unlimited cities or streets sing crowd patches.

6.5.2 Gaze Control

We can improve the realism of a crowd simulation by allowing its pedestrians to be aware of their environment and of the other characters present in this environment. They can even seem to be aware of a user interacting with this environment. We introduced [19] the various setups which allow for crowd characters to gaze at environment objects, other characters or even a user. Finally, we developed a method to add these attention behaviors in order for crowd characters to seem more individual.

The first step is to define the interest points, i.e. the points in space which we consider interesting and which therefore attract the characters' attention. We use several different methods to do this depending on the result we want to obtain:

Fig. 6.11 An example depicting the types of possible gaze behaviors

- The interest points can be defined as regions in space which have been described as interesting. In this case, they will be static.
- They can be defined as characters evolving in space. All characters may then potentially attract the attention of other characters as long as they are in their field of view. In this case, we have dynamic constraints, since the characters move around.
- They can be defined as a user if we track a user interacting with the system. A coupled head- and eye-tracking setup allows us to define the position of the user in the 3D space. Characters may then look at the user.

The second step to obtain the desired attention behaviors consists of computing the displacement map which allows for the current character posture to achieve the gaze posture, i.e. to satisfy the gaze constraints. Once the displacement map has been computed, it is dispatched to the various joints composing the eyes, head, and spine in order for each to contribute to the final posture. Finally, this displacement is propagated in time in order for the looking or looking away motions to be smooth, natural, and human-like. Figure 6.11 shows virtual humans with gaze.

6.5.3 Group Behavior

The behavior of people in a crowd is a fascinating subject: crowds can be very calm but also rise to frenzy, they can lead to joy but also to sorrow. It is quite a common idea that people not only behave differently in crowd situations, but that they undergo some temporary personality change when they form part of

a crowd. Most writers in the field of mass- or crowd-psychology agree that the most discriminating property of crowd situations is that normal cultural rules, norms and organization forms cease to be applicable For instance in a panic situation the normal rule of waiting for your turn, and the concomitant organization form of the queue, are violated and thus become obsolete.

In Musse et al. [20], the model presents a simple method for describing the crowd behavior through the group inter-relationships. Virtual actors only react in the presence of others, e.g., they meet another virtual human, evaluate their own emotional parameters with those of the other one and, if they are similar, they may walk together. The group parameters are specified by defining the goals (specific positions which each group must reach), number of autonomous virtual humans in the group and the level of dominance of each group. This is followed by the creation of virtual humans based on the groups' behavior information. The individual parameters are: a list of goals and individual interests for these goals (originated from the group goals), an emotional status (random number), the level of relationship with the other group members (based on the emotional status of the agents from a same group) and the level of dominance (which follows the group trend). With these rules, we can model the following sociological effects:

- *Grouping* of individuals depending on their inter-relationships and the *domination* effect;
- *Polarization* and the *sharing* effects as the influence of the emotional status and domination parameters; and finally,
- *Adding* in the relationship between autonomous virtual humans and groups.

The group behavior is formed by two behaviors: seek goal, that is the ability of each group to follow the direction of motion specified in its goals, e.g. in the case of a visit to a museum, the agents walk in the sense of its goals; and the flocking (ability to walk together), has been considered as a consequence of the group movement based on the specific goals during a specific time.

Generally, the available computational resources to trigger intelligent behaviors are very limited in crowds, because their navigation, animation, and rendering are already very expensive tasks that are absolutely paramount. Our approach to this problem is to find a trade-off that simulates intelligent behaviors, while remaining computationally cheap. In [10], we describe the various experiments performed to improve pedestrians behaviors. To make crowd movements more realistic, a first important step is to identify the main places where many people tend to go, i.e., places where there is a lot of pedestrian traffic. It can be a shopping mall, a park, a circus, etc. Adding meta-information to key places in an environment has been achieved in many different ways. Our approach is to use the navigation graph of an environment to hold this meta-information, which is a very advantageous solution: instead of tagging the meshes of an environment, or creating a new dedicated informational structure, we directly work on the structure that is already present, and which is used for path planning and pedestrian steering.

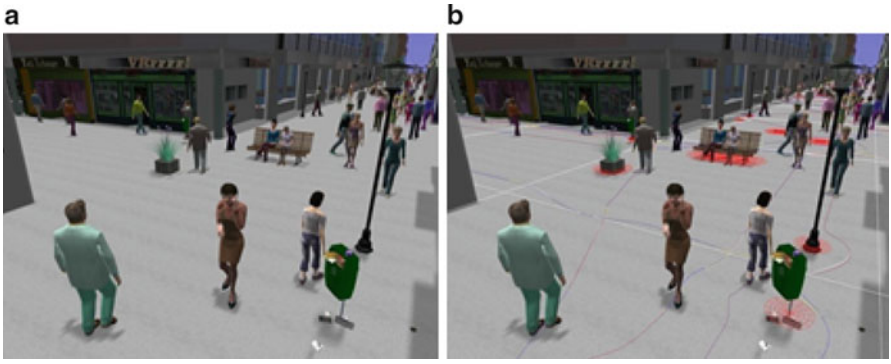

Fig. 6.12 *Left*: A procedurally computed pedestrian street, where patches are generated at runtime. *Right*: The same image revealing the patch borders and their trajectories. (**a**) Density profile. (**b**) Velocity profile

6.5.4 Crowd Patches

We break classical crowd simulation limitations on the environment dimensions: instead of pre-computing a global simulation dedicated to the whole environment, we independently pre-compute the simulation of small areas, called crowd patches [21]. To create virtual populations, the crowd patches are interconnected to infinity from the spectator's point of view. We also break limitations on the usual durations of pre-computed motions: by adapting our local simulation technique, we provide periodic trajectories that can be replayed seamlessly and endlessly in loops over time.

Our technique is based on a set of patch templates, having specific constraints on the patch content, e.g., the type of obstacles in the patch, the human trajectories there, etc. A large variety of different patches can be generated out of a same template, and then be assembled according to designers' directives.

Patches can be pre-computed to populate the empty areas of an existing virtual environment, or generated online with the scene model. In the latter case, some of the patches also contain large obstacles such as the buildings of a virtual city.

Patches are geometric areas with convex polygonal shapes. They may contain static and dynamic objects. Static objects are simple obstacles with its geometry is fully contained inside the patch. Figure 6.12 shows an example.

Larger obstacles, such as buildings, are handled differently. Dynamic objects are animated: they are moving in time according to a trajectory $\tau(t)$. In this context, we want all dynamic objects to have a periodic motion (of period π) in order to be seamlessly repeated in time.

Two categories of dynamic objects may be distinguished: endogenous and exogenous objects. The trajectory of endogenous objects remains inside the geometrical limits of the patch for the whole period. The point's trajectory is fully contained in

the patch and respects the periodicity condition (1). If the animation is looped with a period π, the point appears to be moving endlessly inside the patch. Note that static objects can be considered as endogenous objects, with no animation.

Exogenous objects have a trajectory $\tau(t)$ that goes out of the patch borders at some time, and thus, does not meet the periodicity condition (1). In order to en-force this condition, we impose the presence of another instance of the same exogenous object whose trajectory is $\tau^{'}(t)$. As the two objects are of the same type, i.e., they have an identical kinematics model, their trajectories can be directly compared. Different cases are then to be distinguished and are discussed in [17].

We build environments and their population by assembling patches. Thus, two adjacent patches have at least one common face. They also share identical limit conditions for exogenous objects' trajectories. Indeed, when an exogenous object goes from one patch to an adjacent one, it first follows the trajectory contained by the first patch, and then switches to the one described by the second patch. These two trajectories have to be at least continuous $C0$ to ensure a seamless transition from the first patch to the second one. The patterns between the two adjacent patches allow to share these limit conditions.

6.6 Applications

We can conclude this chapter with a few applications and the challenges associated to them.

6.6.1 Virtual Heritage

Based on archaeological data, we have presented the different steps of our work to generate the ancient city of Pompeii and populate it with virtual romans [22] (see Fig. 6.13). Thanks to the semantic data labeled in the geometry, crowds are able to exhibit particular behaviors relative to their location in the city. Our results and show that we are able to simulate several thousands of virtual characters in the reconstructed city in real-time. The use of a procedural technique for the creation of city models has proven to be very flexible and allows for quick variations and tests not possible with manual editing techniques.

One possible challenging work would be to make the Virtual Romans interact with the model, e.g., opening doors. This would allow creating more intelligent and varied behaviors for crowds.

Fig. 6.13 Roman crowd in Pompeii

6.6.2 Agoraphobia Treatment

We developed an application allowing for characters to perform gazing motions in a real-time virtual crowd in a CAVE environment [23]. Moreover, it allows for users to interact with those crowd characters. It is an adaptation of the model of visual attention described in [16] in order to integrate it in a crowd engine and al-low for the method to function online (in real-time). Certain aspects of the automatic interest point detection have been greatly simplified. The existing architecture has been also modified in order to abide with the limitations induced by the real-time implementation.

The final application consists in a city scene, projected in a CAVE setup, in which a crowd of characters walks around (see Fig. 6.14). The application uses a Phase-space optical motion capture device to evaluate where a user is looking and more specifically, which character he/she is looking at. Finally, we further enhance this setup with an RK-726PCI pupil/corneal reflection tracking device in order to evaluate more precisely where a user is looking. The system then allows for the crowd characters to react to user gaze. For example, since we can determine the user's position and orientation in the virtual world, the characters can look at the user.

6.6.3 Transportation and Urbanism

Virtual crowds are used for simulation of new train stations and airports. A challenge would be to introduce natural motivations to simulate more complex and realistic

Fig. 6.14 Agoraphobia treatment

situations. For example, in an airport, people should not just check in, go to the security then the gate, as in most simulations. They should be able to go to restaurants, cafes, shops, toilets, according to their internal motivations. Such models exist, but the problem is that it will be extremely CPU intensive to introduce them.

Acknowledgements Most of this research has been performed at the VRlab in EPFL, directed by the first author.

References

1. Musse, S.R., Thalmann, D.: A model of human crowd behavior. In: Proceedings of the Eurographics Workshop on Computer Animation and Simulation '97, Budapest, pp. 39–51. Springer, Wien (1997)
2. Thalmann, D., Musse, S.R.: Crowd Simulation, 2nd edn. Springer, London (2012)
3. Dobbyn, S., Hamill, J., O'Conor, K., O'Sullivan, C.: Geopostors: a realtime geometry/impostor crowd rendering system. In: SI3D '05: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, New York, pp. 95–102. ACM (2005)
4. Maïm, J., Yersin, B., Pettré, J., Thalmann, D.: YaQ: an architecture for real-time navigation and rendering of varied crowds. IEEE Comput. Grap. Appl. **29**(4), 44–53 (2009)
5. McDonnell, R., Larkin, M., Dobbyn, S., Collins, S., O'Sullivan, C.: Clone attack! per-ception of crowd variety. ACM Trans. Graph. **27**(3), 1–8 (2008)
6. Glardon, P., Boulic, R.Thalmann, D.: Robust on-line adaptive footplant detection and enforcement for locomotion. Vis. Comput. **22**(3), 194–209 (2006)
7. Glardon, P., Boulic, R., Thalmann, D.: PCA-based walking engine using motion capture data. In: Proceedings of the Computer Graphics International. IEEE Computer Society, Washington, DC, USA (2004)

8. Tecchia, F., Loscos, C., Chrysanthou, Y.: Visualizing crowds in real-time. Comput. Graph. Forum **21**(4), 753–765 (2002)
9. Gosselin, D., Sander, P.V., Mitchell, J.L.: Drawing a crowd. In: Engel, W. (ed.) ShaderX3: Advanced Rendering Techniques in DirectX and OpenGL. Charles River Media, Cambridge (2004)
10. Magnenat-Thalmann, N., Seo, H., Cordier, F.: Automatic modeling of virtual humans and body clothing. In: Proceedings of SIGGRAPH ACM, New York, pp. 19–26 (2003)
11. Yersin, B., Maïm, J., Thalmann, D.: Unique instances for crowds. IEEE Comput. Graph. Appl. **29**(6), 82–90 (2009)
12. Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. Comput. Graph. Forum **23**(3), 509–518 (2004)
13. Pelechano, N., Allbeck, J., Badler, N.: Controlling individual agents in high-density crowd simulation. In: SCA '07, ACM/Eurographics, NY and Geneva (2007)
14. Pettré, J., de Heras Ciechomski, P., Maïm, J., Yersin, B., Laumond, J.-P., Thalmann, D.: Real-time navigating crowds: scalable simulation and rendering. J. Vis. Comput. Animat. **17**(3–4), 445–455 (2006)
15. Pettre, J., Grillon, H., Thalmann, D.: Crowds of moving objects: navigation planning and simulation. In: Proceedings of IEEE International Conference on Robotics and Automation. IEEE Computer Society, Washington, DC, USA, pp. 3062–3067 (2007)
16. Treuille, A., Cooper, S., Popovic, Z.: Continuum crowds. In: Proceedings of the SIGGRAPH 2006, ACM, New York, USA, pp. 1160–1168 (2006)
17. Morini, F., Yersin, B., Maïm, J., Thalmann, D.: Real-time scalable motion planning for crowds. Vis. Comput. **24**(10), 859–870 (2008)
18. Reynolds, C.W.: Steering behaviors for autonomous characters. In: Proceedings of Game Developers Conference, San Jose, pp. 763–782 (1999)
19. Grillon, H., Thalmann, D.: Simulating gaze attention behaviors for crowds. Comput. Animat. Virtual Worlds **3–4**, 111–119 (2009)
20. Musse, S.R., Thalmann, D.: A hierarchical model for real time simulation of virtual human crowds. IEEE Trans. Vis. Comput. Graph. **7**(2), 152–164 (2001)
21. Yersin, B., Maïm, J., Pettré, J., Thalmann, D.: Crowd patches: populating large-scale virtual environments for real-time applications. Proceedings of I3D, ACM, New York (2009)
22. Maïm, J., Haegler, S., Yersin, B., Mueller, P., Thalmann, D., Van Gool, L.: Populating ancient pompeii with crowds of virtual romans. Proceedings of the VAST 2007, Eurographics Association, Geneva, pp. 109–116 (2007)
23. Peternier, A., Cardin, S., Vexo, F., Thalmann, D.: Practical design and implementation of a CAVE environment. In: Proceedings of the 2nd International Conference on Computer Graphics, Theory and Applications GRAPP 2007, Barcelona (2007)