Massive Autonomous Characters: Animation and Interaction

Ingu Kang[1,2] and JungHyun Han[1,⋆]

[1] Game Research Center, College of Information and Communications
Korea University, Seoul, Korea
[2] Nexon Co., Ltd.

Abstract. This article reports the result of an experiment which integrates GPU-accelerated skinning, sprite animation, and character behavior control. The experiment shows that the existing techniques can be neatly integrated: thousands of characters are animated at real-time and the overall motion is natural like fluid. The result is attractive for games, especially where a huge number of non-player characters such as animals or monsters should be animated.

Keywords: game, character animation, skinning, impostor, behavior control, GPU.

1 Introduction

Animation of large crowds is an area of research that has been receiving an increased amount of interest. Especially, it is becoming essential in multi-player online games that can accommodate a huge number of simultaneous players or non-player characters (NPCs). A good example of NPCs is a herd of land animals. Such a herd is made up of autonomous discrete animals, but the overall motion seems fluid. Reynolds observed that the motion is the aggregate result of the actions of individual animals, each acting solely on the basis of its own local perception of the world[1].

This article reports an experiment result for real-time interaction with autonomous discrete NPCs. For rendering a large number of NPCs, the multi-resolution technique proposed by Kang *et al.*[2] is used. The group of NPCs responds to the player character's interaction, as well as to each other and their environment. For controlling the group behavior, the mental models proposed by Reynolds[1,3] have been adopted, which mediate between several conflicting behavioral goals.

The experiment results show that the two existing techniques can be neatly integrated. Such integration is attractive for real-time graphics applications such as games, especially where a huge number of NPCs including animals or monsters should be rendered.

⋆ Corresponding author.

P. Kalra and S. Peleg (Eds.): ICVGIP 2006, LNCS 4338, pp. 333–338, 2006.

2 Rendering of Large Crowds

This section summarizes the multi-resolution rendering technique for large crowds, proposed by Kang *et al.*[2]. Note that an NPC in close proximity with a player character can interact with the player, and the interaction may cause various actions of the NPC. However, an NPC at a distance does not interact with the player character, and performs a limited set of actions. Therefore, different levels of detail in NPC animation are used: *skinning animation*[4] for NPCs at close proximity as well as all player characters, and *impostors* (animated sprites)[6,7] for distant NPCs.

Skinning has been the dominant approach to character animation in real-time applications. The skinning algorithm is based on a hierarchy of bones, and each vertex in the mesh is assigned a set of influencing bones and a blending *weight* for each influence. Then, the deformed vertex position v_c for a configuration c is computed as follows:

$$v_c = \sum_{i=1}^{n} w_i M_{i,c} M_{i,d}^{-1} v_d \tag{1}$$

where w_i is the weight, v_d is dress-pose location of the vertex, $M_{i,c}$ is the transformation matrix associated with the i-th influence in configuration c, and $M_{i,d}^{-1}$ is the inverse of the dress-pose matrix associated with the i-th influence.

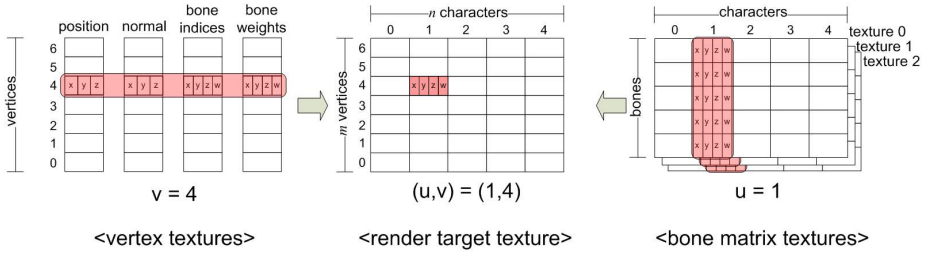

Fig. 1. Skinning and render target texture (from [2])

The four 1D textures in the left side of Fig. 1 show the skinning data for a vertex. A vertex is influenced by 4 bones, and the bone matrices are computed every frame. As shown in the right side of Fig. 1, each row of the 3×4 matrix is recorded in a separate texture. Through a single drawcall, all vertices of all characters are transformed, and written into the *render target texture*, as shown in the middle of Fig. 1. In implementation, the vertex shader renders a quad covering the render target, and the pixel shader fills each texel of the render target texture, where a texel corresponds to a vertex of a character. Then, the render target texture is copied to a vertex buffer object (VBO)[5], and then each character is rendered by the vertex shader using a given index buffer.

The GPU-based skinning algorithm is integrated with sprite animation. A sprite's size is 32×32. For each keyframe, sprites are captured from 256 viewpoints in the spherical coordinates: 8 along the longitude in the range $[0,\pi/2]$, and 32 along the latitude in the range $[0,2\pi]$. The set of 256 sprites is stored in a 512×512 texture. For a character, 11 keyframes are used. The impostors are rendered through hardware-accelerated *point sprites*, for rendering of which both OpenGL and DirectX have standard interfaces.

3 Behavior Control

Reynolds[1] suggested a flocking algorithm which takes a bird as a particle. Three steering behaviors are considered: *separation* for avoiding local flock-mates, *alignment* towards the average heading of local flock-mates, and *cohesion* to move towards the average position of local flock-mates. In addition, *avoidance* behavior is included to avoid obstacles or enemies. In his later work[3], Reynolds also suggested to use *lattices* as spatial directories for the speedup purpose.

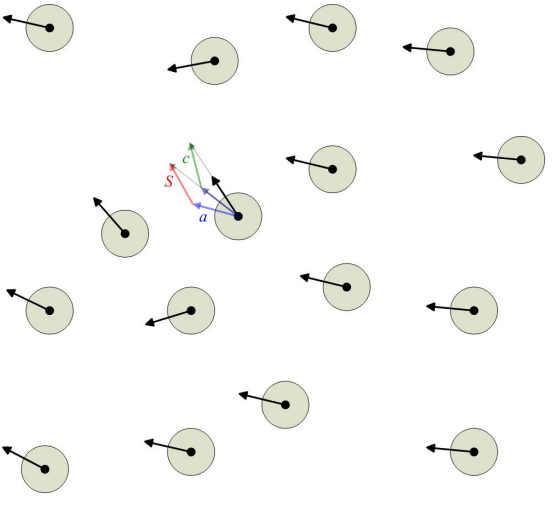

Fig. 2. Vectors for behavior control

In order to control the massive animated NPCs discussed in the previous section, the behavior control algorithm of [1,3] has been adopted. Three steering behaviors (separation, alignment, and cohesion) are used to define the path of each NPC, and the avoidance behavior makes the NPCs flee from the player character and get away from the fences. The four behaviors lead to specific velocity vectors for an NPC, and summed through user-defined weight values. The summed vector determines the combined velocity of an NPC. See Fig. 2 where s, a, and c stand for separation, alignment, and cohesion, respectively.

An NPC's steering behaviors are determined by considering the neighboring NPCs in close proximity. For the sake of the real-time performance, the entire terrain is partitioned into lattices, and only the NPCs in a lattice and neighboring lattices are considered.

In the current experiments, the horses move following the terrain surface, and therefore the moves are restricted basically to 2D. The behavior control algorithm is applicable to 3D space, for example, to determine the behaviors of a flock of birds. Note that, however, behavior control in 2D surface is more difficult than that of 3D space. It is because collisions among the NPCs are harder to handle in the 2D surface. Imagine hundreds of animals in a cage. In the current implementations, the maximum velocity has been set to be high enough for efficient handling of collisions. However, such a high velocity may cause unnatural change of moving directions such as abrupt rotation. The maximum velocity should be properly set according to the features of specific NPCs. For example, the maximum velocity of horses should be greater than that of sheep.

Better performances can be achieved by applying the behavior control algorithm only to the NPCs in close proximity, i.e. the skinning-animated characters. In FPS (first-person shooting) games, for example, the NPCs interacting with the player are those in close proximity, and therefore it is a reasonable choice to apply the behavior control algorithm only to them.

4 Implementation and Result

The algorithms have been implemented in C++, OpenGL and Cg on a PC with 3.2 GHz Intel Pentium4 CPU, 2GB memory, and NVIDIA Geforce 7800GTX 256MB. For experiments, a horse character is used, which is composed of 38 bones, 555 vertices and 1,084 polygons. Table 1 shows how the performance changes in FPS on the ratio of skinning and impostor rendering when 1,000 horses are rendered. Obviously, the rendering performance is improved as the ratio of impostor rendering to skinning is increased. When the behavior control algorithm is executed, the overall FPS is decreased.

Fig. 3 shows snapshots of animating hundreds of horses with behavior control. The three steering behaviors (separation, alignment and cohesion) are implemented. In addition, the human player character is working as an enemy, and causes avoidance behavior of the NPCs (horses). Fig. 4 shows snapshots of

Table 1. Performances

# skinning	# impostor	FPS (no behavior control)	FPS (behavior control)
1000	0	57	52
800	200	70	63
600	400	91	79.5
400	600	124	103.5
200	800	207	155.5

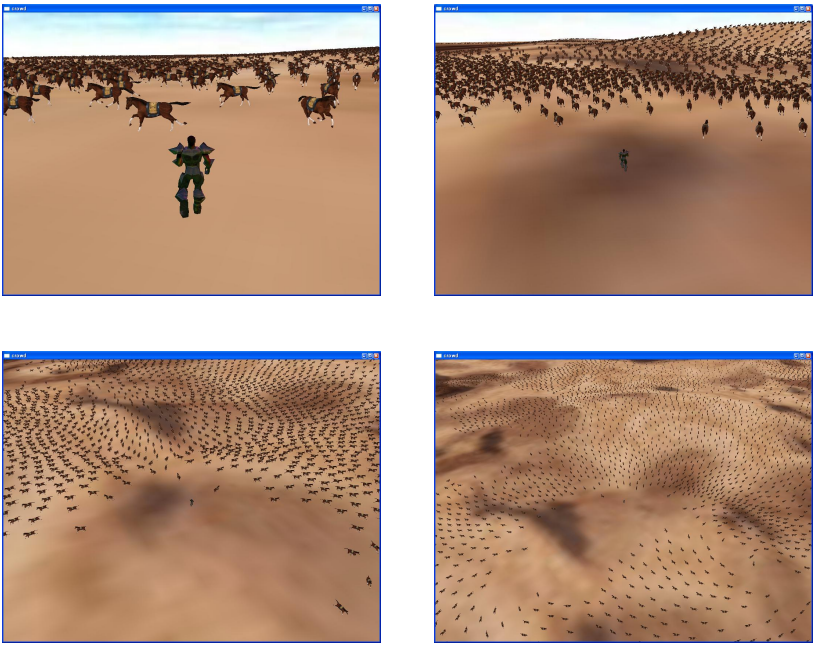

Fig. 3. Large crowd rendering with behavior control

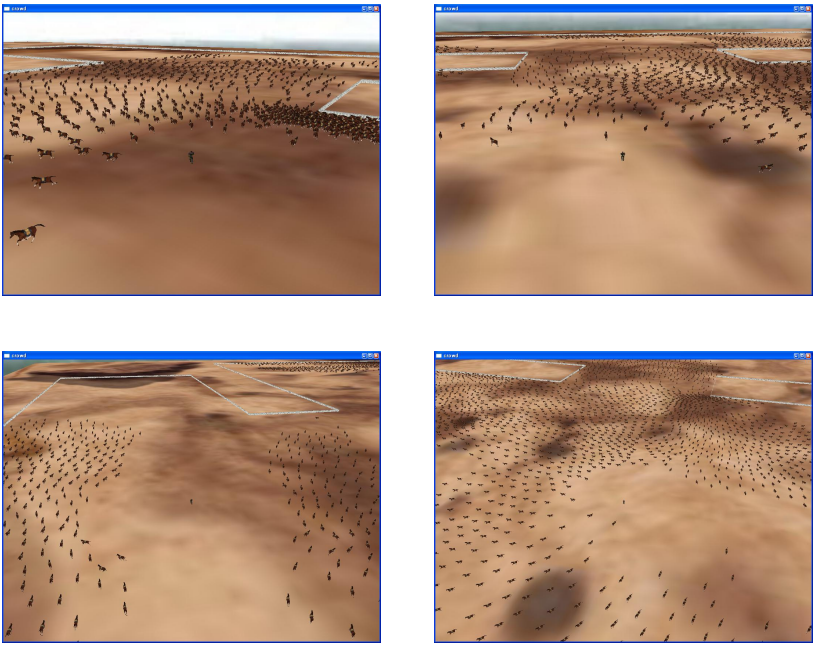

Fig. 4. Large crowd rendering in a scene of fences with behavior control

animating horses in an area with fences, where both of the human player character (enemy) and the fences (obstacles) lead to the avoidance behaviors of the horses.

5 Conclusion

This article presented integration of a multi-resolution technique for real-time animation of large crowds and the character behavior control. The target application of the presented approach is the massively multi-player online role playing games (MMORPGs). The experiment results show that such integration is attractive for MMORPGs, especially where a huge number of NPCs such as animals or monsters should be animated.

Acknowledgements

This research was supported by MIC, Korea under ITRC IITA-2005-(C1090-0501-0019).

References

1. Reynolds, C.: Flocks, Herds, and Schools: A Distributed Behavioral Model. SIGGRAPH87 (1987) 25-34.
2. Kang, I., Eom. Y., Han, J.: A Multi-resolution Technique for Real-time Animation of Large Crowds. International Symposium on Computer and Information Sciences. (2006).
3. Reynolds, C.: Interaction with Groups of Autonomous Characters. Game Developers Conference (2000).
4. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformations: A unified approach to shape interpoalation and skeleton-driven deformation. SIGGRAPH2000 165–172.
5. NVIDIA: Using vertex buffer objects. NVIDIA White Paper. October 2003.
6. Guymon, M.: Pyro-techniques: Playing with fire. Game Developer **7** (2000) 23-27.
7. Maciel, P., Shirley, P.: Visual navigation of large environments using textured clusters. Proc. Symposium of Interactive 3D Graphics(1995).