STUDIA INFORMATICA 2011
Volume 32 Number 1A (94)

Ewa LACH
Politechnika Śląska, Instytut Informatyki

MANAGING VIRTUAL CHARACTERS BEHAVIOUR WITH OPERATIONS CONTROLLERS

Summary. In this paper some of the aspects of creating an animation of a three-dimensional virtual characters are addressed. The open architecture of the animation framework, which enables the user to plug in different independent modules, is described. The concept of the sequence of the operations controllers that manipulate and change character's actions is also presented. The paper also introduces the avatars double buffering method .

Keywords: 3D animation framework, control of virtual character's behaviour

WYKORZYSTANIE KONTROLERÓW OPERACJI DO ZARZĄDZANIA ZACHOWANIEM WIRTUALNYCH POSTACI

Streszczenie. Artykuł przedstawia architekturę środowiska do animacji trójwymiarowych postaci oraz koncepcję sekwencji kontrolerów operacji, które manipulują i zmieniają zachowanie wirtualnych postaci. W artykule została zaproponowana metoda podwójnego buforowania graficznej reprezentacji postaci –awatarów.

Słowa kluczowe: środowisko 3D do animowania postaci, kontrola zachowania wirtualnych postaci

1. Introduction

Virtual characters are widely used in many applications, especially in computer games, movies and educational systems. Currently, there are many powerful tools aimed at creating virtual characters, however the animation of such characters is still a major problem and there is a need to automate this task, especially when the number of involved characters grows. The generation of the virtual character's animation demands solving many constraints that arise from natural limitation of the human-like bodies and the virtual environment. Collision detection and avoidance, interactions with other objects and characters are examples of the challenges facing animators during an animation process. Character's motion should also reflect its personality and emotions. A happy, optimistic person moves differently from a sad and frustrated one. At present there is no ultimate technique that allows for efficient creation of an accurate and natural animation of the human alike characters. However, currently, most of the techniques use a skeleton to generate a motion and a surface representation (called the skin) to draw a character. The skeleton in a character animation has a direct correlation to a human skeleton: it consists of articulated joints and bones, that can be used as a controlling mechanism to deform the attached skin. An animator can use various techniques while creating characters' motion. For instance, forward and inverse kinematics, motion capture, procedural animation or physically-based animation can be applied. They all have their advantages and limitations.

The paper discusses the problem of managing the motion of the three-dimensional characters in the virtual environments. A concept, is presented, that allows dealing with characters motion by means of various controllers that manipulate and change character's actions. The controllers are easily added, removed and substituted in the framework for behavioral animation named fACT (framework for Animation of virtual Characters in Three dimensions), which is also described in the paper.

2. Related work

The problem of controlling a motion of the virtual characters has generated a lot of attention. One of the most researched topics is a collision detection problem [1]. The computational cost of a collision detection algorithm depends not only on the complexity of the basic interference test used, but also on the number of times this test is applied. Therefore for complex scenes comparing all pairs of primitive geometric elements is inefficient. To address this problem many of the approaches have used hierarchies of bounding volumes BVHs [3][4] or spatial decompositions [5]. These techniques reduce the number of pairs of objects that need to be checked for contact by the approximation of objects with bounding volumes or decomposition of the space occupied by objects.

With a bigger number of moving characters a collision avoidance becomes necessity [2]. a reactive obstacle avoidance is an important step in attaining greater autonomy in characters. Collision avoidance not only requires detection of the objects positioning on the character's path but also collision prediction using object dynamics and behaviour analysis.

Another critical issue in the character's animation concerns the constraints imposed by the human body limitations. The simplest method expresses limits of the joints of the skeleton in the terms of the hard limits on the individual rotation angles [6]. It is a very fast solution, but it does not account for dependencies between rotation angles, such as those between the allowable amount of the arm twisting and the arm's position. More complex solution involves a sinus cone [7], which is, currently, one of the most widely-used representations.

Every technique used to generate animations needs to address all problems described above. Depending on the virtual character's level of autonomy an animator has more or less influence on its motion and on how the restrictions of the environment and the character's body are solved. The forward kinematics technique offers animators the most freedom [8]. For example, a hand's motion can be acquired from the animator's specification about amounts of rotation and bending of each joint in an arm. The inverse kinematics technique allows an animator to specify only a new hand's position, when a rotation of the arm's joints are calculated automatically [9][10]. The motion capture technique obtains a character's motion from the human actors [11]. Its adjustment to virtual world can be more or less automatic depending on the applied software configuration. The physically-based animation of the virtual human-like characters could generate a very natural motion. This technique is currently rarely used because of the complexity of the physical model [12]. The behavioural animation can integrate various techniques to obtain virtual characters with capabilities of perceiving their environment and an ability to react and make decisions, depending on the environment's and the character's state [13]. By the character's state we understand not only the posture and the position of the character but also the character's personality, motivation and emotions [14], which make the virtual character more believable.

There are many applications for a character's animation, but their capability for the extension is frequently limited, as the addition of new modules was not considered during system design. Depending on the aim of the animation different features of an animation are having different priorities. For example, in games the real-time generation is more important than a super quality of the character's movements. In films the animation quality is of the greatest importance. Because of that animation frameworks should allow users to choose from many animation techniques the most suited one or add a new technique when it is needed. Barros, Evers and Musse described in [15] a pioneer framework, which integrates different behavioral modules in a client-server architecture. However, a time needed for communication between modules, seems to slow down a process of animation. Behavioural animation framework, described in this paper, permits easy substitution of one behavioural model by another. Additionally, fast performance is its one of main objectives.

3. The fACT framework

The main goal of the fACT framework is providing an environment for testing different methods of a generation and an implementation of the virtual characters' behavioural strategies [16]. The behavioural strategies are understood, in fACT, as the sequences of the operations run successively or in parallel according to character's states or states of his environment to obtain the character's goals. The framework is divided into independent modules focused on their functions (fig. 1). It is easy to modify and substitute them.

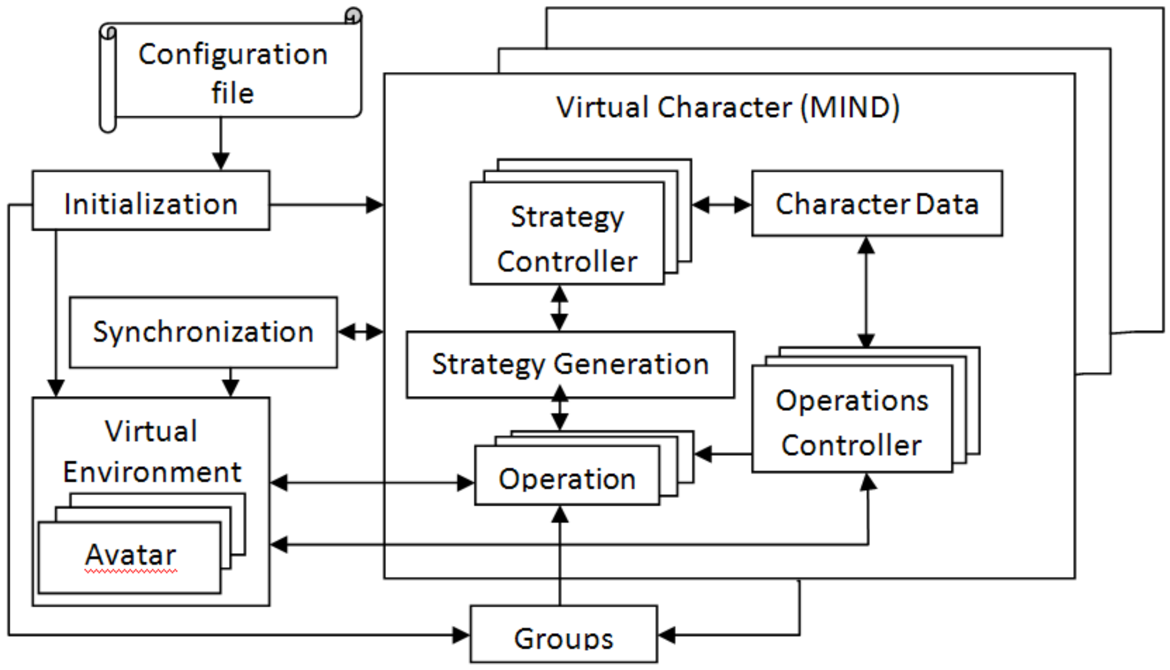

Fig. 1. The architecture of the fACT framework
Rys. 1. Architektura środowiska fACT

In fACT the operations are responsible for modifying the character or the environment of the character, performing transformation on the three-dimensional objects and determining a state of the character, the environment, the objects and other characters placed within the environment. The operations are used to build the behavioural strategies. There are three main types of the operations: simple operations, conditions and complex operations. The simple operations are responsible for the transformation of the three-dimensional objects and the modification of the character's data (e.g. a level of aggression), the environment and the objects within. The conditions are the operations, which acquire various information and determine their membership to the different sets. For example, an operation can determine if an object of interest is on the right, on the left or in the front of an avatar (a graphical representation of a character). Correctly defined conditions allow characters to adjust to the changing environment and to react to the numerous events. The complex operations are built from other operations: simple operations, conditions and complex operations.

During their execution the operations are monitored, restricted and modified by the operations controllers. For example, an operation controller can be responsible for collision detection or for modifying a character's level of happiness. It can also adjust a motion of a character so it mirrors a character's personality. Additionally, the operations controllers can stop an execution of the strategy because, for instance, the character's aim was reached or there was a critical collision.

Information about a size, a position and an orientation of avatars and other three-dimensional objects are stored in the Virtual Environment (VENV) module. The rest of the character's specification is managed by the Character Data (CHD) component, in which each data is identified by its unique name. Stored information details can be dynamically added, removed, modified, reset or accessed by different fACT modules. For example, information about a character's mental state can be managed by CHD. The operation controllers use data from VENV and CHD to manage operations. The operations send instructions to VENV that results in the geometric transformations performed on the three-dimensional objects.

A whole character strategy is managed by the strategy controllers. For example, strategy controller can be used to measure how well the strategy fulfils the character's goals.

The virtual character can be a member of various groups, which are applied in the operations to define the interactions between the characters. For example, virtual characters can be divided into two fighting groups.

For each virtual character there is one dynamically defined the Strategy Generation (SG) module, which is responsible for the generation of character's behaviours composed from the virtual character's operations. The strategies of different characters are generated and executed as different threads and synchronized by the Synchronization (SYN) module.

The fACT framework is fully reconfigurable, all of its components can be specialized through a dynamic polymorphism. The fACT's components are created and initialized by the Initialization (INIT) module in accordance with the configuration files.

4. Character motion controllers

The motion of the avatars in fACT is defined, modified and implemented by various fACT components. At first the SG module generates a behavioural strategy, which is then checked by strategy controllers, which are responsible also for setting initial values for the characters data. After that, the operations of the behavioural strategy are executed. They are controlled by the operations controllers, which can modify them or prevent their execution. At the end of the strategy the strategy controllers are run again. They can, for instance, analyze how the strategy fulfils character's goal.

The operations controllers are responsible for detecting collisions and movements that break body restrictions. They ensure that collision avoidance is executed or that the character's motion mirrors its emotional state. Any technique, which checks a character's motion or wants to modify it can be implemented as an operations controller.

An addition of a new operations controller is very straightforward. a programmer has to define a new C++ sub-class that inherits from a parent abstract class characterizing all operations controllers and then redefine its virtual methods. After a declaration of the class in the configuration file and its registration in the fACT framework (a one line call) a dynamic polymorphism enables an execution of the new controller.

The operations controllers are executed in order consistent with their declaration in the configuration file. They can be declared and implemented more than one time so that they can react to actions of other controllers. They can communicate by the shared data such as a code error. The process of an operation execution can be as follows: one of the controllers allows operation to be run and set execution variable *e* to 1, then another controller checks for collisions if *e* is equal to 1. If collision is detected code error is set and another controller can react to it reversing executed motion or modifying it.

Such architecture makes adding, activating and deactivating various algorithm and techniques quite simple. Another advantage is that the execution of the sequence of the controllers makes possible a distribution of an animation tasks. For example, constructing a collision detection technique a programmer-animator does not have to worry about implementing reactions to the algorithm results. a different controller can do this. It is the solution strengths but it also generates some problems. If a controller detects that an operation modified by an earlier controller is incorrect reversing executed motion can be difficult and uses valuable time. It can also create a problematic situation with other characters, which execute their motion in parallel. They could have already adjust their actions to the one requiring a reversion. The solution to this problem was based on the principle of the double buffering used in the animation visualization. Double buffering is a method that uses a screen buffer to prevent viewer from seeing only partially-updated version of the data. The technique uses two ”pages” or screen buffers. One buffer is shown on screen while the other is updated with the next frame. When the new frame is ready, the pages are flipped.

Based on the double buffering concept one can introduce double buffering for avatars, in which the character can have two avatars: a main avatar and a backup one. The former is created at the beginning of the strategy, together with the character. The later can be created, activated, deactivated and removed during an execution of a strategy. To save time and space some unchanged data, like textures, are shared between avatars. Avatars as the rest of the virtual objects in the ENV module have an attribute *active* that define how 3D objects are treated by others in the virtual scene. If an attribute *active* for an object is set as *true* other objects in the scene sense it, and are influenced by its actions. For example, a collision for an object is checked only with objects that attribute *active* is on. Being active doesn’t mean that object is visible to a camera. Some objects can be active but also invisible. There are objects, with which a collision triggered specified actions, that are not shown in the scene but, nevertheless, active. We can imagine a scene in which a character’s collision with an invisible wall (*a door activation object*), placed in the front of a door, will succeed in an automatic opening of the door. In order to define object’s visibility in visualization of virtual scene second attribute *show* is used. It is set not only for an active objects. For example an object representing a ghost can be visible, but inactive.

These two attributes are used in avatars double buffering method. The main avatar is set as active and visible. It is seen by other virtual objects in the environment and by the human observer watching the rendered animation. The second avatar is inactive and invisible. Other participants in the virtual scene are unaware of its existence. They can move through it without realising that. Because the buffer-avatar is also invisible it does not affect an animation quality from human-watcher points of view. The operations controllers, according to the fACT configuration file, can set one of these two avatars as used by the strategy operations. When the avatars double buffering is applied in the operations controller, the buffer-avatar is created or activated (if already existing), during which its position and orientation is synchronised with the main one. The operations move and rotate the avatar and/or its joints. If the actions inflected on the avatar are incorrect the scene may need to return to its state before the operation’s execution. If avatars double buffering is utilised the operations controller that is responsible for reacting to such a situation can simply deactivate the buffer-avatar. No other action is necessary. If the proposed motion is accepted by all operations controllers, then avatars are swapped: the main avatar of the character become a buffer-avatar, the buffer-avatar become the main avatar. The attributes *active* and *show* are changed accordingly.

Double buffering of avatars speeds up and simplifies creation of avatar’s motion. Even with a synchronization process using a buffer-avatars is faster than reversing actions for only one avatar, especially if applied operations controllers modifies strategy operations. From the visualisation points of view two avatars make it easy to avoid showing incomplete motions. It prevents also problematic situations with other characters, which execute their motion in parallel.

5. Conclusion

The paper introduces the sequence of the operations controllers with double buffering, which is used to control a motion of the virtual characters. The applied solution enables a simple addition of new algorithms and techniques to the process of character's animation. It also breaks up the whole process into smaller tasks.

BIBLIOGRAPHY

1. Kockara S., Halic T., Iqbal K., Bayrak C., Rowe R.: Collision detection: a survey. IEEE International Conference on Systems, Man and Cybernetics, Canada 2007, p. 4046÷4051.
2. Abdullah N.A.B.M., Bin Base A., Kari S.: A Review of Collision Avoidance Technique for Crowd Simulation. International Conference on ICIMT, Jeju Island 2009, p. 388÷392.
3. Liu L., Wang Z., Xia S.: A Volumetric Bounding Volume Hierarchy for Collision Detection. 10th IEEE International Conference on Computer Aided Design and Computer Graphics, Beijing 2007, p. 485÷488.
4. Chang J.W., Wangb W., Kim M.S.: Efficient collision detection using a dual OBB-sphere bounding volume hierarchy. Computer-Aided Design, 2009.
5. Luque R., Comba J., Freitas C.: Broad-phase collision detection using semi-adjusting BSP-trees. Symposium on Interactive 3D graphics and games, Washington 2005, p. 179÷186.
6. Wang X., Maurin M., Mazet F., De Castro Maia N., Voinot K., Verriest J.P, Fayet M.: Three-dimensional modelling of the motion range of axial rotation of the upper arm. Journal of Biomechanics No. 31(10), 1998, p. 899÷908.
7. Maurel W.: 3d modeling of the human upper limb including the biomechanics of joints, muscles and soft tissues. 1998.
8. Benitez A.R., de los Santos G.T, Vallejo D.R.: Forward Kinematics for Virtual Agents. Engineering Letters. No. 15(2), 2008.
9. Li S., Liang J., Liu G. Zhang Y.: Combining analytical inverse kinematics with example postures to generate virtual human whole body reaching postures. System Simulation and Scientific Computing, Beijing 2008, p. 45÷52.

10. Millard J.: Evolutionary motion inverse kinematics. Congress on Evolutionary Compution, Singapore 2007, p. 3671÷3678.
11. ZhiDong X., Zhang J.J., Bell S.: Control of motion in character animation. Eighth International Conference on Information Visualisation, 2004, p. 841÷848.
12. Wrotek P., Jenkins O.Ch., McGuire M.: Dynamo: Dynamic, Data-driven Character Control with Adjustable Balance ACM Sandbox Symposium on Video Games, 2006, p. 61÷70.
13. Hanna J.R.P., Millar R.J., Johnston W.M.: Examining The Generality Of a Behavioural Animation Framework. 2001.
14. Liu Z.: a Personality Based Emotion Model for Intelligent Virtual Agents. ICNC, Jinan 2008, p. 13÷16.
15. Barros L.M., Evers T.F., Musse S.R.: a framework to investigate behavioural models. Journal of WSCG, 2002.
16. Lach E.: fACT - Animation Framework for Generation of Virtual Characters Behaviours. The 1st International Conference on Information Thechnology, Gdańsk 2008, s. 325÷328.

Recenzent: Dr hab. inż. Bogdan Smołka, prof. Pol. Śl.

Wpłynęło do Redakcji 29 października 2010 r.

Omówienie

Obecnie animacja komputerowa pozwala na produkcję prawie realistycznych trójwymiarowych modeli graficznych ekspresyjnych postaci wykonujących różnorodne zadania w ramach wirtualnych światów, które zamieszkują. Mimo to praca, którą muszą wykonać animatorzy, nadal jest imponująca. W artykule przedstawiono jedne z najistotniejszych problemów dotyczących tworzenia animacji trójwymiarowych postaci wirtualnych. Wykrywanie i unikanie kolizji, zachowanie ograniczeń ludzkiego ciała, przedstawienie wewnętrznych odczuć i celi postaci to zadania, dla których nadal nie znaleziono w pełni satysfakcjonujących rozwiązań. W pracy opisano środowisko do animowania trójwymiarowych postaci: fACT, które pozwala łatwo dodawać, podmieniać i usuwać moduły odpowiedzialne za poszczególne etapy procesu animowania postaci. Szczególną uwagę zwrócono na kontrolery operacji, które są odpowiedzialne za zarządzanie sposobem wykonania zadań przez wirtualne postacie. Poszczególne kontrolery wykrywają kolizje, naruszenie ograniczeń stawów postaci, sprawdzają zgodność zachowania postaci z ich stanem wewnętrznym i modyfikują zachowanie postaci zgodnie z przyjętymi założeniami. W artykule zaprezentowano także metodę podwójnego buforowania dla awatarów postaci wirtualnej. Metoda ta umożliwia pominięcie problemu wynikającego z konieczności cofnięcia akcji wykonanych przez postać po przez wykorzystanie awatara tymczasowego, na którym wykonuje się wszelkie skomplikowane przekształcenia. Jeżeli któryś z kontrolerów zgłasza potrzebę unieważnienia wykonanego przez postać zadania, jedyna akcja, którą należy wykonać, to przywrócenie głównemu awatarowi postaci aktywnego statusu. Poprawnie wykonane zadanie wymaga zamianę awatarów miejscami: tymczasowy awatar staje się głównym awatarem postaci. Zaproponowana metoda przyspiesza i ułatwia proces tworzenia animacji wirtualnych postaci.

Address

Ewa LACH: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, ewa.lach @polsl.pl.