Conceptual Farm

Shuen-Huei Guan† Sheng-Yao Cho† Yu-Te Shen†
Rung-Huei Liang† Bing-Yu Chen‡ Ming Ouhyoung*

†{drake,ven,edwards,liang}@cmlab.csie.ntu.edu.tw, ‡robin@ntu.edu.tw, *ming@csie.ntu.edu.tw
Communication and Multimedia Laboratory,
††Dept. of Computer Science and Information Engineering / ‡Dept. of Information Management,
National Taiwan University

Abstract

Conceptual Farm is a virtual reality platform for generating and observing behaviors of different autonomous characters. By providing 1) descriptions for characters’ behaviors and 2) 3D animations and sound, life-like characters in a realistic habitat can be created, modified, and interact both with users and other characters in real time. The flexible, manageable and scalable nature of Conceptual Farm leads to its desirability in zoological research, general education, game and film production, and even decorative arts.

1. Introduction

Imagine you have a paintbrush in your hand. You draw some pigeons with gray feathers in a square, and teach them to walk, to eat, to fly, and so on. Then, the pigeons start wandering around the ground, eating the feed you spread, and flying up from time to time. An idea occurs to you, “Why not add kids to play with the pigeons?” So you draw three children in the square, running amongst the pigeons for fun as soon as running is taught. As new ideas continue to strike you, the picture becomes more and more enriched…

Interactive artificial life as imaged above has been presented so far [2, 3, 5, 6, 12]. Based on biological theories, these systems simulate animals’ behaviors realistically, but, on the other hand, too complicated to be integrated with a compact and systematic interface such that users can create artificial lives easily. Some languages are proposed for describing cognitive behaviors [4, 5, 7], but they are not intuitive enough.

To simplify the process of creating artificial lives, we designed a system, Conceptual Farm, for easy creation, modification, observation of and interaction with virtual autonomous characters. Four important features of Conceptual Farm are:

1. Converting simple descriptions into complex behaviors — Complex autonomous behaviors are created using compact table-based descriptions (Figure 3) and flexible scripts instead of complicated codes.
2. Semi-interactive editing — The behavior is performed in real time as users adjust characters’ properties, except when there is a need to regenerate the animation clips.
3. Extensibility — Conceptual Farm is a platform that can be used to easily realize every type of animal including insects, mammals, fish, and can even be extended to autonomous characters of every type such as aircraft or soccer players.
4. Adoptability for AFX — According to the former features, the design philosophy of Conceptual Farm can be applied to the higher levels in Animation Framework eXtension in MPEG-4, which are not yet clearly defined.

The first two features, which are not easily performed by previously mentioned approaches, are our main contributions, since our system reflects user’s input with characters’ behaviors directly, and users can focus on characters’ behaviors without annoying programming problems.

2. Overview

Figure 1: Flocking pigeons in Conceptual Farm
Based on Conceptual Farm, Dove project creates pigeons in a 3D world that can act autonomously in Figure 1. These autonomous behaviors come from a table-based system, in which the three resources of simulated characters (Plans, Navigating Styles and Appearances) are provided by users.

This paper describes the mechanism which enables an efficient approach to simulate artificial lives as in Dove, the implementation issues, and the impact on AFX.

3. Behavioral Model

In Conceptual Farm, all simulated characters can be viewed as autonomous agents, which repeatedly perceive information from World and perform certain reactions, as illustrated in Figure 2. Each agent consists of Decision Maker, Pilot, and Performer, and has its own resources provided by users. During each simulating process, Decision Maker selects a Navigating Style and an Appearance according to the information obtained from World, and passes them to Pilot and Performer, respectively. Pilot determines new position, velocity, and orientation for the next instant, and Performer outputs proper Appearances to World. In the implementation, World is the union of all the other characters.

Let us take an example of the whole process. There is a dove. At one moment, Decision Maker decides to eat food, then Pilot steps a forward little, and finally Performer plays head-lowering animation and cooing sound.

3.1 Decision Maker

Decision Maker, as a brain with sense organs, perceives information from World and then chooses the best responding action. Influences affecting decision-making include not only events made by users with the system UI or by other characters in the virtual world but also internal factors such as current action, degree of hunger, etc. Following the rules given by users, namely, Plans, Decision Maker will select the action with the highest utility to send to Pilot and Performer.

Plans describe 1) the relationship (occurrence probabilities) between actions and percepts, 2) the mapping of each action to its corresponding Navigating Styles and Appearances, and 3) the scope of each percept. The following is a sample of (1) and (2) of a pigeon’s Plans.

<table>
<thead>
<tr>
<th>Action</th>
<th>Percept</th>
<th>Navigating Style</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wander</td>
<td>Eat</td>
<td>Pursuit</td>
<td>Sound</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0</td>
<td>0.8</td>
<td>wander.wav</td>
</tr>
<tr>
<td>0.7</td>
<td>0.0</td>
<td>0.3</td>
<td>eat.wav</td>
</tr>
<tr>
<td>0.0</td>
<td>0.5</td>
<td>0.5</td>
<td>pursit.wav</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Plans

Decision Maker consists of 2 agents: Percept Agent and Action Agent. We refer to the c4 architecture [2] of the MIT Media Lab.

The Percept Agent senses all external events within the sensory scope for each percept. Internal self-awareness, such as the current action, is also sensed. Users can dynamically add or remove customized percept functions as well as the built-in ones, and the behaviors will change immediately.

Plans not only enable realistic simulation for sophisticated behaviors with the uncertain nature of the probability values, but also, as a table, provide a compact way for easy manipulation.

The Action Agent is a utility-based agent. It uses the percepts stored by Percept Agent and calculates the score for every candidate action using (1). If nothing is sensed within the scope of a percept, the probability of this percept will be set to zero. The resulting probability of each action is proportional to its score.

\[
Score(A_i) = \sum_{i=0}^{N} Prob(P_i, A_i) \cdot \Phi(P_i) \quad (1)
\]

\[A_i:\text{ the } i\text{th action}
\]

\[P_i:\text{ the } i\text{th percept}
\]

\[N:\text{ the number of percepts.}
\]

\[Prob(P_i, A_i):\text{ the occurrence probability of } i\text{th percept when the } j\text{th action happens}
\]

\[\Phi(P_i): 1 \text{ if } P_i \text{ occurs, } 0 \text{ otherwise.}
\]
After selecting the action with the highest score, the Action Agents will send the Navigating Style to the Pilot, while the corresponding animation and sound will be sent to Performer.

### 3.2 Pilot

For each character, *Pilot* determines its own path and orientation according to the Navigating Style (e.g. seek, pursuit and wander) from Decision Maker.

*Pilot* is built based on Reynolds’s OpenSteer library. Extending Reynolds’ 16 common steering styles for autonomous agents [9], we provide 31 built-in Navigating Styles. Although the built-in styles meet the demands of most cases, users can also provide customized Navigating Styles through scripts. We provide a high-level script based on Small, which enhances the flexibility of Conceptual Farm.

For example, when a bird is landing with a built-in Navigating Style, **NS_YParabolicUp**, it will have the unreal velocity-alignment problem as illustrated at the left of Figure 4. The problem can be solved by a script as below:

```java
public doLanding (force, elapsedTime) {
    applyForce (force, elapsedTime)
    velocity[1] = 0
    calculateOrientation (velocity)
}
```

It first calls `applyForce` to get the new position and velocity, and it applies `calculateOrientation` with the y-zeroed velocity to get the reasonable orientation with the result as illustrated on the right side of Figure 4.

![Figure 4: Two landing styles.](image)

### 3.3 Performer

*Perform* is responsible for perceptible outputs of autonomous characters by animation, sound or any expressive media provided by users. Our system handles most kinds of sounds by FMOD.

Several exporters were implemented to allow users to make animations with their favorite tools such as Maya, 3D Studio Max.

### 4. Experimental Results

We spent about one hour using Conceptual Farm to build *Dove*, a square at the Chiang Kai-Shek Memorial Hall in Taipei where pigeons gather to find food, clean their feathers, and where dogs wander around, and user can spread crumbs and run amongst the pigeons in Figure 6. It is much faster than the three-man-week building process of a similar program by scratch, given the same animation and audio files.

Taking the pigeons’ resource as an example, there are 13 actions and 8 percepts from Plans, 12 animations and 4 sounds for Appearances, and 2 scripts for Navigating Styles.

### 5. AFX & Conceptual Farm

According to the characteristics mentioned above, the concept of our system can be applied to enhance current multimedia standards.

As illustrated in Figure 5, MPEG-4 proposed AFX (Animation Framework eXtension) in order to provide a standardized description for computer animation and interaction, similar to video and audio standards. AFX is layered into six components, which are, in a top-down order, cognitive, behavioral, biomechanical, physics, modeling and geometry components. The last four are specified clearly in details, while cognitive and behavioral components are not, since they are AI-intensive and difficult to formalize.

![Figure 5: AFX in MPEG-4](image)

---

2. [http://www.compuphase.com/small.htm](http://www.compuphase.com/small.htm)
3. [http://www.fmod.org](http://www.fmod.org)
To make up for this limitation in current AFX, the concept of our system provides a solution for editing and functioning data in the cognitive and behavioral layers, and connects them with multimedia in the other components of AFX, and even other parts in MPEG-4. The formalized data used to describe characters’ behaviors in Conceptual Farm is also suitable for storage and transmission, which is the characteristic of standardized data.

6. Future Work and Conclusions

Conceptual Farm provides an easy way to create virtual lives. There are, however, some limitations. Characters cannot adapt themselves to the environment because they cannot modify their own Plans. In addition, hardcode cannot be substituted for with our table-based input mechanism to provide user-character interaction, which is simulated-character, environment, and input-device dependent.

In summary, we demonstrate a novel approach to create interactive artificial lives with our virtual reality system, Conceptual Farm. The system simulates character’ behaviors realistically with compact and formalized input descriptions and preserves its flexibility by introducing scripts. It also suggests a practical method for standardizing and easily manipulating the dynamic and real-time properties within the cognitive and behavioral levels of current AFX in MPEG-4.

7. Acknowledgements

This research was supported in part by National Science Council 92-2622-E002-002. We are grateful to Kuei-Yuan Zheng, Ping-Chun Kuo, Wei-Chih Liao, and Tien-Jung Huang (National Taiwan University of Arts) for providing their technical helps. We also thank Wan-Chun Ma for his comments on MPEG-4 AFX.

8. References


4 We use the word “functioning” instead of “playing” because artificial intelligence is concerned in addition to animation.