

Bouncy: An Interactive Life-Like Pet

Paolo Pirjanian, Claus B. Madsen and Erik Granum

Laboratory of Image Analysis, Aalborg University

Fredrik Bajers Vej 7D, DK-9220 Aalborg East, Denmark

E-mail: {paolo, cbm, eg}@vision.auc.dk

November 27, 1998

1 Introduction

In this document we describe Vespa I (Virtual Environment for Staging Plays and (inter)Acting), which is the first of a series of prototypes to be developed at the Laboratory of Image Analysis for the STAGING¹ project.

Vespa I is an interactive virtual reality system with which a user can interact through a multimodal multimedia interface. The main objective of this prototype has been to showcase the necessary tools and technologies for developing such interactive multimedia systems with a special emphasis on intelligent autonomous agents. To this end we have created the character Bouncy which is an animated life-like character with which a user can interact through a multimodal multimedia interface. Bouncy is autonomous in the respect that its behavior is not a function of user commands and interaction only, it is also a function of its own intentions and desires.

2 Vespa I: system description

The system as we see it is illustrated in figure 1 and consists of three main components:

1. *Bouncy* who is an inhabitant of the virtual world,
2. the *user* who resides in the real world,
3. and, the *interface* which defines the media by which the user and Bouncy can interact.

Bouncy (see figure 2) is a 3D animated life-like character with behaviors similar to that of a dog. He moves around by bouncing up and down (hence the name Bouncy) while moving forward and/or

¹STAGING “The Staging of Virtual, Inhabited 3D Spaces” is a collaborative project funded by the Danish Research Councils

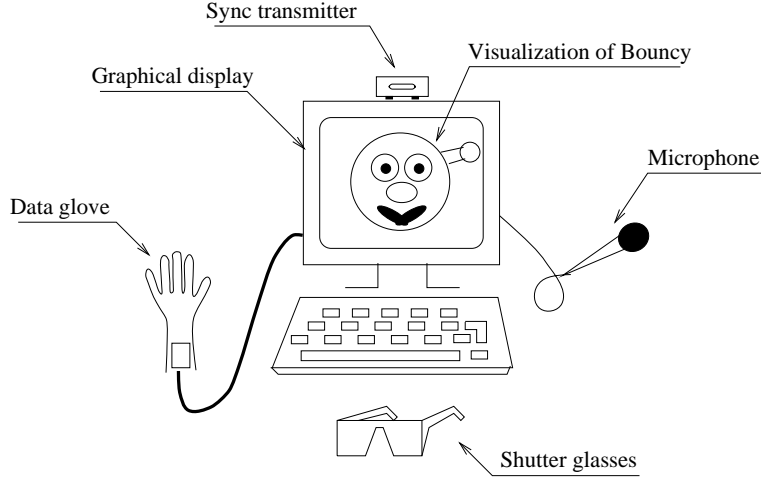


Figure 1: The overall system

turning. He is able to display various facial expressions (happy, sad etc.) by actuating his mouth, eyes and tail.

The user interaction with Bouncy occurs through a set of devices. Bouncy is visualized for the user on a graphical display. The user has the option to wear a pair of shutter glasses to get an stereo-scopic view of Bouncy and its environment. Bouncy perceives the user through a speech interface (the microphone) and a data glove. The data glove allows Bouncy to recognize a set of predefined hand gestures generated by the user. We have attempted to define an interface that feels intuitive for the user. The following table lists how Bouncy perceives the user's behavior:

User's behavior		Bouncy's perception of user
<i>Data glove input</i>	<i>Mic input</i>	
None	Shouting	Master is calling
Pointing	Shouting	Master is scolding
Open hand	Talking	Master is clapping

3 Bouncy the dog

The mechanism for controlling Bouncy follows the behavior-based approach [1]. In this approach the control of an agent is distributed among a set of sensory-motoric units known as behaviors. Each behavior is concerned with a specific and well-defined task such as eating, mating, playing etc. Based on internal and external sensory stimuli each behavior controls the agent in a way to accommodate its task objective(s). This shared control approach might, however, lead to conflict among behavior objectives; thus it is necessary to coordinate the activities of the behaviors to ensure a coherent and rational behavior. In the literature this problem is known as the action selection problem defined in the following: "*How can an agent select 'the most appropriate' or 'the most relevant' next action to take at a particular moment, when facing a particular situation?*" [3]. Thus an action selection mechanism constitutes a major component in Bouncy.



Figure 2: Bouncy.

In the following we describe Bouncy’s behaviors and action selection mechanisms followed by a description of his mental attitude and how that was designed.

3.1 Bouncy’s behavioral repertoire

To keep it simple in its first implementation Bouncy was armed with three main behaviors:

- Play: causes Bouncy to wander about and play,
- Sleep: makes sure that Bouncy gets enough rest,
- Interact: makes Bouncy goto and/or follow the user plus engage in interaction with the user.

Bouncy’s purpose in (virtual) life is to play, interact with and please its owner (the user) and sleep.

A relevant behavior is activated according to Bouncy’s mental state as well as the situation at hand. For example if the user calls Bouncy then the ‘Interact’ behavior should be activated only if Bouncy is interested. If he is not interested he should respond differently when he is called. It is the action selection mechanism that determines which behavior to activate in a given situation.

Once activated a behavior takes control of Bouncy’s motoric and mental capacities. Each behavior is designed to articulate Bouncy so as to manifest the correct attitude and behavior. For example, the ‘Play’ behavior causes Bouncy to “run” and jump in a playful manner. Bouncy’s attitude and behavior is also influenced by its mental state - whether it is happy or sad, excited or not etc.

Since Bouncy’s behavior repertoire is rather limited it was possible to control the activation of its behaviors using a straight forward action selection mechanism known as Discrete Event Systems [2], which is based on the finite state automaton (FSA) formalism. Figure 3 depicts the finite state machine that is used to describe Bouncy’s action selection mechanism. It is seen that there is one state corresponding to each Behavior. State transitions are caused by events denoted `tired`, `rested` etc. How these events are generated will be discussed in the next section.

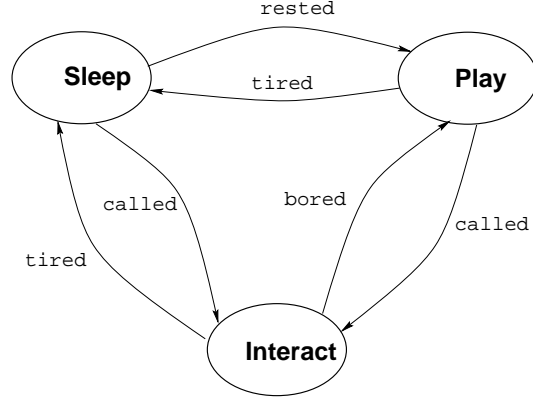


Figure 3: Finite state machine describing the action/behavior selection mechanism for controlling Bouncy.

For example, in the play state the ‘Play’ behavior is activated and Bouncy will start running and jumping around. If Bouncy is called a transition is then made to the interact state and hence the ‘Interact’ behavior is activated. The ‘Interact’ behavior is a more complex behavior which also is described using a finite state formalism as depicted in figure 4. Note that this FSA introduces 3 additional states and hence 3 additional corresponding behaviors.

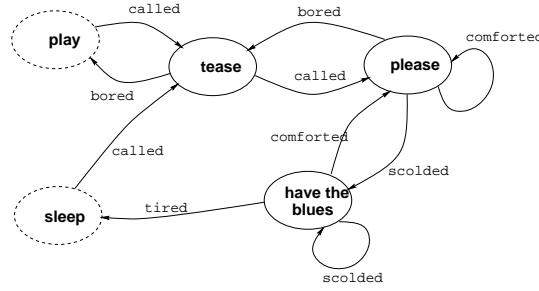


Figure 4: Finite state machine describing the ‘Interact’ behavior.

The interact state in figure 3 could thus be replaced by the finite state machine in figure 4. However, by making this abstraction the overall behavior of Bouncy is much more clearly described in figure 3.

3.2 Bouncy’s mental attitude

Bouncy is designed as a playful creature with a simple yet appealing personality. We chose to design a creature with dog-like characteristics, attributes, and mentality. Given Bouncy’s limited behavior repertoire we chose three parameters to constitute a computational model for Bouncy’s mentality: sleepiness, excitedness and mood.

The value of the attributes range from -1 to $1m$ where -1 would correspond to low and 1 to high,

respectively. The values of these attributes are updated as a function of time and sensory stimuli. Further the attributes are updated differently depending on which activity (behavior) Bouncy is engaged in.

	Sleep		Blues			Play		Tease		Please		
	sound	time	scold	pet	time	sound	time	sound	time	scold	pet	time
excitedness	↑	→ 0	↓			↑	↓	↑	↓		↑	
sleepiness		↓			↑		↑		↑			↑
mood		→ 0	↓	↑	↓		→ 0	↑		↓	↑	

Nomenclature

Increment	: ↑
Decrement	: ↓
Reset	: → 0

Based on the values of these parameters we define a set of perceptions for Bouncy according to the following table:

	excitedness	sleepiness	mood
rested		low	
tired	low	high	low
bored	low		
comforted	low	not low	
scolded		high	low

The **boldfaced** entries indicate that the value of the corresponding attribute should be weighed higher than the other attributes. Thus using this table the value of the internal variables is translated into perceptions for Bouncy. These perceptions cause transition from one state to another and hence activation of a relevant behavior (see figure 3).

3.3 Implementation considerations

The graphical front-end of Bouncy was implemented in Irix Performer 2.0 using the C++ libraries. Performer is based on OpenGL and provides facilities necessary to develop for real-time graphical applications on Silicon Graphics machines.

Bouncy is implemented as a class, called Bouncy with the following motoric and behavioral member functions:

```
// Motion commands (motions are relative)
void moveForward(double speed, double length);
void moveBackward(double speed, double length);
void turn(double speed, double degrees);
```

```

// Motion commands (motions are absolute)
void moveTail(double side_to_side, double up_down);
void moveEyes(double pan, double tilt);
void closeEyes();
void openEyes();
void moveMouth(double scale, double degrees);

// Behaviors
void Breathe();
void Smile();
void donotSmile();
void wagTail(double speed, double intensity);
void Sit();
int approachTarget(pfVec3 *target);
int approachTarget(pfVec3 *target, double v, double w);
void Sleep();
void Play(pfVec3 *);
void HaveTheBlues();
int Tease(pfVec3 *target, double v, double w);

```

Note that the class defines behaviors such as `Smile()` and `wagTail`, which we have not described earlier. These behaviors are implemented to give Bouncy the ability to express emotional states manifesting different moods or mental states. Thus these low-level behaviors are used by the high-level behaviors to express emotional states.

For example the ‘HaveTheBlues’ behavior is defined in the following way:

```

void Bouncy::HaveTheBlues()
{
    moveEyes(0, 0);
    donotSmile();
    Sit();
    wagTail(2.5, 20.0);
    breathe(2.0);
}

```

Thus the ‘HaveTheBlues’ behavior is manifested by 1) looking down `moveEyes(0, 0)`; 2) looking sad, `donotSmile()`; 3) stop bouncing, `Sit()`; 4) wagging tail slower and finally 5) breathing in a nervous manner.

4 Preliminary results

Bouncy has been demonstrated for over 300 individuals with very different backgrounds and in age groups ranging from 7 to over 70 years old. Our subjective observation of people's reactions to Bouncy is that most find him appealing and quite entertaining. Many react with sympathy towards Bouncy when he becomes sad because we scold him under the demonstrations. This suggests that people can relate to Bouncy, this abstract creature with very limited graphical sophistication.

Some of the viewers were allowed to wear the glove and interact with Bouncy themselves. Without further due the users were able to use the interface to interact with Bouncy the way they had seen us do. They felt that the interface was intuitive and user-friendly.

5 Future work

Vespa I with Bouncy comprise the first prototype developed for the STAGING project to demonstrate aspects of autonomous life-like characters that can interact with users through intuitive interfaces.

In the future we plan to develop these ideas into more sophisticated interactive systems for staging entertaining plays. The following is a list of issues that we will address in future prototypes.

1. Extend behavioral complexity: I.e., add a richer behavioral repertoire which also implies a more complex action selection mechanism.
2. Extend interaction capabilities using HMD, Speech recognition, and visually mediated interfaces
3. Learning capabilities: facilities that allow Bouncy to learn from experience to:
 - teach Bouncy new behaviors
 - teach Bouncy new commands and their association with behaviors
4. Refined graphical model of Bouncy:
 - more visually appealing agent
 - a larger set of facial and other expressions of emotions
 - add audio output facilities
5. Virtual environments inhabited with multiple agents: Bouncy+Bouceline+children
6. Scripting: In the future versions of Vespa we'll have to provide scripting facilities that allow the autonomous agents to partake in a play according to a script.

References

- [1] Ronald C. Arkin. *Behavior-Based Robotics*. Intelligent Robotics and Autonomous Agents series. MIT Press, May 1998.

- [2] Jana Košecká and Ruzena Bajcsy. Discrete Event Systems for Autonomous Mobile Agents. . *Proceedings Intelligent Robotic Systems '93 Zakopane*, pages 21–31, July 1993.
- [3] Pattie Maes. How To Do the Right Thing. Technical Report NE 43 - 836, AI-Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, USA, 1989.