CHAPTER

# Seventeen

## Pavlov: Where PBD Meets Macromedia's Director

DAVID WOLBER
*University of San Francisco*

S
R
L

## Abtract

Pavlov and Macromedia's Director both provide mechanisms for creating animated interfaces. Pavlov's design evolved from programming by demonstration (PBD) interface building research, whereas Director evolved from traditional time-line based animation systems. This chapter presents the basic differences in the way the tools are designed, and highlights some advantages of Pavlov's approach.

## 17.1 Introduction

Pavlov (Wolber 1996, 1997, 1998) combines PBD with animation mechanisms similar to those in Macromedia's Director. Instead of writing scripting code to specify interaction, as is done in Director, the Pavlov designer *demonstrates* the events that trigger activity (and the activity itself). The resulting behaviors then appear in a *stimulus-response score,* as shown in the bottom-right corner of Figure 17.1.

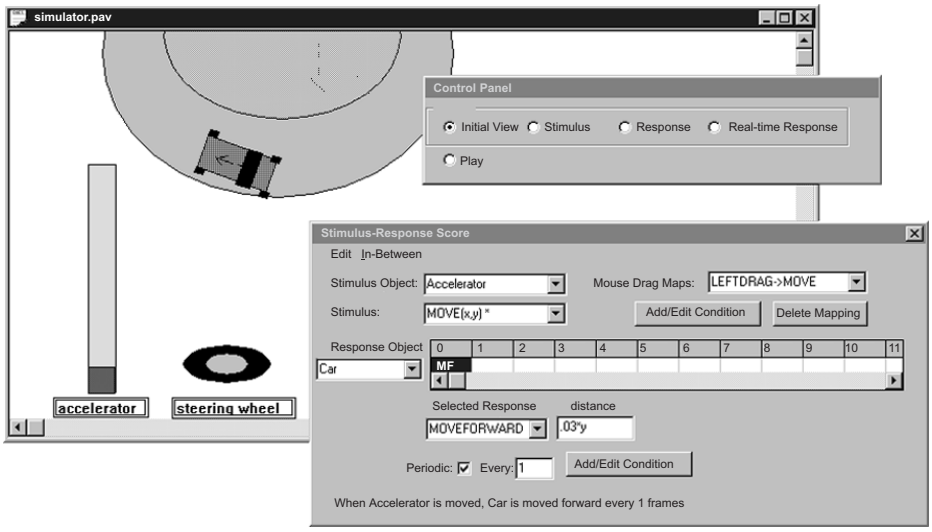
## 17.2 Example

In the example, the designer first specified a direction (nose) for the car by manipulating the special direction vector on it (the vector does not appear at run time). He then demonstrated a stimulus of dragging the accelerator (the small blue rectangle inside the green one) and a response of moving the car. Because the car has a notion of direction, the designer is only able to move it forward or backward on the vector, and the system records the response as a Move Forward instead of a normal move.

The demonstrated behavior is then displayed in the score. When the designer selects a stimulus object and stimulus in the score, the time line displays only the operations that occur in response to that stimulus. As can be seen, the system has inferred that the parameter of the Move Forward operation (distance) be dependent on the vertical movement ($y$) of the accelerator. The designer can modify this parameter or set the behavior as periodic, which has been done in the example (so movement of the accelerator will speed up the car, not just move it once).

S
R
L

FIGURE **17.1**




***Pavlov development of a driving simulator, with a control panel (top right) to tell the system when to record a stimulus or response and the stimulus-response score (bottom right).***

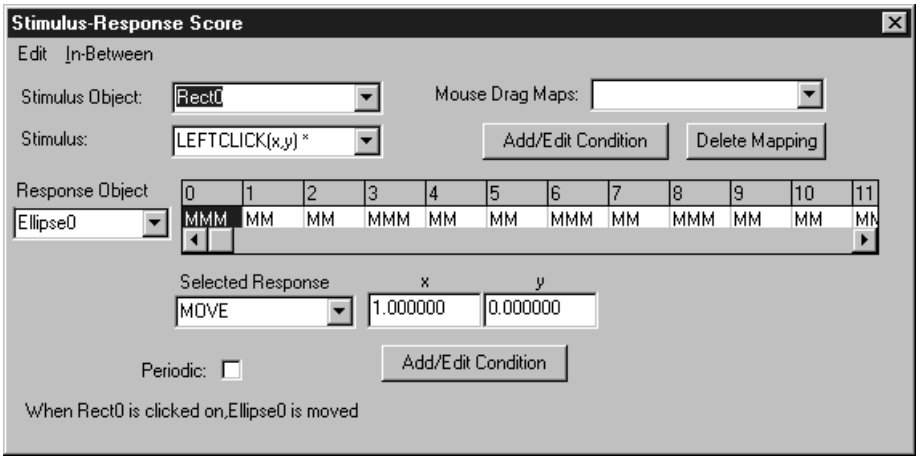
Besides specifying movement as periodic, as in the example, the Pavlov designer can also specify animation paths with in-betweening or with a real-time recording mechanism. With real-time recording, the system records a sequence of time-stamped operations as the designer executes an operation (e.g., moves an object), as shown in Figure 17.2.

Because Pavlov displays a time line for each event, and because operations, not object states, are recorded, the designer can specify interfaces in which objects behave asynchronously. For instance, consider an interface in which two objects follow their respective animation paths in response to clicks on two different buttons.

In Director, the animation path of each object is stored in separate parts of a single time line, as illustrated in Figure 17.3. When a button click occurs, the system changes the global time frame to the corresponding animation path. Whatever animation paths were already in progress are terminated, as each object has a fixed state in the new time frame. The only way to specify interfaces in which objects behave asynchronously is through

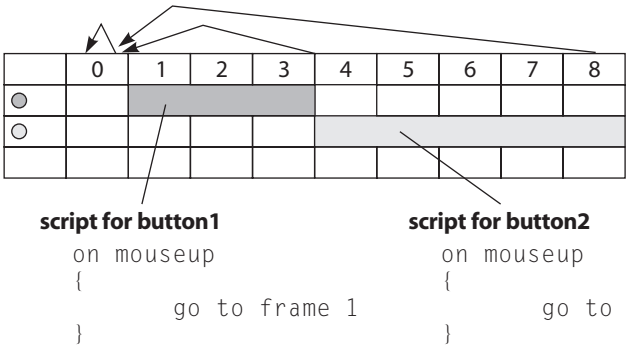
S
R
L

FIGURE 17.2




***A Score showing an animation path as a response to a button click.***

FIGURE 17.3




*A conceptual illustration of the single timeline view. The colored rectangles represent the animation paths of the respective graphics. The arrows represent frame branch statements that set control to the beginning of a path, or back to the waiting state (cell 0). The "on mouseup" scripting code is typical of how interaction is defined in systems like Director.*

bypassing the time line with Lingo *puppet code*. Development becomes more coding and less demonstration.

With Pavlov, the animation paths are stored in separate time lines corresponding to each event (button click). When an event occurs, all the time-stamped operations in a time line are readied for execution, along with other operations from other time lines. Thus, in the example, both objects can follow their paths asynchronously.

## 17.3 Conclusion

Though arguably easier than general programming, script writing is difficult and precludes many from designing interactive animation. PBD along with a stimulus-response based, multiple–time line editor, can eliminate much of the scripting necessary in systems like Director.

## References

Wolber, David. 1997. "An interface builder for designing animated interfaces. In *Transactions on Computer-Human Interface (TOCHI)* (December).

———. 1996. Pavlov: Programming by stimulus-response demonstration. In *Proceedings of the Conference on Human Computer Interface (CHI '96)*.

———. 1998. A multiple timeline editor for designing multi-threaded applications. In *Proceedings of the User Interface and Software Technology (UIST) Conference* (San Francisco).

S
R
L

S
R
L