MicroCosm^{TM} Player Interface

how to control avatars in the MicroCosm universe
by
Chip Morningstar

Lucasfilm Ltd. Games Division
October 17, 1985

Introduction

This document attempts to describe the preliminary player interface design for **MicroCosm™**.

The driving constraint on this design is the nature of the most primitive system that must be expected to support it: a Commodore 64 or similar low-level machine. An ASCII keyboard and the traditional Atari 4-position misery stick are the only available input devices. In spite of this, the interface must enable a wide range of complex behavior to be expressed. It must do this without itself becoming (from the player's viewpoint) complex or cumbersome.

What we wish to do is control the behavior of a **MicroCosm** avatar. The avatar is represented on the screen as an animated humanoid figure. We encourage you, the player, to think of this figure as an extension of yourself. Therefore, the range of behavior we wish the avatar to be capable of should approximate the range of behavior that you might yourself exhibit if you could be projected bodily into the simulation.

Philosophy

The more things an interface tries to let someone do, the more complex it must be, simply because it has to distinguish among numerous options. To get a handle on the enormous repertoire of behaviors that we desire to have available in **MicroCosm**, we look for a mechanism to constrain the number of choices available to the player *at any given time*. This way, even though the range of all possible behaviors will be large, the behaviors that are relevant at any particular moment will be kept down to a manageable number.

Of all the possible behaviors an avatar might exhibit, only a few will be appropriate in any specific circumstance. The key question for us is: How do we determine *which* behaviors these are? Obviously we don't wish to make many prejudgements that reflect a particular social bias about what constitutes ‘‘appropriate’’ and ‘‘inappropriate’’ behavior. We are more interested in determining which behaviors make sense in terms of the underlying physical model. We can get a somewhat better handle on this problem if we consider the associated problem of determining, at any given time, what the situation *is* that we would be selecting behaviors for it. In other words, of all the myriad variables that specify the state of the world model, which are relevant and which are useful for constraining behavior?

We have chosen our answer by looking at the underlying world model. This model is expressed in terms of *objects*. Each object simulates a specific and well-defined component of the world. Each object has a limited set of properties. Only a few things may be done with, to, or by a particular object. What these things are may be different for each possible object, but the particulars can be isolated in the objects' individual definitions.

Thus, if your attention is focused on a particular object, we have already constrained the range of possible behaviors. We believe that this will be enough constraint to bring the range of choices down to a manageable level. The first component of our interface is therefore an *attention selection* mechanism. The details of this mechanism will be discussed in a moment.

Once the focus of attention is determined, we can limit our consideration to a particular object. With each object's definition we associate a piece of code whose job is to determine and then execute the behavior itself based on the state of the world (that object especially) and on your further input. The second component of our interface is then an *action selection* mechanism. These two components taken together define the mechanics of the interface.

Mechanism

The attention selection mechanism is simply a pointer controlled by the joystick. The joystick moves a cursor around on the screen. It does this in the way that joysticks always do. The object beneath the cursor is the focus of attention. (There is a sort of ‘‘background’’ object that covers the whole screen so that *something* will always be selected, even when the cursor isn’t really pointing at anything in particular). Unlike the Macintosh-style interface, no explicit action (such as pressing a button) is required to select something: the object under the cursor is the object of attention, period.

The action selection mechanism involves pressing the joystick button. Actions are expressed in terms of five essential verbs: **do**, **go**, **stop**, **get** and **put**. Of these,**do** is the most important and will be responsible for 70% to 90% of all behavior. One of these five verbs is chosen by pointing the joystick in some particular direction. The pattern is as follows:

$$\begin{tikzcd}
 & \mathbf{stop} & \\
\mathbf{get} & \mathbf{do} \arrow[l] \arrow[r] \arrow[u] \arrow[d] & \mathbf{put} \\
 & \mathbf{go} &
\end{tikzcd}$$

once the joystick is pointed in the direction for the desired verb, releasing the button selects that verb and begins the appropriate action. Note that the center position (i.e., no motion of the joystick) selects **do**. Thus, **do** can be chosen by just pressing and releasing the button. Since**do** will start the majority of actions, most of the time you simply have to point at objects on the screen and push the button.

The interpretation of a particular verb — choosing a specific action to go with it — is done by the object itself. As we mentioned above, each object’s definition contains a piece of code whose purpose is to choose and perform the action to be taken from the repertoire of actions that the object is capable of. This code, called the *object behavior module*, has several different entry points. A list of these entry points is kept as part of the object definition data structure. This list is indexed by verb.

Putting this all together, the operation of the interface (internally) is as follows: You select an object and an action using the joystick and button. The chosen object is looked up in the internal database of object descriptions. From this database entry is extracted the list of action entry points. The selected action (verb) is used as an index into this list to retrieve an address inside the object behavior module. This address is jumped to as a subroutine. The code there determines the appropriate behavior in the present context, executes this behavior, and returns.

The third component of the interface is a set of conventions and standards for the design and coding of the object behavior modules. Clearly, without defined uses, the verbs **do**, **go**, et al, are just meaningless labels. For the interface to be coherent and understandable, a set of rigorous conventions must be specified and then followed. Otherwise, you are thrust into a game of ‘‘guess-what-this-does’’ and the transparency of the interface is violated. We must allow players to develop an intuitive model of the controls so that they think about what they are doing rather than about the way things work.

Here are the conventional meanings for the five verbs:

go

Go is associated with motion. The basic action should be to go to, towards, or through the object or place designated.

If the focus of attention is the background, the specific point designated is significant. Pointing at a location below the horizon commands motion towards the indicated spot. Pointing at a location above the horizon commands motion to simply begin in the indicated direction.

If the focus of attention is not the background but instead a discrete object, the position of the avatar with respect to the object is significant. If the avatar is not adjacent to the object, it is commanded to move to the object's location so that it *is* adjacent to it. If the avatar is already adjacent to the object, it is commanded to go through or into the object, if that is appropriate. For example, if the object is the door to a house, the avatar should walk through the doorway into the house (presuming that the door is unlocked or the avatar has the key). As another example, if the object is a chair, the effect of the **go** action is to make the avatar sit (in the chair). If it is not appropriate to move through or into the object, then the intent is to push against the object from the direction that the avatar faces it.

The means of locomotion when moving is dependent on the state of the avatar. If the avatar is at the controls of a vehicle, then the vehicle is what should move, carrying the avatar (and any other cargo or passengers) with it. Ordinarily, however, the avatar will walk.

A **go** action (usually) starts motion. Once the motion is begun, control is returned to the player. The motion continues independently until it is completed or stopped. Motion may be stopped by explicit command or by encountering an impassable obstacle of some sort.

stop

Stop is associated with the cessation or cancellation of an action, particularly motion. The basic action is to stop whatever the avatar is doing and not do anything else.

Ordinarily, the focus of attention is irrelevant. Any ongoing actions (usually motion) are simply stopped if they are stoppable. **Stop** is also a w ay to prevent any action if you select the wrong object (e.g., your finger accidently slipped onto the joystick button before you pointed at the intended object).

get

Get is associated with obtaining items, delving into containers, receiving transactions, and doffing clothing. The basic action is to move some object from its present location into the avatar's hands or into some other appropriate receptacle.

To perform a **get** action, the focus of attention must be immediately adjacent to the avatar or on the avatar's person. If this is not the case, the **get** action automatically includes an implicit **go** action to the object's location. If the focus of attention is someplace on the background rather than a specific object, the **get** action has no effect.

If the focus of attention is an ordinary object that is small enough to be picked up and carried by the avatar, then the avatar is commanded to pick it up. To pick something up, however, the avatar's hands must be free (i.e., the avatar can't already be holding something else). If the object is too large or too heavy, the **get** action simply fails.

If the focus of attention is a container object, the interpretation depends on whether the container is *open* or *closed*. If the container is open, then the action is to get an item out of the container. If there is more than one item in the container, you are shown a view of the container's contents and given the opportunity to pick the item that is to be grabbed. If the container is closed, then the action is to pick up the container itself, unless the container is inherently not ''getable''. In such a case, the**get** action implicitly opens the container and then behaves the same as it would have if the container had been open all along. If the container is not openable by the avatar the **get** action fails.

If the focus of attention is somewhere on the avatar's person, the interpretation depends on what part of the body is pointed at. Pointing at the hip pockets (or at the backpack if the avatar is wearing one) causes the pockets (or pack) to be treated like an ''ungetable'' container, i.e., they are implicitly opened and their contents retrieved. Pointing at other parts of the body causes the garment associated with that part of the body (pants on the legs, shirt on the chest, hat on the head, etc.) to be removed and placed in the avatar's hands.

If the focus of attention is another avatar, the action is to accept something that the player controlling the other avatar will hand over by executing a **put** action. Your avatar will simply wait for the **put** action to occur. It will wait until either the **put** happens or the you execute some other action (in particular, **stop** can be used to cease such waiting).

put

Put is associated with placing, inserting, giving or throwing items, and with donning clothing. The basic action is to move an object in the avatar's hands to a location somehow suggested by the focus of attention.

To execute a **put** action, there must be some object present in the avatar's hands. If this is not the case, the **put** fails and no action is taken.

If the focus of attention is not immediately adjacent to the avatar nor on the avatar's person, the avatar is commanded to throw the object in its hands to or towards the location designated by the focus of attention.

If the focus of attention is a nearby location, the object in the avatar's hands is simply placed there. If the focus of attention is an ordinary object, the item in the avatar's hands is placed next to or on top of it (whether next to or on top of depends on the kind of object that is the focus of attention and on the kind of object being placed there).

If the focus of attention is a container, the object in the avatar's hands is placed inside it. If the container is closed it is opened implicitly. If the container cannot be opened or if the object being placed inside it is too large to fit, the **put** action fails with no effect.

If the focus of attention is somewhere on the avatar's person, the interpretation depends on what part of the body is pointed at. Pointing at the hip pockets (or at the backpack if the avatar is wearing one) places the item in the avatar's hands into the pockets (or pack) if it will fit (if not the action fails). Pointing at any other part of the body causes the garment in the avatar's hands to be donned on the appropriate part of the body (pants on the legs, shirt on the chest, hat on the head, etc.). If the item in the avatar's hands is not a garment, the action fails. If the avatar is already wearing a garment of the particular type, the garment being worn and the one in the hands are exchanged.

If the focus of attention is another (adjacent) avatar, the action is to hand the item in your avatar's hands to the other avatar. If the player controlling the other avatar has executed a **get** action directed at your avatar, the other player's avatar will be ready to receive the item and it will simply be transferred from your avatar's hands to his. If the other player has not executed a **get**, the item is placed at the other avatar's feet.

do

Do is associated with performing actions of all sorts. The basic action depends on the kind of object that is the focus of attention and sometimes also on the kind of object that is in the avatar's hands (if any).

Usually, to perform a **do** action the focus of attention must be immediately adjacent to the avatar or on the avatar's person. If this is not the case, the **do** action may automatically include an implicit **go** action to the object's location. If the focus of attention is someplace on the background rather than a specific object, the **do** action may be identical to **go**.

If some object is in the avatar's hands, it may be interpreted as an instrument or tool that is to be used or applied on, with, against, or to the object that is the focus of attention. For example, if the avatar is holding a gun, the effect of the **do** action is to fire the gun at the object that is the focus of attention. As another example, the ordinary interpretation of **do** when the focus of attention is a closed door is to open the door, but this action will fail if the door is locked. If, however, the object in the avatar's hands is the key to the door, then the door may be opened even if it is locked. Similarly, if the focus of attention is an open door, the action is to close it. If the avatar is holding the key to the door at the time, the door is closed and locked.

As mentioned, the interpretation of **do** depends on the specific object of attention. This can vary enormously. Since there is no specific rule, I'll simply provide as many examples as I can think of:

If the focus of attention is a door or a container of some sort, the basic action is to open it if it is closed or vice versa. Other objects that have a simple binary state associated with them, such as light switches, music boxes, recording machines, and so on, similarly have their state toggled.

Some objects can be discharged. Some of these objects require another object to be the target of their operation (such as the gun described above). In such a case, the object in the avatar's hand determines the action to be taken and the object that is the focus of attention determines the target. In addition to the gun, objects of this type might include paint sprayers (which change the color of the target), fasteners of various sorts (which attach to the target), and any other sort of tools that manipulate other objects in some manner.

Books: a **do** action applied to a book or other reading material opens the book for reading. The screen displays a view of the first page. Further **do** actions turn the pages. A **put** action is used to put down the book and return to the normal view.

Wastebasket: a wastebasket is a container, and you can put things in it or remove things from it using **put** and **get** just as you can with any other container. However, a **do** action applied to a wastebasket empties it, causing its contents to disappear (is this dangerous?).

Food, drink, and drugs: these items effect an avatar's physical state. A **do** action applied to such an item causes it to be consumed by the avatar.

Communications

In addition to the attention/action selection mechanism with the five essential verbs, an additional mechanism is provided for the purposes of communication between players. We define a sixth essential verb, **speak**.

Speak is different from the first five verbs in that it is not activated by selection with the joystick. Instead, **speak** is the implicit action whenever you type on the keyboard. What you type is transmitted to one or more receivers that are designated by the focus of attention.

If the focus of attention is another avatar, you are speaking to the player controlling it. What you type appears on the other player's screen along with a tag of some sort telling that the message is from you. If the focus of attention is the background near your avatar, you are speaking to any and all nearby avatars. This is just like speaking to an individual, except that there may be multiple recipients. If the focus of attention is the background distant from the avatar, your are speaking to a wider range of players. In essence you are shouting aloud to anyone within hearing distance.

Some objects possess an inherent communications capability. If the focus of attention is one of these objects, you are using that object to communicate to a more distant recipient or to command a complex device. For example, a telephone object lets you contact remote players (ones whose avatars are not near yours). A computer terminal object lets you interact in the traditional time-worn fashion with computers (real or simulated) and associated complicated software systems.

Gestures

The **speak** mechanism can be easily extended to include a range of gestures that avatars can exhibit. We simply define a bunch of gestures (wave the right hand, wave the left hand, give the finger, scratch the head, etc.) and then assign these to the function and control keys on the keyboard. E.g., pressing F3 makes the avatar nod his head; pressing Ctrl-B makes him point to the left. The gestures do not effect the actual state of the avatar but simply alter its graphical representation briefly to convey a simple message. Obviously, if the gesture is not appropriate (e.g., you can't wave when your hands are full) nothing happens.

Display

The above discussion is almost entirely concerned with the handling of player input. Of course, output is also part of the player interface. The detailed aspects of the display are more appropriately discussed in the document on **MicroCosm** graphics, but here is the general layout of things:

The screen is divided into two parts. The upper half contains the graphics window. This is a vistaVision format display running the full width of the screen. It is as high (and only as high) as it needs to be to maintain the proper aspect ratio (2:1?). The graphics window displays a series of fixed-viewpoint, third-person scenes showing the goings on in the world surrounding your avatar. Sometimes the normal display will be replaced by close-up views to show such things as pages of text in books or selections of items in a container.

The lower half of the screen is a text window. This is where communications with the other players takes place. Anything you type is echoed here (so that you can see what you're typing). Any messages from other players are displayed here. This window can be divided into full width subwindows. There will be one such subwindow for each player participating in a conversation, including yourself. Subwindows keep getting added (shrinking the others) up to the point where there are so many people talking that there is no room for more, at which point we have to start time-sharing them. Each subwindow carries a label telling whose speech it is displaying. The subwindows scroll as text is added. Once text scrolls off the top it simply disappears. Perhaps a tape recorder object can be created that will record conversations, but this would be an enhancement to the basic system rather than a fundamental feature.