Virtual Pet: @<https://www.cs.drexel.edu/~mp3492/PetProject.html>

Chart: 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [STATES] | FEED | PET | WALK | CLEAN |
| HUNGRY | eating | angry | angry | angry |
| SLEELING | eating | —— | —— | angry |
| NEUTRAL | eating | HAPPY | walking | angry |
| BATHROOM | angry | angry | walking | angry |
| HAPPY | eating | really happy | walking | soapy |
| DIRTY | —— | —— | —— | soapy |

Chart: 2

|  |  |
| --- | --- |
| 1 Second Action States | Result After Timeout() |
| walking | HAPPY or DIRTY or HUNGRY |
| eating | NEUTRAL or SLEEPING |
| angry | [return to prev. State] |
| really happy | HAPPY |
| soapy | NEUTRAL or HUNGRY |

Chart: 3

|  |  |
| --- | --- |
| 5 Second STATES | Result After Timeout() |
| NEUTRAL | HUNGRY or BATHROOM |
| HAPPY | NEUTRAL |
| SLEEPING | NEUTRAL or BATHROOM |

Back-Story:

The Dog is named Kanobi, we Adopted him at Green Hills farm. He is generic as dogs come. But he does like to stare out the window when he needs to go to the bathroom.

How My Code Works:

My state system has 2 main parts; STATES and ACTION STATES. To illustrate the map that the interaction of buttons and certain STATES create, I have drawn a diagram (see Chart 1). Every time the user presses a button, the STATE changes to a corresponding ACTION STATE (with exception to NUETRAL + Pet => HAPPY). I made this decision because it only made sense for the pet to perform the action before immediately going to the state that the button would inevitably result in. Every ACTION STATE will display for a second before switching the pet into the resultant STATE. STATES will remain constant for an indefinite amount of time with exception of 3 STATES: NEUTRAL; HAPPY; SLEEPING. After 5 seconds, the exceptions will resolve into another STATE.

If we were to look at the code, we would see that we’re are creating 2 “arrays” (one is an array, the other is a map). The first array named STATES, stores all of our STATES; you’ll notice that at the end of the list is a STATE called “Action.” This tells the code to instead look at our second array called SUBSTATES (You can also call this the ACTION STATE Array). The SUBSTATES array stores the action we are performing as a key, and stores the resultant of said action in its value (eg. ReallyHappy is the key, HAPPY is its value). We then have a function called “displayCurrentState” which does exactly what it sounds like, displays the current state to the virtual pet image (note if a parameter is passed to it, it’ll display the parameter instead of the current state.). We then have a function named “setAction,” this function displays the current action, and also sets up a timer to resolve that action into its value (see Chart 2). We then have 4 functions all named similar to the buttons, that’s because the buttons call their respective functions. These functions switch the states to an action. The very last function is called whether the current state is NEUTRAL, SLEEPING, or HAPPY; and which it’ll then set a 5 second timout to switch that state to another state (see Chart 3). I skipped over a few functions because they were more or less helper functions to make the code a little bit more readable. Of course not all of this taught in the course, but I’ve programmed before.