The thought-by-thought development of RPSSL

by Joe Horn

I'm going to need the list of game objects (rock, paper, etc) throughout the entire program, so let's put it on the stack right now, so that whenever we need a copy, it'll be floating on the stack:

```
{ "ROCK" "PAPER" "SCISSORS" "SPOCK" "LIZARD" }
```

That's kind of bulky, so from now on in the stack diagram I'll refer to the list as just {L}. The stack diagram will be shown on the left side of the page, with the program commands to its right in red.

Next I want to make a custom menu from this list. But if I just did TMENU now, I'd lose my only copy of {L}. So I make another copy:

```
{L} {L}                 DUP
```

Now I can do the TMENU. But I want a sixth button that says EXIT, so I add it to the end of {L}.

```
{L} {L} "EXIT"          "EXIT"
{L} {L "EXIT"}          +
{L}                     TMENU
```

I want to start the game with a little prompt message, the custom menu showing, and an otherwise clear screen. CLLCD clears the screen, "PRESS ONE!" would be a sufficient prompt message, and -1 WAIT does two things: it displays the current menu (in this case, our custom menu), and it pauses until a key is pressed, then returns the keycode and resumes program execution. The game should repeat until EXIT is pressed, so start the main loop here.

```
{L}                     CLLCD
{L} "PRESS ONE!"        "PRESS ONE!"
{L}                     1 DISP
{L} 11.1+               WHILE -1 WAIT
```

Keycodes for the menu keys are 11.1, 12.1, 13.1 and so on. To turn these into 1, 2, 3 and so on, we can just subtract 10.1. That yields the number of the object that the user picked, which I'll call X. It'll be 6 if they pressed EXIT.

```
{L} X                   10.1 -
```

Now I want to exit the game if EXIT was pressed.

```
{L} X X                 DUP
{L} X 1/0               5 ≥
{L} X                   REPEAT
```

At this point, I know that X is between 1 and 5, and stands for the game object that the user picked. Let's display it like this: "YOU: ROCK" or "YOU: PAPER", etc. All I have to do is get object X from {L}, append it to "YOU: ", and display it.

```
{L} X {L} X             DUP2
{L} X "X"               GET
{L} X "X" "YOU: "       "YOU: "
{L} X "YOU: " "X"       SWAP
{L} X "YOU: X"          +
{L} X                   1 DISP
```

Now I generate the HP's turn, which I'll call Y.

```
{L} X 0..1              RAND
{L} X Y                 5 * CEIL
```

Let's display it like this: "ME: ROCK" or "ME: PAPER", etc. I need to get object Y from {L}, append it to "ME: ", and display it.

```
{L} X Y {L}             3 PICK
{L} X Y {L} Y           OVER
{L} X Y "Y"             GET
{L} X Y "Y" " ME: "     " ME: "
{L} X Y " ME: " "Y"     SWAP
{L} X Y " ME: Y"        +
{L} X Y                 2 DISP
```

If X and Y are the same, then it was a tie. In that case, we can throw away X and Y, and say that it was a tie.

```
{L} X Y X Y             IF DUP2
{L} X Y T/F             SAME
{L} X Y                 THEN
{L} "TIE!"              DROP2 "TIE!"
```

Leave the message string on the stack; don't show it yet. If it wasn't a tie, then figure out who won. This is done by subtracting X-Y, taking it MOD 5 (to model it around a circle; see diagram elsewhere), and then check to see if it's even (X wins) or odd (Y wins).

```
{L} X-Y                 ELSE -
{L} MOD(X-Y,5)          5 MOD
{L} MOD(MOD(X-Y,5),2)   IF 2 MOD
{L} "YOU"               THEN "YOU"
{L} "I"                 ELSE "I" END
{L} "? WIN!"            " WIN!" + END
```

At this point, "YOU WIN!" or "I WIN!" or "TIE!" is on the stack.

Display it.

```
     «L»                          4 DISP
```

And the game is done. Go back to the beginning of the main loop. If the EXIT key was pressed, then the program jumps down to here:

```
     «L» 0                        END
```

Time to clean up the stack and restore the previous menu.

```
                                  DROP2 0 TMENU
```

And the program is finished!