CSE 3302/5307 Lab Assignment 2

Due July 22, 2015

Goals:

Understanding of JavaScript and compiler/interpreter concepts.

Requirements:

1. Extend the baseline PL/0 compiler/interpreter for:

   a. Emojis (or other Unicode symbols) entered through an additional textarea. Line breaks are not be used.

      1. The total number of Emojis should be available through the PL/0 global variable `emojicount`.
      2. An intrinsic procedure, `emojidraw(emoji#, x, y, size)`, should be available to allow PL/0 programs to place an Emoji on the canvas. The `emoji#` must be in the range 1 . . `emojicount`. `x` and `y` will be in canvas coordinates. `size` is the font size to be used just for the current call.
      3. The set of Emojis is fixed after the Run button is hit. Attempts to change the set at other times are to be ignored.

   b. Changing the canvas cursor to indicate the state of a PL/0 program during execution.

      1. `http://www.w3schools.com/jsref/prop_style_cursor.asp` has an introduction.
      2. When the interpreter is stopped, use `not-allowed`. (A call to `wait` is not considered stopped.)
      3. When the interpreter is blocked by a (synchronous) `cvclick` call, use `help`.
      4. When an asynchronous `cvasclick` handler is active, use `crosshair`.
      5. Otherwise, use `pointer`.

   c. In addition to `cvclick`, which causes the interpreter to block waiting for an `onclick` event on the canvas, provide a `cvasclick` procedure that does not block.

      1. When a PL/0 program calls `cvasclick`, both intrinsic global variables `cvclickx` and `cvclicky` should be assigned a negative value. In addition, an appropriate handler function should be assigned to `canvas.onclick`.
      2. When a mouse click occurs, the handler should set `cvclickx` and `cvclicky` to the event coordinates. In addition, `canvas.onclick` should be removed (`undefined`). A PL/0 program is expected to poll `cvclickx` to know that the asynchronous click occured. Since JavaScript has run-to-completion semantics, it is important for PL/0 programs using `cvasclick` to call `wait` in polling loops.
      3. It is a fatal error for a `cvclick` to be used while a `cvasclick` is still active.
      4. An alert should be given, but execution should continue, if another `cvasclick` occurs while a previous `cvasclick` is still active.

   d. All of the state machine functionality of the "IDE" must be preserved! Be sure to understand the state machine (`http://ranger.uta.edu/~weems/NOTES3302/BASELINE/DIAGRAMS/states.jpg` and `http://ranger.uta.edu/~weems/NOTES3302/LAB1FALL14/lab1fall14.pdf`), especially the code for `cvclick` and `wait`.

2. Submit your *zipped* files on Blackboard by 12:45 p.m. on July 22. Be sure to indicate the browser(s) you tested with.

Getting Started:

1. Useful files, including baseline code and HTML, are at `http://ranger.uta.edu/~weems/NOTES3302/BASELINE/`

2. Each Emoji in a string occupies *two* positions. If the Emoji textarea string has odd length, then use `prompt` to have the user correct the input.

3. Each `table.push()` in `compile()` is for an intrinsic name. Each `varible` (sic) is addressed in `block()`, `initMachine()`, and indirectly in `doInstructions()` through the run-time stack (`s[]`). Each `proc` has code in a large `switch` statement in `doInstructions()`. Those involving canvas output are the simplest, while those involving the interpreter state and events are more complicated. (Why do some have a `break` and others a `return` or `throw`?) Note that the by-value parameters for a procedure call are in stack positions `s[b + 3]`, `s[b + 4]`, `s[b + 5]`, etc.

4. Understand the processing for `cvclick` (along with `cvx` and `cvy`) before attempting `cvasclick`.