FIT100 – Fluency With Information Technology

Lab 11: A Basic Animation

Reference

This lab relies on the content of Chapters 21 and 22 in *FIT*, and the content of Lecture 15 (Wednesday).

Preparation

First, set up the HTML for a JavaScript program, as usual.

Second, get the two GIF files from the files page of the FIT100 homepage, `WhiteBox.gif` and `RedBox.gif`. Save them on your desktop under the names WhiteBox and RedBox.

Goal

The goal of the project is to do a simple animation. The screen will show a series of white boxes. This "box row" will first be filled with red boxes, then be emptied (as shown). The animation will operate not a 30 hertz, but at 120 hertz per frame, so that it moves more slowly than Smooth Action.

Recall the three steps of animations:

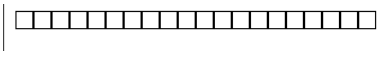
1) Place the initial image(s).
2) Prefetch the images that will over-write the initial images.
3) Set up a timer routine to perform the animation.

Exerices

1. **Place the initial images**. In the JavaScript, declare an iteration variable, `i`, and write a loop (WFI is best) that iterates 20 times. The statement that is iterated is a `document.write` to place the `WhiteBox.gif` image on the page. That is, the HTML to be written to the page would be,

```
<img src="White.Box.gif">
```

Remember that the `document.write` operation must enclose the HTML in quotes, and double quotes have already been used in the file name. (This operation will fill twenty indices (0-19) in `document.images`.)



2. **Prefetch**. There are only two images to prefetch, the white box and the red box. So, the array that you store the prefetched images in will only have two elements. Declare that array. Next, set up those two array elements so that they can store images, i.e. assign both elements the object: `new Image`. Notice that we have used a loop for this operation in Lecture 15, but since there are only two images to save, a loop is not necessary. Finally, assign the `.src` property of the 0th element `WhiteBox.gif` and the `.src` property of the 1th element `RedBox.gif`. Again, a loop is not appropriate.

3. **Set up the timer**. There are several substeps to this part.
a) The first step is to declare the `timerID` variable, as well as two other variables, `pos=0`, and `hue=1`. Notice the initializations. The variable pos will keep track of the position in the sequence of 20 boxes that is to be updated. The hue is to keep track of whether the position should be updated to be white or red.
b) Next, define the `animate()` function. Its goal is to fill in the next box in sequence. So, the variable `pos` must be incremented. To prevent it from getting too large, and to allow it to "wrap around to 0", the variable should limited to a maximum of 19 by the mod operator.
c) The `pos`th image in `document.images` should be updated to be the opposite color. The first time, this is red, i.e. the index 1 element of the prefetch array. Use the variable `hue` to refer to that element. (We have not changed colors from red back to white yet.)
d) The last step of the animation function is to set the timer to wake up in 120 milliseconds. (The needed statement will be similar to the one in step (e), but with a different time.) That completes the animation function for the moment.
e) At the end of the JavaScript start the timer. That is, place the statement

```
timerID = setTimeout("animate()", 1000);
```

which starts the timer one second after loading. Test the program.

4. **Alternate red and white**. So far the row of cells changes to red, but doesn't change back. To make it change back we need to change the element of the prefetch array that we use to update document images with. The easiest way to do this is to include an `if`-statement in the `animate()` function after the position has been incremented. If the `pos` is back to 0, then change `hue` to have the value `1-hue`. (Notice that if `hue` is 1 it changes to 0, and if `hue` is 0 it changes to 1.) Test the program.

5. **Embellishments**. There are a variety of changes that can be made to the program. The boxes can be placed vertically (put a `<br>` after each image), and the red can go up and down in the vertical stack, making it look like a thermometer. Notice that this means that the colors have to be drawn from the bottom up and returned to white from the top down. The time can be adjusted to make the thermometer operate slower or faster. Finally, the user might enter input (a number between 0 and 19) that is to be the maximum height of the colored boxes.