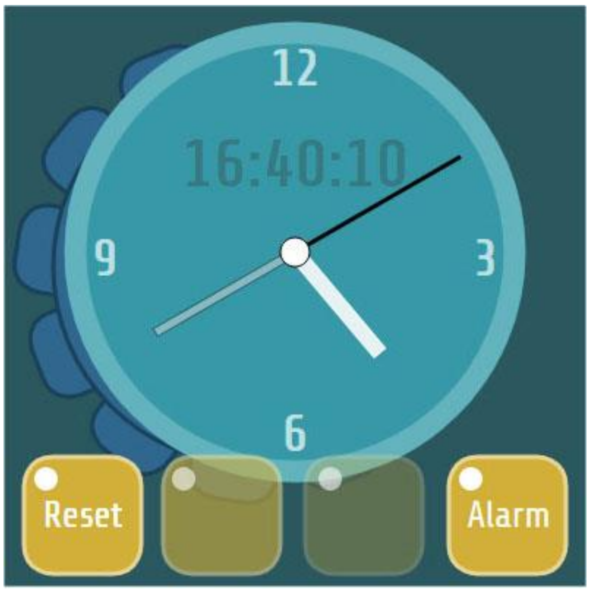
Multimedia Design and Applications

Assignment 01

Candidate No. 178492

Multimedia clock development in Javascript



Introduction

For this assignment I decided to create a simply styled clock face with a rotary control to set the time. An alarm function is to be implemented which can be set using the same rotary control and a multi use button. Controls with multiple use should limit the amount of buttons needed on the interface and therefore decrease interface clutter.

Media elements used

Most of the elements used for this assignment have been created with javascript using the scalable vector graphics in the ‘Raphael’ library. The shapes used for the clock face are rectangles and circles which are relatively easy to produce and transform using Raphael’s shapes creation methods. The buttons are slightly more complicated to produce. To achieve this I used a ‘path’ method which is a string of translations and vector coordinates that dictate the shape of the path.

The only bitmap used is the icon indicating that the alarm has been set. A bitmap was used instead of vectors because I thought that I would spend too much time working out the considerably more complex shape of the alarm bell graphic, but with more time, this graphic can be achieved using the path method in Raphael.

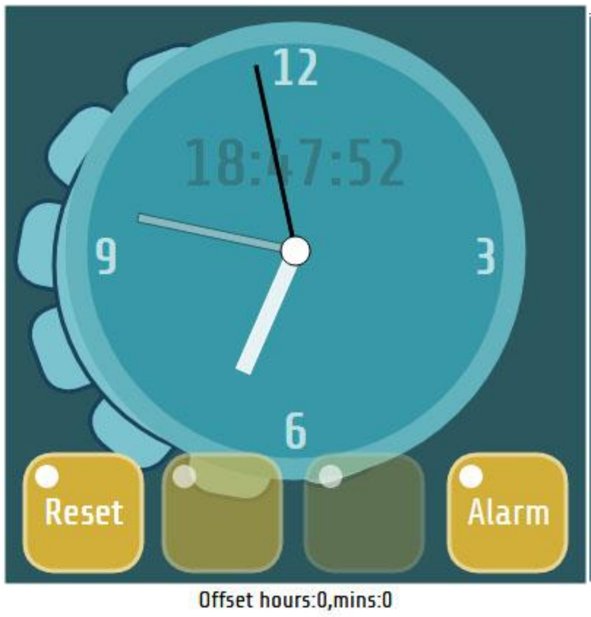
The typeface used is ‘Share Tech, sans-serif’ which is a free ‘Google Font’ linked to the web page and will only display if connected to the internet.

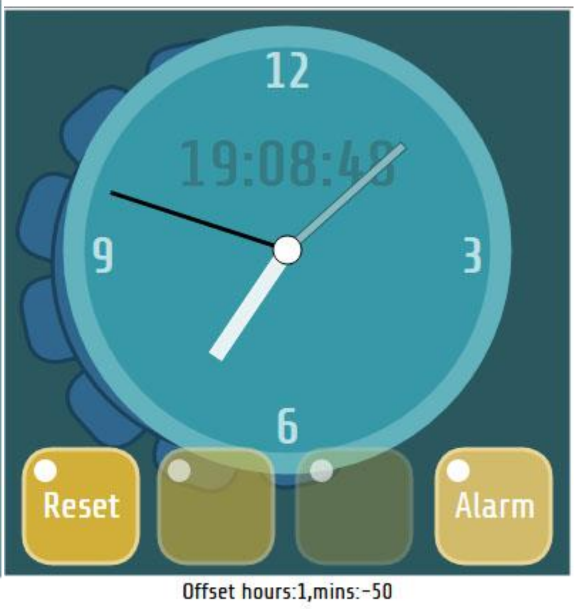
Javascript functionality

Once the basic clock face was set up visually, the time data is extracted from the system and the clock face updated to reflect the data retrieved in the ‘startTime()’ function. In the original code this data was updated every second, but I changed the ‘setTimeout()’ to call the ‘startTime()’ function every 300 milliseconds. This makes the interaction with the clock controls smoother. Because of the way Raphael draws its rectangles, I had to apply an offset to the hour, minute and second hands of 180 degrees so the analogue time read correctly.

The hours grabbed from the system are in 24 hour format so this translates straight onto the digital 24 hour read out. The numbers 1 to 12 are only needed for the analogue read out so this hour variable will cycle twice.

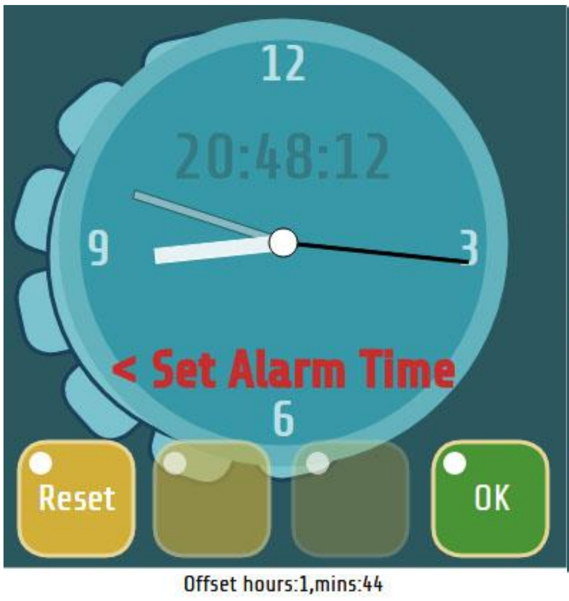
The first function I implemented was the ‘set time’ function. I wanted to use a rotary control system instead of increase and decrease buttons. The rotary control graphic itself is created using the path method in Raphael’s scalable vector graphics. All the vector elements used are then grouped in a ‘set’ which allows all the elements to be rotated around a common pivot point at once. The rotary control is animated using the mouse Y movement on the screen after it has been clicked. To grab the mouse events a separate ‘listener’ has been set up which overlays the rotary control graphic. This listener has all the ‘onmouse’ listeners assigned to it.

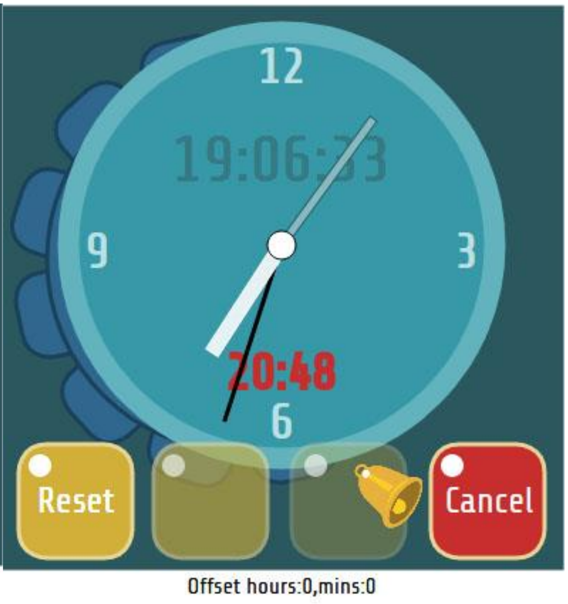




Using the mouse Y position, the difference between the initial Y position and the current Y position is used to calculate the rotation of the rotary control graphic. This difference is then translated to an offset adding to or subtracting from the angle of the minute and hour hands.The initial mouse down Y value is taken by using ‘event.clientY’ and passed to the function controlling the rotary movement. This functionality is terminated when the user releases the mouse click. To avoid weird behaviour the code stops updating the minutes and hour variables from the system while the mouse click down is held. As soon as the user releases the mouse click the code resumes updating the time variables from the system but with the off set variables added to the minutes and the hours.

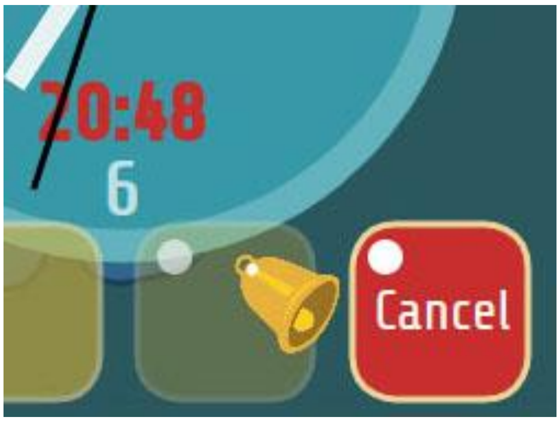
This ‘off set’ function came up against problems when the system time plus the off sets started to produce negative values when the clock was left to run freely for seemingly random lengths of time. A debug system was set up which catches any negative or over 60 minute values and logs variable values at the time of the error to help with a solution, but when these bad numbers are caught resetting the offset value appropriately seemed to solve the problem.





The second function I created was an alarm function. A button on the interface selects setting and ok's setting of the alarm time while the actual setting of the alarm time is controlled by the rotary function on the left side of the clock face.

The rotary control behaves in the same manor as when the user is setting the time. After the alarm button is pressed, rotating the rotary control selects an alarm time. When this is achieved, the alarm / ok button is pressed again which stores the alarm time and switches the clock face back to reading time from the system.



The program alerts the user that the alarm is set by displaying a bitmap icon on the interface. When the alarm is triggered the clock face flashes and a mp3 sound is played. The flashing is created by cycling through an array of colours set in the alarmClr[ ] array using a 'setTimeout' method. The colours are set using the relevant Raphael element's attributes method every 80 milliseconds. The sound is played by simply calling the 'play()' method in the audio object created. The audio object is created by instantiating from the Audio() class built into the Javascript library.

Finally there is a 'Reset' button to set everything to zero and sync back to the system time. This is a simple function and was found to be very handy while testing.

References

Most of the reference used for this assignment, outside of the initial code and lab tutorials, came from 'W3 Schools (www.w3schools.com)' and the forums on 'Stack Overflow (stackoverflow.com)'. I find these sources very helpful for syntax and answers to similar problems that I encountered throughout the assignment. The only images that are used in this assignment are original images created by myself.