Animating the Bingo Ball Matrix

The pseudo code below shows how a basic 9x10 (9 Rows of 10 balls) matrix is rendered using the HTML5 Canvas.

```
for (var i = 0; i < 9; i++) {
    for (j = 0; j < 10; j++) {
        //ctx code here than renders a single ball
        Increment x-ordinate
    }
    Reset x-ordinate
    Increment y-ordinate
}
```

I wanted to be able to delay each ball for added aesthetic effect. Now the only functions Javascript provides for timing is setInerval() and setTimeout(). So I created test.html and .js files to create this effect. I started with a single line of 10 balls. Let's assume the function is called drawCircle(). The code looked something like this;

```
for (j = 0; j < 10; j++) {
        setTimeout(function() {
                drawCircle(x, y ,r);
        }, 1000);
        Increment x-ordinate
}
```

I wrapped the drawCircle() function in an anonymous function as I was passing parameters to it. I then set the timeout to 1s (1000ms). What I then realised was that the setTimeout() function was 'non-blocking' and meant the loop would continue executing as normal. This meant that 10 x setTimeout() functions were being executed in mere milliseconds. Giving the impression that they were all rendering at the same time. So I multiplied the delay time by the loop variable i. This 'queued' the timers to execute every (1 x 1000), (2x 1000), (3x1000), (nx1000)… to give the impression they were rendering every 1 second. This can be seen demonstrated here: https://jsfiddle.net/TomPlum/w60x4e4f/

I then tried to replicate this effect with the full matrix by using the nested for loop again. But of course we need to be resetting the x-ordinate back to the start and incrementing our y-ordinate in the outer loop (I.e. every row of 10 balls). To achieve this I used setInterval() at the start of the renderMatrix() function to perform the required increments every time 10 balls had finished rendering. A demonstration can be seen here: https://jsfiddle.net/TomPlum/nozzsdk5/2/

Creating the Bingo Card Number Selector

I wanted to add extra user-interaction to the Bingo Game that went beyond the requirements specified in the Assignment Brief. Since I'd already covered the ball generation and display, I wanted to add a function that allowed a user to play themselves, instead of just having a ball generator. So I needed a way for users to click and toggle on/off a number of balls in a displayed matrix. I also added a button that generates 9 random balls as well.

I started by creating a test page (pickball.html and pickball.js) in which I copied my coloured ball matrix from the Assignment. I then wrote a function that stores the users cursor co-ordinates and passes them into a function I wrote called checkSelection(). I added an event listener '***myCanvas.addEventListener("mousedown", measureCursor, false);***' for this to work. The measureCursor() function can be seen below.

```
function measureCursor(e) {
   myCanvas_x = e.pageX;
   myCanvas_y = e.pageY;
   checkSelection(myCanvas_x, myCanvas_y);
}
```

So each time the user clicks, their cursor click position is passed into the checkSelection() function which uses a nested for loop to check a square area over each ball. A series of if-else statements then verifies which ball it's on and pushes it into an array called selection[]. The only issue with this was that the matrix I had rendered in the pickball.html page was at the top-left hand corner of the page. Therefore, since I was logging pageX and pageY, it worked relative the (0, 0) position the canvas started at. Since my Bingo Game uses a pop-up modal window at the start, the co-ordinates of the canvas were not (0, 0). I therefore had to modify my measure() cursor function to measure the cursor co-ordinates of the selected *canvas* not the current *page.* As seen here;

```
function measureCursor(e) {
   var rect = modal_canvas.getBoundingClientRect();
   modal_canvas_x = e.clientX - rect.left;
   modal_canvas_y = e.clientY - rect.top;
   checkSelection(modal_canvas_x, modal_canvas_y);
}
```

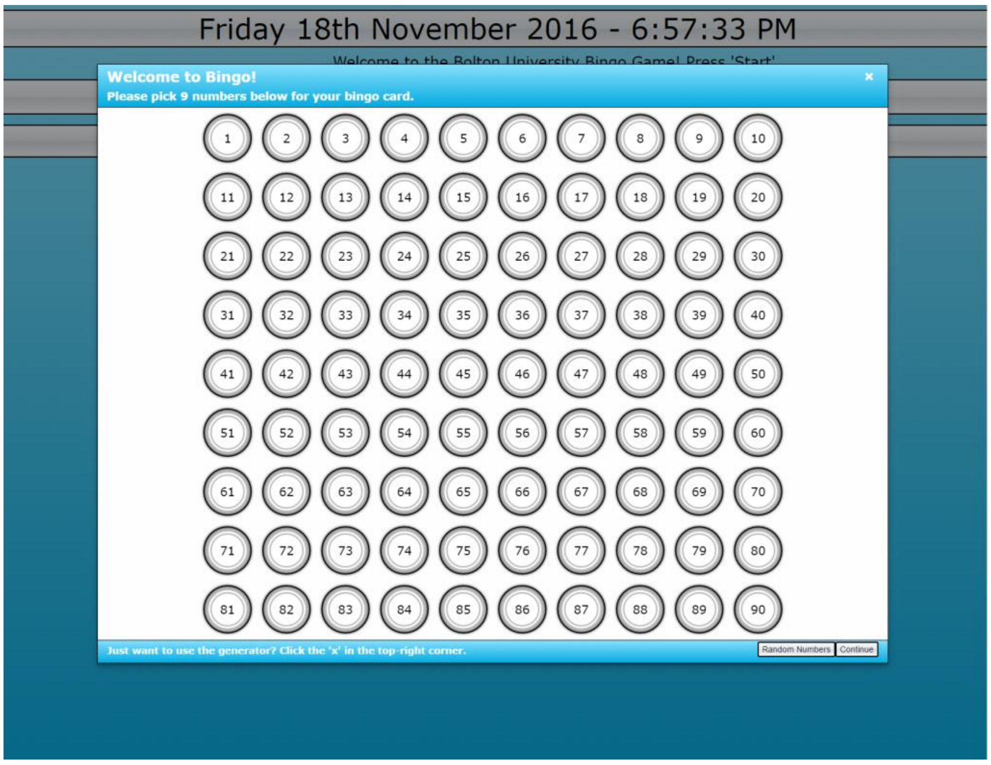
A working demonstration of pickball.html and pickball.js can be seen here: https://jsfiddle.net/TomPlum/off1rsgL/1/

This example does not 'grey-out' the ball on toggle, it merely adds/removes it from the array string at the bottom of the screen. As you will notice from the assignment, I then displayed the selected balls in a Number Card once the game is started.

Note:

Please note then when I was writing this Bingo Game, I was using a 3440x1440 21:9 monitor. The 1440p height gave me extra screen real-estate when designing it. It was only towards then end that I realised the project will be viewed on a 1920x1080 16:9 monitor. I've managed to compress a lot of the padding and margin in the CSS to ensure the matrix just fits in view on this resolution.

The pop-up modal on the load screen will now be usable if you scroll down to the continue button at the bottom-right hand corner. Although this may cause inconvenience and may be considered a problem, given the retirement home scenario in the assignment brief. I didn't have time to then add responsiveness to the page using bootstrap.

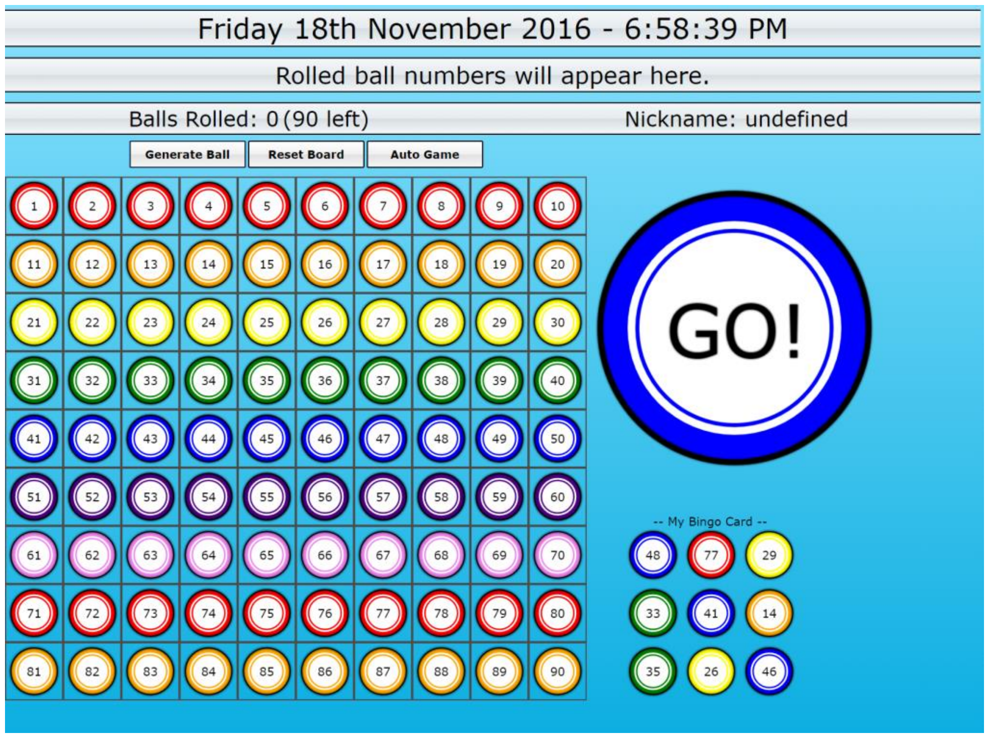


Friday 18th November 2016 - 6:57:33 PM

Welcome to the Bolton University Bingo Game! Press 'Start'

Welcome to Bingo!
Please pick 9 numbers below for your bingo card.

Just want to use the generator? Click the 'x' in the top-right corner.



Friday 18th November 2016 - 6:58:39 PM

Rolled ball numbers will appear here.

Balls Rolled: 0 (90 left)   Nickname: undefined