BUBU GAME

presented by

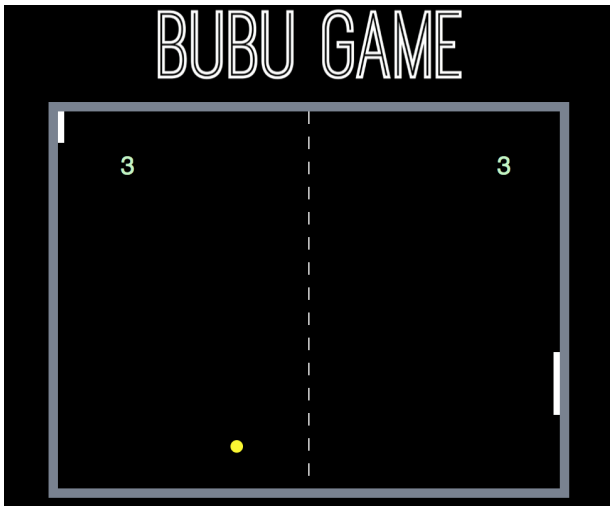
Nahuel Pregot

Final Project for CIS 61 Fall 2017 (Student ID: 10953303)

Table of contents

0.0 How to Run & Play the Game

Bubu is a simple straightforward game. The User will have to win against the computer to beat the machine in a paddle game, the first player scoring 5 points will be the winner of the match.



To play the game simply open the file Game.html with your favorite browser. The game will start immediately and you will control the left paddle with your mouse.

Please note that this game won't execute in Internet Explorer 8 or older, since they do not support canvas.

0.1 Files in the project

Game.html	Open this file to play the game
game.css	Some styling for the front end (for more details check out the front-end section)
js.js	File containing the code (for more details check out the back-end section)
title.jpeg	It is picture containing the title for BUBU GAME
Proposal_Bubu.pdf	Document outlining the project

1.0 The Back End (js.js file)

BUBU GAME is written from scratch without using game libraries or frameworks in JavaScript. All the gaming happens inside a canvas first defined in the Game.html file.

1.1 Variables

```
var canvas;
var canvasContext;
var ball_x = 400;  // position of the ball for the x coordinate
var ball_y = 300;  // position of the ball for the y coordinate
var ball_speed_x = 5;  // ball's speed for the x coordinate
var ball_speed_y = 5;  // ball's speed for the y coordinate
```

var canvas = Is the variable of the canvas, the program is executed inside of it
var canvasContext = Whatever happens inside the canvas

By modifying the ball speed for the coordinates X & Y will increase or decrease the difficulty of the game.

```
var paddle1Y = 250; // Initial position of paddle1 relative to the y
var paddle2Y = 250; // Initial position of paddle2 relative to the y
const PADDLE_HEIGHT = 100;
const PADDLE_THICKNESS = 10;
```

In the above is where the paddles are defined. “const” is used to determine a fix size for the paddles.

```
var playerLeft = 0; // Starting score left player
var playerRight = 0; // Starting score right player
const WINNING_SCORE = 5; // Score needed to win
```

The variables playerLeft/Right will store the score of the players starting at zero. WINNING_SCORE is a constant value that will not change during the execution.

```
var showingWinScreen = false; // Winning page showing the winner & option to play again
```

This variable will become true when the match has ended

1.2 Functions

1.21 Paddle functions

The movement of the paddles are controlled by two functions, one controlling the paddle moved by the User and the other managing the movement of the opponent paddle.

1.21a - calculateMousePosition( )

This function tracks the movement of the mouse.

```
// To allow the paddle to move with the mouse
function calculateMousePosition (evt) {
    var rect = canvas.getBoundingClientRect();
    var root = document.documentElement;
    var mouseX = evt.clientX - rect.left - root.scrollLeft;
    var mouseY = evt.clientY - rect.top - root.scrollTop;
    return {x:mouseX, y:mouseY};
    }
```

```
canvas.addEventListener('mousemove',
    function(evt) {
            var mousePos = calculateMousePosition(evt);
            paddle1Y = mousePos.y - (PADDLE_HEIGHT / 2);
```

The function is executed by creating an addEventListener with ‘mousemove’, assigning paddle1Y allows to control the left paddle. PADDLE_HEIGHT / 2 keeps the pointer in the middle of the paddle.

1.21b - computerMovement( )

The location of the right paddle will change as the ball moves across the field. In this case we only take in consideration the Y value. For each movement of the ball it will move 7 frames. The value of Y for the ball is stored in the variable ball_y. In the conditional - 35 is used to create a smoother movement of the right paddle.
By modifying the amount of frames moved by the right paddle (now 7) and the -35 will make the computer a stronger or weaker player.

```
// Function controlling the movement of the paddle handled by the computer
function computerMovement(){
    var paddle2YCenter = paddle2Y + (PADDLE_HEIGHT / 2);
     if(paddle2YCenter < ball_y - 35){
        paddle2Y += 7;
        }
    else if (paddle2YCenter > ball_y - 35){
        paddle2Y -= 7;
    }
}
```

1.22 The Ball

Here below are illustrated the functions managing & designing the ball.

1.22a - colorCircle( )

The circle is drawn using the mathematical formula to define a circle, centers are the coordinates defining the center of the ball for the respective x & y values.

```
// next function draws the ball
function colorCircle (centerX, centerY, radius, drawColor){
    canvasContext.fillStyle = drawColor;
    canvasContext.beginPath();
    canvasContext.arc(centerX, centerY, radius, 0, Math.PI*2, true);
    canvasContext.fill();
    }
```

1.22b - ballReset( )

Every time the ball is not caught by one of the paddles it will overpass the left or right side of the canvas and 1 point will be assigned to one of the players. Then the ball needs to be brought again in the middle of the field to continue the game.
The values of the variables “ball_speed_x”, “ball_x”, “ball_y” are changed in order to change the direction of the ball each time, this is a solution that avoids to create repetitiveness in the game.

```
// The ball is set to beginning position after a point is scored
function ballReset() {
    if(playerLeft >= WINNING_SCORE || playerRight >= WINNING_SCORE){
        showingWinScreen = true;

    }
    ball_speed_x = - ball_speed_x;
    ball_x = canvas.width/2;
    ball_y = canvas.height/2;

}
```

1.23 Elements of the canvas

The functions drawing the net and the rectangle of the game

1.23a - drawNet( )

A for loop is executed to repeatedly draw a series of lines in the middle of the field. The function is then called inside the drawEverything( ) function.

```
// For loop drawing the middle net
function drawNet(){
    for(var i=0; i<canvas.height; i+= 40)
    colorRect(canvas.width / 2 - 1, i, 2, 20, 'white');
}
```

1.23b - colorRect ( )

This function, as suggested by the name, creates rectangles. This function is used to draw the paddles and the rectangle where the game is happening. As for drawNet( ) and colorRect( ) is executed inside the function drawEverything( )

```
// "colorRect()" is the "template" function to draw elements in the canvas
function colorRect(leftX, topY, width, height, drawColor){
    canvasContext.fillStyle = drawColor;
    canvasContext.fillRect(leftX, topY, width, height);
    }
```

1.24 Main Functions

The code is composed by variables, and micro and macro functions. The Ones listed below are the main ones. The mini functions listed above are called inside the functions listed below.

1.24a - drawEverything( )

As a first step the function colorRect( ) is called to draw the black rectangle.

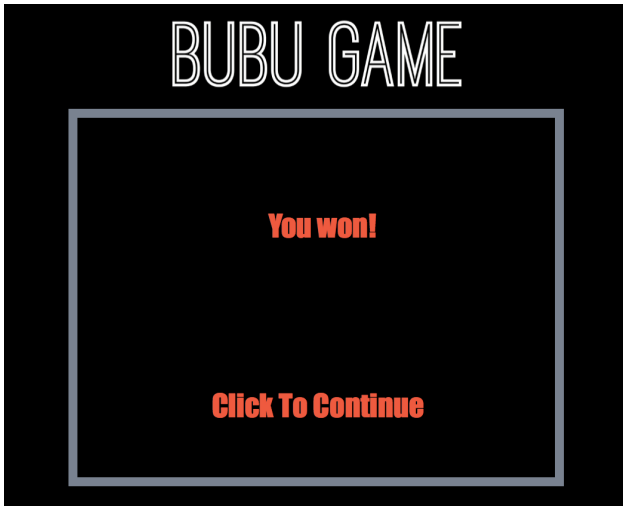
```
// IMPORTANT: For canvas' dimension look up Ref.1 in HTML file
colorRect(0,0,canvas.width,canvas.height, 'black');  // the canvas (black box)
```

Second, if “showingWinScreen” is set to true, the following code is executed and the below screen is displayed.

```
        if(showingWinScreen){

            canvasContext.font = "50px Impact";
            canvasContext.fillStyle = '#f45c42';
            if(playerLeft >= WINNING_SCORE) {canvasContext.fillText("You won!", canvas.width / 2 - 80, 200)
            else if(playerRight >= WINNING_SCORE) {canvasContext.fillText("You lost!", canvas.width / 2 - 8

            //canvasContext.fillStyle = 'red';
            canvasContext.fillText("Click To Continue", canvas.width / 2 - 175, 500);
            return;
        }
```



The conditional will display a different message depending on the result of the game. Their position and content is defined inside the methods, where their style is determined in .font and .fillStyle.

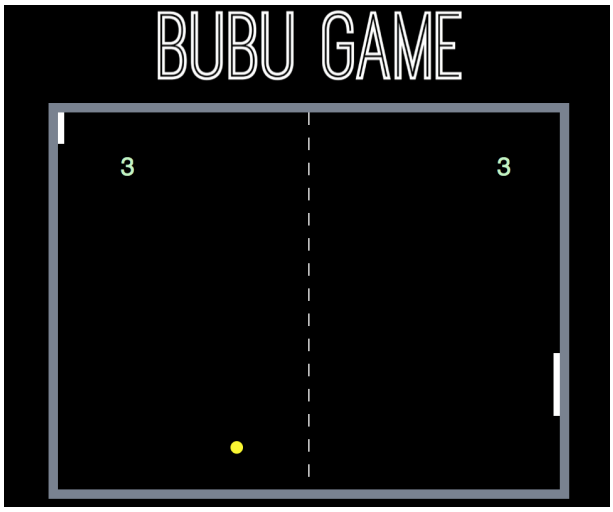
If “showingWinScreen” is false then the game is still playing. The function drawNet( ) displays the net, colorRect( ) is called to draw the paddles, colorCircle( ) to draw the yellow ball.

For canvasContext, .fillStyle, .font and .fillText will determine the location, content and style of the scoring points.

```
drawNet();

colorRect(0,paddle1Y,PADDLE_THICKNESS,100,'white');  // the white paddle on the left
colorRect(canvas.width - PADDLE_THICKNESS,paddle2Y,PADDLE_THICKNESS,100,'white'); // the white paddle o
// colorRect(ball_x,ball_y,25,25, 'red');  // the ball as rectangle(red)
colorCircle(ball_x, ball_y, 10, 'yellow');

canvasContext.fillStyle = '#c5f7c6';
canvasContext.font = "40px Helvetica";
canvasContext.fillText(playerLeft, 100, 100,);
canvasContext.fillText(playerRight, canvas.width - 100, 100);
}
```



1.24b - moveEverything( )

This function manages the logic of the movements happening inside the field. When the match has ended the “showingWinScreen” is set to true.

The ball has an initial position, then its position is increased each time. This is the logic behind the movement of the element. By increasing “ball_speed_x or y” it is increased the amount of frames that the ball moves each time and this results in a higher speed.

```
if(showingWinScreen){
    return;
    }
computerMovement();
ball_x += ball_speed_x;
ball_y += ball_speed_y;
```

The direction of the ball needs to be inverted each time it touches one of the sides, or it needs to get into reset mode each time it reaches the left or right walls.

This is when the ball touches the bottom or the top of the field

```
    }
    // bounces the ball from the BOTTOM side of the canvas
    if (ball_y > canvas.height - 5){
        ball_speed_y = - ball_speed_y;



    }
    // bounces the ball from the TOP side of the canvas
    if (ball_y < 5){
        ball_speed_y = - ball_speed_y;
```

Managing when the ball touches the left or right sides is a little bit more complex.
When the ball touches the paddle its direction needs to be inverted.

If the ball touches the extremes of the paddle its speed is increased. This solution allows the player to have more control on the direction and speed of the game, and it is more fun and challenging.

Instead if the ball reaches the ends without getting hit by the paddle, the “else” condition is executed. A point is recorded for one of the players and then the ball is brought back in the middle of the field with the ballReset ( ) function.

```
if (ball_x > canvas.width - 5){  // "canvas.width" is the right limit x = 800
    // ball_speed_x = - ball_speed_x;
    if(ball_y > paddle2Y && ball_y < paddle2Y + PADDLE_HEIGHT){
        ball_speed_x = - ball_speed_x;

        var deltaY = ball_y - (paddle2Y + PADDLE_HEIGHT / 2);
        ball_speed_y = deltaY * 0.35;


        }
    else {
        playerLeft ++; // must be BEFORE ballReset()
        ballReset();
        }
```

```
// bounce the ball from the LEFT side of the canvas
if (ball_x < 5){ // "0" is left limit x = 0

    if(ball_y > paddle1Y && ball_y < paddle1Y + PADDLE_HEIGHT){
        ball_speed_x = - ball_speed_x;

        var deltaY = ball_y - (paddle1Y + PADDLE_HEIGHT / 2);
        ball_speed_y = deltaY * 0.35;

        }
    else {
        playerRight ++; // must be BEFORE ballReset()

        ballReset();

        }
```

Another way to increase the difficulty of the game is by increasing the value of 0.35, which is increase of speed of the ball when hitting the edges of the paddle.

1.24c - window.onload function ( )

This is a function acts like a container of all the functions. As suggested by the name is executed when the page is loaded.

It displays the canvas by selecting its id and it runs functions inside of the elements. The function “setInterval” defines the speed of execution in milliseconds (set at 7), smaller the value higher is the speed of execution for the functions drawEverything ( ) and moveEverything( ).

As explained before the EventListeners are used to track the click of the mouse when starting a new match and track the movement of the mouse to control the paddle used by the User.

```
window.onload = function() {
    // "gameCanvas" is the id of element in HTML file, look up Ref.1
    canvas = document.getElementById('gameCanvas');
    // getContext() will allow to create elements in the canvas
    canvasContext = canvas.getContext('2d');
    // setInterval() calls functions every so often,
    // "7" is interval in milliseconds
    setInterval(function(){
            drawEverything();
            moveEverything();
    }, 7);

    canvas.addEventListener('mousedown', handleMouseCLick);

    canvas.addEventListener('mousemove',
        function(evt) {
                var mousePos = calculateMousePosition(evt);
                paddle1Y = mousePos.y - (PADDLE_HEIGHT / 2);
    });
```

2.0 The Front End (game.css file)

Majority of the front end is defined within the JavaScript code. The content and some styling are instead set in the file game.css file, here below there is a screenshot of the content of the file.

Each parameter is explained with a comment, therefore no further explanation is required here in the documentation.

```
html {
    background-color: black; /* gives a black color to the html page*/
}

#title {
    display: block; /* To display the element in center of the page*/
    margin: auto; /* To display the element in center of the page*/
}

#gameCanvas {
    padding: 15px; /* Thickness of the grey frame around the canvas */
    display: block; /* To display the element in center of the page*/
    margin: 0 auto; /* To display the element in center of the page*/
    background-color: #7c8593; /* The color of the frame  */

}
```