```
-----------------------------------
Appendix II : HTML5 board viewer
View dependencies & input file setup at
https://github.com/gabyhaddock/Submarine/tree/master/html
-----------------------------------



<html>
<head>
<script src="sub.js" > </script>
<script src="handlebars-v1.3.0.js"> </script>
<script src="jquery-1.11.0.min.js"> </script>
<link rel="stylesheet" type="text/css" href="playSub.css" />

</head>
<body>

<div style="display:none;">
<img id="starter-board" src="starter-board.jpg" />
<img id="fire-token" src="fire-token.png" />
<img id="high-flood-token" src="high-flood-token.png" />
<img id="low-flood-token" src="low-flood-token.png" />
<img id="hatch-block-token" src="fire-token.png" />
<img id="gnome" src="red-star2.png" />
</div>


<script type="x-template" id="game-state-template">
    <div class="game-state">
        <div class="header">
            <ul>
                <li>Start in room: {{startRoom.number}} </li>
                <li>End in room: {{finalRoom.number}}</li>
                <li>Total cost: {{totalCost}}</li>
            </ul>
        </div>
        <canvas width="1024" height="512" style="float:left;"> </canvas>

        <table class="moves-list" style="float:left;">
            <tbody>
                <tr><th>Action</th><th>Cost</th> </tr>
                  {{#each actions}}
                <tr>
                    {{#ifMove}}
                        <td>Move to Room {{room.number}} </td> <td> {{cost}}  </td>
                    {{else}}
                        {{#ifFlood}}
                            <td colspan="2">Flood rooms {{room1.number}} and
{{room2.number}} </td>
                        {{else}}
                            <td>Open Hatch {{hatch.number}} </td>  <td> {{cost}} </td>
                        {{/ifFlood}}
                    {{/ifMove}}
```

```
 <!--                <td>{{this.type}} </td> <td> {{this.cost}} </td> -->
                </tr>
                  {{/each}}
                <tr><th>Total cost:</th> <th> {{totalCost}} </th> </tr>
            </tbody>
        </table>


    </div>
</script>


    <div id="nav">
        Move to room  (# of choices) :
        <ul id="nav-list">
         <!--  <li> <a href="#results-1">1</a>  (1) </li>-->
        </ul>
    </div>

<div id="starter-sub" >
    <div class="header">
             <ul>
                <li>Starter submarine</li>
            </ul>
    </div>
    <canvas id="canvas" width="1024" height="512""></canvas>
</div>


<div id="results-1">

</div>

<div id="results-2">
</div>

<div id="results-3">
</div>

<div id="results-4">
</div>

<div id="results-5">
</div>

<div id="results-6">
</div>

<div id="results-7">
</div>

<div id="results-8">
</div>
```

```
<div id="results-9">
</div>

<div id="results-10">
</div>

<script>


    var allRoomCoords = [
                        {x: 250, y: 275} , //1
                        {x: 365, y: 215},     //2
                        {x: 350 , y: 350 },   //3
                        {x: 510 , y: 390 },   //4
                        {x: 555 , y: 245 },   //5
                        {x: 585 , y: 160 },   //6
                        {x: 620 , y: 390 },   //7
                        {x: 710 , y: 260 },   //8
                        {x: 735 , y: 370 },   //9
                        {x: 855 , y: 320 }   //10
                    ] ;

    var allHatchCoords = {
        "1-2" : {x: 320, y: 280},
        "1-3" : {x: 320, y: 355},
        "2-3" : {x: 375, y: 320},
        "2-4" : {x: 480, y: 340},
        "2-5" : {x: 480, y: 270},
        "3-4" : {x: 480, y: 410},
        "4-5" : {x: 525, y: 320},
        "5-6" : {x: 615, y: 245},
        "5-7" : {x: 655, y: 320},
        "5-8" : {x: 700, y: 270},
        "7-8" : {x: 700, y: 345},
        "7-9" : {x: 700, y: 405},
        "8-9" : {x: 790, y: 330},
        "8-10" : {x: 830, y: 300},
        "9-10" : {x: 830, y: 370}
    }



var board = document.getElementById("starter-board");
var fireToken = document.getElementById("fire-token");
var highFloodToken = document.getElementById("high-flood-token");
var lowFloodToken = document.getElementById("low-flood-token");
var hatchBlockToken = document.getElementById("hatch-block-token");
var gnome = document.getElementById("gnome");

//Set up the handlebars template for the game state
var gameStateTemplate = $("#game-state-template").html();
var compiledTemplate = Handlebars.compile(gameStateTemplate);

Handlebars.registerHelper('ifMove', function(options) {
```

```
    if(this.type == "Move") {
        return options.fn(this);
    } else {
        return options.inverse(this);
    }
});

Handlebars.registerHelper('ifFlood', function(options) {
    if(this.type == "Flood") {
        return options.fn(this);
    } else {
        return options.inverse(this);
    }
});

Handlebars.registerHelper('actionRow', function(options) {
    if(this.type === "Move") {
        return options.fn(this);
    } else if(this.type === "OpenHatch") {
        return options.fn(this);
    } else if(this.type === "Flood") {
        return options.fn(this);
    } else {
        console.log("Could not understand type " + this.type);
    }
});

function renderSub(context, gameState) {
    //Draw the game board
    context.drawImage(board, 0, 0);

    var rooms = gameState.rooms;
    var hatches = gameState.hatches;

    //Render rooms
    for (var i = 0; i < rooms.length ; i++) {
        var roomState = rooms[i].state;
        var roomCoords = allRoomCoords[rooms[i].number - 1];  //uses list index to get the
coords from allRoomCoords


        switch(roomState)
        {
            case "Clear" :
                break;
            case "Fire" :
                context.drawImage(fireToken, roomCoords.x, roomCoords.y);
                break;
            case "HighFlood" :
                context.drawImage(highFloodToken, roomCoords.x, roomCoords.y);
                break;
            case "LowFlood" :
                context.drawImage(lowFloodToken, roomCoords.x, roomCoords.y);
                break;
```

```
         }
    }

    //Render hatches
    context.font = "bold 40px sans-serif";
    context.textAlign = "center";

    for (var i = 0; i < hatches.length ; i++) {
        var hatchState = hatches[i].state;
        var hatchCoords = allHatchCoords[hatches[i].number];

        switch (hatchState)
        {
            case "Open":
                context.fillStyle = 'green';
                context.fillText("O", hatchCoords.x, hatchCoords.y);
                break;
            case "Closed":
/*              --Do nothing? */
                break;
            case "Blocked":
                context.fillStyle = 'red';
                context.fillText("X", hatchCoords.x, hatchCoords.y);
                break;

        }
    }

    //Draw the gnome
     var roomCoords = allRoomCoords[gameState.finalRoom.number - 1];  //uses list index to
get the coords from allRoomCoords
     context.drawImage(gnome, roomCoords.x, roomCoords.y);

    //Render the path for the actions

    //Add a placeholder function for browsers that don't have setLineDash()
    //From : http://www.rgraph.net/blog/2013/january/html5-canvas-dashed-lines.html
    if (!context.setLineDash) {
        context.setLineDash = function () {}
    }


    var actions = gameState.actions;

    if (actions.length > 0) {
        context.beginPath();
            //Start the line at the coordinate for the first room, which is not in the
action list
            var firstCoord = actionToCoord(gameState.startRoom.number);
            context.moveTo(firstCoord.x, firstCoord.y);

            //Draw lines to the rest of the coordinates
            for (var i = 0; i < actions.length; i++) {
                var coord = actionToCoord(actions[i]);
```

```
                if (coord) {
                    context.lineTo(coord.x, coord.y);
                }
            }

            context.strokeStyle = "#D2EDFC";
            context.lineWidth = 5;
            context.setLineDash([5]);
        context.stroke();
    }


}

function renderResult(gameState) {

    //Bind the gameState object to the handlebars template (defined globally) to show
labels and the list of moves.
    var boundHTML = compiledTemplate(gameState);

    //Insert into the document in the results list that corresponds to the final room
    var template = $(boundHTML).appendTo($("#results-" + gameState.finalRoom.number));

    //After binding with handlebars, render the sub and path on the canvas
    var canvas = template.find("canvas")[0];
    var context = canvas.getContext("2d");

    renderSub(context, gameState);



}

function actionToCoord(action) {
    if (action.type === "Move" ){
        var roomCoords = allRoomCoords[action.room.number - 1]; //use list index in
allRoomCoords
        //For rooms, the path should be adjusted so it ends up in the center of the icon
        return {x : roomCoords.x + 25, y : roomCoords.y + 25 };
    }
    else if (action.type === "OpenHatch"){
        var hatchCoords = allHatchCoords[action.hatch.number];
        return {x : hatchCoords.x, y : hatchCoords.y };
    }
    else if (action.type === "Flood"){
        return null; //Detect this return type in the rendering
    }
    else if (action.type === undefined){ //Can pass in just a number with no type - in this
case, interpret as a room number
        var roomCoords = allRoomCoords[action- 1]; //use list index in allRoomCoords
        //For rooms, the path should be adjusted so it ends up in the center of the icon
        return {x : roomCoords.x + 25, y : roomCoords.y + 25 };
    }
    else {
```

```
        console.log("Could not understand action type " + action.type);
    }
}

window.onload = function(){

    //Render the starter submarine
    var canvas = $("#starter-sub canvas")[0];
    var context = canvas.getContext("2d");
    renderSub(context, starterSub);

    //Render final game states (with path and description of actions
    $.each(results, function(){
        renderResult(this);
    });


  //  var resultsSummary = [];

    for (var i = 1; i <= 10 ; i++){
        var numResults = $("#results-" + i + " .game-state").length;
        if (numResults > 0) {
             //   resultsSummary.push({room = i, count = numResults});
                var link = '<a href="#results-' + i + '">' + i + '</a> '
                link = link + '(' + numResults + ') '
                $("<li>").appendTo("#nav-list").append(link);
        }
    }
}


</script>
</body>
</html>
```