<script>

    import { onMount } from 'svelte';

    import { fade } from 'svelte/transition';

*let* players = [];

*let* ticks = [];

*let* windowHeight = 300;

*let* playerNumber = 100;

*let* startingAmount = 1000;

*let* worldrecord = 0;

*let* highestNumber = 0;

*let* maxWager = 20;

    export *let* round = 0;

    export *let* roundLimit;

*let* chartWidth;

    export *let* increment;

*let* running = true;

*let* stepHeight = 600;

*let* stepWidth = 600;

*let* panelHeight = stepWidth \* 0.8;

*let* chartHeight = panelHeight - 50;

*function* generatePlayers() {

        players = [];

        for (*let* i in [...Array(playerNumber).keys()]) {

*let* player = {

                "wealth": startingAmount,

                "id": i,

                "order": 0,

                "height": 0,

                "is\_it\_me": 0

            }

            players.push(player);

        }

    }

*function* playGame() {

        round = 0;

        setInterval(*function*() {

            running = false;

            if (round < roundLimit) {

                running = true;

                for (*var* i = 0; i < increment; i++) {

                    playRounds();

                }

                sortPlayers();

            } else if (round != roundLimit) {

                round = 0;

            }

        }, 10);

    }

*function* playRounds() {

        players = shuffle(players); // randomize the array order to get random pairings

        for (*let* i = 0; i < players.length; i+=2) { // iterate through all 50 games. each game gets players[i] and players[i+1].

*let* player1 = players[i];

*let* player2 = players[i+1];

*let* playerOrder = player1.wealth > player2.wealth ? [player1, player2] : [player2, player1]; // array with [wealthierPlayer, poorerPlayer]

*let* wager = playerOrder[1].wealth \* maxWager/100; // the second player in the playerOrder array is poorer

*let* payouts = Math.random() > 0.5 ? [wager, -wager] : [-wager, wager]; // randomize who wins, and return array of payouts

            players[i].wealth = players[i].wealth + payouts[0];

            players[i + 1].wealth = players[i + 1].wealth + payouts[1];

        }

        round+= 1;

    }

*function* sortPlayers() {

        worldrecord = 0;

        highestNumber = 0;

        players.sort(dynamicSort("wealth"));

        players.forEach(*function*(*d*) {

            if (d.wealth > worldrecord) {

                worldrecord = d.wealth;

            }

        });

        // setting the ticks based on the highest numerical value

        if (worldrecord > 20000) {

            highestNumber = 120000;

        } else if (worldrecord > 9000) {

            highestNumber = 25000;

        } else if (worldrecord > 4800) {

            highestNumber = 10000;

        } else if (worldrecord > 2400) {

            highestNumber = 5000;

        } else {

            highestNumber = 2500;

        }

        ticks = [];

        for (*var* i = 0; i < highestNumber; i += highestNumber/5) {

*const* tick\_y = getHeight(i);

            ticks.push([i, tick\_y]);

        }

        for (*var* i = 0; i < players.length; i++) {

            players[i].is\_it\_me = i == 0 ? 1 : 0;

            players[i].order = i;

            players[i].height = chartHeight - getHeight(players[i].wealth);

            players[i].height = players[i].height < 2 ? 2 : players[i].height;

        }

    }

    $: {

        stepWidth = stepWidth;

        stepHeight = stepHeight;

        panelHeight = stepWidth \* 0.85;

        chartHeight = panelHeight;

        increment = increment;

        roundLimit = roundLimit;

        generatePlayers();

        sortPlayers();

        playGame();

    }

    // UTILTIES

*function* dynamicSort(*property*) {

*let* sortOrder = 1;

        if(property[0] === "-") {

            sortOrder = -1;

            property = property.substr(1);

        }

        return *function* (*a*,*b*) {

*let* result = (a[property] < b[property]) ? -1 : (a[property] > b[property]) ? 1 : 0;

            return result \* sortOrder;

        }

    }

*function* getHeight(*h*) {

        return chartHeight - (h / highestNumber \* chartHeight);

    }

*function* shuffle(*array*) {

*let* currentIndex = array.length,  randomIndex;

        while (currentIndex != 0) {

            randomIndex = Math.floor(Math.random() \* currentIndex);

            currentIndex--;

            [array[currentIndex], array[randomIndex]] = [

                array[randomIndex], array[currentIndex]];

        }

        return array;

    }

*function* comma(*x*) {

        return x.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",");

    }

</script>

<div class="ysm\_container interactive\_container"  bind:clientHeight={stepHeight} bind:clientWidth={stepWidth}>

    <div class="chartArea" bind:clientWidth={chartWidth} style="height:{panelHeight}px;">

        {#if chartWidth > 0}

        <svg>

            {#each ticks as tick}

            <line x1=0 x2={chartWidth} y1={ tick[1] } y2={ tick[1] }></line>

            <text class="chartText" x=0 y={tick[1] - 5}>${comma(tick[0])}</text>

            {/each}

            {#each players as player}

            <rect class="player player{player.order}" x={player.order \* ((chartWidth-50) /playerNumber) + 50 } width={chartWidth / 200} height={player.height} y={chartHeight - player.height}></rect>

            {/each}

            {#if !running}

                {#each players as player}

                    {#if player.order == 0}

                    <text class="player1Text" x={player.order \* ((chartWidth-50) /playerNumber) + 45 } y={chartHeight - player.height - 7} in:fade={{ delay: 200 }}>Poorest: ${comma(Math.round(player.wealth))}</text>

                    {/if}

                    {#if player.order == 99 }

                    <text class="player2Text" width={200} x={player.order \* ((chartWidth-50) /playerNumber) + 50} y={chartHeight - player.height - 7} in:fade={{ delay: 200 }}>Richest: ${comma(Math.round(player.wealth))}</text>

                    {/if}

                {/each}

            {/if}

        </svg>

        {/if}

    </div>

    <div class="toolbar ysm\_data">

        <div class="toolItem roundItem">

            <div class="toolLabel">Round: <strong><span class="toolValue">{comma(round)}</span></strong> </div>

        </div>

    </div>

</div>

<style>

    .toolbar {

*position*: absolute;

*left*:  10px;

*top*: 20px;

*width*: 100%;

*height*: 20px;

*margin-top*: 10px;

    }

    .toolLabel {

*display*: inline-block;

*font-size*: 16px;

*line-height*: 1.3em;

    }

    .ysm\_container { *padding*:  20px; *width*: 100%; *height*:  100%;}

    .chartArea {  *width*: 100%; *margin-bottom*: 10px; }

    svg { *width*: 100%; *height*: 100%; }

    .ysm\_container svg .player1Text, .ysm\_container svg .player2Text {

*stroke*: #c8becf;

    }

    .toolLabel { *margin-bottom*: 20px; }

    rect.player {

*fill*:  var(--category-purple2);

    }

</style>