```
<!DOCTYPE html>
<html>
<head>


<!-- ********************************************************************************************************************** -->
<!-- *                                                                                                                    * -->
<!-- *                                      _____                 _ _ _                                                   * -->
<!-- *                                     / ____|               | (_) |                                                  * -->
<!-- *                                     | |     _ __ ___   __| |_| |_ ___                                             * -->
<!-- *                                     | |    | '__/ _ \ / _` | | __/ __|                                            * -->
<!-- *                                     | |____| | |  __/ (_| | | |_\__ \                                            * -->
<!-- *                                     \_____|_|  \___|\__,_|_|\__|___/                                            * -->
<!-- *                                                                                                                    * -->
<!-- *  ____________________________________________________________________________________________________________  * -->
<!-- *  |                                          JAVASCRIPT                                                      |  * -->
<!-- *  | Game development library (devlib.js) provided by our teacher                                             |  * -->
<!-- *  | Various stackoverflow answers:                                                                           |  * -->
<!-- *  |   | https://stackoverflow.com/questions/4565112/javascript-how-to-find-out-if-the-user-browser-is-chrome     |  * -->
<!-- *  |   | https://stackoverflow.com/questions/109086/stop-setinterval-call-in-javascript                           |  * -->
<!-- *  |   | https://stackoverflow.com/questions/7812514/drawing-a-dot-on-html5-canvas                                |  * -->
<!-- *  |   | https://stackoverflow.com/questions/22097747/how-to-fix-getimagedata-error-the-canvas-has-been-tainted-by-    |  * -->
<!-- *  |                                                                                  cross-origin-data/58648632#58648632 |  * -->
<!-- *  |   | https://stackoverflow.com/questions/20058242/html-5-canvas-causes-browser-to-lag-when-switching-tabs     |  * -->
<!-- *  |                                                                                                          |  * -->
<!-- *  |__________________________________________________________________________________________________________|  * -->
<!-- *                                                                                                                    * -->
<!-- *  ____________________________________________________________________________________________________________  * -->
<!-- *  |                                            CSS                                                           |  * -->
<!-- *  | Flashing play again button: https://codepen.io/arximughal/pen/LEpoOJ                                     |  * -->
<!-- *  | Fonts: my own font library - https://jackcrane.github.io/static/cdn/jfonts.css                           |  * -->
<!-- *  |__________________________________________________________________________________________________________|  * -->
<!-- *                                                                                                                    * -->
<!-- *  ____________________________________________________________________________________________________________  * -->
<!-- *  |                                           Images                                                         |  * -->
<!-- *  | All images were made by me in Piskelapp or MS Paint                                                      |  * -->
<!-- *  |__________________________________________________________________________________________________________|  * -->
<!-- *                                                                                                                    * -->
<!-- *                                                &                                                                   * -->
<!-- *                                            .((((&@@@                                                               * -->
<!-- *                    *&@@@@@@@@@@,            ,@@@@@@@@@@,                                                            * -->
<!-- *                %@@@@@@@@@@@@@@@@@@@*       #@@@@@@@@@@                          LUNAR LANDER                       * -->
<!-- *             .@@@@@@@&%&@@@@@@/     &@@@@.  . @@@@@@@&.                                                             * -->
<!-- *            @@@@@@@%     ,@@@%     &@@% /@@@%                          Created Under the instruction of Mike Thomas    * -->
<!-- *          .@@@@@@@@@     @@@@@@@@@@ /@@@@%                             At the Summit Country Day School                * -->
<!-- *         @@@@@@@@@@@@@@@@@@@@@@,.@.,@@@ &@@%                                                                      * -->
<!-- *        @@@@    &@@@@@@@@@@@@ (@@@@%  @@@@@@@                          AP Computer Science Principles                  * -->
<!-- *        @@@       &@@@@@@@@@* @@@@@# &@      .@@&                                                                   * -->
<!-- *        @@@@&. *@@@@@@@@@ ,@@@@& *@@@@      @@@.                                                                  * -->
<!-- *        *@@@@@@@@@@,   %@@@@, @@@@@@@@@@@@@@@@@%                          Sophomore Year                                  * -->
<!-- *          @@@@@@#     @@@@( &@@@@@@@@@@@@@@@@@@&                                                                   * -->
<!-- *                (@@@%   @@@@@@@@@@      &@@@@#                                                                    * -->
<!-- *              @@@@    &@@@@@@@@@@%      @@@@                                                                      * -->
<!-- *            ,@@@,    @@@@@@@%*/@@@@@&(#@@@@@.                                                                      * -->
<!-- *           /@@#      @@@@@@@      @@@@@@@@@@                                                                      * -->
<!-- *         (@%         #@@@@@@%    #@@@@@@@&                                                                        * -->
<!-- *                        #@@@@@@@@@@@@%.                                                                            * -->
<!-- *                                                                                                                    * -->
<!-- *                                                                                                                    * -->
<!-- ********************************************************************************************************************** -->


    <title>Jack Crane | Lunar Lander</title>
    <!-- <script src="https://jackcrane.github.io/static/cdn/jconsole.js" type="text/javascript"></script> -->
    <script src="devlib.js" type="text/javascript"></script>
    <link href="https://jackcrane.github.io/static/cdn/jfonts.css" rel="stylesheet">
    <!-- <link href="https://fonts.googleapis.com/css?family=Roboto+Mono&display=swap" rel="stylesheet"> -->
    <!-- <script src="cdlImage.js" type="text/javascript"></script> -->
    <style>
        a {
            color:inherit;
            border:1px solid;
            padding:1%;
            margin:1%;
            background-color:transparent;
            cursor:pointer;
        }
        #canvas {
```

```
            border:1px solid;

        }
        .canvasParent{
            /*box-shadow: -1px 0px 9px 3px #FFF;*/
            overflow:auto;
            display:inline-block;
        }
        .blinking {

            padding:4% 45%;
            background-color:white;
            -webkit-animation: glowing 1500ms infinite;
            -moz-animation: glowing 1500ms infinite;
            -o-animation: glowing 1500ms infinite;
            animation: glowing 1500ms infinite;
        }
        @-webkit-keyframes glowing {
          0% { background-color: #B20000; -webkit-box-shadow: 0 0 3px #B20000; }
          50% { background-color: #FF0000; -webkit-box-shadow: 0 0 40px #FF0000; }
          100% { background-color: #B20000; -webkit-box-shadow: 0 0 3px #B20000; }
        }

        @-moz-keyframes glowing {
          0% { background-color: #B20000; -moz-box-shadow: 0 0 3px #B20000; }
          50% { background-color: #FF0000; -moz-box-shadow: 0 0 40px #FF0000; }
          100% { background-color: #B20000; -moz-box-shadow: 0 0 3px #B20000; }
        }

        @-o-keyframes glowing {
          0% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
          50% { background-color: #FF0000; box-shadow: 0 0 40px #FF0000; }
          100% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
        }

        @keyframes glowing {
          0% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
          50% { background-color: #FF0000; box-shadow: 0 0 40px #FF0000; }
          100% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
        }
    </style>
    <script>
        var darkMode = true
        function darkModeToggle() {
            if(!darkMode) {
                document.body.style.backgroundColor='black';
                document.body.style.color='white';
                darkMode = true;
                ctx.filter = 'invert(100%)';
                document.getElementsByClassName('canvasParent')[0].setAttribute('style','box-shadow:-1px 0px 9px 3px white')
            } else {
                document.body.style.backgroundColor='white';
                document.body.style.color='black';
                darkMode = false;
                ctx.filter = 'invert(0%)';
                document.getElementsByClassName('canvasParent')[0].setAttribute('style','box-shadow:-1px 0px 9px 3px black')
            }
        }

var isChromium = window.chrome;
var winNav = window.navigator;
var vendorName = winNav.vendor;
var isOpera = typeof window.opr !== "undefined";
var isIEedge = winNav.userAgent.indexOf("Edge") > -1;
var isIOSChrome = winNav.userAgent.match("CriOS");

if (isIOSChrome) {
   // is Google Chrome on IOS
} else if(
  isChromium !== null &&
  typeof isChromium !== "undefined" &&
  vendorName === "Google Inc." &&
  isOpera === false &&
  isIEedge === false
) {
    alert("Unfortunately (and i have no idea why) this game does not work in Google Chrome. Please switch to Firefox. The game will show in Edge, but is unplayable")
} else {
   // not Google Chrome
}
```

```
        // console.image("https://raw.githubusercontent.com/jackcrane/jackcrane.github.io/f1dfd25b6d300ee740261084215121eb2dc35511/JS.jpg")
    </script>
</head>
<body style="/*background-color:black;color:white;*/font-family:'Space Age';transition:.2s">
<center><h1>Moon Lander</h1></center>

<center><div class="canvasParent"><canvas id="canvas" width="800" height="520" style="transition:.2s"></canvas></div></center>
<br><br>
<br><br>
<p id="sconsole" style="font-family:sans-serif;display:none"></p>
<div style='display: none'>
  <img id='background' src='images/terrain_v2_dark.png'>
  <img id="lander" src="images/spritesheets/ll_eng_idle.png">
</div><br>
<p style="font-family:sans-serif;">To allow audio: go to about:preferences#privacy in a new tab then scroll to "autoplay" (a little more than halfway down the page) and click "settings". Either whitelist this page, or clic
<a class="blinking" id="playAgain" hidden onclick="document.location.reload()">Play again!</a>
<script src="baseDeclaration.js" type="text/javascript">// Explanation inside file.</script>
<script>

    // Variable declration

    var speedMultiplier = 1

    var paused = false

    var alive = true;

    var keys = new KeysPresses()

    var killGameInstance = false;

    var drawInt = null;

    var won = false

    var thrusterSound;

    document.addEventListener("keydown", keyDownHandler, false);
    document.addEventListener("keyup", keyUpHandler, false);
    document.addEventListener("mousemove", mouseMoveHandler, false);
    document.addEventListener("mousedown", mouseDownHandler, false);
    document.addEventListener("mouseup", mouseUpHandler, false);
    document.addEventListener("mouseclick", mouseClickHandler, false);


    function keyDownHandler(e) {
          keys.processDownKey(e);
    }
    function keyUpHandler(e) {
        keys.processUpKey(e);
    }
    function mouseMoveHandler(e) {
        keys.processMouseMove(e);
    }
    function mouseUpHandler(e) {
        keys.processMouseUp(e);
    }
    function mouseDownHandler(e) {
        keys.processMouseDown(e);
    }
    function mouseClickHandler(e) {
        keys.processMouseClick(e);
    }

// BELOW: The ugly base code:
// This was the only way i could get to work so the color detection could work, as chrome was being annoying and "protecting" me from cross-origin images.

    var backgroundImg = BackgroundPre;
    var lander = {
        engUp : '',
        engDown : '',
        engIdle : EngIdlePre,
        lander:null,
        engUpSS:null,
        engDownSS:null,
        engIdleSS:null,
        frameSD:0,
        frameSkip:0,
```

```
}

var gameStats = {
    frameCount:0,
}

var background = new Image();
background.crossOrigin = "Anonymous"
background.src = backgroundImg;

var health = {
    remainingFuel : 1000,
    remainingRetro : 500,
    remainingOxygen : 1000,

}

var deathCause = null;

var randomNumber = 0;

var canvas = document.getElementById('canvas');
var ctx = canvas.getContext("2d");
ctx.crossOrigin = "Anonymous";

var game = new GameMaster();

// Method creation

function drawText() {
    var text = new GameText();
    text.font = "35px Arial";
    if(!darkMode) {
        text.fillStyle = "Black"
    } else {text.fillStyle = "White"}
    text.x = 100;
    text.y = 100;
    text.draw("Loading...");
}

function killPlayer() {
    health.remainingOxygen = 1;
    // used for debugging
}

function drawScoreText() {
    var text = new GameText();
    text.font = "20px Roboto Mono";
    text.fillStyle = "white";

    text.x = 10;
    text.y = 20;
    if(gameStats.frameCount%2||gameStats.frameCount%3) {
        text.draw("Remaining:");
    }

    text.x = 25;
    text.y = 45;
    if(gameStats.frameCount%2||gameStats.frameCount%3) {
        text.draw("Fuel: " + health.remainingFuel + "");
    }

    text.x = 25;
    text.y = 70;
    if(gameStats.frameCount%2||gameStats.frameCount%3) {
        text.draw("Retro: " + health.remainingRetro);
    }

    text.x = 25;
    text.y = 95;
    if(gameStats.frameCount%2||gameStats.frameCount%3) {
        text.draw("Oxygen: " + health.remainingOxygen);
    }

    text.x = 600;
    text.y = 20;
    text.draw("Frames: " + gameStats.frameCount);
}

```

```
    var fullY;
    var colDetX;

    function getCrashedByColor(x,y) {
        // Desperate attempt
        x = Math.round(x);
        y = Math.round(y);
        // console.log("Checking "+x+","+y)
        var pixelData = ctx.getImageData(x , y, 5, 5).data;
            var r = pixelData[0]
            var g = pixelData[1]
            var b = pixelData[2]
            var rgb = r.toString()+g.toString()+b.toString()
            // console.log(r.toString()+g.toString()+b.toString())
            // console.log("R=" +pixelData[0] + " G=" + pixelData[1] + " B = " + pixelData[2])
            if(r==g && g==b && r!=0) {
                return "crashed"
            } else if(rgb==2372836) {
                return "landed"
            }
    }

    function colorColDetection(x,y,lander) {

        // very computationally heavy way to determine collision based on color

        fullY = y + 30;
        colDetX = x;



        if(getCrashedByColor(colDetX,fullY-6)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY-6)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX,fullY-5)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY-5)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX,fullY-4)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY-4)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX,fullY-3)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY-3)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX,fullY-2)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY-2)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX,fullY-1)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY-1)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+1,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+2,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+3,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+4,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+5,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+6,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+7,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+8,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+9,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+10,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+11,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+12,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+13,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+14,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+15,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+16,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+17,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+18,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+19,fullY)=="crashed"){console.log("crashed");alive=false;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+20,fullY)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+20,fullY-1)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+20,fullY-2)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+20,fullY-3)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+20,fullY-4)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        if(getCrashedByColor(colDetX+20,fullY-5)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}
        // if(getCrashedByColor(colDetX+20,fullY-6)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fullY)=="landed") {console.log("landed");won=true;}

        // var pixelData = ctx.getImageData(colDetX , fullY, 1, 1).data;
//           var r = pixelData[0]
//           var g = pixelData[1]
//           var b = pixelData[2]
//           console.log("R=" +pixelData[0] + " G=" + pixelData[1] + " B = " + pixelData[2])


    }

    function drawOutline() {
        // used to debug where collision detection would be watched for


        // ctx.fillStyle = "red"
```

```
        // ctx.fillRect(colDetX,fullY-6,2,2);
        // ctx.fillRect(colDetX,fullY-5,2,2);
        // ctx.fillRect(colDetX,fullY-4,2,2);
        // ctx.fillRect(colDetX,fullY-3,2,2);
        // ctx.fillRect(colDetX,fullY-2,2,2);
        // ctx.fillRect(colDetX,fullY-1,2,2);
        // ctx.fillRect(colDetX,fullY,2,2);
        // ctx.fillRect(colDetX+1,fullY,2,2);
        // ctx.fillRect(colDetX+2,fullY,2,2);
        // ctx.fillRect(colDetX+3,fullY,2,2);
        // ctx.fillRect(colDetX+4,fullY,2,2);
        // ctx.fillRect(colDetX+5,fullY,2,2);
        // ctx.fillRect(colDetX+6,fullY,2,2);
        // ctx.fillRect(colDetX+7,fullY,2,2);
        // ctx.fillRect(colDetX+8,fullY,2,2);
        // ctx.fillRect(colDetX+9,fullY,2,2);
        // ctx.fillRect(colDetX+10,fullY,2,2);
        // ctx.fillRect(colDetX+11,fullY,2,2);
        // ctx.fillRect(colDetX+12,fullY,2,2);
        // ctx.fillRect(colDetX+13,fullY,2,2);
        // ctx.fillRect(colDetX+14,fullY,2,2);
        // ctx.fillRect(colDetX+15,fullY,2,2);
        // ctx.fillRect(colDetX+16,fullY,2,2);
        // ctx.fillRect(colDetX+17,fullY,2,2);
        // ctx.fillRect(colDetX+18,fullY,2,2);
        // ctx.fillRect(colDetX+19,fullY,2,2);
        // ctx.fillRect(colDetX+20,fullY,2,2);
        // ctx.fillRect(colDetX+20,fullY-6,2,2);
        // ctx.fillRect(colDetX+20,fullY-5,2,2);
        // ctx.fillRect(colDetX+20,fullY-4,2,2);
        // ctx.fillRect(colDetX+20,fullY-3,2,2);
        // ctx.fillRect(colDetX+20,fullY-2,2,2);
        // ctx.fillRect(colDetX+20,fullY-1,2,2);
    }

    function dir(a) {
        return game.imageDir + a;
    }

    function makeBackgroundOnce() {
        // var background = new Image();
        // background.src = backgroundImg;
        // var rand = randomNumber;

        // THIS DARN SCROLLING BACKGROUND WONT START AT THE RANDOM NUMBER

        let bg = new ScrollingBackGround(randomNumber,0,13000,2080,4550,520,background);
        bg.draw();

    }

    function makeBackground() {
        makeBackgroundOnce()
    }

    function calcLife() {
        if(gameStats.frameCount%13===0) {
            health.remainingOxygen--;
        }
    }

    function aliveF() {
        if(health.remainingOxygen >= 0) {
            return true
        } else {
            return false
        }
    }

    function deathCauseFunct() {
        if(health.remainingOxygen <= 0) {
            return "ran out of oxygen"
        }
        console.log('deathCause run')
    }

    function drawDeathScreen() {
        var text = new GameText();
        text.font = "50px Hacked";
```

```
        text.fillStyle = "white";
        text.x = 10;
        text.y = 120;
        text.align = 'center';
        text.draw("You crashed!")
        document.getElementById("playAgain").hidden = false
        // console.log(text)
    }

    function drawVictoryScreen() {
        // function that shows text congratulating winner
        var text = new GameText();
        text.font = "50px Hacked";
        text.fillStyle = "white";
        text.x = 10;
        text.y = 120;
        text.align = 'center';
        text.draw("You won!")
        document.getElementById("playAgain").hidden = false
        // console.log(text)
    }

    function createLander() {
        var img = new Image();
        img.crossOrigin = "Anonymous"
        img.src = lander.engIdle;

        thrusterSound = new sound("sound/lander.mp3")

        // console.log(img)
        lander.engIdleSS = new SpriteSheet (100,100,256,256,50,50, 0,0,img);
    }

    function checkCollission(x,y) {
        canvas.crossOrigin = "Anonymous";
        lander.lander.crossOrigin = "Anonymous";

        ctx.crossOrigin = "Anonymous";
        // var hex = getColor(x,y)

        // console.log(lander.lander.width)
    }

    function sconsole(e) {
        document.getElementById("sconsole").innerHTML = e
    }

    function landerRefresh() {



        lander.lander = lander.engIdleSS

        if(keys.upArrow) {
            // lander.lander.y = lander.lander.y - 0.1
            var tempX = lander.lander.x
            var tempY = lander.lander.y
            // lander.lander = lander.engUpSS;
            lander.lander.x = tempX
            lander.lander.y = tempY

            if(gameStats.frameCount % 5 == 0) {
                if(health.remainingFuel >= 0) {
                    health.remainingFuel -= 1
                }
            }

        } else  {
            // lander.lander = lander.engIdleSS;
            // lander.lander.currentFrame = 0;
        }

        if(!keys.upArrow) {
            // lander.engUpSS.currentFrame = 0;
            lander.lander.y = lander.lander.y + 0.1 * speedMultiplier
            // console.log(lander.lander.y)
        } else {
            if(health.remainingFuel >= 0) {
                lander.lander.y = lander.lander.y - 0.1 *speedMultiplier
```

```
            thrusterSound.play()
        } else {
            lander.lander.y = lander.lander.y + 0.1 *speedMultiplier
        }
    }

    if(keys.leftArrow) {
        lander.lander.x = lander.lander.x - 0.1 *speedMultiplier
        // console.log(lander.lander.dX)

    } else if (keys.rightArrow) {
        lander.lander.x = lander.lander.x + 0.1 *speedMultiplier
    }

    lander.frameSD ++
    if(lander.frameSD == 50) {
        lander.frameSkip ++
        lander.lander.advanceFrame()
        sconsole(lander.lander.x)
        colorColDetection(lander.lander.x,lander.lander.y,lander)
        // console.log(lander.frameSD)
        lander.frameSD = 0;

    }




}

function checkDeadKeys() {
    if(keys.key_P){}
}

// createLander()

function draw() {
    gameStats.frameCount++;
    ctx.fillStyle = "black";
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    calcLife();
    if(aliveF()&&alive==true&&won==false) {
        drawText();
        makeBackground();
        drawScoreText();
        landerRefresh();

        lander.lander.draw()
        checkCollission(lander.lander.x,lander.lander.y)
        ctx.fillStyle = "red";
        drawOutline();
    } else if(alive==false) {
        drawDeathScreen();
        checkDeadKeys();
        clearInterval("drawInt")
    } else if (won==true) {
        drawVictoryScreen()
        clearInterval("drawInt")
    }

    if(killGameInstance) {
        clearInterval("drawInt")
    }

    drawInt = setInterval(draw, 1000/30)



}

    makeBackgroundOnce()
    createLander()

    draw();
```

```
        randomNumber = Math.floor(Math.random() * 1200) + 1;

    darkModeToggle()



    // ***************************************************************************************************************************************************************
    // *                                                                                                                                                             *
    // *                                    The following is a Javascript Game Development library written by our teacher                                            *
    // *                                                                                                                                                             *
    // ***************************************************************************************************************************************************************

function GameMaster(){
    this.imageDir = ".\\images\\";
    this.soundDir = ".\\sounds\\";
    this.libraryDir = ".\\lib\\";
    this.CurrentScreen = "MainMenu";
    this.debugstr = "";
    this.bgDX = 0;
    this.bgDY = 0;
    this.gameIterator=0;
    this.gameIterate = function(){
        this.gameIterator++;
        //** just in case of an overflow
        if (this.gameIterator < 0){
            this.gameIterate = 0;
        }
    }
    this.moveObjectWithBackground = function(o){
        o.x += this.bgDX;
        o.y += this.bgDY;
        if (o.x < o.useWidth*-1){
            o.x = canvas.width;
        }
    }
    this.debugAppend = function(str){
        this.debugstr = this.debugstr +  str
    }
    this.debug = function(){
        document.getElementById("debug").innerHTML = this.debugstr;
        this.debugstr = "";
    }
}
function GameText(){
    this.font = "48px Arial";
    this.fillStyle = "Blue";
    this.x = 800;
    this.y = 45;
    this.alpha = 0.5;
    this.draw = function(inText) {
        ctx.font = this.font;
        ctx.fillStyle = this.fillStyle;
        ctx.globalAlpha = this.alpha;
        ctx.fillText(inText, this.x, this.y);
        ctx.globalAlpha = 1.0;
    }
}
// For creating a single image Sprite
function Sprite (x,y,width,height, useWidth, useHeight,image) {
        this.x = x;
        this.y = y;
        this.dX = 0;
        this.dY = 0;
        this.width = width;
        this.height = height;
        this.useWidth = useWidth;
        this.useHeight = useHeight;
        this.image = image;
        this.alpha = 1.0; //transparency of image whe drawn
        this.collision = false;
        this.visible = true;
        this.moveWithBackground = false;

        this.draw = function() {
            if (this.visible) {
                if (this.moveWithBackground = true){
                    game.moveObjectWithBackground(this);
                }
                ctx.globalAlpha = this.alpha;
```

```
                ctx.drawImage(this.image,0,0,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
                ctx.globalAlpha = 1.0;
            }
        }
        this.checkCollisions = function(obj2) {
            var bot = BottomCollision(this, obj2);
            var top = TopCollision(this,obj2);
            var left = LeftCollision(this,obj2);
            var right = RightCollision(this,obj2);
            if ( bot || top || left || right){
                this.collision = true;
                return true;
            } else {
                this.collision = false;
                return false;
            }
        }

        //This one just checks if "this" moves one more spot down whether its bottom
        //will collide with the passed object
        this.checkBottomCollision = function(obj2) {
            return BottomCollision(this, obj2);
        }
        //This one just checks if "this" moves one more spot up whether its top
        //will collide with the passed object
        this.checkTopCollision = function(obj2){
            return TopCollision(this,obj2);
       }
        //This one just checks if "this" moves one more spot right whether its right
        //will collide with the passed object
        this.checkRightCollision = function(obj2) {
            return RightCollision(this,obj2);
        }
        //This one just checks if "this" moves one more spot right whether its left
        //will collide with the passed object
        this.checkLeftCollision = function(obj2) {
            return LeftCollision(this,obj2);
        }
    }
// For creating a Sprite that uses a spritesheet for motion
// note - assumes each row represents a direction and each column a frame
function SpriteSheet (x,y,width,height, useWidth, useHeight, dir, frame, image) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.useWidth = useWidth;
        this.useHeight = useHeight;
        this.dX = 0;
        this.dY = 0;
        this.dirMax = dir; //Max number of "directions" the Sprite can move in
        this.currentDir = 0; //start the Sprite with the first "direction"
        this.maxFrame = frame; //Max number of Frames the Sprite animation takes
        this.currentFrame = 0; //start on current Frame
        //this.sheet = image; //deprecated sheet - was inconsistent with other objects
        this.image = image;
        this.alpha = 1.0; //transparency of image whe drawn
        this.collision = false;
        this.visible = true;
        this.moveWithBackground = false;
        this.canJump = false;
        this.gravity = 0; //How fast he falls - should be set for any character that can fall/jump
        this.isJumping = false; // set to true once jump starts and is not set back nutil is "grounded" - this keeps from jumping in mid-air
        this.jumpMax = 40;
        this.jumpCount = 0; // Sets the number of refresh cycles the character can jump - once set to zero (probably by some collision) -  this value is set to zero

        this.jump = function(){
            if (this.canJump && this.isJumping == false) {
                this.isJumping = true;
                this.jumpCount = this.jumpMax;
            }
        }
        this.stopJump = function(){
            this.jumpCount = 0;
            myKeys.space = false;
        }
        this.touchingColor = function(ctx,r,g,b){
            var data = ctx.getImageData(this.x,this.y,this.width,this.height);
            console.log(data)
```

```
            for(var i=0;i<data.length;i+=4){
                if(
                    data[i+0]==r&&
                    data[i+1]==g&&
                    data[i+2]==b
                ){
                    return true;
                }
            }
            return false;
        }

        this.checkJump = function(){
            if (this.isJumping && this.jumpCount > 0){
                this.y -= (this.gravity + 1);
                this.jumpCount --;
            } else{
                this.stopJump();
            }
        }
        this.applyGravity = function(){
            this.y += this.gravity;
        }
        this.draw = function() {
            if (this.visible) {
                ctx.globalAlpha = this.alpha; //This command sets the opacity to half - meaning you can see though any images painted after this
                ctx.drawImage(this.image,this.currentFrame * this.width,this.currentDir * this.height,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
                ctx.globalAlpha = 1.0; //Setting the opacity back to 1 - which mean solid
            }
        }
        // Increments the Sprite animation frame.  If already at the max goes back to zero position
        this.advanceFrame = function() {
            this.currentFrame++;
            if (this.currentFrame > this.maxFrame) {
                this.currentFrame = 0;
            }
        }
        // Changes the Sprite "direction" or row of animation - will only change to a valid value
        this.changeDir = function(dir) {
            if (dir <= this.dirMax && dir >= 0) {
                this.currentDir = dir;
            }
        }
        this.checkCollisions = function(obj2) {
            var bot = BottomCollision(this, obj2);
            var top = TopCollision(this,obj2);
            var left = LeftCollision(this,obj2);
            var right = RightCollision(this,obj2);
            if ( bot || top || left || right){
                this.collision = true;
                return true;
            } else {
                this.collision = false;
                return false;
            }
        }
        //This one just checks if "this" moves one more spot down whether its bottom
        //will collide with the passed object
        this.checkBottomCollision = function(obj2) {
            return BottomCollision(this, obj2);
        }
        //This one just checks if "this" moves one more spot up whether its top
        //will collide with the passed object
        this.checkTopCollision = function(obj2){
            return TopCollision(this,obj2);
       }
        //This one just checks if "this" moves one more spot right whether its right
        //will collide with the passed object
        this.checkRightCollision = function(obj2) {
            return RightCollision(this,obj2);
        }
        //This one just checks if "this" moves one more spot right whether its left
        //will collide with the passed object
        this.checkLeftCollision = function(obj2) {
            return LeftCollision(this,obj2);
        }
    }
    //end of SpriteSheet
    //SpriteArray is when instead of a spritesheet you must create an array of sprites
```

```
    //from seprate images.  At thsi time SpriteArrays only support 1-dimension arrays
    //if you want to use 2dimenstions you'll simply create multiple arrays and change
    //the inArray to a differnt array of images when appropriate
    function SpriteArray (x,y,width,height, useWidth, useHeight,dX, dY, inArray) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.useWidth = useWidth;
        this.useHeight = useHeight;
        this.dX = dX;
        this.dY = dY;
        this.inArray = inArray;
        this.maxFrame = inArray.length-1;
        this.alpha = 1.0; //transparency of image whe drawn
        this.moveWithBackground = false;


        this.currentFrame = 0; //initial frame is passed

        this.collision = false;
        this.visible = true;
        // called to increment to the next image in the array
        this.advanceFrame = function(){
            if( (this.currentFrame +1) >= this.maxFrame){
                this.currentFrame = 0;
            } else {
                this.currentFrame++;
            }
        }
        this.draw = function() {
            if (this.visible) {
                ctx.globalAlpha = this.alpha; //This command sets the opacity to half - meaning you can see though any images painted after this
                tmpImg = new Image();
                tmpImg.src = this.inArray[this.currentFrame];

                ctx.drawImage(tmpImg,0,0,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
                ctx.globalAlpha = 1.0;
            }
        }
        this.checkCollisions = function(obj2) {
            bot = BottomCollision(this, obj2);
            top = TopCollision(this,obj2);
            left = LeftCollision(this,obj2);
            right = RightCollision(this,obj2);
            if ( bot || top || left || right){
                this.collision = true;
                return true;
            } else {
                this.collision = false;
                return false;
            }
        }
        //This one just checks if "this" moves one more spot down whether its bottom
        //will collide with the passed object
        this.checkBottomCollision = function(obj2) {
            return BottomCollision(this, obj2);
        }
        //This one just checks if "this" moves one more spot up whether its top
        //will collide with the passed object
        this.checkTopCollision = function(obj2){
            return TopCollision(this,obj2);
       }
        //This one just checks if "this" moves one more spot right whether its right
        //will collide with the passed object
        this.checkRightCollision = function(obj2) {
            return RightCollision(this,obj2);
        }
        //This one just checks if "this" moves one more spot right whether its left
        //will collide with the passed object
        this.checkLeftCollision = function(obj2) {
            return LeftCollision(this,obj2);
        }
    }
//This is a special Sprite for the background.  It will have capabilities to have it
//to be a scrolling background as certain games need this
function ScrollingBackGround(x,y,width,height, useWidth, useHeight,image){
    this.image = image;
    this.alpha = 1.0;
```

```
    this.visible = true;
    this.width = width;
    this.height = height;
    this.useWidth = useWidth;
    this.useHeight = useHeight;
    this.x = 0;
    this.y = 0;
    this.advanceBackground = true;
    // idea here is to load all terrain that will move along with the background.
    //As the background moves the terrain elements included in the background will
    //also move by changing their x & y values in lockstep
    //this.terrainArray = inArray;

    this.draw = function() {
        //*!*!*!*!Need to add the logic that will cause image to "wrap" around if scrolling
        if (this.visible) {
            if (this.advanceBackground){
                this.x += game.bgDX;
                this.y += game.bgDY;
            }
            ctx.globalAlpha = this.alpha; //This command sets the opacity to half - meaning you can see though any images painted after this

            ctx.drawImage(image,0,0,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
            //console.log (this.x);
            var tmpX = 0;
            var tmpY = 0;
            if (this.x < 0){
                tmpX = this.useWidth+this.x;
                if (this.x == -this.useWidth){
                    this.x = 0;
                }
            } else if (this.x > 0){
                tmpX = this.x-this.useWidth;
                if (this.x == this.useWidth){
                    this.x = 0;
                }
            }
            if (this.y < 0){
                tmpY = this.useHeight+this.y;
                if (this.y == -this.useHeight){
                    this.y = 0;
                }
            } else if (this.y > 0){
                tmpY = this.y-this.useHeight;
                if (this.y == this.useHeight){
                    this.y = 0;
                }
            }
             ctx.drawImage(image, 0,0,this.width,this.height, tmpX,tmpY,this.useWidth,this.useHeight);

            ctx.globalAlpha = 1.0;
        }

        }
    }

function createAudio(src) {
      var audio = document.createElement('audio');
      audio.src    = src;
      audio.play();
      //return audio;


    }

//** Used to help "read" the color of a specific pixel on the canvas.  Often you use
//** this to help decide if a collision has occurred this function actually takes the
//** three color componenents (RGB) and converts them to a hexadecimal color value. I
//** find it easier to just compare to this one value than all 3
function rgbToHex(r, g, b) {
if (r > 255 || g > 255 || b > 255)
    throw "Invalid color component";
return ((r << 16) | (g << 8) | b).toString(16);
}
function getColor(x, y) {
    var pixelData = ctx.getImageData(x, y, 1, 1).data;
    hex = "#" + ("000000" + rgbToHex(pixelData[0], pixelData[1], pixelData[2])).slice(-6);
    return hex;
}
```

```
// This is shared by all objects that want to check for a bottom collision
function BottomCollision (obj1, obj2) {
    if(obj1.x + obj1.useWidth > obj2.x && obj1.x < obj2.x + obj2.useWidth) {
        if (obj1.y  < obj2.y + obj2.useHeight && obj1.y + obj1.useHeight + 1 > obj2.y ){
            obj1.collision = true;
            //obj1.stopJump();
            return true;
        }
    }
    return false;
}
function TopCollision (obj1, obj2) {
    if(obj1.x + obj1.useWidth > obj2.x && obj1.x < obj2.x + obj2.useWidth) {
        if (obj1.y  - 1 < obj2.y + obj2.useHeight && obj1.y  > obj2.y ){
            obj1.collision = true;
            //obj1.stopJump();
            return true;
        }
            }
            return false;
}
function RightCollision (obj1, obj2) {
    obj1.collision = false;
    if(obj1.y + obj1.useHeight > obj2.y && obj1.y < obj2.y + obj2.useHeight) {
        if (obj1.x + obj1.useWidth + 1  > obj2.x && obj1.x < obj2.x + obj2.useWidth){
            obj1.collision = true;
            //obj1.stopJump();
            return true;
        }
    }
    return false;
}
function LeftCollision(obj1, obj2) {
    obj1.collision = false;
    if(obj1.y + obj1.useHeight > obj2.y && obj1.y < obj2.y + obj2.useHeight) {
        if (obj1.x + obj1.useWidth   > obj2.x && obj1.x -1 < obj2.x + obj2.useWidth){
            obj1.collision = true;
            //obj1.stopJump();
            return true;
        }
    }
    return false;
}
//This Class can be used to create an object to keep track of keyboard presses and releases
//I don't currently define each key, but they are automatically defined once pressed
//I'll likley add these however so they show up in code complete
function KeysPresses(){
    this.processDownKey = function(e) {
        //console.log(e.keycode);
        switch(e.keyCode) {
            case 8:
                this.backSpace = true;
                break;
            case 9:
                this.tab = true;
                break;
            case 12:
                this.numpadCenter = true;
                break;
            case 13:
                this.enter = true;
                break;
            case 16:
                this.shift = true;
                break;
            case 17:
                this.ctrl = true;
                break;
            case 18:
                this.alt = true;
                break;
            case 19:
                this.pauseBreak = true;
                break;
            case 20:
                this.capsLock = true;
                break;
            case 27:
                this.escape = true;
```

```
                break;
            case 32:
                this.space = true;
                break;
            case 33:
                this.pageUp = true;
                break;
            case 34:
                this.pageDown = true;
                break;
            case 35:
                this.end = true;
                break;
            case 36:
                this.home = true;
                break;
            case 37:
                this.leftArrow = true;
                break;
            case 38:
                this.upArrow = true;
                break;
            case 39:
                this.rightArrow = true;
                break;
            case 40:
                this.downArrow = true;
                break;
            case 45:
                this.insert = true;
                break;
            case 46:
                this.delete = true;
                break;
            case 48:
                this.key_0 = true;
                break;
            case 49:
                this.key_1 = true;
                break;
            case 50:
                this.key_2 = true;
                break;
            case 51:
                this.key_3 = true;
                break;
            case 52:
                this.key_4 = true;
                break;
            case 53:
                this.key_5 = true;
                break;
            case 54:
                this.key_6 = true;
                break;
            case 55:
                this.key_7 = true;
                break;
            case 56:
                this.key_8 = true;
                break;
            case 57:
                this.key_9 = true;
                break;
            case 65:
                this.key_a = true;
                break;
            case 66:
                this.key_b = true;
                break;
            case 67:
                this.key_c = true;
                break;
            case 68:
                this.key_d = true;
                break;
            case 69:
                this.key_e = true;
                break;
```

```
            case 70:
                this.key_f = true;
                break;
            case 71:
                this.key_g = true;
                break;
            case 72:
                this.key_h = true;
                break;
            case 73:
                this.key_i = true;
                break;
            case 74:
                this.key_j = true;
                break;
            case 75:
                this.key_k = true;
                break;
            case 76:
                this.key_l = true;
                break;
            case 77:
                this.key_m = true;
                break;
            case 78:
                this.key_n = true;
                break;
            case 79:
                this.key_o = true;
                break;
            case 80:
                this.key_p = true;
                break;
            case 81:
                this.key_q = true;
                break;
            case 82:
                this.key_r = true;
                break;
            case 83:
                this.key_s = true;
                break;
            case 84:
                this.key_t = true;
                break;
            case 85:
                this.key_u = true;
                break;
            case 86:
                this.key_v = true;
                break;
            case 87:
                this.key_w = true;
                break;
            case 88:
                this.key_x = true;
                break;
            case 89:
                this.key_y = true;
                break;
            case 90:
                this.key_z = true;
                break;
            case 91:
                this.leftWindowKey = true;
                break;
            case 92:
                this.rightWindowKey = true;
                break;
            case 93:
                this.selectKey = true;
                break;
            case 96:
                this.numpad0 = true;
                break;
            case 97:
                this.numpad1 = true;
                break;
            case 98:
```

```
                this.numpad2 = true;
                break;
            case 99:
                this.numpad3 = true;
                break;
            case 100:
                this.numpad4 = true;
                break;
            case 101:
                this.numpad5 = true;
                break;
            case 102:
                this.numpad6 = true;
                break;
            case 103:
                this.numpad7 = true;
                break;
            case 104:
                this.numpad8 = true;
                break;
            case 105:
                this.numpad9 = true;
                break;
            case 106:
                this.multiplyKey = true;
                break;
            case 107:
                this.addKey = true;
                break;
            case 109:
                this.subtractKey = true;
                break;
            case 110:
                this.decimalPoint = true;
                break;
            case 111:
                this.divideKey = true;
                break;
            case 112:
                e.preventDefault();
                this.f1 = true;
                break;
            case 113:
                this.f2 = true;
                break;
            case 114:
                this.f3 = true;
                break;
            case 115:
                this.f4 = true;
                break;
            case 116:
                this.f5 = true;
                break;
            case 117:
                this.f6 = true;
                break;
            case 118:
                this.f7 = true;
                break;
            case 119:
                this.f8 = true;
                break;
            case 120:
                this.f9 = true;
                break;
            case 121:
                this.f10 = true;
                break;
            case 122:
                this.f11 = true;
                break;
            case 123:
                this.f12 = true;
                break;
            case 144:
                this.numLock = true;
                break;
            case 145:
```

```
                this.scrollLock = true;
                break;
            case 186:
                this.semiColon = true;
                break;
            case 187:
                this.equalSign = true;
                break;
            case 188:
                this.comma = true;
                break;
            case 189:
                this.dash = true;
                break;
            case 190:
                this.period = true;
                break;
            case 191:
                this.forwardSlash = true;
                break;
            case 192:
                this.graveAccent = true;
                break;
            case 219:
                this.graveAccent = true;
                break;
            case 220:
                this.backSlash = true;
                break;
            case 221:
                this.closeBracket = true;
                break;
            case 222:
                this.singleQuote = true;
                break;
            default:
                //No action
        }

    }
    this.processUpKey = function(e) {
        switch(e.keyCode) {
            case 8:
                this.backSpace = false;
                break;
            case 9:
                this.tab = false;
                break;
            case 12:
                this.numpadCenter = false;
                break;
            case 13:
                this.enter = false;
                break;
            case 16:
                this.shift = false;
                break;
            case 17:
                this.ctrl = false;
                break;
            case 18:
                this.alt = false;
                break;
            case 19:
                this.pauseBreak = false;
                break;
            case 20:
                this.capsLock = false;
                break;
            case 27:
                this.escape = false;
                break;
            case 32:
                this.space = false;
                break;
            case 33:
                this.pageUp = false;
                break;
            case 34:
```

```
                this.pageDown = false;
                break;
            case 35:
                this.end = false;
                break;
            case 36:
                this.home = false;
                break;
            case 37:
                this.leftArrow = false;
                break;
            case 38:
                this.upArrow = false;
                break;
            case 39:
                this.rightArrow = false;
                break;
            case 40:
                this.downArrow = false;
                break;
            case 45:
                this.insert = false;
                break;
            case 46:
                this.delete = false;
                break;
            case 48:
                this.key_0 = false;
                break;
            case 49:
                this.key_1 = false;
                break;
            case 50:
                this.key_2 = false;
                break;
            case 51:
                this.key_3 = false;
                break;
            case 52:
                this.key_4 = false;
                break;
            case 53:
                this.key_5 = false;
                break;
            case 54:
                this.key_6 = false;
                break;
            case 55:
                this.key_7 = false;
                break;
            case 56:
                this.key_8 = false;
                break;
            case 57:
                this.key_9 = false;
                break;
            case 65:
                this.key_a = false;
                break;
            case 66:
                this.key_b = false;
                break;
            case 67:
                this.key_c = false;
                break;
            case 68:
                this.key_d = false;
                break;
            case 69:
                this.key_e = false;
                break;
            case 70:
                this.key_f = false;
                break;
            case 71:
                this.key_g = false;
                break;
            case 72:
                this.key_h = false;
```

```
                break;
            case 73:
                this.key_i = false;
                break;
            case 74:
                this.key_j = false;
                break;
            case 75:
                this.key_k = false;
                break;
            case 76:
                this.key_l = false;
                break;
            case 77:
                this.key_m = false;
                break;
            case 78:
                this.key_n = false;
                break;
            case 79:
                this.key_o = false;
                break;
            case 80:
                this.key_p = false;
                break;
            case 81:
                this.key_q = false;
                break;
            case 82:
                this.key_r = false;
                break;
            case 83:
                this.key_s = false;
                break;
            case 84:
                this.key_t = false;
                break;
            case 85:
                this.key_u = false;
                break;
            case 86:
                this.key_v = false;
                break;
            case 87:
                this.key_w = false;
                break;
            case 88:
                this.key_x = false;
                break;
            case 89:
                this.key_y = false;
                break;
            case 90:
                this.key_z = false;
                break;
            case 91:
                this.leftWindowKey = false;
                break;
            case 92:
                this.rightWindowKey = false;
                break;
            case 93:
                this.selectKey = false;
                break;
            case 96:
                this.numpad0 = false;
                break;
            case 97:
                this.numpad1 = false;
                break;
            case 98:
                this.numpad2 = false;
                break;
            case 99:
                this.numpad3 = false;
                break;
            case 100:
                this.numpad4 = false;
                break;
```

```
            case 101:
                this.numpad5 = false;
                break;
            case 102:
                this.numpad6 = false;
                break;
            case 103:
                this.numpad7 = false;
                break;
            case 104:
                this.numpad8 = false;
                break;
            case 105:
                this.numpad9 = false;
                break;
            case 106:
                this.multiplyKey = false;
                break;
            case 107:
                this.addKey = false;
                break;
            case 109:
                this.subtractKey = false;
                break;
            case 110:
                this.decimalPoint = false;
                break;
            case 111:
                this.divideKey = false;
                break;
            case 112:
                this.f1 = false;
                break;
            case 113:
                this.f2 = false;
                break;
            case 114:
                this.f3 = false;
                break;
            case 115:
                this.f4 = false;
                break;
            case 116:
                this.f5 = false;
                break;
            case 117:
                this.f6 = false;
                break;
            case 118:
                this.f7 = true;
                break;
            case 119:
                this.f8 = false;
                break;
            case 120:
                this.f9 = false;
                break;
            case 121:
                this.f10 = false;
                break;
            case 122:
                this.f11 = false;
                break;
            case 123:
                this.f12 = false;
                break;
            case 144:
                this.numLock = false;
                break;
            case 145:
                this.scrollLock = false;
                break;
            case 186:
                this.semiColon = false;
                break;
            case 187:
                this.equalSign = false;
                break;
            case 188:
```

```
                this.comma = false;
                break;
            case 189:
                this.dash = false;
                break;
            case 190:
                this.period = false;
                break;
            case 191:
                this.forwardSlash = false;
                break;
            case 192:
                this.graveAccent = false;
                break;
            case 219:
                this.graveAccent = false;
                break;
            case 220:
                this.backSlash = false;
                break;
            case 221:
                this.closeBracket = false;
                break;
            case 222:
                this.singleQuote = false;
                break;
            default:
                //No action
        }

    }
    this.processMouseMove = function(e) {
        this.mouseX = e.clientX;
        this.mouseY = e.clientY;
        // console.log("X:" + this.mouseX + " Y: " + this.mouseY);
        // colorColDetection(this.mouseX,this.mouseY)
    }
    this.processMouseClick = function(e) {
        //console.log("Mousebutton: " + e.which + " Click");
        switch(e.which) {
            case 1:
                this.mouseLeftButtonClick = true;
                break;
            case 2:
                this.mouseMiddleButtonClick = true;
                break;
            case 2:
                this.mouseRightButtonClick = true;
                break;
        }
    }
    this.processMouseUp = function(e) {
        //console.log("Mouse Button " + e.which + " Up X:" + this.mouseX + " Y: " + this.mouseY);
        switch(e.which) {
            case 1:
                this.mouseLeftButtonUp = true;
                break;
            case 2:
                this.mouseMiddleButtonUp = true;
                break;
            case 2:
                this.mouseRightButtonUp = true;
                break;
        }
    }
    this.processMouseDown = function(e) {
        //console.log("Mouse Button " + e.which + " Down X:" + this.mouseX + " Y: " + this.mouseY);
        switch(e.which) {
            case 1:
                this.mouseLeftButtonDown = true;
                break;
            case 2:
                this.mouseMiddleButtonDown = true;
                break;
            case 2:
                this.mouseRightButtonDown = true;
                break;
        }
    }
```

```
}

function sound(src) {
  this.sound = document.createElement("audio");
  this.sound.src = src;
  this.sound.setAttribute("preload", "auto");
  this.sound.setAttribute("controls", "none");
  this.sound.style.display = "none";
  document.body.appendChild(this.sound);
  this.play = function(){
    this.sound.play();
  }
  this.stop = function(){
    this.sound.pause();
  }
}

</script>


</body>
</html>
```