


```

79     border:1px solid;
80 }
81 .canvasParent{
82     /*box-shadow: -1px 0px 9px 3px #FFF; */
83     overflow:auto;
84     display:inline-block;
85 }
86 .blinking {
87
88     padding:4% 45%;
89     background-color:white;
90     -webkit-animation: glowing 1500ms infinite;
91     -moz-animation: glowing 1500ms infinite;
92     -o-animation: glowing 1500ms infinite;
93     animation: glowing 1500ms infinite;
94 }
95 }
96 @-webkit-keyframes glowing {
97     0% { background-color: #B20000; -webkit-box-shadow: 0 0 3px #B20000; }
98     50% { background-color: #FF0000; -webkit-box-shadow: 0 0 40px #FF0000; }
99     100% { background-color: #B20000; -webkit-box-shadow: 0 0 3px #B20000; }
100 }
101
102 @-moz-keyframes glowing {
103     0% { background-color: #B20000; -moz-box-shadow: 0 0 3px #B20000; }
104     50% { background-color: #FF0000; -moz-box-shadow: 0 0 40px #FF0000; }
105     100% { background-color: #B20000; -moz-box-shadow: 0 0 3px #B20000; }
106 }
107
108 @-o-keyframes glowing {
109     0% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
110     50% { background-color: #FF0000; box-shadow: 0 0 40px #FF0000; }
111     100% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
112 }
113
114 @keyframes glowing {
115     0% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
116     50% { background-color: #FF0000; box-shadow: 0 0 40px #FF0000; }
117     100% { background-color: #B20000; box-shadow: 0 0 3px #B20000; }
118 }
119 
120 <script>
121     var darkMode = true
122     function darkModeToggle() {
123         if(!darkMode) {
124             document.body.style.backgroundColor='black';
125             document.body.style.color='white';
126             darkMode = true;
127             ctx.filter = 'invert(100%)';
128             document.getElementsByClassName('canvasParent')[0].setAttribute('style','box-shadow:-1px 0px 9px 3px white')
129         } else {
130             document.body.style.backgroundColor='white';
131             document.body.style.color='black';
132             darkMode = false;
133             ctx.filter = 'invert(0%)';
134             document.getElementsByClassName('canvasParent')[0].setAttribute('style','box-shadow:-1px 0px 9px 3px black')
135         }
136     }
137
138 var isChromium = window.chrome;
139 var winNav = window.navigator;
140 var vendorName = winNav.vendor;
141 var isOpera = typeof window.opr !== "undefined";
142 var isIEedge = winNav.userAgent.indexOf("Edge") > -1;
143 var isIOSChrome = winNav.userAgent.match("CriOS");
144
145 if (isIOSChrome) {
146     // is Google Chrome on iOS
147 } else if(
148     isChromium !== null &&
149     typeof isChromium !== "undefined" &&
150     vendorName === "Google Inc." &&
151     isOpera === false &&
152     isIEedge === false
153 ) {
154     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")
155 } else {
156     // not Google Chrome
157 }
```

```
158 // console.image("https://raw.githubusercontent.com/jackcrane/jackcrane.github.io/f1dfd25b6d300ee740261084215121eb2dc35511/JS.jpg")
159 </script>
160 </head>
161 <body style="background-color:black;color:white;font-family:'Space Age';transition:.2s">
162 <center><h1>Moon Lander</h1></center>
164
165 <center><div class="canvasParent"><canvas id="canvas" width="800" height="520" style="transition:.2s"></canvas></div></center>
166 <br><br>
167 <br><br>
168 <p id="console" style="font-family:sans-serif;display:none"></p>
169 <div style="display: none">
170   <img id='background' src='images/terrain_v2_dark.png'>
171   
172 </div><br>
173 <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 click
174 <a class="blinking" id="playAgain" hidden onclick="document.location.reload()">Play again!</a>
175 <script src="baseDeclaration.js" type="text/javascript">// Explanation inside file.</script>
176 <script>
177
178 // Variable declaration
179
180 var speedMultiplier = 1
181
182 var paused = false
183
184 var alive = true;
185
186 var keys = new KeysPresses()
187
188 var killGameInstance = false;
189
190 var drawInt = null;
191
192 var won = false
193
194 var thrusterSound;
195
196 document.addEventListener("keydown", keyDownHandler, false);
197 document.addEventListener("keyup", keyUpHandler, false);
198 document.addEventListener("mousemove", mouseMoveHandler, false);
199 document.addEventListener("mousedown", mouseDownHandler, false);
200 document.addEventListener("mouseup", mouseUpHandler, false);
201 document.addEventListener("mouseclick", mouseClickHandler, false);
202
203
204 function keyDownHandler(e) {
205   keys.processDownKey(e);
206 }
207 function keyUpHandler(e) {
208   keys.processUpKey(e);
209 }
210 function mouseMoveHandler(e) {
211   keys.processMouseMove(e);
212 }
213 function mouseUpHandler(e) {
214   keys.processMouseUp(e);
215 }
216 function mouseDownHandler(e) {
217   keys.processMouseDown(e);
218 }
219 function mouseClickedHandler(e) {
220   keys.processMouseClicked(e);
221 }
222
223 // BELOW: The ugly base code:
224 // 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.
225
226 var backgroundImg = BackgroundPre;
227 var lander = {
228   engUp : '',
229   engDown : '',
230   engIdle : EngIdlePre,
231   lander:null,
232   engUpSS:null,
233   engDownSS:null,
234   engIdleSS:null,
235   frameSD:0,
236   frameSkip:0,
```

```
237 }
238
239 var gameStats = {
240   frameCount:0,
241 }
242
243 var background = new Image();
244 background.crossOrigin = "Anonymous"
245 background.src = backgroundImg;
246
247 var health = {
248   remainingFuel : 1000,
249   remainingRetro : 500,
250   remainingOxygen : 1000,
251 }
252
253 var deathCause = null;
254
255 var randomNumber = 0;
256
257 var canvas = document.getElementById('canvas');
258 var ctx = canvas.getContext("2d");
259 ctx.crossOrigin = "Anonymous";
260
261 var game = new GameMaster();
262
// Method creation
263
264 function drawText() {
265   var text = new GameText();
266   text.font = "35px Arial";
267   if(!darkMode) {
268     text.fillStyle = "Black"
269   } else {text.fillStyle = "White"}
270   text.x = 100;
271   text.y = 100;
272   text.draw("Loading...");
273 }
274
275 function killPlayer() {
276   health.remainingOxygen = 1;
277   // used for debugging
278 }
279
280
281 function drawScoreText() {
282   var text = new GameText();
283   text.font = "20px Roboto Mono";
284   text.fillStyle = "white";
285
286   text.x = 10;
287   text.y = 20;
288   if(gameStats.frameCount%2||gameStats.frameCount%3) {
289     text.draw("Remaining:");
290   }
291
292   text.x = 25;
293   text.y = 45;
294   if(gameStats.frameCount%2||gameStats.frameCount%3) {
295     text.draw("Fuel: " + health.remainingFuel + "");
296   }
297
298   text.x = 25;
299   text.y = 70;
300   if(gameStats.frameCount%2||gameStats.frameCount%3) {
301     text.draw("Retro: " + health.remainingRetro);
302   }
303
304   text.x = 25;
305   text.y = 95;
306   if(gameStats.frameCount%2||gameStats.frameCount%3) {
307     text.draw("Oxygen: " + health.remainingOxygen);
308   }
309
310   text.x = 600;
311   text.y = 20;
312   text.draw("Frames: " + gameStats.frameCount);
313 }
314
315 }
```

```

316 var fully;
317 var colDetX;
318
319 function getCrashedByColor(x,y) {
320     // Desperate attempt
321     x = Math.round(x);
322     y = Math.round(y);
323     // console.log("Checking "+x+", "+y)
324     var pixelData = ctx.getImageData(x , y, 5, 5).data;
325     var r = pixelData[0]
326     var g = pixelData[1]
327     var b = pixelData[2]
328     var rgb = r.toString()+g.toString()+b.toString()
329     // console.log(r.toString()+g.toString()+b.toString())
330     // console.log("R=" +pixelData[0] + " G=" + pixelData[1] + " B = " + pixelData[2])
331     if(r==g && g==b && r!=0)
332         return "crashed"
333     } else if(rgb==2372836) {
334         return "landed"
335     }
336 }
337
338 function colorColDetection(x,y,lander) {
339
340     // very computationally heavy way to determine collision based on color
341
342     fully = y + 30;
343     colDetX = x;
344
345
346     if(getCrashedByColor(colDetX,fully-6)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-5)=="landed") {console.log("landed");won=true;}
347     // if(getCrashedByColor(colDetX,fully-5)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-4)=="landed") {console.log("landed");won=true;}
348     if(getCrashedByColor(colDetX,fully-4)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-3)=="landed") {console.log("landed");won=true;}
349     // if(getCrashedByColor(colDetX,fully-3)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-2)=="landed") {console.log("landed");won=true;}
350     if(getCrashedByColor(colDetX,fully-2)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-1)=="landed") {console.log("landed");won=true;}
351     // if(getCrashedByColor(colDetX,fully-1)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully)=="landed") {console.log("landed");won=true;}
352     if(getCrashedByColor(colDetX,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+1)=="landed") {console.log("landed");won=true;}
353     if(getCrashedByColor(colDetX+1,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+2)=="landed") {console.log("landed");won=true;}
354     // if(getCrashedByColor(colDetX+2,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+3)=="landed") {console.log("landed");won=true;}
355     if(getCrashedByColor(colDetX+2,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+4)=="landed") {console.log("landed");won=true;}
356     // if(getCrashedByColor(colDetX+3,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+5)=="landed") {console.log("landed");won=true;}
357     if(getCrashedByColor(colDetX+4,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+6)=="landed") {console.log("landed");won=true;}
358     // if(getCrashedByColor(colDetX+5,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+7)=="landed") {console.log("landed");won=true;}
359     if(getCrashedByColor(colDetX+6,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+8)=="landed") {console.log("landed");won=true;}
360     // if(getCrashedByColor(colDetX+7,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+9)=="landed") {console.log("landed");won=true;}
361     if(getCrashedByColor(colDetX+8,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+10)=="landed") {console.log("landed");won=true;}
362     // if(getCrashedByColor(colDetX+9,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+11)=="landed") {console.log("landed");won=true;}
363     if(getCrashedByColor(colDetX+10,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+12)=="landed") {console.log("landed");won=true;}
364     // if(getCrashedByColor(colDetX+11,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+13)=="landed") {console.log("landed");won=true;}
365     if(getCrashedByColor(colDetX+12,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+14)=="landed") {console.log("landed");won=true;}
366     // if(getCrashedByColor(colDetX+13,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+15)=="landed") {console.log("landed");won=true;}
367     if(getCrashedByColor(colDetX+14,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+16)=="landed") {console.log("landed");won=true;}
368     // if(getCrashedByColor(colDetX+15,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+17)=="landed") {console.log("landed");won=true;}
369     if(getCrashedByColor(colDetX+16,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+18)=="landed") {console.log("landed");won=true;}
370     // if(getCrashedByColor(colDetX+17,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+19)=="landed") {console.log("landed");won=true;}
371     if(getCrashedByColor(colDetX+18,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+20)=="landed") {console.log("landed");won=true;}
372     // if(getCrashedByColor(colDetX+19,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+21)=="landed") {console.log("landed");won=true;}
373     // if(getCrashedByColor(colDetX+20,fully)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully+22)=="landed") {console.log("landed");won=true;}
374     if(getCrashedByColor(colDetX+20,fully-1)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully)=="landed") {console.log("landed");won=true;}
375     // if(getCrashedByColor(colDetX+20,fully-2)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-1)=="landed") {console.log("landed");won=true;}
376     if(getCrashedByColor(colDetX+20,fully-3)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-2)=="landed") {console.log("landed");won=true;}
377     // if(getCrashedByColor(colDetX+20,fully-4)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-3)=="landed") {console.log("landed");won=true;}
378     if(getCrashedByColor(colDetX+20,fully-5)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully)=="landed") {console.log("landed");won=true;}
379     // if(getCrashedByColor(colDetX+20,fully-6)=="crashed"){console.log("crashed");alive=false;;} else if(getCrashedByColor(colDetX,fully-4)=="landed") {console.log("landed");won=true;}
380
381     // var pixelData = ctx.getImageData(colDetX , fully, 1, 1).data;
382     // var r = pixelData[0]
383     // var g = pixelData[1]
384     // var b = pixelData[2]
385     // console.log("R=" +pixelData[0] + " G=" + pixelData[1] + " B = " + pixelData[2])
386
387
388 }
389
390 function drawOutline() {
391     // used to debug where collision detection would be watched for
392
393
394     // ctx.fillStyle = "red"

```

```

395    // ctx.fillRect(colDetX,fULLY-6,2,2);
396    // ctx.fillRect(colDetX,fULLY-5,2,2);
397    // ctx.fillRect(colDetX,fULLY-4,2,2);
398    // ctx.fillRect(colDetX,fULLY-3,2,2);
399    // ctx.fillRect(colDetX,fULLY-2,2,2);
400    // ctx.fillRect(colDetX,fULLY-1,2,2);
401    // ctx.fillRect(colDetX,fULLY,2,2);
402    // ctx.fillRect(colDetX+1,fULLY,2,2);
403    // ctx.fillRect(colDetX+2,fULLY,2,2);
404    // ctx.fillRect(colDetX+3,fULLY,2,2);
405    // ctx.fillRect(colDetX+4,fULLY,2,2);
406    // ctx.fillRect(colDetX+5,fULLY,2,2);
407    // ctx.fillRect(colDetX+6,fULLY,2,2);
408    // ctx.fillRect(colDetX+7,fULLY,2,2);
409    // ctx.fillRect(colDetX+8,fULLY,2,2);
410    // ctx.fillRect(colDetX+9,fULLY,2,2);
411    // ctx.fillRect(colDetX+10,fULLY,2,2);
412    // ctx.fillRect(colDetX+11,fULLY,2,2);
413    // ctx.fillRect(colDetX+12,fULLY,2,2);
414    // ctx.fillRect(colDetX+13,fULLY,2,2);
415    // ctx.fillRect(colDetX+14,fULLY,2,2);
416    // ctx.fillRect(colDetX+15,fULLY,2,2);
417    // ctx.fillRect(colDetX+16,fULLY,2,2);
418    // ctx.fillRect(colDetX+17,fULLY,2,2);
419    // ctx.fillRect(colDetX+18,fULLY,2,2);
420    // ctx.fillRect(colDetX+19,fULLY,2,2);
421    // ctx.fillRect(colDetX+20,fULLY,2,2);
422    // ctx.fillRect(colDetX+20,fULLY-6,2,2);
423    // ctx.fillRect(colDetX+20,fULLY-5,2,2);
424    // ctx.fillRect(colDetX+20,fULLY-4,2,2);
425    // ctx.fillRect(colDetX+20,fULLY-3,2,2);
426    // ctx.fillRect(colDetX+20,fULLY-2,2,2);
427    // ctx.fillRect(colDetX+20,fULLY-1,2,2);
428  }
429
430  function dir(a) {
431    return game.imageDir + a;
432  }
433
434  function makeBackgroundOnce() {
435    // var background = new Image();
436    // background.src = backgroundImg;
437    // var rand = randomNumber;
438
439    // THIS DARN SCROLLING BACKGROUND WONT START AT THE RANDOM NUMBER
440
441    let bg = new ScrollingBackGround(randomNumber,0,13000,2080,4550,520,background);
442    bg.draw();
443
444  }
445
446  function makeBackground() {
447    makeBackgroundOnce()
448  }
449
450  function calcLife() {
451    if(gameStats.frameCount%13==0) {
452      health.remainingOxygen--;
453    }
454  }
455
456  function aliveF() {
457    if(health.remainingOxygen >= 0) {
458      return true
459    } else {
460      return false
461    }
462  }
463
464  function deathCauseFunct() {
465    if(health.remainingOxygen <= 0) {
466      return "ran out of oxygen"
467    }
468    console.log('deathCause run')
469  }
470
471  function drawDeathScreen() {
472    var text = new GameText();
473    text.font = "50px Hacked";

```

```

474     text.fillStyle = "white";
475     text.x = 10;
476     text.y = 120;
477     text.align = 'center';
478     text.draw("You crashed!")
479     document.getElementById("playAgain").hidden = false
480     // console.log(text)
481 }
482
483 function drawVictoryScreen() {
484     // function that shows text congratulating winner
485     var text = new GameText();
486     text.font = "50px Hacked";
487     text.fillStyle = "white";
488     text.x = 10;
489     text.y = 120;
490     text.align = 'center';
491     text.draw("You won!")
492     document.getElementById("playAgain").hidden = false
493     // console.log(text)
494 }
495
496 function createLander() {
497     var img = new Image();
498     img.crossOrigin = "Anonymous"
499     img.src = lander.engIdle;
500
501     thrusterSound = new sound("sound/lander.mp3")
502
503     // console.log(img)
504     lander.engIdleSS = new SpriteSheet (100,100,256,256,50,50, 0,0,img);
505 }
506
507 function checkCollision(x,y) {
508     canvas.crossOrigin = "Anonymous";
509     lander.lander.crossOrigin = "Anonymous";
510
511     ctx.crossOrigin = "Anonymous";
512     // var hex = getColor(x,y)
513
514     // console.log(lander.Lander.width)
515 }
516
517 function sconsole(e) {
518     document.getElementById("sconsole").innerHTML = e
519 }
520
521 function landerRefresh() {
522
523
524     lander.lander = lander.engIdleSS
525
526     if(keys.upArrow) {
527         // Lander.Lander.y = Lander.Lander.y - 0.1
528         var tempX = lander.lander.x
529         var tempY = lander.lander.y
530         // Lander.Lander = Lander.engUpSS;
531         lander.lander.x = tempX
532         lander.lander.y = tempY
533
534         if(gameStats.frameCount % 5 == 0) {
535             if(health.remainingFuel >= 0) {
536                 health.remainingFuel -= 1
537             }
538         }
539     }
540
541 } else {
542     // Lander.Lander = Lander.engIdleSS;
543     // Lander.Lander.currentFrame = 0;
544 }
545
546 if(!keys.upArrow) {
547     // Lander.engUpSS.currentFrame = 0;
548     lander.lander.y = lander.lander.y + 0.1 * speedMultiplier
549     // console.log(lander.lander.y)
550 } else {
551     if(health.remainingFuel >= 0) {
552         lander.lander.y = lander.lander.y - 0.1 *speedMultiplier

```

```

553         thrusterSound.play()
554     } else {
555         lander.lander.y = lander.lander.y + 0.1 *speedMultiplier
556     }
557 }
558
559 if(keys.leftArrow) {
560     lander.lander.x = lander.lander.x - 0.1 *speedMultiplier
561     // console.log(lander.lander.dx)
562
563 } else if (keys.rightArrow) {
564     lander.lander.x = lander.lander.x + 0.1 *speedMultiplier
565 }
566
567 lander.frameSD ++
568 if(lander.frameSD == 50) {
569     lander.frameSkip ++
570     lander.lander.advanceFrame()
571     sconsole(lander.lander.x)
572     colorColDetection(lander.lander.x,lander.lander.y,lander)
573     // console.log(lander.frameSD)
574     lander.framesSD = 0;
575 }
576
577
578
579
580
581
582 }
583
584 function checkDeadKeys() {
585     if(keys.key_P){}
586 }
587
588 // createLander()
589
590 function draw() {
591     gameStats.frameCount++;
592     ctx.fillStyle = "black";
593     ctx.fillRect(0, 0, canvas.width, canvas.height);
594
595     calcLife();
596     if(aliveF()&&alive==true&&won==false) {
597         drawText();
598         makeBackground();
599         drawScoreText();
600         landerRefresh();
601
602         lander.lander.draw()
603         checkCollision(lander.lander.x,lander.lander.y)
604         ctx.fillStyle = "red";
605         drawOutline();
606     } else if(alive==false) {
607         drawDeathScreen();
608         checkDeadKeys();
609         clearInterval("drawInt")
610     } else if (won==true) {
611         drawVictoryScreen()
612         clearInterval("drawInt")
613     }
614
615     if(killGameInstance) {
616         clearInterval("drawInt")
617     }
618
619     drawInt = setInterval(draw, 1000/30)
620
621
622
623 }
624
625 makeBackgroundOnce()
626 createLander()
627
628 draw();
629
630
631

```

```

632     randomNumber = Math.floor(Math.random() * 1200) + 1;
633
634     darkModeToggle()
635
636
637
638 // ****
639 // *
640 // *          The following is a Javascript Game Development library written by our teacher
641 // *
642 // ****
643
644 function GameMaster(){
645     this.imageDir = ".\\images\\";
646     this.soundDir = ".\\sounds\\";
647     this.libraryDir = ".\\lib\\";
648     this.CurrentScreen = "MainMenu";
649     this.debugstr = "";
650     this.bgDX = 0;
651     this.bgDY = 0;
652     this.gameIterator=0;
653     this.gameIterate = function(){
654         this.gameiterator++;
655         /** just in case of an overflow
656         if (this.gameIterator < 0){
657             this.gameIterate = 0;
658         }
659     }
660     this.moveObjectWithBackground = function(o){
661         o.x += this.bgDX;
662         o.y += this.bgDY;
663         if (o.x < o.useWidth*-1){
664             o.x = canvas.width;
665         }
666     }
667     this.debugAppend = function(str){
668         this.debugstr = this.debugstr + str
669     }
670     this.debug = function(){
671         document.getElementById("debug").innerHTML = this.debugstr;
672         this.debugstr = "";
673     }
674 }
675 function GameText(){
676     this.font = "48px Arial";
677     this.fillStyle = "Blue";
678     this.x = 800;
679     this.y = 45;
680     this.alpha = 0.5;
681     this.draw = function(inText) {
682         ctx.font = this.font;
683         ctx.fillStyle = this.fillStyle;
684         ctx.globalAlpha = this.alpha;
685         ctx.fillText(inText, this.x, this.y);
686         ctx.globalAlpha = 1.0;
687     }
688 }
689 // For creating a single image Sprite
690 function Sprite (x,y,width,height, useWidth, useHeight,image) {
691     this.x = x;
692     this.y = y;
693     this.dX = 0;
694     this.dY = 0;
695     this.width = width;
696     this.height = height;
697     this.useWidth = usewidth;
698     this.useHeight = useHeight;
699     this.image = image;
700     this.alpha = 1.0; //transparency of image whe drawn
701     this.collision = false;
702     this.visible = true;
703     this.moveToBackground = false;
704
705     this.draw = function() {
706         if (this.visible) {
707             if (this.moveToBackground = true){
708                 game.moveObjectWithBackground(this);
709             }
710             ctx.globalAlpha = this.alpha;

```

```

711         ctx.drawImage(this.image,0,0,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
712         ctx.globalAlpha = 1.0;
713     }
714
715     this.checkCollisions = function(obj2) {
716         var bot = BottomCollision(this, obj2);
717         var top = TopCollision(this,obj2);
718         var left = LeftCollision(this,obj2);
719         var right = RightCollision(this,obj2);
720         if ( bot || top || left || right){
721             this.collision = true;
722             return true;
723         } else {
724             this.collision = false;
725             return false;
726         }
727     }
728
729 //This one just checks if "this" moves one more spot down whether its bottom
730 //will collide with the passed object
731 this.checkBottomCollision = function(obj2) {
732     return BottomCollision(this, obj2);
733 }
734 //This one just checks if "this" moves one more spot up whether its top
735 //will collide with the passed object
736 this.checkTopCollision = function(obj2){
737     return TopCollision(this,obj2);
738 }
739 //This one just checks if "this" moves one more spot right whether its right
740 //will collide with the passed object
741 this.checkRightCollision = function(obj2) {
742     return RightCollision(this,obj2);
743 }
744 //This one just checks if "this" moves one more spot right whether its left
745 //will collide with the passed object
746 this.checkLeftCollision = function(obj2) {
747     return LeftCollision(this,obj2);
748 }
749 }
750 // For creating a Sprite that uses a spritesheet for motion
751 // note - assumes each row represents a direction and each column a frame
752 function SpriteSheet (x,y,width,height, useWidth, useHeight, dir, frame, image) {
753     this.x = x;
754     this.y = y;
755     this.width = width;
756     this.height = height;
757     this.useWidth = useWidth;
758     this.useHeight = useHeight;
759     this.dX = 0;
760     this.dY = 0;
761     this.dirMax = dir; //Max number of "directions" the Sprite can move in
762     this.currentDir = 0; //start the Sprite with the first "direction"
763     this.maxFrame = frame; //Max number of Frames the Sprite animation takes
764     this.currentFrame = 0; //start on current Frame
765     //this.sheet = image; //deprecated sheet - was inconsistent with other objects
766     this.image = image;
767     this.alpha = 1.0; //transparency of image whe drawn
768     this.collision = false;
769     this.visible = true;
770     this.moveToBackground = false;
771     this.canJump = false;
772     this.gravity = 0; //How fast he falls - should be set for any character that can fall/jump
773     this.isJumping = false; // set to true once jump starts and is not set back until is "grounded" - this keeps from jumping in mid-air
774     this.jumpMax = 40;
775     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
776
777     this.jump = function(){
778         if (this.canJump && this.isJumping == false) {
779             this.isJumping = true;
780             this.jumpCount = this.jumpMax;
781         }
782     }
783     this.stopJump = function(){
784         this.jumpCount = 0;
785         myKeys.space = false;
786     }
787     this.touchingColor = function(ctx,r,g,b){
788         var data = ctx.getImageData(this.x,this.y,this.width,this.height);
789         console.log(data)

```

```

790     for(var i=0;i<data.length;i+=4){
791         if(
792             data[i+0]==r&&
793             data[i+1]==g&&
794             data[i+2]==b
795         ){
796             return true;
797         }
798     }
799     return false;
800 }
801
802 this.checkJump = function(){
803     if (this.isJumping && this.jumpCount > 0){
804         this.y -= (this.gravity + 1);
805         this.jumpCount--;
806     } else{
807         this.stopJump();
808     }
809 }
810 this.applyGravity = function(){
811     this.y += this.gravity;
812 }
813 this.draw = function() {
814     if (this.visible) {
815         ctx.globalAlpha = this.alpha; //This command sets the opacity to half - meaning you can see though any images painted after this
816         ctx.drawImage(this.image,this.currentFrame * this.width,this.currentDir * this.height,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
817         ctx.globalAlpha = 1.0; //Setting the opacity back to 1 - which mean solid
818     }
819 }
820 // Increments the Sprite animation frame. If already at the max goes back to zero position
821 this.advanceFrame = function() {
822     this.currentFrame++;
823     if (this.currentFrame > this.maxFrame) {
824         this.currentFrame = 0;
825     }
826 }
827 // Changes the Sprite "direction" or row of animation - will only change to a valid value
828 this.changeDir = function(dir) {
829     if (dir <= this.dirMax && dir >= 0) {
830         this.currentDir = dir;
831     }
832 }
833 this.checkCollisions = function(obj2) {
834     var bot = BottomCollision(this, obj2);
835     var top = TopCollision(this,obj2);
836     var left = LeftCollision(this,obj2);
837     var right = RightCollision(this,obj2);
838     if (bot || top || left || right){
839         this.collision = true;
840         return true;
841     } else {
842         this.collision = false;
843         return false;
844     }
845 }
846 //This one just checks if "this" moves one more spot down whether its bottom
847 //will collide with the passed object
848 this.checkBottomCollision = function(obj2) {
849     return BottomCollision(this, obj2);
850 }
851 //This one just checks if "this" moves one more spot up whether its top
852 //will collide with the passed object
853 this.checkTopCollision = function(obj2){
854     return TopCollision(this,obj2);
855 }
856 //This one just checks if "this" moves one more spot right whether its right
857 //will collide with the passed object
858 this.checkRightCollision = function(obj2) {
859     return RightCollision(this,obj2);
860 }
861 //This one just checks if "this" moves one more spot right whether its left
862 //will collide with the passed object
863 this.checkLeftCollision = function(obj2) {
864     return LeftCollision(this,obj2);
865 }
866 }
867 //end of SpriteSheet
868 //SpriteArray is when instead of a spritesheet you must create an array of sprites

```

```

869 //from seprate images. At thsi time SpriteArrays only support 1-dimension arrays
870 //if you want to use 2dimentions you'll simply create multiple arrays and change
871 //the inArray to a differnt array of images when appropriate
872 function SpriteArray (x,y,width,height, useWidth, useHeight,dX, dY, inArray) {
873     this.x = x;
874     this.y = y;
875     this.width = width;
876     this.height = height;
877     this.useWidth = useWidth;
878     this.useHeight = useHeight;
879     this.dX = dX;
880     this.dY = dY;
881     this.inArray = inArray;
882     this.maxFrame = inArray.length-1;
883     this.alpha = 1.0; //transparency of image whe drawn
884     this.moveWithBackground = false;
885
886
887     this.currentFrame = 0; //initial frame is passed
888
889     this.collision = false;
890     this.visible = true;
891     // called to increment to the next image in the array
892     this.advanceFrame = function(){
893         if( (this.currentFrame +1) >= this.maxFrame){
894             this.currentFrame = 0;
895         } else {
896             this.currentFrame++;
897         }
898     }
899     this.draw = function() {
900         if (this.visible) {
901             ctx.globalAlpha = this.alpha; //This command sets the opacity to half - meaning you can see though any images painted after this
902             tmpImg = new Image();
903             tmpImg.src = this.inArray[this.currentFrame];
904
905             ctx.drawImage(tmpImg,0,0,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
906             ctx.globalAlpha = 1.0;
907         }
908     }
909     this.checkCollisions = function(obj2) {
910         bot = BottomCollision(this, obj2);
911         top = TopCollision(this,obj2);
912         left = LeftCollision(this,obj2);
913         right = RightCollision(this,obj2);
914         if ( bot || top || left || right){
915             this.collision = true;
916             return true;
917         } else {
918             this.collision = false;
919             return false;
920         }
921     }
922     //This one just checks if "this" moves one more spot down whether its bottom
923     //will collide with the passed object
924     this.checkBottomCollision = function(obj2) {
925         return BottomCollision(this, obj2);
926     }
927     //This one just checks if "this" moves one more spot up whether its top
928     //will collide with the passed object
929     this.checkTopCollision = function(obj2){
930         return TopCollision(this,obj2);
931     }
932     //This one just checks if "this" moves one more spot right whether its right
933     //will collide with the passed object
934     this.checkRightCollision = function(obj2) {
935         return RightCollision(this,obj2);
936     }
937     //This one just checks if "this" moves one more spot right whether its left
938     //will collide with the passed object
939     this.checkLeftCollision = function(obj2) {
940         return LeftCollision(this,obj2);
941     }
942 }
943 //This is a special Sprite for the background. It will have capabilities to have it
944 //to be a scrolling background as certain games need this
945 function ScrollingBackGround(x,y,width,height, useWidth, useHeight,image){
946     this.image = image;
947     this.alpha = 1.0;

```

```

948 this.visible = true;
949 this.width = width;
950 this.height = height;
951 this.useWidth = useWidth;
952 this.useHeight = useHeight;
953 this.x = 0;
954 this.y = 0;
955 this.advanceBackground = true;
956 // idea here is to load all terrain that will move along with the background.
957 // As the background moves the terrain elements included in the background will
958 // also move by changing their x & y values in lockstep
959 //this.terrainArray = inArray;
960
961 this.draw = function() {
962 ///*!*!*!Need to add the logic that will cause image to "wrap" around if scrolling
963 if (this.visible) {
964     if (this.advanceBackground){
965         this.x += game.bgDX;
966         this.y += game.bgDY;
967     }
968     ctx.globalAlpha = this.alpha; //This command sets the opacity to half - meaning you can see through any images painted after this
969
970     ctx.drawImage(image,0,0,this.width,this.height,this.x,this.y,this.useWidth,this.useHeight);
971     //console.log (this.x);
972     var tmpX = 0;
973     var tmpY = 0;
974     if (this.x < 0){
975         tmpX = this.useWidth+this.x;
976         if (this.x == -this.useWidth){
977             this.x = 0;
978         }
979     } else if (this.x > 0){
980         tmpX = this.x-this.useWidth;
981         if (this.x == this.useWidth){
982             this.x = 0;
983         }
984     }
985     if (this.y < 0){
986         tmpY = this.useHeight+this.y;
987         if (this.y == -this.useHeight){
988             this.y = 0;
989         }
990     } else if (this.y > 0){
991         tmpY = this.y-this.useHeight;
992         if (this.y == this.useHeight){
993             this.y = 0;
994         }
995     }
996     ctx.drawImage(image, 0,0,this.width,this.height, tmpX,tmpY,this.useWidth,this.useHeight);
997     ctx.globalAlpha = 1.0;
998 }
999 }
1000 }
1001 }
1002 }
1003
1004 function createAudio(src) {
1005     var audio = document.createElement('audio');
1006     audio.src   = src;
1007     audio.play();
1008     //return audio;
1009
1010 }
1011 }
1012
1013 /** Used to help "read" the color of a specific pixel on the canvas. Often you use
1014 /** this to help decide if a collision has occurred this function actually takes the
1015 /** three color components (RGB) and converts them to a hexdecimal color value. I
1016 /** find it easier to just compare to this one value than all 3
1017 function rgbToHex(r, g, b) {
1018 if (r > 255 || g > 255 || b > 255)
1019     throw "Invalid color component";
1020 return ((r << 16) | (g << 8) | b).toString(16);
1021 }
1022 function getColor(x, y) {
1023     var pixelData = ctx.getImageData(x, y, 1, 1).data;
1024     hex = "#" + ("000000" + rgbToHex(pixelData[0], pixelData[1], pixelData[2])).slice(-6);
1025     return hex;
1026 }

```

```

1027 // This is shared by all objects that want to check for a bottom collision
1028 function BottomCollision (obj1, obj2) {
1029     if(obj1.x + obj1.useWidth > obj2.x && obj1.x < obj2.x + obj2.useWidth) {
1030         if (obj1.y < obj2.y + obj2.useHeight && obj1.y + obj1.useHeight + 1 > obj2.y ){
1031             obj1.collision = true;
1032             //obj1.stopJump();
1033             return true;
1034         }
1035     }
1036     return false;
1037 }
1038 function TopCollision (obj1, obj2) {
1039     if(obj1.x + obj1.useWidth > obj2.x && obj1.x < obj2.x + obj2.useWidth) {
1040         if (obj1.y - 1 < obj2.y + obj2.useHeight && obj1.y > obj2.y ){
1041             obj1.collision = true;
1042             //obj1.stopJump();
1043             return true;
1044         }
1045     }
1046     return false;
1047 }
1048 function RightCollision (obj1, obj2) {
1049     obj1.collision = false;
1050     if(obj1.y + obj1.useHeight > obj2.y && obj1.y < obj2.y + obj2.useHeight) {
1051         if (obj1.x + obj1.useWidth + 1 > obj2.x && obj1.x < obj2.x + obj2.useWidth){
1052             obj1.collision = true;
1053             //obj1.stopJump();
1054             return true;
1055         }
1056     }
1057     return false;
1058 }
1059 function LeftCollision(obj1, obj2) {
1060     obj1.collision = false;
1061     if(obj1.y + obj1.useHeight > obj2.y && obj1.y < obj2.y + obj2.useHeight) {
1062         if (obj1.x + obj1.useWidth > obj2.x && obj1.x -1 < obj2.x + obj2.useWidth){
1063             obj1.collision = true;
1064             //obj1.stopJump();
1065             return true;
1066         }
1067     }
1068     return false;
1069 }
1070 //This Class can be used to create an object to keep track of keyboard presses and releases
1071 //I don't currently define each key, but they are automatically defined once pressed
1072 //I'll likley add these however so they show up in code complete
1073 function KeysPresses(){
1074     this.processDownKey = function(e) {
1075         //console.log(e.keyCode);
1076         switch(e.keyCode) {
1077             case 8:
1078                 this.backSpace = true;
1079                 break;
1080             case 9:
1081                 this.tab = true;
1082                 break;
1083             case 12:
1084                 this.numpadCenter = true;
1085                 break;
1086             case 13:
1087                 this.enter = true;
1088                 break;
1089             case 16:
1090                 this.shift = true;
1091                 break;
1092             case 17:
1093                 this.ctrl = true;
1094                 break;
1095             case 18:
1096                 this.alt = true;
1097                 break;
1098             case 19:
1099                 this.pauseBreak = true;
1100                 break;
1101             case 20:
1102                 this.capsLock = true;
1103                 break;
1104             case 27:
1105                 this.escape = true;

```

```
1106         break;
1107     case 32:
1108         this.space = true;
1109         break;
1110     case 33:
1111         this.pageUp = true;
1112         break;
1113     case 34:
1114         this.pageDown = true;
1115         break;
1116     case 35:
1117         this.end = true;
1118         break;
1119     case 36:
1120         this.home = true;
1121         break;
1122     case 37:
1123         this.leftArrow = true;
1124         break;
1125     case 38:
1126         this.upArrow = true;
1127         break;
1128     case 39:
1129         this.rightArrow = true;
1130         break;
1131     case 40:
1132         this.downArrow = true;
1133         break;
1134     case 45:
1135         this.insert = true;
1136         break;
1137     case 46:
1138         this.delete = true;
1139         break;
1140     case 48:
1141         this.key_0 = true;
1142         break;
1143     case 49:
1144         this.key_1 = true;
1145         break;
1146     case 50:
1147         this.key_2 = true;
1148         break;
1149     case 51:
1150         this.key_3 = true;
1151         break;
1152     case 52:
1153         this.key_4 = true;
1154         break;
1155     case 53:
1156         this.key_5 = true;
1157         break;
1158     case 54:
1159         this.key_6 = true;
1160         break;
1161     case 55:
1162         this.key_7 = true;
1163         break;
1164     case 56:
1165         this.key_8 = true;
1166         break;
1167     case 57:
1168         this.key_9 = true;
1169         break;
1170     case 65:
1171         this.key_a = true;
1172         break;
1173     case 66:
1174         this.key_b = true;
1175         break;
1176     case 67:
1177         this.key_c = true;
1178         break;
1179     case 68:
1180         this.key_d = true;
1181         break;
1182     case 69:
1183         this.key_e = true;
1184         break;
```

```
1185     case 70:
1186         this.key_f = true;
1187         break;
1188     case 71:
1189         this.key_g = true;
1190         break;
1191     case 72:
1192         this.key_h = true;
1193         break;
1194     case 73:
1195         this.key_i = true;
1196         break;
1197     case 74:
1198         this.key_j = true;
1199         break;
1200     case 75:
1201         this.key_k = true;
1202         break;
1203     case 76:
1204         this.key_l = true;
1205         break;
1206     case 77:
1207         this.key_m = true;
1208         break;
1209     case 78:
1210         this.key_n = true;
1211         break;
1212     case 79:
1213         this.key_o = true;
1214         break;
1215     case 80:
1216         this.key_p = true;
1217         break;
1218     case 81:
1219         this.key_q = true;
1220         break;
1221     case 82:
1222         this.key_r = true;
1223         break;
1224     case 83:
1225         this.key_s = true;
1226         break;
1227     case 84:
1228         this.key_t = true;
1229         break;
1230     case 85:
1231         this.key_u = true;
1232         break;
1233     case 86:
1234         this.key_v = true;
1235         break;
1236     case 87:
1237         this.key_w = true;
1238         break;
1239     case 88:
1240         this.key_x = true;
1241         break;
1242     case 89:
1243         this.key_y = true;
1244         break;
1245     case 90:
1246         this.key_z = true;
1247         break;
1248     case 91:
1249         this.leftWindowKey = true;
1250         break;
1251     case 92:
1252         this.rightWindowKey = true;
1253         break;
1254     case 93:
1255         this.selectKey = true;
1256         break;
1257     case 96:
1258         this.numpad0 = true;
1259         break;
1260     case 97:
1261         this.numpad1 = true;
1262         break;
1263     case 98:
```

```
1264         this.numpad2 = true;
1265         break;
1266     case 99:
1267         this.numpad3 = true;
1268         break;
1269     case 100:
1270         this.numpad4 = true;
1271         break;
1272     case 101:
1273         this.numpad5 = true;
1274         break;
1275     case 102:
1276         this.numpad6 = true;
1277         break;
1278     case 103:
1279         this.numpad7 = true;
1280         break;
1281     case 104:
1282         this.numpad8 = true;
1283         break;
1284     case 105:
1285         this.numpad9 = true;
1286         break;
1287     case 106:
1288         this.multiplyKey = true;
1289         break;
1290     case 107:
1291         this.addKey = true;
1292         break;
1293     case 109:
1294         this.subtractKey = true;
1295         break;
1296     case 110:
1297         this.decimalPoint = true;
1298         break;
1299     case 111:
1300         this.divideKey = true;
1301         break;
1302     case 112:
1303         e.preventDefault();
1304         this.f1 = true;
1305         break;
1306     case 113:
1307         this.f2 = true;
1308         break;
1309     case 114:
1310         this.f3 = true;
1311         break;
1312     case 115:
1313         this.f4 = true;
1314         break;
1315     case 116:
1316         this.f5 = true;
1317         break;
1318     case 117:
1319         this.f6 = true;
1320         break;
1321     case 118:
1322         this.f7 = true;
1323         break;
1324     case 119:
1325         this.f8 = true;
1326         break;
1327     case 120:
1328         this.f9 = true;
1329         break;
1330     case 121:
1331         this.f10 = true;
1332         break;
1333     case 122:
1334         this.f11 = true;
1335         break;
1336     case 123:
1337         this.f12 = true;
1338         break;
1339     case 144:
1340         this.numLock = true;
1341         break;
1342     case 145:
```

```
1343         this.scrollLock = true;
1344         break;
1345     case 186:
1346         this.semiColon = true;
1347         break;
1348     case 187:
1349         this.equalSign = true;
1350         break;
1351     case 188:
1352         this.comma = true;
1353         break;
1354     case 189:
1355         this.dash = true;
1356         break;
1357     case 190:
1358         this.period = true;
1359         break;
1360     case 191:
1361         this.forwardSlash = true;
1362         break;
1363     case 192:
1364         this.graveAccent = true;
1365         break;
1366     case 219:
1367         this.graveAccent = true;
1368         break;
1369     case 220:
1370         this.backSlash = true;
1371         break;
1372     case 221:
1373         this.closeBracket = true;
1374         break;
1375     case 222:
1376         this.singleQuote = true;
1377         break;
1378     default:
1379         //No action
1380     }
1381 }
1382 this.processUpKey = function(e) {
1383     switch(e.keyCode) {
1384         case 8:
1385             this.backSpace = false;
1386             break;
1387         case 9:
1388             this.tab = false;
1389             break;
1390         case 12:
1391             this.numpadCenter = false;
1392             break;
1393         case 13:
1394             this.enter = false;
1395             break;
1396         case 16:
1397             this.shift = false;
1398             break;
1399         case 17:
1400             this.ctrl = false;
1401             break;
1402         case 18:
1403             this.alt = false;
1404             break;
1405         case 19:
1406             this.pauseBreak = false;
1407             break;
1408         case 20:
1409             this.capsLock = false;
1410             break;
1411         case 27:
1412             this.escape = false;
1413             break;
1414         case 32:
1415             this.space = false;
1416             break;
1417         case 33:
1418             this.pageUp = false;
1419             break;
1420         case 34:
```

```
1422         this.pageDown = false;
1423         break;
1424     case 35:
1425         this.end = false;
1426         break;
1427     case 36:
1428         this.home = false;
1429         break;
1430     case 37:
1431         this.leftArrow = false;
1432         break;
1433     case 38:
1434         this.upArrow = false;
1435         break;
1436     case 39:
1437         this.rightArrow = false;
1438         break;
1439     case 40:
1440         this.downArrow = false;
1441         break;
1442     case 45:
1443         this.insert = false;
1444         break;
1445     case 46:
1446         this.delete = false;
1447         break;
1448     case 48:
1449         this.key_0 = false;
1450         break;
1451     case 49:
1452         this.key_1 = false;
1453         break;
1454     case 50:
1455         this.key_2 = false;
1456         break;
1457     case 51:
1458         this.key_3 = false;
1459         break;
1460     case 52:
1461         this.key_4 = false;
1462         break;
1463     case 53:
1464         this.key_5 = false;
1465         break;
1466     case 54:
1467         this.key_6 = false;
1468         break;
1469     case 55:
1470         this.key_7 = false;
1471         break;
1472     case 56:
1473         this.key_8 = false;
1474         break;
1475     case 57:
1476         this.key_9 = false;
1477         break;
1478     case 65:
1479         this.key_a = false;
1480         break;
1481     case 66:
1482         this.key_b = false;
1483         break;
1484     case 67:
1485         this.key_c = false;
1486         break;
1487     case 68:
1488         this.key_d = false;
1489         break;
1490     case 69:
1491         this.key_e = false;
1492         break;
1493     case 70:
1494         this.key_f = false;
1495         break;
1496     case 71:
1497         this.key_g = false;
1498         break;
1499     case 72:
1500         this.key_h = false;
```

```
1501         break;
1502     case 73:
1503         this.key_i = false;
1504         break;
1505     case 74:
1506         this.key_j = false;
1507         break;
1508     case 75:
1509         this.key_k = false;
1510         break;
1511     case 76:
1512         this.key_l = false;
1513         break;
1514     case 77:
1515         this.key_m = false;
1516         break;
1517     case 78:
1518         this.key_n = false;
1519         break;
1520     case 79:
1521         this.key_o = false;
1522         break;
1523     case 80:
1524         this.key_p = false;
1525         break;
1526     case 81:
1527         this.key_q = false;
1528         break;
1529     case 82:
1530         this.key_r = false;
1531         break;
1532     case 83:
1533         this.key_s = false;
1534         break;
1535     case 84:
1536         this.key_t = false;
1537         break;
1538     case 85:
1539         this.key_u = false;
1540         break;
1541     case 86:
1542         this.key_v = false;
1543         break;
1544     case 87:
1545         this.key_w = false;
1546         break;
1547     case 88:
1548         this.key_x = false;
1549         break;
1550     case 89:
1551         this.key_y = false;
1552         break;
1553     case 90:
1554         this.key_z = false;
1555         break;
1556     case 91:
1557         this.leftWindowKey = false;
1558         break;
1559     case 92:
1560         this.rightWindowKey = false;
1561         break;
1562     case 93:
1563         this.selectKey = false;
1564         break;
1565     case 96:
1566         this.numpad0 = false;
1567         break;
1568     case 97:
1569         this.numpad1 = false;
1570         break;
1571     case 98:
1572         this.numpad2 = false;
1573         break;
1574     case 99:
1575         this.numpad3 = false;
1576         break;
1577     case 100:
1578         this.numpad4 = false;
1579         break;
```

```
1580     case 101:
1581         this.numpad5 = false;
1582         break;
1583     case 102:
1584         this.numpad6 = false;
1585         break;
1586     case 103:
1587         this.numpad7 = false;
1588         break;
1589     case 104:
1590         this.numpad8 = false;
1591         break;
1592     case 105:
1593         this.numpad9 = false;
1594         break;
1595     case 106:
1596         this.multiplyKey = false;
1597         break;
1598     case 107:
1599         this.addKey = false;
1600         break;
1601     case 109:
1602         this.subtractKey = false;
1603         break;
1604     case 110:
1605         this.decimalPoint = false;
1606         break;
1607     case 111:
1608         this.divideKey = false;
1609         break;
1610     case 112:
1611         this.f1 = false;
1612         break;
1613     case 113:
1614         this.f2 = false;
1615         break;
1616     case 114:
1617         this.f3 = false;
1618         break;
1619     case 115:
1620         this.f4 = false;
1621         break;
1622     case 116:
1623         this.f5 = false;
1624         break;
1625     case 117:
1626         this.f6 = false;
1627         break;
1628     case 118:
1629         this.f7 = true;
1630         break;
1631     case 119:
1632         this.f8 = false;
1633         break;
1634     case 120:
1635         this.f9 = false;
1636         break;
1637     case 121:
1638         this.f10 = false;
1639         break;
1640     case 122:
1641         this.f11 = false;
1642         break;
1643     case 123:
1644         this.f12 = false;
1645         break;
1646     case 144:
1647         this.numLock = false;
1648         break;
1649     case 145:
1650         this.scrollLock = false;
1651         break;
1652     case 186:
1653         this.semiColon = false;
1654         break;
1655     case 187:
1656         this.equalSign = false;
1657         break;
1658     case 188:
```

```
1659         this.comma = false;
1660         break;
1661     case 189:
1662         this.dash = false;
1663         break;
1664     case 190:
1665         this.period = false;
1666         break;
1667     case 191:
1668         this.forwardSlash = false;
1669         break;
1670     case 192:
1671         this.graveAccent = false;
1672         break;
1673     case 194:
1674         this.graveAccent = false;
1675         break;
1676     case 220:
1677         this.backSlash = false;
1678         break;
1679     case 221:
1680         this.closeBracket = false;
1681         break;
1682     case 222:
1683         this.singleQuote = false;
1684         break;
1685     default:
1686         //No action
1687     }
1688 }
1689
1690 this.processMouseMove = function(e) {
1691     this.mouseX = e.clientX;
1692     this.mouseY = e.clientY;
1693     // console.log("X:" + this.mouseX + " Y: " + this.mouseY);
1694     // colorColDetection(this.mouseX, this.mouseY)
1695 }
1696 this.processMouseClick = function(e) {
1697     //console.Log("Mousebutton: " + e.which + " Click");
1698     switch(e.which) {
1699         case 1:
1700             this.mouseLeftButtonClick = true;
1701             break;
1702         case 2:
1703             this.mouseMiddleButtonClick = true;
1704             break;
1705         case 2:
1706             this.mouseRightButtonClick = true;
1707             break;
1708     }
1709 }
1710 this.processMouseUp = function(e) {
1711     //console.log("Mouse Button " + e.which + " Up X:" + this.mouseX + " Y: " + this.mouseY);
1712     switch(e.which) {
1713         case 1:
1714             this.mouseLeftButtonUp = true;
1715             break;
1716         case 2:
1717             this.mouseMiddleButtonUp = true;
1718             break;
1719         case 2:
1720             this.mouseRightButtonUp = true;
1721             break;
1722     }
1723 }
1724 this.processMouseDown = function(e) {
1725     //console.Log("Mouse Button " + e.which + " Down X:" + this.mouseX + " Y: " + this.mouseY);
1726     switch(e.which) {
1727         case 1:
1728             this.mouseLeftButtonDown = true;
1729             break;
1730         case 2:
1731             this.mouseMiddleButtonDown = true;
1732             break;
1733         case 2:
1734             this.mouseRightButtonDown = true;
1735             break;
1736     }
1737 }
```

```
1738 }
1739
1740 function sound(src) {
1741     this.sound = document.createElement("audio");
1742     this.sound.src = src;
1743     this.sound.setAttribute("preload", "auto");
1744     this.sound.setAttribute("controls", "none");
1745     this.sound.style.display = "none";
1746     document.body.appendChild(this.sound);
1747     this.play = function(){
1748         this.sound.play();
1749     }
1750     this.stop = function(){
1751         this.sound.pause();
1752     }
1753 }
1754
1755 </script>
1756
1757
1758 </body>
1759 </html>
```