

```
1  /*
2  Hi! I already know coding and tried following coursework requirements as
   closely as possible, apologizing for any unintentionally advanced
   implementations used.
3  I carefully watched the plagiarism warning video because I was scared of
   getting flagged for academic misconduct.
4  Please check my GitHub profile (github.com/itzmaniss) to see my prior
   programming experience and understanding of these concepts.
5  In case of any issues, I am happy to discuss and clarify my work.
6  */
7
8  // Game character variables
9  var gameChar_x = 200;
10 var gameChar_y = 300;
11 var gameChar_width = 30;
12 var velocityX = 0;
13 var velocityY = 0;
14 var facing = "front";
15
16 // Movement state variables
17 var isLeft = false;
18 var isRight = false;
19 var isFalling = false;
20 var isWalking = false;
21 var isPlummeting = false;
22
23 // Game world variables
24 var floorPos_y = 300;
25 var cameraPosX = 0;
26 var coinCount = 0;
27
28 // Game object arrays
29 var clouds = [];
30 var trees = [];
31 var mountainRange = [];
32 var collectables = [];
33 var canyons = [];
34
35 // Movement constants
36 var cloudSpeed = 0.25;
37
38 function setup() {
39     createCanvas(1024, 576);
40     noStroke();
41
42     // Create clouds at different x positions
43     for (var i = 0; i < 50; i++) {
44         clouds[i] = {x_pos: i * 200 - 300, y_pos: 80};
```

```
45     }
46
47     // Create trees at different x positions
48     for (var i = 0; i < 100; i++) {
49
50         trees[i] = {x_pos: i * 150 - 200, y_pos: 240};
51     }
52
53     // Create mountains at different x positions
54     for (var i = 0; i < 100; i++) {
55         mountainRange[i] = {x_pos: i * 400 - 300, y_pos: 300};
56     }
57
58     // Create collectables at different x positions
59     for (var i = 0; i < 80; i++) {
60         collectables[i] = {x_pos: i * 300 - 100, y_pos: 175, size: 30, isFound:
61         false};
62     }
63
64     // Create canyons at different x positions
65     for (var i = 0; i < 40; i++) {
66         canyons[i] = {x_pos: i * 800 + 400, width: 85};
67     }
68 }
69
70 function draw() {
71     // Update camera to follow character
72     cameraPosX = gameChar_x - width/2;
73
74     // Draw sky background
75     background(135, 206, 235);
76
77     // Draw extended ground
78     fill(34, 139, 34);
79     rect(-width, floorPos_y, width * 3, height - floorPos_y);
80
81     // Apply camera transformation for world objects
82     push();
83     translate(-cameraPosX, 0);
84
85     // Draw mountains
86     for (var i = 0; i < mountainRange.length; i++) {
87         if (mountainRange[i].x_pos > cameraPosX - 200 && mountainRange[i].x_pos
88         < cameraPosX + width + 200) {
89             fill(125, 125, 125);
90             noStroke();
91             triangle(mountainRange[i].x_pos - 60, mountainRange[i].y_pos,
92             mountainRange[i].x_pos, mountainRange[i].y_pos - 240, mountainRange[i].x_pos +
93             60, mountainRange[i].y_pos);
94         }
95     }
96 }
```

```
89         triangle(mountainRange[i].x_pos - 120, mountainRange[i].y_pos,
mountainRange[i].x_pos - 60, mountainRange[i].y_pos - 180,
mountainRange[i].x_pos, mountainRange[i].y_pos);
90         triangle(mountainRange[i].x_pos, mountainRange[i].y_pos,
mountainRange[i].x_pos + 60, mountainRange[i].y_pos - 150,
mountainRange[i].x_pos + 120, mountainRange[i].y_pos);
91     }
92 }
93
94 // Draw and animate clouds
95 for (var i = 0; i < clouds.length; i++) {
96     if (clouds[i].x_pos > cameraPosX - 200 && clouds[i].x_pos < cameraPosX
+ width + 200) {
97         fill(255);
98         noStroke();
99         ellipse(clouds[i].x_pos, clouds[i].y_pos, 60, 40);
100        ellipse(clouds[i].x_pos - 20, clouds[i].y_pos + 10, 50, 30);
101        ellipse(clouds[i].x_pos + 20, clouds[i].y_pos + 10, 50, 30);
102        ellipse(clouds[i].x_pos - 10, clouds[i].y_pos - 10, 40, 20);
103        ellipse(clouds[i].x_pos + 10, clouds[i].y_pos - 10, 40, 20);
104    }
105
106    // Move clouds to the left
107    clouds[i].x_pos = clouds[i].x_pos - cloudSpeed;
108
109    // Reset cloud position when it goes off screen
110    if (clouds[i].x_pos < -100) {
111        clouds[i].x_pos = clouds[i].x_pos + 10000;
112    }
113 }
114
115 // Draw trees
116 for (var i = 0; i < trees.length; i++) {
117     if (trees[i].x_pos > cameraPosX - 200 && trees[i].x_pos < cameraPosX +
width + 200) {
118         fill(139, 69, 19);
119         rect(trees[i].x_pos, trees[i].y_pos, 20, 60);
120         fill(34, 139, 34);
121         ellipse(trees[i].x_pos + 10, trees[i].y_pos - 30, 60, 60);
122         ellipse(trees[i].x_pos - 20, trees[i].y_pos - 10, 50, 50);
123         ellipse(trees[i].x_pos + 40, trees[i].y_pos - 10, 50, 50);
124         ellipse(trees[i].x_pos + 10, trees[i].y_pos - 50, 40, 40);
125         ellipse(trees[i].x_pos + 10, trees[i].y_pos - 70, 40, 40);
126     }
127 }
128
129 // Draw collectable coins
130 for (var i = 0; i < collectables.length; i++) {
```

```
131     if (collectables[i].isFound == false && collectables[i].x_pos >
cameraPosX - 200 && collectables[i].x_pos < cameraPosX + width + 200) {
132         fill(255, 215, 0, 100);
133         ellipse(collectables[i].x_pos, collectables[i].y_pos, 34, 34);
134         fill(255, 215, 0);
135         ellipse(collectables[i].x_pos, collectables[i].y_pos, 30, 30);
136         fill(255, 235, 0);
137         ellipse(collectables[i].x_pos, collectables[i].y_pos, 24, 24);
138         fill(255, 255, 200);
139         push();
140         translate(collectables[i].x_pos - 5, collectables[i].y_pos - 5);
141         rotate(PI / 4);
142         ellipse(0, 0, 5, 12);

143         pop();
144         textAlign(CENTER, CENTER);
145         textSize(16);
146         fill(0);
147         text('$', collectables[i].x_pos, collectables[i].y_pos);
148     }
149 }

150
151 // Draw canyons
152 for (var i = 0; i < canyons.length; i++) {
153     if (canyons[i].x_pos > cameraPosX - 200 && canyons[i].x_pos <
cameraPosX + width + 200) {
154         fill(40, 40, 40);
155         rect(canyons[i].x_pos, floorPos_y, canyons[i].width, height -
floorPos_y);
156         fill(139, 69, 19);
157         rect(canyons[i].x_pos - 5, floorPos_y, 5, height - floorPos_y);
158         rect(canyons[i].x_pos + canyons[i].width, floorPos_y, 5, height -
floorPos_y);
159         for(var j = 0; j < 5; j++) {
160             fill(0, 0, 0, 50 - j * 10);
161             rect(canyons[i].x_pos, floorPos_y + j * 5, canyons[i].width,
5);
162         }
163     }
164 }

165
166 // Check coin collection
167 for (var i = 0; i < collectables.length; i++) {
168     if (collectables[i].isFound == false) {
169         var charLeft = gameChar_x - 15;
170         var charRight = gameChar_x + 15;
171         var charTop = gameChar_y - 60;
172         var charBottom = gameChar_y;
173     }
```

```
174         var coinLeft = collectables[i].x_pos - 15;
175         var coinRight = collectables[i].x_pos + 15;
176         var coinTop = collectables[i].y_pos - 15;
177         var coinBottom = collectables[i].y_pos + 15;
178
179         if (charRight > coinLeft && charLeft < coinRight && charBottom >
coinTop && charTop < coinBottom) {
180             collectables[i].isFound = true;
181             coinCount = coinCount + 1;
182         }
183     }
184 }
185
186 // Check canyon collision
187 for (var i = 0; i < canyons.length; i++) {
188     if (gameChar_x > canyons[i].x_pos && gameChar_x < canyons[i].x_pos +
canyons[i].width && gameChar_y >= floorPos_y && isPlummeting == false) {
189         isPlummeting = true;
190     }
191 }
192
193 // Handle canyon falling
194 if (isPlummeting == true) {
195     gameChar_y = gameChar_y + 5;
196     velocityX = 0;
197     velocityY = 0;
198
199     for (var i = 0; i < canyons.length; i++) {
200         if (gameChar_x > canyons[i].x_pos && gameChar_x < canyons[i].x_pos
+ canyons[i].width) {
201             if (gameChar_x < canyons[i].x_pos) {
202                 gameChar_x = canyons[i].x_pos;
203             }
204             if (gameChar_x > canyons[i].x_pos + canyons[i].width) {
205                 gameChar_x = canyons[i].x_pos + canyons[i].width;
206             }
207         }
208     }
209 } else {
210     // Apply gravity and movement
211     velocityY = velocityY + 0.8;
212     gameChar_x = gameChar_x + velocityX;
213     gameChar_y = gameChar_y + velocityY;
214
215     // Check ground collision
216     if (gameChar_y >= floorPos_y) {
217         gameChar_y = floorPos_y;
218         velocityY = 0;
```

```
219     velocityX = 0;
220     isFalling = false;
221   }
222 }
223
224 // Draw character based on current state
225 if (isFalling == true) {
226   if (facing === "right") {
227     push();
228     translate(gameChar_x, gameChar_y);
229     fill('#FFFF00');
230     ellipse(0, -35, 35, 35);
231     ellipse(3, -60, 30, 30);
232     fill('#FFFFFF');
233     ellipse(11, -62, 10, 10);
234     fill('#00BFFF');
235     ellipse(12, -62, 7, 7);
236     fill('#000000');
237     ellipse(13, -62, 4, 4);
238     fill("#BA8E23");
239     triangle(16, -62, 23, -58, 16, -54);
240     stroke(0);
241     strokeWeight(3);
242     line(-5, -18, -3, -10);
243     line(-3, -10, 2, -10);
244     noStroke();
245     pop();
246   } else if (facing === "left") {
247     push();
248     translate(gameChar_x, gameChar_y);
249     scale(-1, 1);
250     fill('#FFFF00');
251     ellipse(0, -35, 35, 35);
252     ellipse(3, -60, 30, 30);
253     fill('#FFFFFF');
254     ellipse(11, -62, 10, 10);
255     fill('#00BFFF');
256     ellipse(12, -62, 7, 7);
257     fill('#000000');
258     ellipse(13, -62, 4, 4);
259     fill("#BA8E23");
260     triangle(16, -62, 23, -58, 16, -54);
261     stroke(0);
262     strokeWeight(3);
263     line(-5, -18, -3, -10);
264     line(-3, -10, 2, -10);
265     noStroke();
266     pop();
```

```
267     } else {
268         push();
269         translate(gameChar_x, gameChar_y);
270         fill('#FFFF00');
271         ellipse(0, -35, 35, 35);
272         ellipse(0, -60, 30, 30);
273         fill('#FFFFFF');
274         ellipse(-6, -62, 8, 8);
275         fill('#00BFFF');
276         ellipse(-6, -62, 6, 6);
277         fill('#000000');
278         ellipse(-6, -62, 3, 3);
279         fill('#FFFFFF');
280         ellipse(6, -62, 8, 8);
281         fill('#00BFFF');
282         ellipse(6, -62, 6, 6);
283         fill('#000000');
284         ellipse(6, -62, 3, 3);
285         fill("#BA8E23");
286         triangle(0, -55, -4, -50, 4, -50);
287         stroke(0);
288
289         strokeWeight(3);
290         noFill();
291         line(-8, -18, -12, -10);
292         line(-12, -10, -15, -10);
293         line(8, -18, 12, -10);
294         line(12, -10, 15, -10);
295         pop();
296     }
297 } else if (facing === "left") {
298     push();
299     translate(gameChar_x, gameChar_y);
300     scale(-1, 1);
301     fill('#FFFF00');
302     ellipse(-3, -25, 35, 35);
303     ellipse(0, -50, 30, 30);
304     fill('#FFFFFF');
305     ellipse(8, -52, 10, 10);
306     fill('#00BFFF');
307     ellipse(9, -52, 7, 7);
308     fill('#000000');
309     ellipse(10, -52, 4, 4);
310     fill("#BA8E23");
311     triangle(13, -52, 20, -48, 13, -44);
312     stroke(0);
313     strokeWeight(3);
314     line(-8, -8, -8, -2);
```

```
314     line(-8, -2, -3, -2);
315     noStroke();
316     pop();
317 } else if (facing === "right") {
318     push();
319     translate(gameChar_x, gameChar_y);
320     fill('#FFFF00');
321     ellipse(-3, -25, 35, 35);
322     ellipse(0, -50, 30, 30);
323     fill('#FFFFFF');
324     ellipse(8, -52, 10, 10);
325     fill('#00BFFF');
326     ellipse(9, -52, 7, 7);
327     fill('#000000');
328     ellipse(10, -52, 4, 4);
329     fill("#BA8E23");
330     triangle(13, -52, 20, -48, 13, -44);
331     stroke(0);
332     strokeWeight(3);
333     line(-8, -8, -8, -2);
334     line(-8, -2, -3, -2);
335     noStroke();
336
337     pop();
338 } else {
339     push();
340     translate(gameChar_x, gameChar_y);
341     fill('#FFFF00');
342     ellipse(0, -25, 35, 35);
343     ellipse(0, -50, 30, 30);
344     fill('#FFFFFF');
345     ellipse(-6, -52, 8, 8);
346     fill('#00BFFF');
347     ellipse(-6, -52, 6, 6);
348     fill('#000000');
349     ellipse(-6, -52, 3, 3);
350     fill('#FFFFFF');
351     ellipse(6, -52, 8, 8);
352     fill('#00BFFF');
353     ellipse(6, -52, 6, 6);
354     fill('#000000');
355     ellipse(6, -52, 3, 3);
356     fill("#BA8E23");
357     triangle(0, -45, -4, -40, 4, -40);
358     stroke(0);
359     strokeWeight(3);
360     noFill();
361     line(-8, -8, -15, -2);
```



```
361     line(-15, -2, -18, -2);
362     line(8, -8, 15, -2);
363     line(15, -2, 18, -2);
364     noStroke();
365     pop();
366 }
367
368 // End camera transformation
369 pop();
370
371 // Draw UI elements (coin counter)
372 fill(255, 215, 0);
373 textAlign(LEFT, TOP);
374 textSize(24);
375 text('Coins: ' + coinCount, 20, 20);
376
377 // Handle horizontal movement
378 if (isPlummeting == false) {
379     if (isLeft == true) {
380         velocityX = -5;
381     } else if (isRight == true) {
382         velocityX = 5;
383     }
384 }
385 }
386
387 function keyPressed() {
388     if (isPlummeting == false) {
389         if (key === 'A' || key === 'a' || keyCode == 37) {
390             if (isRight == false) {
391                 isLeft = true;
392                 facing = "left";
393                 isWalking = true;
394             }
395         } else if (key === 'D' || key === 'd' || keyCode == 39) {
396             if (isLeft == false) {
397                 isRight = true;
398                 facing = "right";
399                 isWalking = true;
400             }
401         } else if (key === ' ' || keyCode == 38) {
402             if (isFalling == false) {
403                 velocityY = -12;
404                 isFalling = true;
405                 isWalking = false;
406             }
407         }
408     }
```

```
409 }
410
411 function keyReleased() {
412     if (isPlummeting == false) {
413         if (key === 'A' || key === 'a' || keyCode == 37) {
414             isLeft = false;
415             if (isRight == true) {
416                 facing = "right";
417                 isWalking = true;
418             } else {
419                 velocityX = 0;
420                 isWalking = false;
421                 facing = "front";
422             }
423         } else if (key === 'D' || key === 'd' || keyCode == 39) {
424             isRight = false;
425             if (isLeft == true) {
426                 facing = "left";
427                 isWalking = true;
428             } else {
429                 velocityX = 0;
430                 isWalking = false;
431                 facing = "front";
432             }
433         }
434     }
435 }
```