JavaScript Final Project

Dog Volleyball - 0716050 吳泓毅

Motivation

在想專題要做什麼的時候想到小時候很風靡的一款經典遊戲--皮卡丘打排球，因為看起來應該是我能應付的難度，所以就選定了這個主題。

Goal

做類似於皮卡丘打排球的遊戲，皮卡丘用兩隻狗代替，排球和背景也都是網路上找的。

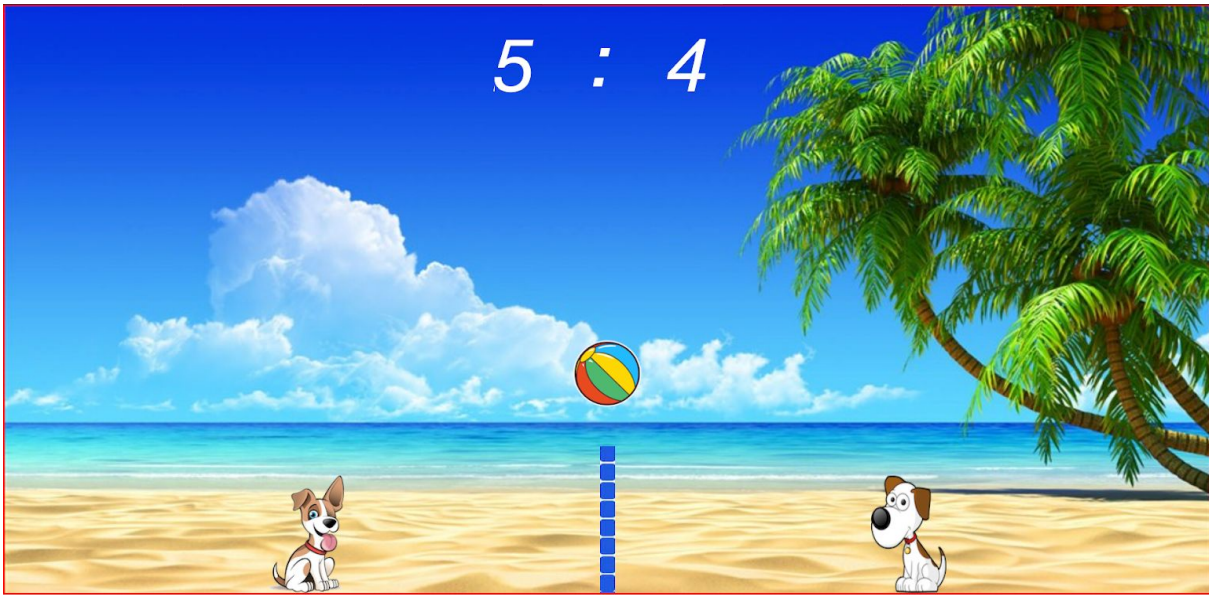
Current Progress

目前的進度算是較簡化的版本，求可以順順地移動，狗可以上下左右，但上還不是跳，只是單純向上懸在空中。球和狗的碰撞已經完成，可以順利反彈。

Development

是在MacOS Big Sur下開發，用的開發工具是Microsoft的Visual Studio Code。JavaScript是內嵌在HTML中，沒有使用其他的框架。

About My Project



包括的檔案：

html, css, js, images/

架構：

7個objects，一顆球、兩隻狗、一個網子、三個和計分板有關的物件。

遊戲說明：

由左邊的狗先發球，之後由上一回合贏的狗發球。球落地則落地的另一側的狗贏，先得到11分的狗贏得比賽。

程式說明：

建立container，接這建立其他物件後加在container中。

```
245   // 設一個container當球移動的最大範圍的block
246   var container = document.createElement("div");
247   container.setAttribute("id", "container");
248   document.body.appendChild(container);
249
250   // scoreboard
251   // left score
252   let scorel = document.createElement("div");
253   let sscorel = new Score(scorel, 650, 20, 0);
254   container.appendChild(scorel);
255
256   // right score
257   let scorer = document.createElement("div");
258   let sscorer = new Score(scorer, 880, 20, 0);
259   container.appendChild(scorer);
260
261   // :
262   let scorem = document.createElement("div");
263   let sscorem = new Score(scorem, 780, 13, ':');
264   container.appendChild(scorem);
265
266   // net
267   let net = document.createElement("div");
268   let nnet = new Net(net);
269   container.appendChild(net);
270
271   // dog
272   // left dog
273   let dogl = document.createElement("div");
274   let doglo = new Dog(dogl, 350, 640, "url('images/dog4.png')");
275   container.appendChild(dogl);
276
277   // right dog
278   let dogr = document.createElement("div");
279   let dogro = new Dog(dogr, 1150, 640, "url('images/dog2.png')");
280   container.appendChild(dogr);
281
282   // create ball
283   let node = document.createElement("div");
284   let bball = new Ball(node);
285   container.appendChild(node);
```

球的constructor：半徑50，起始座標350, 0，起始變量0, -1(向下)。

```
105     constructor(node) {
106         // 隨機設定球的半徑大小
107         this.radius = 50;
108 
109         // 隨機設定球在block中的起始座標
110         this.coor = {
111             x: 350,
112             y: 0
113         };
114 
115         // 隨機設定球移動的變量大小
116         this.offset = {
117             x: 0,
118             y: -1
119         };
120 
121         // 設定球的長、高、和顏色
122         this.node = node;
123         this.node.style.width = this.radius * 2 + "px";
124         this.node.style.height = this.radius * 2 + "px";
125         this.node.style.backgroundImage = "url('images/ball.png')";
126         this.node.setAttribute("class", "ball");
127 
128         // 設定球的起始座標
129         this.node.style.left = this.coor.x + "px";
130         this.node.style.top = this.coor.y + "px";
131     }
```

狗的constructor：起始座標x, y是傳入的參數，因為左右兩隻狗不同。

```
51     constructor(dog, x, y, image) {
52         this.coor = {
53             x: x,
54             y: y
55         };
56 
57         // 設定狗的長、高、和顏色
58         this.dog = dog;
59         this.dog.style.width = 100 + "px";
60         this.dog.style.height = 160 + "px";
61         this.dog.style.backgroundImage = image;
62         this.dog.setAttribute("class", "dog");
63 
64         // 設定狗的起始座標
65         this.dog.style.left = this.coor.x + "px";
66         this.dog.style.top = this.coor.y + "px";
67     }
```

網子的constructor：起始位置790, 600

```
36      constructor(net) {
37          this.net = net;
38          this.net.style.width = 20 + "px";
39          this.net.style.height = 200 + "px";
40          // this.net.style.backgroundColor = "#a86f06";
41          this.net.style.backgroundImage = "url('images/1.jpeg')";
42          this.net.setAttribute("class", "net");
43
44          this.net.style.left = 790 + "px";
45          this.net.style.top = 600 + "px";
46      }
```

球的移動：

利用requestAnimationFrame持續Recursive Call讓球一直移動。requestAnimationFrame()可以讓畫面以每個frame更新一次的速度更新，藉此來令球可以移動的很平順沒有頓點。

```
287   // 利用requestAnimationFrame在更新frame時移動球
288   requestAnimationFrame(Move);
289   function Move() {
290       bball.move();
291       requestAnimationFrame(Move);
292   }
```

鍵盤控制：

鍵盤可以控制狗的移動，上、下、左、右，左右不能超出自己的界線，下不能超出地板，上不能超出一定高度。狗的移動一樣是使用requestAnimationFrame讓狗可以移動的很平順沒有頓點，而這裡是判斷鍵盤是被按下還是放開來修改一個布林變數，此布林變數用來控制狗是否要移動，如此一來就不會有鍵盤按一下動一下頓頓的感覺。

```
332  // keydown detection
333  document.onkeydown = function (e) {
334      switch (e.code) {
335          case "ArrowLeft":
336              rLEFT = true;
337              break;
338          case "ArrowRight":
339              rRIGHT = true;
340              break;
341          case "ArrowUp":
342              rUP = true;
343              break;
344          case "ArrowDown":
345              rDOWN = true;
346              break;
347
348          case "KeyA":
349              lLEFT = true;
350              break;
351          case "KeyD":
352              lRIGHT = true;
353              break;
354          case "KeyW":
355              lUP = true;
356              break;
357          case "KeyS":
358              lDOWN = true;
359              break;
360      }
361  }
```

```
363  // keyup detection
364  document.onkeyup = function (e) {
365      switch (e.code) {
366          case "ArrowLeft":
367              rLEFT = false;
368              break;
369          case "ArrowRight":
370              rRIGHT = false;
371              break;
372          case "ArrowUp":
373              rUP = false;
374              break;
375          case "ArrowDown":
376              rDOWN = false;
377              break;
378
379          case "KeyA":
380              lLEFT = false;
381              break;
382          case "KeyD":
383              lRIGHT = false;
384              break;
385          case "KeyW":
386              lUP = false;
387              break;
388          case "KeyS":
389              lDOWN = false;
390              break;
391      }
392  }
```

```
requestAnimationFrame(DogLMove);
function DogLMove() {
    if (lLEFT)
        doglo.move('A');
    if (lRIGHT)
        doglo.move('D');
    if (lUP)
        doglo.move('W');
    if (lDOWN)
        doglo.move('S');
    requestAnimationFrame(DogLMove);
}

requestAnimationFrame(DogRMove);
function DogRMove() {
    if (rLEFT)
        dogro.move('l');
    if (rRIGHT)
        dogro.move('r');
    if (rUP)
        dogro.move('u');
    if (rDOWN)
        dogro.move('d');
    requestAnimationFrame(DogRMove);
}
```

Ball class中的function：

init(side)：在一回合結束後依照傳入參數重設球的位置和偏移量，如果參數是left，則左邊發球，球從左邊落下，如果參數是right，則右邊發球，球從右邊落下。

```
    init(side) {
        // this.coor.x = 350;
        // this.coor.y = 0;
        if (side == 'l') {
            this.coor = {
                x: 350,
                y: 0
            };
        }

        else if (side == 'r') {
            this.coor = {
                x: 1150,
                y: 0
            };
        }

        this.offset = {
            x: 0,
            y: -1
        };
        this.node.style.left = this.coor.x + "px";
        this.node.style.top = this.coor.y + "px";
    }
```

move()：用來偵測碰撞和一些事件，若無發生則照原本的移動。
偵測此回合贏了沒，若是則依據球的落點加分數及reset球和狗的位置。
其中又有判定是否game over，若是則跳出訊息及reset計分板。

```
159         // winning detection
160         if (this.coor.y + 100 >= 800 && this.coor.x + 100 < 790) {
161             sscorer.set(parseInt(parseInt(sscorer.get()) + 1));
162 
163             bball.init('r');
164             doglo.init(350, 640);
165             dogro.init(1150, 640);
166 
167             if (parseInt(sscorer.get()) == 11) {
168                 alert("Right Wins");
169                 sscorel.set(0);
170                 sscorer.set(0);
171             }
172             sleep(1000);
173         }
```

偵測球和狗的碰撞以反彈。

```
190     // dog and ball collision detect
191     if (this.coor.x >= doglo.coor.x && this.coor.x <= doglo.coor.x + 100 && this.coor.y >= doglo.coor.y - 50 && this.coor.y <= doglo.coor.y + 50) {
192         this.offset.x = 10;
193         this.offset.y = -this.offset.y;
194     }
195 
196     if (this.coor.x >= doglo.coor.x - 50 && this.coor.x <= doglo.coor.x && this.coor.y >= doglo.coor.y - 50 && this.coor.y <= doglo.coor.y + 50) {
197         this.offset.x = -10;
198         this.offset.y = -this.offset.y;
199     }
200 
201     if (this.coor.x >= dogro.coor.x && this.coor.x <= dogro.coor.x + 80 && this.coor.y >= dogro.coor.y - 50 && this.coor.y <= dogro.coor.y + 50) {
202         this.offset.x = 10;
203         this.offset.y = -this.offset.y;
204     }
205 
206     if (this.coor.x >= dogro.coor.x - 50 && this.coor.x <= dogro.coor.x && this.coor.y >= dogro.coor.y - 50 && this.coor.y <= dogro.coor.y + 50) {
207         this.offset.x = -10;
208         this.offset.y = -this.offset.y;
209     }
```

偵測球和邊界及網子的碰撞以反彈。

```
211         // border & net collision detection
212         // 如果從左邊碰到網子則反彈，原本offset.x正的變負的
213         if (this.coor.x == 690 && this.coor.y > 480)
214             this.offset.x = -10;
215 
216         // 如果從右邊碰到網子則反彈，原本offset.x負的變正的
217         if (this.coor.x == 810 && this.coor.y > 480)
218             this.offset.x = 10;
219 
220         // 如果碰到右邊牆壁則反彈，原本offset.x正的變負的
221         if (this.coor.x + 100 >= 1600)
222             this.offset.x = -10;
223 
224         // 如果碰到下面牆壁則反彈，原本offset.y正的變負的
225         if (this.coor.y + 100 >= 800)
226             this.offset.y = -10;
227 
228         // 如果碰到左邊牆壁則反彈，原本offset.x負的變正的
229         if (this.coor.x <= 0)
230             this.offset.x = 10;
231 
232         // 如果碰到上面牆壁則反彈，原本offset.y負的變正的
233         if (this.coor.y <= 0)
234             this.offset.y = 10;
235 
236         // 將移動變量加上本來的座標來移動球的位置
237         this.coor.x = (this.coor.x + this.offset.x);
238         this.coor.y = (this.coor.y + this.offset.y);
239 
240         this.node.style.left = this.coor.x + "px";
241         this.node.style.top = this.coor.y + "px";
242     }
243 }
```

Dog class中的function：
init(x, y)：每回合結束時重設狗的位置。

```
93      init(x, y) {
94          this.coor = {
95              x: x,
96              y: y
97          };
98          this.dog.style.left = this.coor.x + "px";
99          this.dog.style.top = this.coor.y + "px";
100     }
```

move(direction)：依照傳入參數決定移動方向，同時考慮是否超過邊界。

```
69      move(direction) {
70          if (direction == 'l' && this.coor.x > 820)
71              this.coor.x -= 10;
72          else if (direction == 'r' && this.coor.x < 1480)
73              this.coor.x += 10;
74          else if (direction == 'u' && this.coor.y > 350)
75              this.coor.y -= 10;
76          else if (direction == 'd' && this.coor.y < 640)
77              this.coor.y += 10;
78
79          if (direction == 'A' && this.coor.x > 30)
80              this.coor.x -= 10;
81          else if (direction == 'D' && this.coor.x < 660)
82              this.coor.x += 10;
83          else if (direction == 'W' && this.coor.y > 350)
84              this.coor.y -= 10;
85
86          else if (direction == 'S' && this.coor.y < 640)
87              this.coor.y += 10;
88
89          this.dog.style.left = this.coor.x + "px";
90          this.dog.style.top = this.coor.y + "px";
91      }
```

Score class中的function：
get()：回傳分數
set(i)：設定分數

```
26      set(i) {
27          this.value.nodeValue = i;
28      }
29      get() {
30          return this.value.nodeValue;
31      }
```