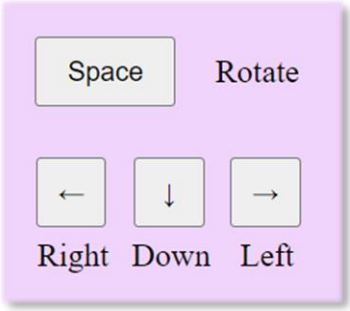
Javascript term project: Tetris

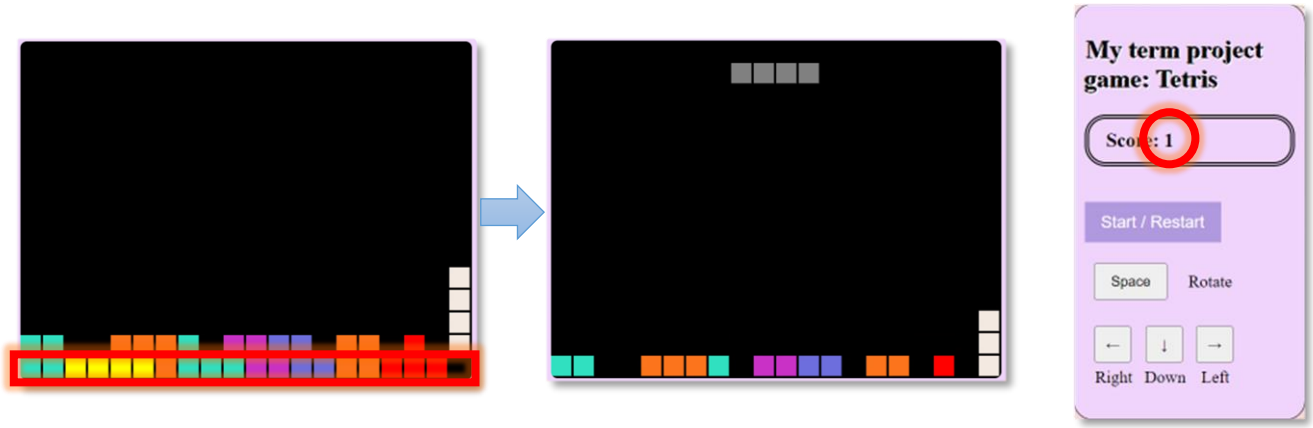
110704039 許甄芸

- 系統功能/摘要

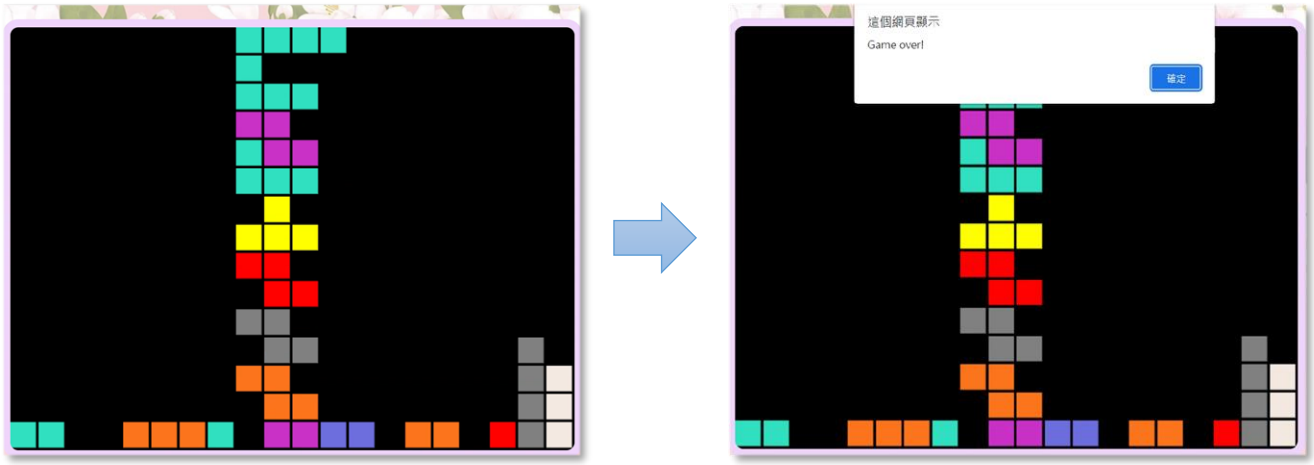
  操作: 遊戲開始後會有方塊自動落下，而使用者可使用空白鍵旋轉方塊、方向鍵←↓→分別讓方塊左移向下和右移。



  計分: 當方塊排滿一列就會消去，其餘方塊向下落一格，並且在計分板上 score 顯示加 1。



  遊戲結束: 若方塊在任何一列疊滿了一行，也就是觸碰到了上邊界，那麼網頁就會跳出“Game over!”的視窗，並且在計分的地方改為顯示 Game Over! 的字樣。



- 系統開發平台

  開發平台:Notepad++

  瀏覽器:Google Chrome.Ink

  包含檔案:Term_110704039.html

  Term_110704039.css

  Term_110704039.js

  圖片 *1(jpg 檔)

CPU type	AMD Ryzen 7 4700U with Radeon Graphics 2.00 GHz
Memory size	16.0GB
Kernel version	Windows 11 家用版
C version	Dev-C++
Machine type	64 位元作業系統，x64 型處理器

➢ 程式說明

- Term_110704039.html:

使用 canvas 創建畫布

```
<head>
    <meta charset="UTF-8">
    <title>俄羅斯方塊</title>
    <link rel = "stylesheet" type = "text/css" href = "Term_110704039.css">
</head>
<body>
    <canvas width="700" height="525" style="border:10px solid #F0D4FC;border-radius: 20px;" id="tetris"></canvas>
    <h2>My term project game: Tetris</h2>
```

使用 div 建立記分板上的文字以及各個按鍵，其中包括 h2、h3、star-button、roatate、left、down、right。

```
    <div class="info">
    <h2>My term project game: Tetris</h2>
    <h3>Score: <span id="score">0</span></h3>
    <button id="start-button">Start / Restart</button>
    <div class="intro">
      <div class="keyBoard0" disabled>
        <button class="rotate">Space</button><span>Rotate</span></div>
      <div class="keyBoard1" disabled>
        <div>
          <button class="left">←</button>
          <P>Right</P>
        </div>
        <div>
          <button class="down">↓</button>
          <P>Down</P>
        </div>
        <div>
          <button class="right">→</button>
          <P>Left</P>
        </div>
      </div>

    </div>
  </div>
```

- Term_110704039.css:

設定遊戲區域的位置

```
#tetris{
    position:fixed;
    margin: 10px 250px;
}
```

記分板的排版設計

```css
.info {
    width: 200px;
    position:fixed;
    top: 100px;
    left: 1000px;
    border: 1px solid black;
    text-indent: 0.1em;
    padding: 10px;
    border-radius: 20px;
    background-color: #F0D4FC;
}
```

Start 按鍵的樣式設計以及點擊後的反應

```css
#start-button {
  position: relative;
  background-color: rgb(177, 153, 223);
  border: none;
  font-size: 16px;
  color: #ffffff;
  padding: 10px;
  width: 130px;
  text-align: center;
  transition-duration: 0.4s;
  text-decoration: none;
  overflow: hidden;
  cursor: pointer;
  margin-top: 15px;
}
```

```css
#start-button:after {
  content: "";
  background: #f1f1f1;
  display: block;
  position: absolute;
  padding-top: 300%;
  padding-left: 350%;
  margin-left: -20px;
  margin-top: -120%;
  opacity: 0;
  transition: all 0.8s;
}
```

```css
#start-button:active:after {
  padding: 0;
  margin: 0;
  opacity: 1;
  transition: 0s;
}
```

- Term_110704039.js:

使用到的函數

gameStart()	judgeCollision_down()
init()	judgeCollision_other()
run()	down_speed_up()
initBackground()	move()
initBlock()	up_change_direction()
drawBlock()	clearUnderBlcok()
drawStaticBlock()	gameover()
createRandom()	

變數的假設與用途

```
class tetris{
    constructor(side=35, width=20, height=15, speed=400){
    this.side = side                // 每個方塊邊長
    this.width = width              // 一行包含的方塊數
    this.height = height            // 一列包含的方塊數
    this.speed = speed              // 方塊下落移動速度
    this.num_blcok                  // 當前方塊類型的數字變量
    this.type_color                 // 當前顏色類型的字符串變量
    this.ident                      // setInterval的標認
    this.direction = 1              // 方塊方向,初始化為1，默認狀態
    this.grade = 0                  // 用來計算分數
    this.over = false               // 遊戲是否結束
    this.arr_bX = []                // 存放當前方塊的X坐標
    this.arr_bY = []                // 存放當前方塊的Y坐標
    this.arr_store_X = []           // 存放到達底部所有方塊的X坐標
    this.arr_store_Y = []           // 存放到達底部所有方塊的Y坐標
    this.arr_store_color = []       // 存放到達底部所有方塊的顏色
    this.paints = document.getElementById('tetris').getContext('2d')  //獲取畫筆
    self = this
}
```

使用 fillstyle 將背景地圖填充為黑色

```
// 初始化地圖
initBackground(){
    this.paints.beginPath()
    this.paints.fillStyle='#000000'          //地圖填充顏色為黑色
    for(let i = 0; i < this.height; i++){
        for(let j = 0; j < this.width; j++){
            this.paintfr(j, i)
        }
    }
    this.paints.closePath()
}
```

使用 Math.random 生成隨機數決定方塊的類型，並以 switch case 一一列出

```
94  // 生成隨機數返回方塊類型或顏色類型
95  createRandom(type){
96      let temp = this.width/2-1
97
98      if (type == 'rBlock'){          //如果是方塊類型
99          this.num_blcok = Math.round(Math.random()*4+1)
100         switch(this.num_blcok){
101             case 1:
102                 this.arr_bX.push(temp,temp-1,temp,temp+1)
103                 this.arr_bY.push(0,1,1,1)
104                 break
105             case 2:
106                 this.arr_bX.push(temp,temp-1,temp-1,temp+1)
107                 this.arr_bY.push(1,0,1,1)
108                 break
109             case 3:
110                 this.arr_bX.push(temp,temp-1,temp+1,temp+2)
111                 this.arr_bY.push(0,0,0,0)
112                 break
113             case 4:
114                 this.arr_bX.push(temp,temp-1,temp,temp+1)
115                 this.arr_bY.push(0,0,1,1)
116                 break
117             case 5:
118                 this.arr_bX.push(temp,temp+1,temp,temp+1)
119                 this.arr_bY.push(0,0,1,1)
120                 break
121         }
122     }
```

以下為顏色的類型，也適用 switch case 一一列出

```
123     if (type == 'rColor'){          //如果是顏色類型
124         let num_color = Math.round(Math.random()*8+1)
125 
126         switch(num_color){
127             case 1:
128                 this.type_color='#3EF72A'
129                 break
130             case 2:
131                 this.type_color='yellow'
132                 break
133             case 3:
134                 this.type_color='#2FE0BF'
135                 break
136             case 4:
137                 this.type_color='red'
138                 break
139             case 5:
140                 this.type_color='gray'
141                 break
142             case 6:
143                 this.type_color='#C932C6'
144                 break
145             case 7:
146                 this.type_color= '#FC751B'
147                 break
148             case 8:
149                 this.type_color= '#6E6EDD'
150                 break
151             case 9:
152                 this.type_color= '#F4E9E1'
153                 break
154         }
155     }
```

由於方塊可以變換方向，因此就會有變換過後碰觸到邊界或碰觸到已經堆疊好的方塊等問題，因此需另外寫函數判斷碰觸。我選擇用 if else 來檢查，並包在 for 迴圈中以確保沒有遺漏之處，以下為下邊界判斷的程式碼

```
158 // 判斷方塊之間是否碰撞(下)，以及變形時是否越過下邊界
159 judgeCollision_down(){
160     for(let i = 0; i < this.arr_bX.length; i++){
161         if (this.arr_bY[i] + 1 == this.height){ //變形時是否越過下邊界
162             return false
163         }
164         if (this.arr_store_X.length != 0) {    //判斷方塊之間是否碰撞(下)
165             for(let j = 0; j < this.arr_store_X.length; j++){
166                 if (this.arr_bX[i] == this.arr_store_X[j]) {
167                     if (this.arr_bY[i] + 1 == this.arr_store_Y[j]) {
168                         return false
169                     }
170                 }
171 
172             }
173         }
174     }
175     return true
176 }
```

以下為左右邊界判斷的程式碼，和下邊界判斷用的是同個方法

```
//判斷方塊之間是否碰撞(左右)，以及變形時是否越過左右邊界
judgeCollision_other(num){
    for(let i = 0; i < this.arr_bX.length; i++){
        if (num == 1) {             //變形時是否越過右邊界
            if (this.arr_bX[i] == this.width - 1)
                return false
        }
        if (num == -1) {            //變形時是否越過左邊界
            if (this.arr_bX[i] == 0)
                return false
        }
        if (this.arr_store_X.length != 0) {                    //判斷方塊之間是否碰撞(左右)
            for(let j = 0; j < this.arr_store_X.length; j++){
                if (this.arr_bY[i] == this.arr_store_Y[j]) {
                    if (this.arr_bX[i] + num == this.arr_store_X[j]) {
                        return false
                    }
                }
            }
        }
    }
    return true;
}
```

clearUnderBlock()函數是專門處理方塊排滿一排時的反應，首先要處理排滿的那層將它刪除

```
//一行滿了清空方塊，上面方塊Y坐標+1
clearUnderBlock(){
    //刪除低層方塊
    let arr_row=[]
    let line_num
    if (this.arr_store_X.length != 0) {
        for(let j = this.height-1; j >= 0; j--){
            for(let i = 0; i < this.arr_store_color.length; i++){
                if (this.arr_store_Y[i] == j) {
                    arr_row.push(i)
                }
            }
            if (arr_row.length == this.width) {
                line_num = j
                break
            }else{
                arr_row.splice(0, arr_row.length)
            }
        }
    }
```

接下來要將分數加 1，並且讓上面一層的方塊往下掉一格:使用 for 迴圈跑每格方塊並將它們的 Y 座標+1 (this.arr_store_Y[i] = this.arr_store_Y[i] + 1)

```
399         if (arr_row.length == this.width) {
400             //計算成績grade
401             this.grade++
402 
403             document.getElementById('score').innerHTML = this.grade
404             for(let i = 0; i < arr_row.length; i++){
405                 this.arr_store_X.splice(arr_row[i]-i, 1)
406                 this.arr_store_Y.splice(arr_row[i]-i, 1)
407                 this.arr_store_color.splice(arr_row[i]-i, 1)
408             }
409 
410             //讓上面的方塊往下掉一格
411             for(let i = 0; i < this.arr_store_color.length; i++){
412                 if (this.arr_store_Y[i] < line_num) {
413                     this.arr_store_Y[i] = this.arr_store_Y[i]+1
414                 }
415             }
416         }
417 }
```

遊戲結束使用 gameover()函數來判斷。使用 for 迴圈跑每個 X 座標，並判斷 Y 軸是否排滿方塊，只要任一 X 座標的 Y 符合條件就判定為遊戲結束，網頁將會跳出”Game over!”的視窗、記分板的 score:也會顯示”Game Over!”字樣。

```
419 //判斷遊戲結束
420 gameover(){
421     for(let i=0; i < this.arr_store_X.length; i++){
422         if (this.arr_store_Y[i] == 0) {
423             clearInterval(this.ident);
424             this.over = true;
425             alert("Game over!");
426             score.innerText = "Game Over ! ";
427         }
428     }
429 
430 }
431 }
```

方向鍵以及空格的控制，使用 document.onkeydown 來捕捉”按鍵按下”的事件，並用 switch case 一一列舉，其中會使用到程式前面寫好的各個函數，像是 down_speed_up()、initBackground()、up_change_direction()…等等。

```
436  //方向鍵功能函數
437  document.onkeydown = (e) => {
438      if (tetrisObj.over)
439          return
440
441      switch(e.keyCode){
442          case 40:  // 方向為下
443              tetrisObj.down_speed_up()
444              break
445          case 32:  // 空格換方向
446              tetrisObj.initBackground()          //重畫地圖
447              tetrisObj.up_change_direction(tetrisObj.num_blcok)
448              tetrisObj.drawBlock(tetrisObj.type_color)
449              break
450          case 37:  // 方向為左
451              tetrisObj.initBackground()
452              tetrisObj.move(-1)
453              tetrisObj.drawBlock(tetrisObj.type_color)
454              break
455          case 39:  // 方向為右
456              tetrisObj.initBackground()
457              tetrisObj.move(1)
458              tetrisObj.drawBlock(tetrisObj.type_color)
459              break
460      }
461  }
```

- 結論與心得

這學期下來真的覺得 Javascript 是個很奧妙的程式語言，跟之前接觸過的都不太一樣，而寫網頁也是我第一次接觸，因此還有很多很多不足和需要學系的地方，每次上課時看到老師總是能夠很輕鬆理所當然地打出一段程式碼，總是會想自己甚麼時候也能學到那樣的程度。這次期末專題做出的成果看似精美但其實花了我很多很多的時間思考和 debug，且仍然有許多功能還能做的更多、更好，雖說整個網頁遊戲的重點都是在 js 的部分，但我最喜歡的是 css 的排版過程，能夠設計出自己想要的網頁的樣子真的是很快樂的事，這學期下來的每次作業我都花了不少時間設計與排版，希望老師看得出我的用心 XD，除此之外在寫 js 時終於寫出了通暢的邏輯並成功執行的那一刻也會想興奮地大叫哈哈~而在這個小遊戲中我有設計了一個特別的小地方，那就是記分板上的 Srart/Restart 按鍵，點擊時會有很療癒的動畫反應 XD 而這個設計是寫在 css 中的#start-button。總之謝謝老師這學期下來的教導，也很感謝當初讓我加簽這門課的您，我現在才能學到這些知識並有這些實作經驗!