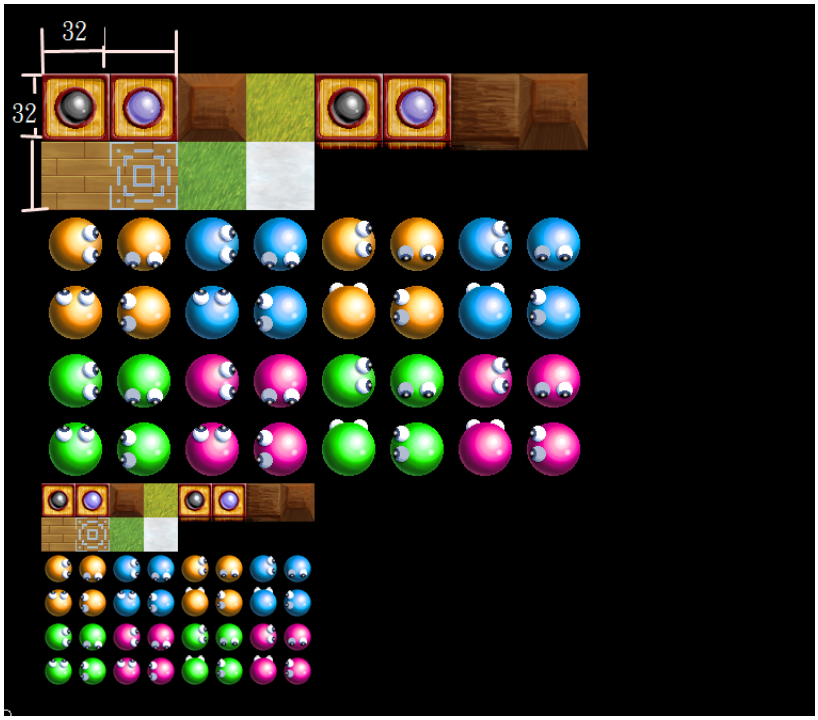
```
/**
 *  @file       index.js
 *  @brief      The entry file of Sokoban.
 *  @author     Yiwei Chiao (ywchiao@gmail.com)
 *  @date       11/17/2017 created.
 *  @date       01/05/2018 last modified.
 *  @version    0.1.0
 *  @since      0.1.0
 *  @copyright  MIT, © 2017-2018 Yiwei Chiao
 *  @details
 *
 *  The entry file of Sokoban.
 */
'use strict';

/**
 * Sokoban 符號常數
 *
 *  #    牆壁 (wall)
 *  @    玩家 (player)
 *  $    箱子 (box)
 *  .    目標點 (goal)
 *  +    玩家站在目標點上 (player on goal square)
 *  *    箱子在目標點上 (box on goal square)
 *  空白 地板 (floor)
 */
//TS Add
 var GoalNo=0; //計算關卡有幾個箱子要歸位
 var BoxOnGoalNo=0; //先在已經有幾個箱子歸位。

 const SOKOBAN = {
  BOX: '$',
  BOX_ON_GOAL: '*',
  FLOOR: ' ',
  GOAL: '.',
  GROUND: '-',
  MAN: '@',
  MAN_ON_GOAL: '+',
  WALL: '#',
};

/**
 * Sokoban 關卡描述
 */
let levels = [
  [
    "############",
    "#          .#",
    "#           #",
    "#           #",
    "#    ####   #",
    "#           #",
    "#           #",
    "#     $     #",
    "#     @     #",
    "#           #",
    "#           #",
    "############"
  ],

  [
     "------------",
     "------------",
     "--#######---",
     "--# ..$ #---",
     "--# # $ #---",
     "--# # # #---",
     "--# $@# #---",
     "--#.$   #---",
```

```
        "--#.#####---",
        "--###-------",
        "------------",
        "------------"
      ],
  ];

  /**
   * 將 'str' 的第 'x' 字元換成 'ch'。
   *
   */
  let replaceAt = (str, x, ch) => {
    let arrayOfChar = str.split('');

    arrayOfChar[x] = ch;

    return arrayOfChar.join('');
  };

  /**
   * 準備繪圖用的 sprites 資料。
   *
   * @returns sprites 集合物件。
   */
  let tileset = { 定義圖形
    src: 'SokobanClone_byVellidragon.png', 圖形存放的檔案名稱

    tile: {
      box: {
        x: 0,
        y: 0,
        width: 32,
        height: 32,
      },
      boxOnGoal: {
        x: 32,
        y: 0,
        width: 32,
        height: 32,
      },
      wall: {
        x: 64,
        y: 0,
        width: 32,
        height: 32,
      },

      floor: {
        x: 0,
        y: 32,
        width: 32,
        height: 32,
      },
      goal: {
        x: 32,
        y: 32,
        width: 32,
        height: 32,
      },
      ground: {
        x: 64,
        y: 32,
        width: 32,
        height: 32,
      },

      faceRight: {
        x: 0,
        y: 64,
```

box圖形由SokobanClone_byVellidragon.png的左上角(0,0)的位置取長寬32*32的像素



```
        width: 32,
        height: 32,
      },
      faceDown: {
        x: 32,
        y: 64,
        width: 32,
        height: 32,
      },

      faceUp: {
        x: 0,
        y: 96,
        width: 32,
        height: 32,
      },
      faceLeft: {
        x: 32,
        y: 96,
        width: 32,
        height: 32,
      },
    },
  };

  /**
   * 貼地磚函式
   */
  let tile = function (tileset, { x, y, width, height }) {
    this.brush.drawImage(
      tileset,
      x, y, width, height,
      0, 0, width, height
    );
  };

  /**
   * Sokoban 遊戲狀態物件的 prototype (原形)
   */
  let prototypeGameState = {
    isBox: function ({x, y}) {
      return (this.level[y].charAt(x) == SOKOBAN.BOX) ||
        (this.level[y].charAt(x) == SOKOBAN.BOX_ON_GOAL);
    },

    isBoxOnGoal: function ({x, y}) {
      return (this.level[y].charAt(x) == SOKOBAN.BOX_ON_GOAL);
    },

    isGoal: function ({x, y}) {
      return (this.level[y].charAt(x) == SOKOBAN.GOAL);
    },

    isMan: function ({x, y}) {
      return (this.level[y].charAt(x) == SOKOBAN.MAN) ||
        (this.level[y].charAt(x) == SOKOBAN.MAN_ON_GOAL);
    },

    isManOnGoal: function ({x, y}) {
      return (this.level[y].charAt(x) == SOKOBAN.MAN_ON_GOAL);
    },

    isVacant: function ({x, y}) {
      return (this.level[y].charAt(x) == SOKOBAN.FLOOR) ||
        (this.level[y].charAt(x) == SOKOBAN.GOAL) ||
        (this.level[y].charAt(x) == SOKOBAN.GROUND);
    },

    cellDown: function ({x, y}) {
```

```
208       return {
209         x: x,
210         y: ((y + 1) < this.level.length) ? (y + 1) : y
211       };
212     },
213
214     cellLeft: function ({x, y}) {
215       return {
216         x: (x > 0) ? (x - 1) : x,
217         y: y
218       };
219     },
220
221     cellRight: function ({x, y}) {
222       return {
223         x: ((x + 1) < this.level.length) ? (x + 1) : x,
224         y: y
225       };
226     },
227
228     cellUp: function ({x, y}) {
229       return {
230         x: x,
231         y: (y > 0) ? (y - 1) : y,
232       };
233     },
234
235     moveBox: function (oldCell, newCell) {
236       return this
237         .moveBoxOut(oldCell)
238         .moveBoxIn(newCell);
239     },
240
241     moveBoxIn: function (cell) {
242       if (this.isGoal(cell)) {
243         this.putBoxOnGoal(cell);
244       }
245       else {
246         this.putBox(cell);
247       };
248
249       return this;
250     },
251
252     moveBoxOut: function (cell) {
253       if (this.isBoxOnGoal(cell)) {
254         this.putGoal(cell);
255       }
256       else {
257         this.putFloor(cell);
258       };
259
260       return this;
261     },
262
263     moveMan: function (oldCell, newCell) {
264       return this
265         .moveManOut(oldCell)
266         .moveManIn(newCell);
267     },
268
269     moveManIn: function (cell) {
270       if (this.isGoal(cell)) {
271         this.putManOnGoal(cell);
272       }
273       else {
274         this.putMan(cell);
275       };
276
```

```
277       return this;
278     },
279
280     moveManOut: function (cell) {
281       if (this.isManOnGoal(cell)) {
282         this.putGoal(cell);
283       }
284       else {
285         this.putFloor(cell);
286       };
287
288       return this;
289     },
290
291     moveManDown: function (cell) {
292       let manCell = this.cellUp(cell);
293       let newCell = this.cellDown(cell);
294
295       if (
296         this.isBox(cell) &&
297         this.isVacant(newCell)
298       ) {
299         return this.pushBoxDown(cell);
300       }
301
302       if (this.isVacant(cell)) {
303         return this.moveMan(manCell, cell);
304       }
305
306       return this;
307     },
308
309     moveManLeft: function (cell) {
310       let manCell = this.cellRight(cell);
311       let newCell = this.cellLeft(cell);
312
313       if (
314         this.isBox(cell) &&
315         this.isVacant(newCell)
316       ) {
317         return this.pushBoxLeft(cell);
318       }
319
320       if (this.isVacant(cell)) {
321         return this.moveMan(manCell, cell);
322       }
323
324       return this;
325     },
326
327     moveManRight: function (cell) {
328       let manCell = this.cellLeft(cell);
329       let newCell = this.cellRight(cell);
330
331       if (
332         this.isBox(cell) &&
333         this.isVacant(newCell)
334       ) {
335         return this.pushBoxRight(cell);
336       }
337
338       if (this.isVacant(cell)) {
339         return this.moveMan(manCell, cell);
340       }
341
342       return this;
343     },
344
345     moveManUp: function (cell) {
```

```
346       let manCell = this.cellDown(cell);
347       let newCell = this.cellUp(cell);
348
349       if (
350         this.isBox(cell) &&
351         this.isVacant(newCell)
352       ) {
353         return this.pushBoxUp(cell);
354       }
355
356       if (this.isVacant(cell)) {
357         return this.moveMan(manCell, cell);
358       }
359
360       return this;
361     },
362
363     pushBoxDown: function (cell) {
364       let manCell = this.cellUp(cell);
365       let boxCell = this.cellDown(cell);
366
367       return this
368         .moveBox(cell, boxCell)
369         .moveMan(manCell, cell);
370     },
371
372     pushBoxLeft: function (cell) {
373       let manCell = this.cellRight(cell);
374       let boxCell = this.cellLeft(cell);
375
376       return this
377         .moveBox(cell, boxCell)
378         .moveMan(manCell, cell);
379     },
380
381     pushBoxRight: function (cell) {
382       let manCell = this.cellLeft(cell);
383       let boxCell = this.cellRight(cell);
384
385       return this
386         .moveBox(cell, boxCell)
387         .moveMan(manCell, cell);
388     },
389
390     pushBoxUp: function (cell) {
391       let manCell = this.cellDown(cell);
392       let boxCell = this.cellUp(cell);
393
394       return this
395         .moveBox(cell, boxCell)
396         .moveMan(manCell, cell);
397     },
398
399     putBox: function ({x, y}) {
400       this.level[y] = replaceAt(this.level[y], x, SOKOBAN.BOX);
401
402       return this;
403     },
404
405     putBoxOnGoal: function ({x, y}) {
406       this.level[y] = replaceAt(this.level[y], x, SOKOBAN.BOX_ON_GOAL);
407       //BoxOnGoalNo++;
408       //alert(GoalNo);
409       //if (GoalNo == 0)
410       //alert("you win");
411       return this;
412     },
413
414     putFloor: function ({x, y}) {
```

```
415       this.level[y] = replaceAt(this.level[y], x, SOKOBAN.FLOOR);
416
417       return this;
418     },
419
420     putGoal: function ({x, y}) {
421       this.level[y] = replaceAt(this.level[y], x, SOKOBAN.GOAL);
422
423       return this;
424     },
425
426     putMan: function ({x, y}) {
427       this.level[y] = replaceAt(this.level[y], x, SOKOBAN.MAN);
428
429       return this;
430     },
431
432     putManOnGoal: function ({x, y}) {
433       this.level[y] = replaceAt(this.level[y], x, SOKOBAN.MAN_ON_GOAL);
434
435       return this;
436     }
437   };
438
439   /**
440    * 繪出盤面上的格線
441    *
442    * @param 'ctx' : 繪圖 context 物件
443    * @returns {undefined}
444    */
445   let drawBoardGrid = (ctx) => {
446     // 準備一支可以畫 _斷續線_ 的畫筆
447     ctx.strokeStyle = 'black';
448     // 斷續線由連續 4px，再空白 4px構成
449     ctx.setLineDash([4, 4]);
450
451     // 開始記錄格線的 paths
452     ctx.beginPath();
453
454     // 畫 12 條鉛直斷續線
455     for (var c = 1; c < 12; c ++) {
456       ctx.moveTo(c * 32, 0);
457       ctx.lineTo(c * 32, 32*12);
458     }
459
460     // 畫 12 條水平斷續線
461     for (var r = 1; r < 12; r ++) {
462       ctx.moveTo( 0, r * 32);
463       ctx.lineTo(640, r * 32);
464     }
465
466     // 繪出格線
467     ctx.stroke();
468   };
469
470   /**
471    * Sokoban 遊戲物件
472    */
473   let sokoban = {
474     /**
475      * 依滑鼠事件 (click)，改變遊戲資料
476      *
477      * @returns {undefined}
478      */
479     move: function (e) {
480       let cell = {
481         x: Math.floor(e.offsetX / 32),
482         y: Math.floor(e.offsetY / 32),
483       };
```

```
484
485       if (this.isMan(this.cellDown(cell))) {
486         this.man = this.faceUp;
487         this.moveManUp(cell);
488       }
489
490       if (this.isMan(this.cellLeft(cell))) {
491         this.man = this.faceRight;
492         this.moveManRight(cell);
493       }
494
495       if (this.isMan(this.cellRight(cell))) {
496         this.man = this.faceLeft;
497         this.moveManLeft(cell);
498       }
499
500       if (this.isMan(this.cellUp(cell))) {
501         this.man = this.faceDown;
502         this.moveManDown(cell);
503       }
504     },
505
506     /**
507      * 依遊戲狀態，繪出盤面
508      *
509      * @returns {undefined}
510      */
511     paint: function () {
512       let height = this.level.length;
513       GoalNo=0;
514        for (let x = 0; x < height; x ++) {
515         for (let y = 0; y < height; y ++) {
516           this.brush.save();
517           this.brush.translate(32*x, 32*y);
518
519           Object.entries(SOKOBAN).some(([key, value]) => {
520             if (value == this.level[y].charAt(x)) {
521               switch (value) {
522                 case SOKOBAN.MAN:
523                   this.floor();
524                   break;
525
526                 case SOKOBAN.MAN_ON_GOAL:
527                   this.goal();
528                   break;
529               };
530
531               this[this.tiling[key]]();
532               //不能用，因為任何移動都是用pain來重新繪製，GoalNo會每次增加該關所有的goal
533               //TS add start
534               if (key=="GOAL"||key=="MAN_ON_GOAL") GoalNo++;
535               //TS add End
536               return true;
537             };
538           });
539
540           this.brush.restore();
541         };
542       };
543     },
544
545
546     /**
547      * 依傳入的遊戲關卡編號，初始遊戲
548      *
549      * @returns {undefined}
550      */
551     start: function (level) {
552       this.level = JSON.parse(JSON.stringify(levels[level]));
```

每一次重新產生時，就要將GoalNo記數器歸零

在重新繪圖的同時計算有多少個位置是GOAL或是MAN_ON_GOAL，就是尚有幾個目標尚未放上箱子。設定GoalNo變數計算目前螢幕仍有幾個Goal

```
553       //GoalNo=0; // TS Add 每次開新關卡時將 GoalNo 有幾個箱子要歸位設為 0 重新計算。
554       //BoxOnGoalNo=0;//TS add 同上
555       this.paint();
556     },
557
558     /**
559      * 貼圖函式和指令的對應表
560      */
561     tiling: {
562       BOX: 'box',
563       BOX_ON_GOAL: 'boxOnGoal',
564       FLOOR: 'floor',
565       GOAL: 'goal',
566       GROUND: 'ground',
567       MAN: 'man',
568       MAN_ON_GOAL: 'man',
569       WALL: 'wall',
570     },
571
572     /**
573      * 遊戲更新介面函式
574      *
575      * @returns {undefined}
576      */
577     update: function (e) {
578       this.move(e);
579       this.paint();
580       //alert(GoalNo);
581       if (GoalNo == 0)
582
583           alert("you win");
584       //alert(You win);
585       //[0,1,2].forEach(n=>{console.log(n);)};
586
587     },
588   };
589
590   /**
591    * 設定關卡按鈕
592    *
593    * @param 'sokoban' : 遊戲物件
594    * @returns HTML 'section' 物件，含有關卡選擇按鈕
595    */
596   let controlPane = (sokoban) => {
597     let choices = [ '第一關', '第二關', '第三關' ];
598
599     let section = document.createElement('section');
600     section.style.gridArea = '5 / 2 / 6 / 5';
601
602     choices.forEach((text, level) => {
603       let btn = document.createElement('button');
604
605       btn.style.backgroundColor = '#007fff5f';
606       btn.style.color = '#051268cf';
607       btn.style.fontSize = '2rem';
608
609       btn.textContent = text;
610       btn.value = level;
611
612       btn.addEventListener('click', e => {
613         sokoban.start(e.target.value);
614       });
615
616       section.appendChild(btn);
617     });
618
619     return section;
620   }
621
```

選定第幾關就是由此function將圖形繪出

第一次產生時呼叫Paint

Paint是主要的繪圖產生程式

每一次滑鼠移動產生時都要再一次呼叫Paint

若GoalNo變數=0時，代表一前螢幕上已無Goal，表示箱子均放置完成

```
622  /**
623   * 初始化遊戲物件
624   *
625   * @param 'ctx' : 繪圖用的 context 物件
626   * @param 'tileset': 貼圖用的 tileset 物件
627   *
628   * @returns Game 物件
629   */
630  let newGame = (ctx, tileset) => {
631    let game = Object.create(sokoban);
632    Object.setPrototypeOf(sokoban, prototypeGameState);
633
634    let spriteSheet = new Image();
635    spriteSheet.src = tileset.src;
636
637    Object.keys(tileset.tile).forEach(key => {
638      tileset.tile[key].y += 6 * 64;
639
640      game[key] = tile.bind(
641        game, spriteSheet, tileset.tile[key]
642      );
643    });
644
645    game.brush = ctx;
646    game.man = game.faceUp;
647
648    return game;
649  };
650
651  /**
652   * sokoban 程式進入點
653   *
654   * @callback
655   * @param 'load' : DOM 事件名
656   * @returns {undefined}
657   */
658  window.addEventListener('load', () => {
659    console.log("Sokoban.js loaded");
660
661    let gameTitle = document.createElement('span');
662    gameTitle.textContent = 'Sokoban';
663
664    let gameHeader = document.createElement('header');
665    gameHeader.className = 'card_header';
666
667    gameHeader.appendChild(gameTitle);
668
669    let sokobanCanvas = document.createElement('canvas');
670    let ctxPaint = sokobanCanvas.getContext('2d');
671
672    // 設定繪圖圖紙的寬高
673    sokobanCanvas.width = 32*12
674    sokobanCanvas.height = 32*12;
675
676    // 將圖紙填滿背景色
677    ctxPaint.fillStyle = 'mintcream';
678    ctxPaint.fillRect(0, 0, sokobanCanvas.width, sokobanCanvas.height);
679
680    // 繪出遊戲盤面上的格線
681    drawBoardGrid(ctxPaint);
682
683    let sokobanBoard = document.createElement('div');
684    sokobanBoard.style.gridArea = '1 / 2 / 4 / 5';
685
686    sokobanBoard.appendChild(sokobanCanvas);
687
688    let gameBoard = document.createElement('article');
689    gameBoard.className = 'card_content';
690
```

```
691     gameBoard.appendChild(sokobanBoard);
692
693     let sokoban = newGame(ctxPaint, tileset);
694
695     gameBoard.appendChild(controlPane(sokoban));
696
697     sokobanBoard.addEventListener(
698       'click',
699       sokoban.update.bind(sokoban)
700     );
701
702     let gameDesktop = document.createElement('section');
703     gameDesktop.className = 'card';
704
705     gameDesktop.appendChild(gameHeader);
706     gameDesktop.appendChild(gameBoard);
707
708     let desktop = document.querySelector('.site_body')
709     desktop.appendChild(gameDesktop);
710
711     /**
712      * 滑鼠游標移動追踪
713      *
714      * @callback
715      * @param 'mousemove' : DOM 事件名
716      * @param e : DOM event 物件
717      * @returns {undefined}
718      */
719     desktop.addEventListener('mousemove', (e) => {
720       document.getElementById('cursor_x').textContent = e.clientX;
721       document.getElementById('cursor_y').textContent = e.clientY;
722     });
723   });
724
725   // index.js
726
```

滑鼠座標顯示在browser