```javascript
/**
 *  @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',
```

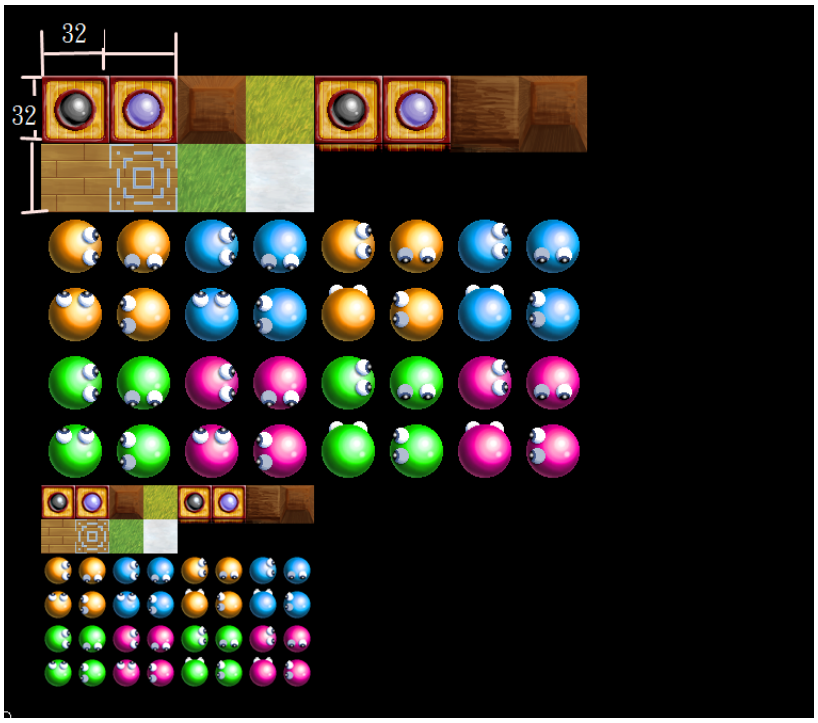
圖形存放的檔案名稱

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

```
  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,
```



```
        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}) {
      return {
        x: x,
        y: ((y + 1) < this.level.length) ? (y + 1) : y
      };
    },

    cellLeft: function ({x, y}) {
      return {
        x: (x > 0) ? (x - 1) : x,
        y: y
      };
    },

    cellRight: function ({x, y}) {
      return {
        x: ((x + 1) < this.level.length) ? (x + 1) : x,
        y: y
      };
    },

    cellUp: function ({x, y}) {
      return {
        x: x,
        y: (y > 0) ? (y - 1) : y,
      };
    },

    moveBox: function (oldCell, newCell) {
      return this
        .moveBoxOut(oldCell)
        .moveBoxIn(newCell);
    },

    moveBoxIn: function (cell) {
      if (this.isGoal(cell)) {
        this.putBoxOnGoal(cell);
      }
      else {
        this.putBox(cell);
      };

      return this;
    },

    moveBoxOut: function (cell) {
      if (this.isBoxOnGoal(cell)) {
        this.putGoal(cell);
      }
      else {
        this.putFloor(cell);
      };

      return this;
    },

    moveMan: function (oldCell, newCell) {
      return this
        .moveManOut(oldCell)
        .moveManIn(newCell);
    },

    moveManIn: function (cell) {
      if (this.isGoal(cell)) {
        this.putManOnGoal(cell);
      }
      else {
        this.putMan(cell);
      };

      return this;
    },

    moveManOut: function (cell) {
      if (this.isManOnGoal(cell)) {
        this.putGoal(cell);
      }
      else {
        this.putFloor(cell);
      };

      return this;
    },

    moveManDown: function (cell) {
      let manCell = this.cellUp(cell);
      let newCell = this.cellDown(cell);

      if (
        this.isBox(cell) &&
        this.isVacant(newCell)
      ) {
        return this.pushBoxDown(cell);
      }

      if (this.isVacant(cell)) {
        return this.moveMan(manCell, cell);
      }

      return this;
    },

    moveManLeft: function (cell) {
      let manCell = this.cellRight(cell);
      let newCell = this.cellLeft(cell);

      if (
        this.isBox(cell) &&
        this.isVacant(newCell)
      ) {
        return this.pushBoxLeft(cell);
      }

      if (this.isVacant(cell)) {
        return this.moveMan(manCell, cell);
      }

      return this;
    },

    moveManRight: function (cell) {
      let manCell = this.cellLeft(cell);
      let newCell = this.cellRight(cell);

      if (
        this.isBox(cell) &&
        this.isVacant(newCell)
      ) {
        return this.pushBoxRight(cell);
      }

      if (this.isVacant(cell)) {
        return this.moveMan(manCell, cell);
      }

      return this;
    },

    moveManUp: function (cell) {
      let manCell = this.cellDown(cell);
      let newCell = this.cellUp(cell);

      if (
        this.isBox(cell) &&
        this.isVacant(newCell)
      ) {
        return this.pushBoxUp(cell);
      }

      if (this.isVacant(cell)) {
        return this.moveMan(manCell, cell);
      }

      return this;
    },

    pushBoxDown: function (cell) {
      let manCell = this.cellUp(cell);
      let boxCell = this.cellDown(cell);

      return this
        .moveBox(cell, boxCell)
        .moveMan(manCell, cell);
    },

    pushBoxLeft: function (cell) {
      let manCell = this.cellRight(cell);
      let boxCell = this.cellLeft(cell);

      return this
        .moveBox(cell, boxCell)
        .moveMan(manCell, cell);
    },

    pushBoxRight: function (cell) {
      let manCell = this.cellLeft(cell);
      let boxCell = this.cellRight(cell);

      return this
        .moveBox(cell, boxCell)
        .moveMan(manCell, cell);
    },

    pushBoxUp: function (cell) {
      let manCell = this.cellDown(cell);
      let boxCell = this.cellUp(cell);

      return this
        .moveBox(cell, boxCell)
        .moveMan(manCell, cell);
    },

    putBox: function ({x, y}) {
      this.level[y] = replaceAt(this.level[y], x, SOKOBAN.BOX);

      return this;
    },

    putBoxOnGoal: function ({x, y}) {
      this.level[y] = replaceAt(this.level[y], x, SOKOBAN.BOX_ON_GOAL);
      //BoxOnGoalNo++;
      //alert(GoalNo);
      //if (GoalNo == 0)
      //alert("you win");
      return this;
    },

    putFloor: function ({x, y}) {
      this.level[y] = replaceAt(this.level[y], x, SOKOBAN.FLOOR);

      return this;
    },

    putGoal: function ({x, y}) {
      this.level[y] = replaceAt(this.level[y], x, SOKOBAN.GOAL);

      return this;
    },

    putMan: function ({x, y}) {
      this.level[y] = replaceAt(this.level[y], x, SOKOBAN.MAN);

      return this;
    },

    putManOnGoal: function ({x, y}) {
      this.level[y] = replaceAt(this.level[y], x, SOKOBAN.MAN_ON_GOAL);

      return this;
    }
  };

  /**
   * 繪出盤面上的格線
   *
   * @param 'ctx' : 繪圖 context 物件
   * @returns {undefined}
   */
  let drawBoardGrid = (ctx) => {
    // 準備一支可以畫 _斷續線_ 的畫筆
    ctx.strokeStyle = 'black';
    // 斷續線由連續 4px，再空白 4px構成
    ctx.setLineDash([4, 4]);

    // 開始記録格線的 paths
    ctx.beginPath();

    // 畫 12 條鉛直斷續線
    for (var c = 1; c < 12; c ++) {
      ctx.moveTo(c * 32, 0);
      ctx.lineTo(c * 32, 32*12);
    }

    // 畫 12 條水平斷續線
    for (var r = 1; r < 12; r ++) {
      ctx.moveTo( 0, r * 32);
      ctx.lineTo(640, r * 32);
    }

    // 繪出格線
    ctx.stroke();
  };

  /**
   * Sokoban 遊戲物件
   */
  let sokoban = {
    /**
     * 依滑鼠事件 (click)，改變遊戲資料
     *
     * @returns {undefined}
     */
    move: function (e) {
      let cell = {
        x: Math.floor(e.offsetX / 32),
        y: Math.floor(e.offsetY / 32),
      };

      if (this.isMan(this.cellDown(cell))) {
        this.man = this.faceUp;
        this.moveManUp(cell);
      }

      if (this.isMan(this.cellLeft(cell))) {
        this.man = this.faceRight;
        this.moveManRight(cell);
      }

      if (this.isMan(this.cellRight(cell))) {
        this.man = this.faceLeft;
        this.moveManLeft(cell);
      }

      if (this.isMan(this.cellUp(cell))) {
        this.man = this.faceDown;
        this.moveManDown(cell);
      }
    },

    /**
     * 依遊戲狀態，繪出盤面
     *
     * @returns {undefined}
     */
    paint: function () {
      let height = this.level.length;
      GoalNo=0;
```

每一次重新產生時，就要將GoalNo記數器歸零

```
       for (let x = 0; x < height; x ++) {
        for (let y = 0; y < height; y ++) {
          this.brush.save();
          this.brush.translate(32*x, 32*y);

          Object.entries(SOKOBAN).some(([key, value]) => {
            if (value == this.level[y].charAt(x)) {
              switch (value) {
                case SOKOBAN.MAN:
                  this.floor();
                  break;

                case SOKOBAN.MAN_ON_GOAL:
                  this.goal();
                  break;
              };

              this[this.tiling[key]]();
              //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
              //TS add start
              if (key=="GOAL"||key=="MAN_ON_GOAL") GoalNo++;
              //TS add End
              return true;
            };
          });

          this.brush.restore();
        };
      };
    },
```

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

```


    /**
     * 依傳入的遊戲關卡編號，初始遊戲
     *
     * @returns {undefined}
     */
    start: function (level) {
      this.level = JSON.parse(JSON.stringify(levels[level]));
      //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。
      //BoxOnGoalNo=0;//TS add 同上
      this.paint();
    },
```

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

```
  /**
   * 貼圖函式和指令的對應表
   */
  tiling: {
    BOX: 'box',
    BOX_ON_GOAL: 'boxOnGoal',
    FLOOR: 'floor',
    GOAL: 'goal',
    GROUND: 'ground',
    MAN: 'man',
    MAN_ON_GOAL: 'man',
    WALL: 'wall',
  },

  /**
   * 遊戲更新介面函式
   *
   * @returns {undefined}
   */
  update: function (e) {
    this.move(e);
    this.paint();
    //alert(GoalNo);
    if (GoalNo == 0)

        alert("you win");
    //alert(You win);
    //[0,1,2].forEach(n=>{console.log(n);)};

  },
};
```

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

```
/**
 * 設定關卡按鈕
 *
 * @param 'sokoban' : 遊戲物件
 * @returns HTML 'section' 物件，含有關卡選擇按鈕
 */
let controlPane = (sokoban) => {
  let choices = [ '第一關', '第二關', '第三關' ];

  let section = document.createElement('section');
  section.style.gridArea = '5 / 2 / 6 / 5';

  choices.forEach((text, level) => {
    let btn = document.createElement('button');

    btn.style.backgroundColor = '#007fff5f';
    btn.style.color = '#051268cf';
    btn.style.fontSize = '2rem';

    btn.textContent = text;
    btn.value = level;

    btn.addEventListener('click', e => {
      sokoban.start(e.target.value);
    });

    section.appendChild(btn);
  });

  return section;
}
```

```javascript
/**
 * 初始化遊戲物件
 *
 * @param 'ctx' : 繪圖用的 context 物件
 * @param 'tileset': 貼圖用的 tileset 物件
 *
 * @returns Game 物件
 */
let newGame = (ctx, tileset) => {
  let game = Object.create(sokoban);
  Object.setPrototypeOf(sokoban, prototypeGameState);

  let spriteSheet = new Image();
  spriteSheet.src = tileset.src;

  Object.keys(tileset.tile).forEach(key => {
    tileset.tile[key].y += 6 * 64;

    game[key] = tile.bind(
      game, spriteSheet, tileset.tile[key]
    );
  });

  game.brush = ctx;
  game.man = game.faceUp;

  return game;
};

/**
 * sokoban 程式進入點
 *
 * @callback
 * @param 'load' : DOM 事件名
 * @returns {undefined}
 */
window.addEventListener('load', () => {
  console.log("Sokoban.js loaded");

  let gameTitle = document.createElement('span');
  gameTitle.textContent = 'Sokoban';

  let gameHeader = document.createElement('header');
  gameHeader.className = 'card_header';

  gameHeader.appendChild(gameTitle);

  let sokobanCanvas = document.createElement('canvas');
  let ctxPaint = sokobanCanvas.getContext('2d');

  // 設定繪圖圖紙的寬高
  sokobanCanvas.width = 32*12
  sokobanCanvas.height = 32*12;

  // 將圖紙埴滿背景色
  ctxPaint.fillStyle = 'mintcream';
  ctxPaint.fillRect(0, 0, sokobanCanvas.width, sokobanCanvas.height);

  // 繪出遊戲盤面上的格線
  drawBoardGrid(ctxPaint);

  let sokobanBoard = document.createElement('div');
  sokobanBoard.style.gridArea = '1 / 2 / 4 / 5';

  sokobanBoard.appendChild(sokobanCanvas);

  let gameBoard = document.createElement('article');
  gameBoard.className = 'card_content';

```

```
    gameBoard.appendChild(sokobanBoard);

    let sokoban = newGame(ctxPaint, tileset);

    gameBoard.appendChild(controlPane(sokoban));

    sokobanBoard.addEventListener(
      'click',
      sokoban.update.bind(sokoban)
    );

    let gameDesktop = document.createElement('section');
    gameDesktop.className = 'card';

    gameDesktop.appendChild(gameHeader);
    gameDesktop.appendChild(gameBoard);

    let desktop = document.querySelector('.site_body')
    desktop.appendChild(gameDesktop);

    /**
     * 滑鼠游標移動追踪
     *
     * @callback
     * @param 'mousemove' : DOM 事件名
     * @param e : DOM event 物件
     * @returns {undefined}
     */
    desktop.addEventListener('mousemove', (e) => {
      document.getElementById('cursor_x').textContent = e.clientX;
      document.getElementById('cursor_y').textContent = e.clientY;
    });
  });

  // index.js
```

滑鼠座標顯示在browser