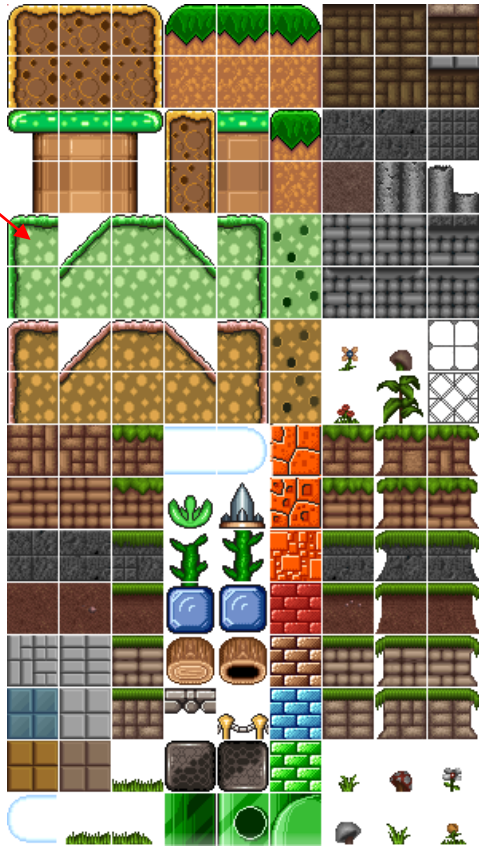
TileMap Creare come creare la scena del gioco

Adesso che sappiamo come far muovere il nostro player in una ipotetica mappa vediamo come realizzare uno scenario utilizzando una **tilemap**: una texture unica, composta da piccole porzioni dette “tiles” (mattonelle, piastrelle) che vengono assemblate per creare l’ambiente risparmiando risorse grafiche.



Caricare le tiles

Predisponiamo l’utizzo di una mappa di gioco caricando, in main.s, l’immagine delle tiles:

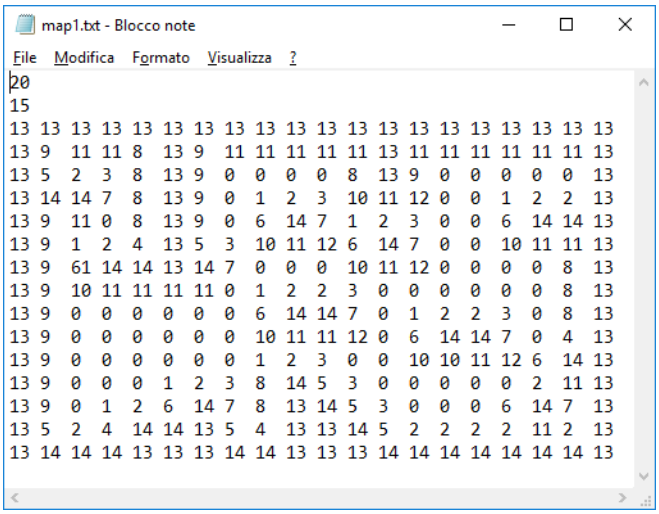
```
this.imgTiles =
this.gr.LoadSprite("immagini/tileMia.png", 1);
```

e nel metodo ResetLevel inizializziamo l’array delle tiles:

```
this.tiles = [];
```

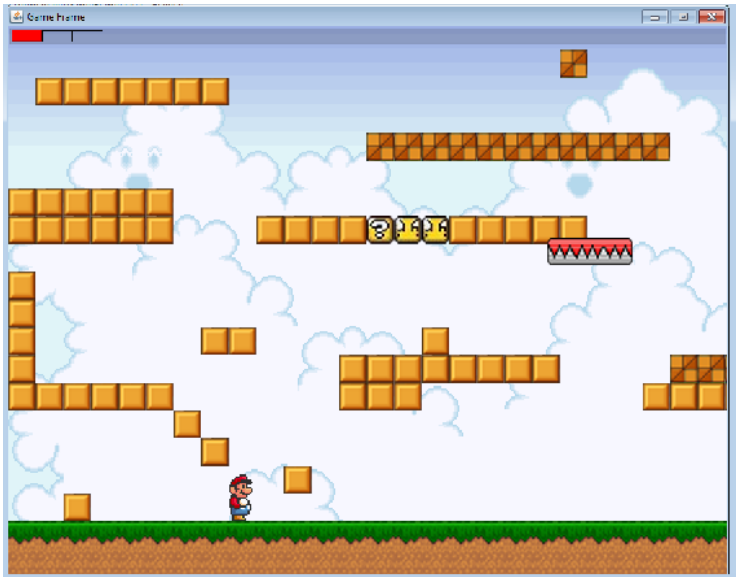
Prima di andare avanti è opportuno soffermarci ad immaginare come rappresentare dal punto di vista dei dati la nostra mappa per far corrispondere allo scenario che vogliamo realizzare le mattonelle del file che rappresentano le tessere del mosaico che vogliamo realizzare. Per esempio nel classico gioco di Super Mario:

Come si vede dalla figura qui a fianco essa contiene tutte le tessere della mappa del gioco. In fase di inizializzazione del gioco si caricherà la mappa che rappresenta la disposizione di ogni mattonella sullo schermata di gioco. Questa normalmente è contenuta in un file di testo che rappresenta una matrice NxM dove in ogni cella è contenuto un byte che rappresenta la posizione lineare della mattonella sulla tileMap. Nelle prime due righe della mappa sono contenute le dimensioni di colonne e righe. Quindi una volta caricata in una matrice NxM tutti i dati numerici corrispondenti alla nostra mappa dobbiamo ridisegnarla sul video ad ogni rendering semplicemente prelevando dalla matrice il numero di mattonella corrente ritagliarlo dalla tileMap e disegnarlo sullo schermo nella posizione x,y della nostra mappa .



Tanto per avere un’idea di fianco è riportato un possibile scenario semplificato. Infatti nella matrice non dovranno essere presenti solo i codici delle mattonelle ma anche quelli degli attori del gioco.

E qui sotto riporto il rendering ipotetico sullo schermo



Purtroppo questo approccio non è gestibile in modo soddisfacente in javascript. Infatti con Javascript è possibile leggere un file di testo con un oggetto filereader ma quello che otteniamo è un'unica stringa che contiene tutto il contenuto del file e quindi poi sarebbe abbastanza complicato splittare la stringa per ricostruire la matrice di tiles.

Quindi utilizzeremo un approccio completamente diverso sfruttando la potenzialità di gestire gli array di javascript. In javascript un array è una lista di oggetti che può essere semplicemente creato, scrivendo per un array di stringhe ad esempio:

```
var frutta = [];
 frutta.push('banana', 'mela', 'pera');// Il metodo push() aggiunge uno o più elementi alla
                                       //fine di un array e ne restituisce la nuova lunghezza.
 console.log(fruits.length); // 3
```

la numerosità è dinamica e quindi si può aggiungere, in qualsiasi momento, un nuovo frutto:

```
frutta.push('kiwi');
```

ovviamente è possibile modificare un elemento con il codice classico di un vettore

```
frutta[2] = 'mango';
```

e la nuova configurazione dell'array sarebbe:

```
['banana', 'mela', 'mango', 'kiwi']
```

ora siccome un elemento del vettore può essere qualsiasi oggetto, può essere anche un vettore e quindi si può costruire ad esempio una matrice come un array di array. Ad esempio una scacchiera potrebbe essere rappresentata come segue:

```
var scacchiera = [
                ['T','C','A','Q','K','A','C','T'],
                ['P','P','P','P','P','P','P','P'],
                [' ',' ',' ',' ',' ',' ',' ',' '],
                [' ',' ',' ',' ',' ',' ',' ',' '],
                [' ',' ',' ',' ',' ',' ',' ',' '],
                [' ',' ',' ',' ',' ',' ',' ',' '],
                ['p','p','p','p','p','p','p','p'],
                ['t','c','a','q','k','a','c','t']
                ];
```

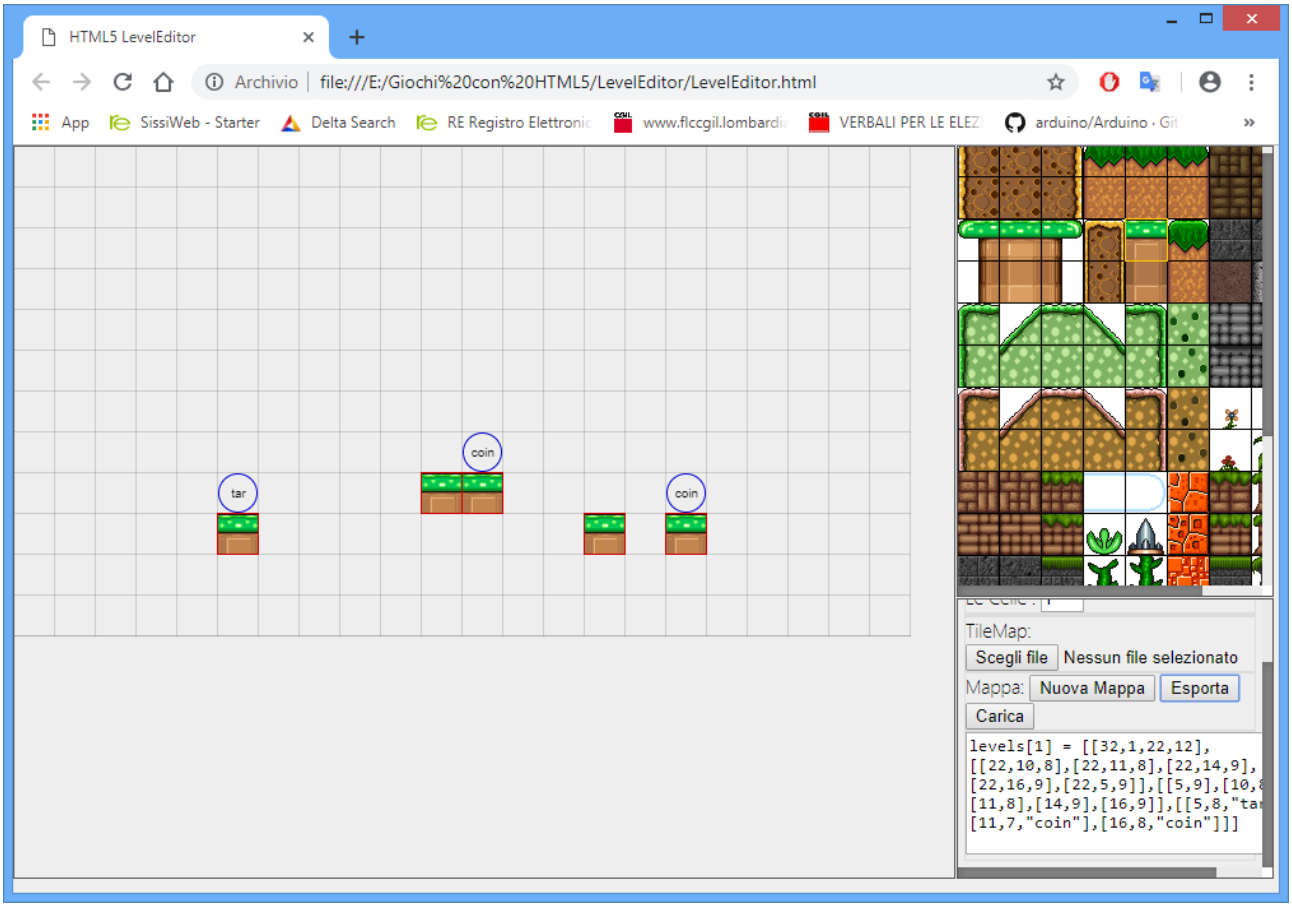
in pratica ogni cella del vettore scacchiera è a sua volta un vettore e si può quindi referenziare una cella con la notazione classica ad esempio scacchiera[0][2] contiene 'A'.

con questo approccio le celle del vettore più esterno possono contenere anche vettori di lunghezza variabile con contenuto eterogeneo Ad esempio:

```
var mappa = [
  [2,3,4,5],
  [20,12],
  [12,13,'Pippo']];
```

Sfruttando questa caratteristica possiamo rappresentare la mappa di un livello del gioco come un vettore di vettori dove i vettori interni possono contenere varie informazioni della mappa di gioco come ad esempio: un array di array di tiles dove ogni sottoarray contiene l'indice lineare della tile nella tileImage e gli indici di colonna e riga di rappresentazione sulla mappa, un array per contenere i blocchi di collisione, un array per contenere i nemici ecc. Per editare l'array della mappa io mi sono costruito una piccola applicazione tileEdit utilizzabile al seguente indirizzo: http://www.prof.accarino.altervista.org/TileEdit/TileEdit.html.

Ad esempio se guardiamo lo screenshot dell'applicazione:



Si vede come è possibile editare la nostra mappa posizionando le tile sullo schema della mappa e rappresentare i blocchi di collisione e i nemici ecc. Se guardiamo in basso a destra si vede come viene esportato un livello. Esso altro non è che una cella di un vettore di livelli dove la prima cella di questo sottovettore contiene un vettore di 4 elementi che rappresentano le informazioni sulla tileMap e nello specifico 32 dimensione di una tile 1 spazio tra le tile e 22 e 12 sono il numero di colonne e di righe della mappa. La seconda cella è un vettore di vettori dove ognuno contiene numero di tile e sua posizione sulla mappa col,rig. la terza cella contiene un array riguardanti le informazioni dei blocchi di collisione e la quarta cella contiene un array per i nemici.

esaminiamo nel dettaglio il vettore:

levels[1] = [[32,1,22,12], `//la prima cella è un vettore dimensione cella, spazio tra celle, numero colonne e numero righe`

[[22,10,8],[22,11,8],[22,14,9],[22,16,9],[22,5,9]], `//la seconda cella è una matrice dove ogni riga rappresenta il numero di tiles e le sue coordinate col,rig`

[[5,9],[10,8],[11,8],[14,9],[16,9]], `//la terza cella è una matrice dove ogni riga rappresenta le coordinate dei blocchi collidenti e l'ultima cella contiene la matrice degli altri oggetti`

[[5,8,"tar"],[11,7,"coin"],[16,8,"coin"]]]fine cella vettore

Io utilizzerò questa struttura e inserirò i vettori dei livelli in un ulteriore file di nome livelli.js.

Di seguito è riportato il codice da inserire in questo file in cui si dichiara un vettore di livelli e si imposta come primo livello che io ho realizzato utilizzando la mia applicazione per editare la mappa.

Ecco il codice:

```
levels = [];

levels[1] =
[[64,2,80,24],[[27,0,24],[27,1,24],[27,2,24],[5,9,20],[1,22,18],[1,22,19],[1,22,20],[1,22,21],[1,22,22],[1,14,22],[1,14,21],[1,14,20],[1,14,19],[1,14,18],[1,20,18],[1,21,18],[1,21,19],[1,21,20],[1,15,20],[1,15,19],[1,15,18],[1,16,18],[1,17,18],[1,18,18],[1,19,18],[1,20,19],[1,16,19],[1,15,17],[1,21,17],[1,20,17],[1,19,17],[1,18,17],[1,17,17],[1,16,17],[42,15,23],[42,15,22],[42,16,22],[42,16,21],[42,17,21],[42,17,20],[42,18,20],[42,19,20],[42,19,21],[42,20,21],[42,21,22],[42,22,24],[42,21,24],[42,20,23],[42,19,23],[42,17,23],[42,16,23],[42,17,22],[42,18,22],[42,18,21],[42,19,22],[42,20,22],[42,21,23],[34,11,23],[34,11,22],[34,11,21],[34,11,20],[71,17,14],[71,18,14],[71,19,14],[53,20,13],[53,16,13],[53,17,13],[53,18,13],[53,19,13],[42,16,14],[42,20,14],[42,2,23],[42,3,23],[42,3,22],[42,4,22],[42,5,23],[42,4,23],[42,9,24],[42,10,24],[42,11,24],[42,12,24],[42,13,24],[42,14,24],[42,15,24],[70,15,14],[70,21,14],[61,13,16],[42,18,23],[42,20,24],[42,26,24],[42,25,24],[42,27,24],[42,30,24],[42,32,24],[42,30,23],[42,29,23],[42,29,22],[42,28,22],[42,28,23],[42,27,23],[42,31,23],[42,30,21],[42,30,20],[42,30,22],[42,27,21],[42,27,22],[34,29,17],[34,33,17],[52,33,16],[52,32,16],[52,31,16],[52,30,16],[52,29,16],[13,28,18],[10,34,20],[42,32,23],[42,33,23],[42,33,22],[42,32,22],[42,32,21],[42,33,21],[42,31,22],[70,28,17],[34,36,25],[34,37,25],[43,58,24],[10,46,24],[6,49,24],[6,50,24],[42,39,23],[42,40,23],[42,41,23],[42,42,23],[42,43,23],[42,44,23],[42,45,23],[42,46,22],[42,46,21],[42,46,20],[42,47,20],[42,48,20],[42,48,19],[42,49,19],[42,49,21],[42,50,21],[42,50,20],[42,50,22],[42,50,23],[42,49,23],[42,48,23],[42,47,23],[42,46,23],[42,47,21],[42,48,21],[42,48,22],[42,50,19],[42,38,23],[42,40,22],[42,41,22],[42,41,21],[43,47,22],[43,49,20],[43,49,22],[43,49,17],[43,49,18],[43,44,21],[43,45,21],[43,44,22],[43,45,22],[43,42,22],[1,48,24],[34,53,16],[34,53,17],[34,53,18],[34,53,21],[34,53,22],[34,53,23],[34,53,24],[34,58,16],[34,58,17],[34,58,18],[34,58,19],[34,58,20],[34,58,21],[34,58,22],[34,58,23],[10,57,24],[10,56,24],[10,55,24],[34,53,15],[34,58,15],[42,69,23],[42,70,23],[42,71,22],[42,71,21],[42,72,21],[42,72,22],[42,72,23],[42,71,23],[42,71,24],[42,72,24],[42,73,22],[42,74,22],[42,73,21],[42,74,23],[42,75,22],[42,75,23],[42,78,23],[43,73,23],[43,68,22],[34,55,16],[34,56,16],[34,56,17],[34,56,18],[34,55,18],[34,55,19],[34,55,20],[34,55,21],[34,55,22],[34,56,22],[34,56,23],[34,56,21],[34,56,20],[34,56,19],[34,55,17],[34,55,23],[34,55,25],[34,55,11],[34,55,10],[34,56,10],[34,56,11],[10,79,21],[10,79,20],[10,79,19],[10,79,18],[10,79,17],[10,79,
```

16],[10,79,15],[10,79,14],[10,80,11],[10,79,13],[10,79,12],[10,79,11],[10,79,10],[10,79,9],[10,79,8],[10,79,7],[10,79,6],[10,79,5],[10,79,4],[10,79,3],[10,79,2],[10,79,1],[10,79,0],[10,79,0],[10,79,2],[34,78,14],[34,78,15],[34,78,16],[34,78,17],[34,78,18],[42,79,23],[42,80,23],[42,80,24],[42,77,24],[34,11,19],[34,63,6],[34,63,5],[34,63,4],[34,63,3],[34,63,2],[34,63,1],[34,63,0],[34,63,2],[34,63,3],[34,63,0],[34,64,0],[34,64,2],[43,53,14],[43,58,14],[44,54,15],[44,55,15],[44,56,15],[44,57,15],[17,57,16],[17,54,16],[16,54,17],[16,54,18],[16,54,19],[16,54,20],[16,54,23],[16,54,22],[16,54,21],[16,57,23],[16,57,22],[16,57,21],[16,57,20],[16,57,19],[16,57,18],[16,57,17],[34,53,19],[34,53,20],[44,53,13],[44,54,13],[44,55,13],[44,56,13],[44,57,13],[44,58,13],[69,75,17],[69,40,18],[70,45,18],[34,11,18],[0,1,19],[79,0,19],[80,1,19],[72,0,20],[73,1,20],[73,1,21],[73,1,22],[72,0,21],[72,0,22],[82,2,21],[82,5,21],[82,6,21],[82,6,22],[82,7,22],[82,7,21],[82,8,21],[82,8,22],[82,8,23],[82,9,21],[82,9,22],[82,9,23],[4,2,20],[4,3,20],[4,4,20],[4,5,20],[4,6,20],[4,7,20],[4,8,20],[142,3,19],[142,6,19],[142,7,19],[142,9,19],[1,14,17],[0,13,17],[1,22,17],[2,23,17],[54,14,16],[56,15,16],[56,16,16],[56,17,16],[56,18,16],[56,19,16],[56,20,16],[56,21,16],[58,22,16],[143,14,15],[141,22,15],[134,17,11],[134,19,11],[97,16,12],[96,17,12],[96,18,12],[96,19,12],[98,20,12],[115,15,15],[114,16,15],[114,17,15],[114,18,15],[114,19,15],[114,20,15],[116,21,15],[9,13,18],[9,13,19],[9,13,20],[9,13,21],[9,13,22],[9,13,23],[11,23,18],[11,23,19],[11,23,20],[11,23,21],[11,23,22],[11,23,23],[119,17,19],[119,18,19],[119,19,19],[119,16,20],[119,15,21],[119,14,23],[119,20,20],[119,21,21],[119,22,23],[119,0,23],[119,1,23],[119,2,22],[119,3,21],[119,4,21],[119,5,22],[119,6,23],[119,7,23],[128,29,14],[128,31,14],[128,30,14],[128,32,14],[128,33,14],[79,29,15],[78,30,15],[78,31,15],[78,32,15],[80,33,15],[62,30,17],[62,31,17],[62,32,17],[3,26,16],[5,27,16],[13,26,17],[13,27,17],[13,26,18],[13,27,18],[13,26,19],[13,27,19],[13,26,20],[13,26,21],[13,26,22],[13,26,23],[13,28,19],[13,29,18],[13,30,18],[13,31,18],[13,32,18],[13,33,18],[13,29,19],[13,29,20],[13,28,20],[13,27,20],[13,28,21],[13,29,21],[13,30,19],[13,31,19],[13,32,19],[13,33,19],[13,34,19],[13,33,20],[13,32,20],[13,31,20],[13,31,21],[13,34,21],[13,34,22],[13,34,23],[5,34,18],[79,38,21],[72,38,22],[73,39,21],[73,39,22],[73,40,21],[73,39,20],[79,39,19],[73,40,20],[73,41,20],[73,42,20],[73,42,21],[73,43,21],[73,43,22],[73,44,20],[73,43,20],[74,40,19],[136,41,18],[137,42,18],[74,41,19],[74,42,19],[80,43,19],[79,45,19],[79,46,18],[79,47,17],[79,48,16],[79,49,14],[80,50,14],[138,45,20],[138,46,19],[138,47,19],[138,47,18],[138,48,18],[138,48,17],[138,49,16],[138,49,15],[138,50,15],[138,50,16],[138,50,17],[138,50,18],[121,54,14],[121,55,14],[121,56,14],[121,57,14],[84,54,11],[84,57,11],[97,55,9],[98,56,9],[97,53,12],[96,54,12],[96,55,12],[96,56,12],[96,57,12],[98,58,12],[77,60,9],[77,59,10],[77,61,14],[77,60,15],[77,61,16],[77,60,17],[77,61,18],[77,60,19],[77,61,20],[77,60,21],[77,61,22],[77,60,23],[31,63,23],[31,63,22],[31,63,21],[31,63,20],[31,63,19],[31,63,18],[31,63,17],[31,63,16],[31,63,15],[31,63,14],[22,63,13],[22,64,14],[31,64,15],[31,64,16],[31,64,17],[31,64,18],[31,64,19],[31,64,20],[31,64,21],[31,64,22],[31,64,23],[18,61,7],[19,62,7],[19,63,7],[19,64,7],[20,65,7],[18,61,8],[19,62,8],[19,63,8],[19,64,8],[20,65,8],[137,77,11],[137,76,11],[4,78,11],[4,77,12],[4,76,12],[3,75,12],[80,74,16],[80,75,18],[80,73,15],[80,72,14],[70,73,14],[142,70,9],[101,69,10],[101,70,10],[106,68,10],[107,71,10],[106,67,12],[141,67,11],[100,68,11],[100,69,11],[100,70,11],[100,71,11],[100,71,12],[100,70,12],[100,69,12],[100,68,12],[100,68,13],[100,67,13],[100,67,14],[100,67,15],[100,67,16],[100,67,17],[100,67,18],[100,67,19],[100,68,19],[100,68,20],[100,69,20],[100,69,21],[100,70,21],[100,70,20],[100,69,19],[100,68,18],[100,68,17],[100,68,16],[100,68,15],[100,68,14],[100,69,13],[100,71,13],[100,70,13],[100,70,14],[100,69,14],[100,69,15],[100,69,16],[100,69,17],[100,69,18],[100,70,15],[100,71,14],[100,72,15],[100,71,15],[100,72,16],[100,71,16],[10

0,70,16],[100,70,17],[100,70,18],[100,70,19],[100,71,17],[100,71,18],[100,71,19],[100,72,19],[100,72,18],[100,72,17],[100,73,16],[100,73,17],[100,73,18],[100,73,19],[100,74,19],[100,74,18],[100,74,17],[100,75,19],[100,78,20],[100,78,21],[100,77,22],[100,76,22],[100,76,21],[100,76,20],[100,75,20],[100,74,20],[100,77,20],[100,77,21],[128,59,9],[128,60,8],[128,63,12],[128,64,13],[121,60,14],[121,61,13],[129,61,15],[129,60,16],[129,61,17],[129,60,18],[129,61,19],[129,60,20],[129,61,21],[129,60,22],[129,61,23],[93,62,6],[93,64,6],[17,75,13],[17,76,13],[17,77,13],[17,78,13],[4,78,12],[101,76,19],[101,77,19],[101,78,19],[44,68,21],[44,71,20],[44,72,20],[44,73,20],[44,74,21],[44,75,21],[44,69,22],[44,70,22],[44,76,23],[44,77,23],[44,78,22],[44,79,22]],[[68,10],[69,10],[70,10],[71,10],[68,11],[67,12],[67,13],[67,14],[67,15],[67,16],[67,17],[67,18],[65,8],[64,8],[63,8],[62,8],[61,8],[71,11],[71,12],[71,13],[72,14],[73,15],[74,16],[74,17],[74,18],[75,19],[78,19],[79,18],[79,17],[79,16],[79,15],[79,14],[78,13],[77,13],[76,13],[75,13],[75,12],[76,12],[77,12],[78,11],[79,10],[79,9],[79,8],[79,7],[79,6],[79,5],[79,4],[79,3],[79,2],[63,13],[63,14],[63,15],[63,16],[63,17],[63,18],[58,17],[58,19],[53,15],[54,15],[55,15],[56,15],[57,15],[58,15],[58,16],[58,18],[58,20],[58,21],[58,22],[58,23],[53,16],[53,17],[53,18],[53,19],[53,20],[50,19],[50,18],[50,17],[50,16],[50,15],[49,15],[48,16],[47,17],[46,18],[45,19],[44,20],[43,19],[50,20],[50,21],[50,22],[50,23],[42,19],[41,19],[40,19],[39,19],[39,20],[39,21],[38,22],[38,23],[34,23],[34,22],[34,21],[34,20],[34,19],[34,18],[33,18],[32,18],[31,18],[30,18],[29,18],[28,18],[27,18],[27,17],[27,16],[26,16],[26,17],[26,18],[26,19],[26,20],[26,21],[26,22],[26,23],[29,15],[30,15],[31,15],[32,15],[33,15],[33,16],[32,16],[31,16],[30,16],[29,16],[23,17],[23,18],[23,19],[23,20],[23,21],[23,22],[23,23],[22,16],[21,15],[20,15],[19,15],[18,15],[17,15],[16,15],[20,13],[19,13],[18,13],[17,13],[16,13],[16,12],[17,12],[18,12],[19,12],[20,12],[15,15],[14,16],[13,17],[13,18],[13,19],[13,20],[13,21],[13,22],[13,23],[11,18],[11,19],[11,20],[11,21],[11,22],[11,23],[9,20],[9,21],[9,22],[9,23],[9,24],[0,23],[0,22],[0,21],[0,20],[1,20],[2,20],[3,20],[4,20],[5,20],[6,20],[7,20],[8,20],[53,12],[54,12],[55,12],[56,12],[57,12],[58,12],[55,9],[56,9],[50,14],[38,21],[49,14],[77,19],[75,18],[64,18],[64,17],[64,16],[64,15],[64,14],[60,9],[59,10],[0,19],[1,19],[60,15],[61,14],[76,19]],[[17,14,"coin"],[18,14,"coin"],[19,14,"coin"],[30,17,"coin"],[31,17,"coin"],[32,17,"coin"],[54,14,"coin"],[55,14,"coin"],[56,14,"coin"],[57,14,"coin"],[75,11,"coin"],[76,11,"coin"],[77,11,"coin"],[68,8,"coin"],[67,7,"coin"],[65,6,"coin"],[66,6,"coin"],[29,14,"coin"],[30,14,"coin"],[31,14,"coin"],[32,14,"coin"],[33,14,"coin"],[38,20,"coin"],[44,19,"coin"],[46,17,"coin"],[48,15,"coin"],[49,13,"coin"],[50,13,"coin"],[5,19,"coin"],[6,19,"coin"],[7,19,"coin"],[11,17,"coin"],[53,11,"coin"],[52,10,"coin"],[51,9,"coin"],[50,8,"coin"],[23,16,"coin"],[26,15,"coin"],[27,15,"coin"],[63,10,"coin"],[76,18,"spikes"],[77,18,"spikes"],[76,17,"coin"],[77,17,"coin"],[77,16,"coin"],[77,15,"coin"],[77,14,"coin"],[76,14,"coin"],[76,15,"coin"],[76,16,"coin"],[72,8,"umbrella"],[41,16,"umbrella"],[25,15,"umbrella"],[78,10,"portale"],[1,18,"inizio"]]];

nel vettore sono già inseriti nemici, monete posizione di pertenza del player e punto di ultimazione del livello. Queste informazioni le utilizzeremo in seguito, per adesso limitiamoci a copiare tutto il codice precedente nel file levels.js e ovviamente lo importiamo in Index.html.
Proviamo a vedere se la mappa viene visualizzata correttamente. In questa fase visualizzeremo anche i blocchi cosiddetti duri cioè quelli che provocano al player una collisione fermandone il movimento.

Adesso che sappiamo come muoverci torniamo al nostro gioco e inseriamo `In` *LoadLevel*,

al posto del codice precedentemente inserito per provare le collisioni disegnando dei semplici blocchi rettangolari:

```
for(i=0; i<10; i++){
    this.blocks.push(new Block(150 + i*64, 600));
    }
 this.blocks.push(new Block(150 , 536));
 this.blocks.push(new Block(576+150 , 536));
 this.blocks.push(new Block(350 , 350));
```

andiamo ad inserire il seguente codice:

```
// carica le info sul livello
var settings = levels[lev][0];//[64,2,22,12], primo vettore
this.cellSize = settings[0];//64
this.spacing = settings[1];//2
this.areaW = settings[2] * this.cellSize;//22*64
this.areaH = settings[3] * this.cellSize;//12*64
```

In questo modo, otteniamo le informazioni sul livello che stiamo caricando.

Ricaviamo quindi le informazioni sui tiles e inseriamole nell'array this.tiles:

```
// dati sui tiles
 var tiles = levels[lev][1]; //prelevo il vettore delle tile
 var cs = this.cellSize + this.spacing; //dimensione totale con lo spazio
 //Math.ceil() ritorna il più piccolo intero più grande di o uguale a un dato numero
var cellsX = Math.ceil(this.imgTiles.width / cs); //numero di colonne della tileImage
 var cellsY = Math.ceil(this.imgTiles.height / cs); //numero di righe della tileImage

 // creo la tilelist
     for (var i = 0; i < tiles.length; i++) {
         // coordinate del tile relative alla texture
       // Math.floor() restituisce il numero intero, arrotondato per difetto
         var cy = Math.floor(tiles[i][0] / cellsX);//numero di riga su tileImage
         var cx = tiles[i][0] - cy * cellsX;//numero di colonna su tileImage
         // aggiunge il tile contenente le coordinate sulla mappa e sull'immagine
         this.tiles.push([tiles[i][1] * this.cellSize, tiles[i][2] * this.cellSize, cx * cs,
cy * cs]);
     }
```

Fatto ciò, istanziamo i blocchi di collisione.

```
// resetta l'array dei blocchi
 this.blocks = [];
 // dati sui blocchi di collisione
 var blocks = levels[lev][2];
 //creiamo le varie istanze di Block
 for (var i = 0; i < blocks.length; i++) {
     this.blocks.push(new Block(blocks[i][0] * this.cellSize, blocks[i][1] *
this.cellSize));
 }
```

Disegnare la tilemap

Nell'evento draw dell'oggetto Game, appena sopra il codice per disegnare il Player, aggiungiamo:

```
// salva le impostazioni context
 this.ctx.save();
 // trasla il context per mostrare gli oggetti in view
 this.ctx.translate(-this.viewX, -this.viewY);
 var cs = this.cellSize;
 // disegna le tiles se sono dentro la view
 var vx1 = this.viewX - this.cellSize;
 var vy1 = this.viewY - this.cellSize;
 var vx2 = this.viewX + this.canvas.width;
 var vy2 = this.viewY + this.canvas.height;

// renderizza tutte le tiles
for (var i = 0; i < this.tiles.length; i++) {
    if (this.tiles[i][0] > vx1 && this.tiles[i][1] > vy1 &&
```

```
          this.tiles[i][0] < vx2 && this.tiles[i][1] < vy2)
        this.ctx.drawImage(this.imgTiles,
                        this.tiles[i][2],
                        this.tiles[i][3],
                        cs, cs,
                        this.tiles[i][0],
                        this.tiles[i][1],
                        cs, cs);
}

// ripistina il context per eliminare la traslazione
 this.ctx.restore();
```

La view (effetto telecamera)

Prima di proseguire, facciamo in modo che la view (area visualizzata) segua il personaggio. Per rendere meno fastidioso lo spostamento, aggiungiamo qualche funzione di supporto per rendere più dolci i movimenti della telecamera. Creiamo un nuovo file utils.js e in esso aggiungiamo 2 funzioni all'oggetto Math di JavaScript

```
// limita il valore di x tra un minimo e un massimo scelto
Math.xMinMax = function (x, min, max) {
    return x < min ? min : (x > max ? max : x);
};
// interpolazione lineare serve a calcolare un valore intermedio x tra a e b x è un valore
compreso tra 0 e 1 se x=0 restituisce l'estremo inferiore se x=1 restituisce l'estremo
superiore altrimenti restituisce un valore intermedio
Math.lerp = function (a, b, x) {
    return (1 - x) * a + x * b;
};
```

Aggiungiamo anche altre funzioni che utilizzeremo in seguito per la riproduzione del suono e per la gestione della messa in pausa del gioco:

```
//questa funzione riproduce un suono indipendentemente dal tipo di browser
function AudioPlay(source){
       if(window.ActiveXObject != undefined){
              source.pause();
              source.currentTime = 0;
       }else{
              source.load();
       }
       source.play();
}
```

Aggiungiamo anche:

```
//Questa funzione ci servirà per il ridimensionamento della finestra
window.addEventListener('resize', function() {
  //
}, true);

//Questa funzione ascolta l'evento load ed avvia il gioco
window.addEventListener('load', function() {
  StartGame();
}, true);

Il codice precedente era già stato inserito alla fine del file main adesso possiamo anche
toglierlo.
//Questa funzione funzione scolta l'evento focus (il browser è in primo piano) servirà alla
gestione di uscita dallo stato in pausa
window.addEventListener('focus', function() {
    game.OnFocus();
});
//Questa funzione funzione scolta l'evento blur (il browser non è in primo piano) servirà
alla gestione di entrata nello stato in pausa
window.addEventListener('blur', function() {
    game.OnBlur();
});
```

aggiungiamo all'oggetto game subito dopo l'istanza del gestore delle risorse le due funzioni:

```
//istanza del gestore delle risorse
this.gr = new GestioneRisorse();

this.OnBlur = function () {
    if (this.level > 0) {
        this.paused = true;
    }
    this.sndMusic.pause();
}

this.OnFocus = function () {
    this.sndMusic.play();
}
//dimensione cella di gioco
```

……

Cambiamo le coordinate di inizio del player:

```
this.xStart = 720;
this.yStart =  300;
this.x = this.xStart;
this.y = this.yStart;
```

In fondo all'evento Update del Player aggiungiamo

```
// imposta il target da raggiungere, utilizzando la funzione che abbiamo precedentemente
//definito. In pratica vengono calcolate due coordinate che si troveranno al centro
//dell'area visibile
var targetX = Math.xMinMax(this.x - game.canvas.width / 2, 0, game.areaW - game.canvas.width);
var targetY = Math.xMinMax (this.y - game.canvas.height / 2, 0, game.areaH - game.canvas.height);

// si muove la telecamera verso il puntotarget, in modo smussato
game.viewX = Math.floor(Math.lerp(game.viewX, targetX, 0.2));
game.viewY = Math.floor(Math.lerp(game.viewY, targetY, 0.2));
```

importiamo in index.html il file utils.js e il file levels.js che contiene i vettori dei livelli costruiti con il tool tileEditor:

```
<script language="javascript" src="levels.js" ></script>
<script language="javascript" src="utils.js" ></script>
```

Proviamo a far ripartire il gioco e se tutto è andato bene dovremmo ottenere: