Création d'un petit jeu de plateforme
Initiation à la Programmation Orienté Objet

Principe du jeu : Un personnage dont le mouvement est commandé par le clavier doit se frayer un chemin dans un labyrinthe parsemé de policiers afin de récupérer un maximum de sacs d'argent. Le nombre de « vies » diminue lorsque le personnage rencontre un policier.

I. La réalisation du background
II. La création et l'animation du personnage
III. La création et l'animation des policiers
IV. Le fonctionnement du jeu : détection des collisions et comptage des points

Le jeu utilisera une page game.html, un fichier game.js et un fichier game.css.
L'image tileset.png se trouvera dans le même dossier.
Le fichier game.html ci-dessous indique que l'on utilise la superposition de deux canvas de même taille 1024×640, l'un pour le décor « background » et l'autre pour l'animation des personnages.

```
<!DOCTYPE html>
<html>
 <head>
 <link rel="stylesheet" href="game.css" />
 <title>Game</title>
 </head>
 <body>
 <canvas id="background" width="1024" height="640"></canvas>
 <canvas id="sprite" width="1024" height="640"></canvas>
 <script type="text/javascript" src="game.js"></script>
 </body>
</html>
```

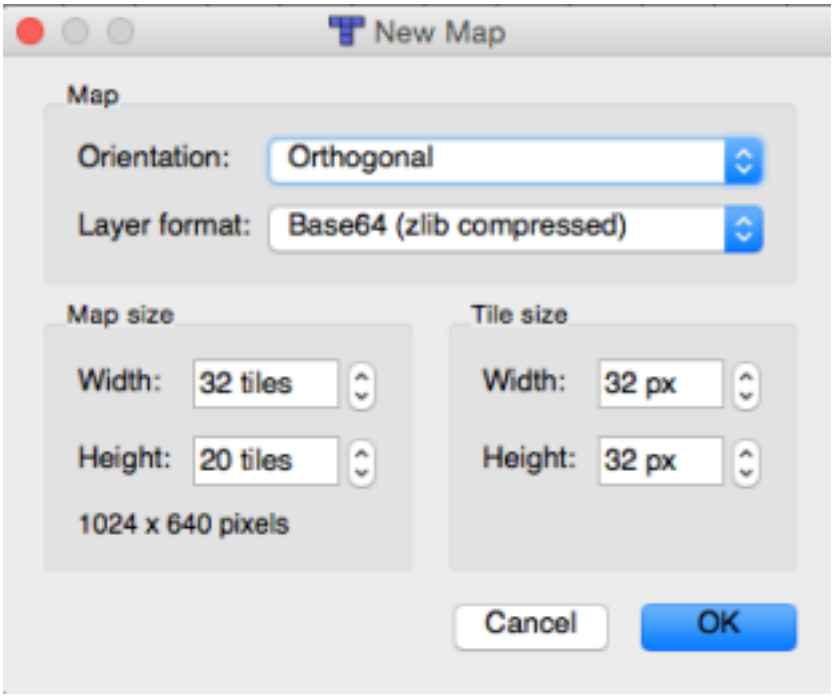
Le fichier game.css ci-dessous se limite à indiquer l'ordre de superposition des canvas, le sprite devant se mettre au-dessus du background.

```
canvas /* Propriétés communes aux deux canvas */
{
border:2px solid blue;
top:2%;
left:10%;
position: absolute;
}

/* Propriété spécifique au canvas sprite */
#sprite {
                z-index: 0;
                }

/* Propriété spécifique au canvas background */
#background {
                z-index: -1;
                }
```

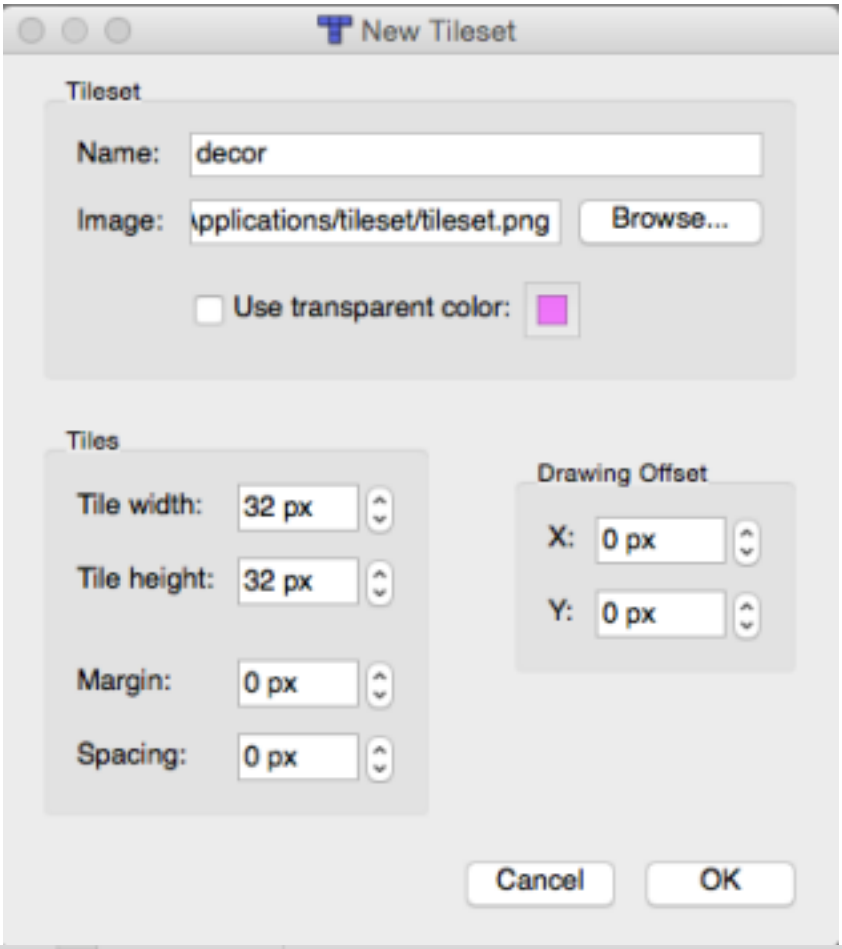
I. La réalisation du background

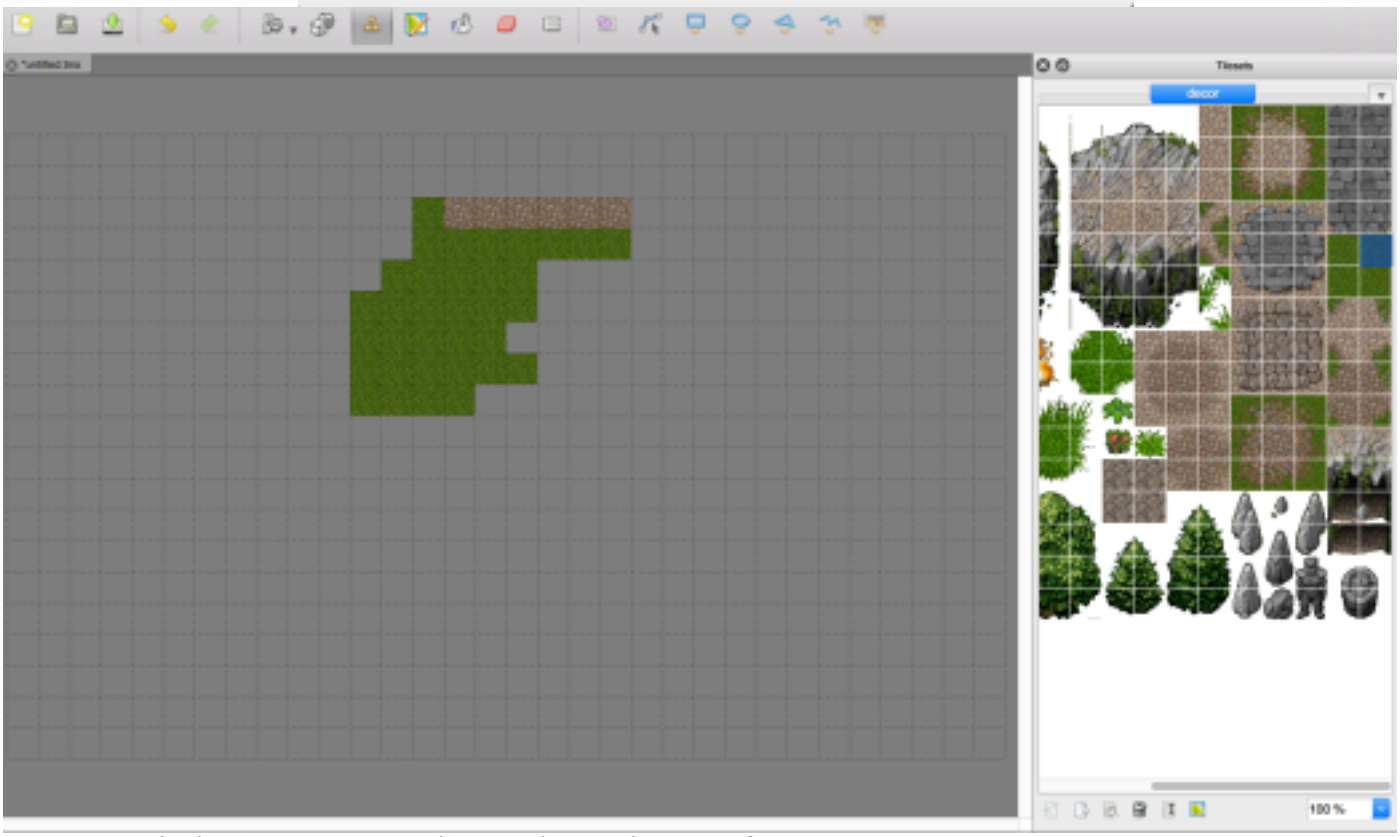
Pour réaliser le décor, nous utilisons le logiciel open-source appelé Tiled : http://www.mapeditor.org/ qui est un map editor que l'on trouve aussi en ligne : http://apps.elias.media/Online-Tile-Map-Editor

L'image http://isnangellier.alwaysdata.net/tileset.png de taille 512px · 512px est constituée de 256 dalles de taille 32px × 32 px chacune.
Notre décor sera constitué de dalles extraites de cette image pour former sur notre canvas un carrelage de 32 · 20 dalles correspondant bien à une largeur « width » de 20 × 32 = 640 px et une hauteur « height » de 32×32 = 1024 px comme indiqué sur la page html.



Sélectionner Map New Tileset et aller chercher l'image





On pourra ainsi créer son propre décor et l'enregistrer au format text : map.txt

map.txt

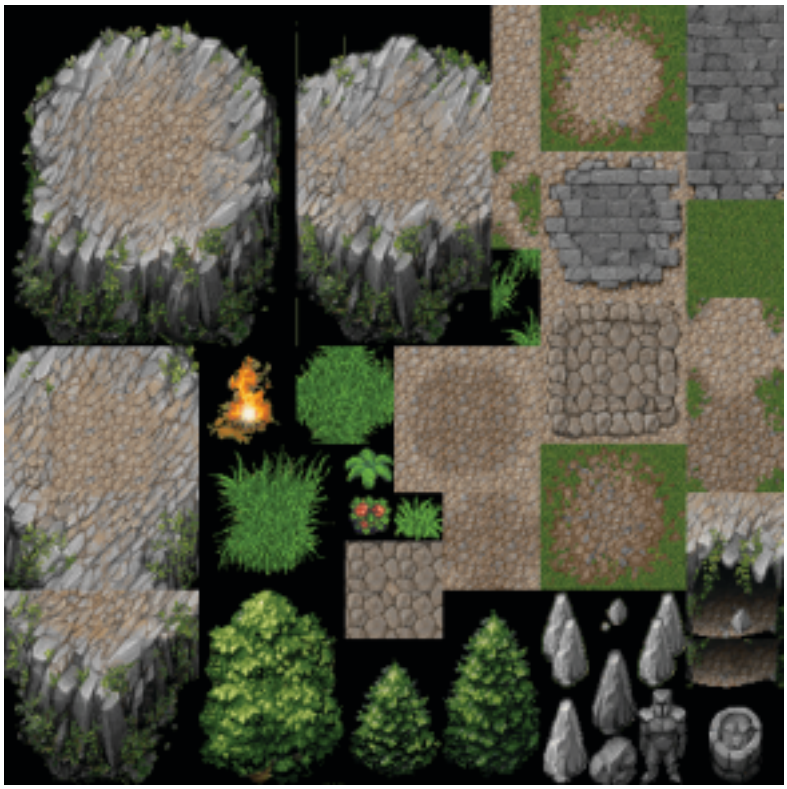
```
[header]
width=32
height=20
tilewidth=32
tileheight=32

[tilesets]
tileset=../../../../../Applications/tileset/tileset.png,32,32,0,0

[layer]
type=Tile Layer 1
data=
95,95,95,95,95,124,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,124,95,95,95,95,95,
124,95,95,95,124,124,124,95,124,124,124,124,124,124,95,124,124,124,124,124,124,124,124,124,124,95,124,95,95,124,124,95,
124,95,124,95,95,95,95,95,95,124,95,95,95,95,95,124,95,95,95,95,95,95,95,124,95,95,124,124,95,95,95,95,
124,95,124,124,95,124,95,95,95,124,95,124,124,124,124,124,95,124,124,124,124,124,124,124,124,95,95,95,124,124,124,124,95,
95,95,124,124,95,124,124,124,124,124,95,95,95,95,95,124,95,95,95,95,124,95,95,124,95,95,95,95,124,95,95,95,95,
124,124,124,95,95,95,95,95,95,124,124,124,124,124,95,95,95,95,95,124,95,124,95,95,95,95,124,124,124,95,95,124,95,
95,95,124,95,124,95,95,95,95,95,95,95,124,124,124,124,124,95,124,124,124,95,124,95,95,95,95,95,95,124,124,95,
95,95,95,95,124,95,95,124,124,95,95,95,95,95,95,95,124,95,95,95,95,95,124,95,95,124,95,95,124,95,95,95,
95,95,95,124,124,95,95,95,124,124,124,124,124,124,124,124,124,124,124,124,124,124,124,124,95,95,95,95,95,124,95,95,124,95,95,95,
95,95,95,95,124,124,124,95,124,95,95,95,95,95,95,95,05,95,95,95,95,124,95,124,124,124,95,124,124,124,124,124,124,124,95,
95,124,95,95,95,95,124,124,124,95,95,95,124,95,95,124,95,124,95,124,95,124,124,95,95,95,95,124,95,95,95,95,
124,124,95,124,95,95,124,95,124,95,124,124,124,95,124,124,95,124,124,124,95,124,124,95,95,124,124,124,95,95,95,95,95,
95,95,95,124,95,95,124,95,95,95,124,95,95,95,95,124,95,95,95,95,95,95,124,124,95,95,124,95,95,124,124,95,
95,124,95,95,95,95,124,95,124,124,124,95,124,124,124,124,124,124,124,124,124,124,95,95,95,95,95,95,124,95,95,124,95,95,
95,124,95,124,95,95,95,95,95,95,124,95,124,95,95,95,95,124,95,95,95,95,124,124,95,124,124,95,95,124,95,95,
95,124,95,124,95,124,124,124,95,95,124,95,95,95,95,95,95,124,124,124,124,124,124,124,95,95,95,124,95,95,124,95,95,
95,124,95,124,95,95,124,95,95,124,124,124,124,124,124,124,124,124,124,95,95,95,95,124,95,95,95,124,124,124,124,95,95,
124,124,95,124,124,95,124,95,95,95,95,95,124,95,95,95,95,124,95,124,95,95,124,95,95,95,124,95,95,95,95,95,
95,95,95,95,124,95,95,124,124,124,124,124,124,95,124,95,95,95,95,124,124,124,124,124,95,95,95,124,124,95,95,124,124,95,
95,95,124,124,124,95,95,95,95,95,95,95,95,95,124,124,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95
```

Récupérer les données data pour former le tableau background constitué de 20 tableaux, un tableau pour chaque ligne, chacun des tableaux étant constitué lui-même de 32 éléments, un élément par colonne. A l'aide de ce tableau, on va pouvoir constituer notre image de 20 lignes et 32 colonnes, chaque cellule représentant une dalle particulière du carrelage correspondant à l'image tileset.png représentée ci-dessous :

x

94 95 96 97 98 99 100 101 102 103 104 105 106 107

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16



y

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

108 109 110 111 112

113 114 115 116 117 118 119 120 121 122 123 124 125

126 127 128 129 130 131 132 133 134 135 136 137 138

139 140 141 142 143 144

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57

58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75

76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93

1451 46 147 148 149 150 151 152 153 154 155 156 157

158 159 160 161 162 163 164 165 166 167 168 169 170

171 172 173 174 175 176 177 178 179 180 181 182 183

184 185 186 187 188 189 190 191 192 193 194 195 196225 226 227 228 229 230 231 232 233 234 235 236 237
197 198 199 200 201 202 203 204 205 206 207 208 209238 239 240 241 242 242 243 245 246 247 248 249 250
210 211 212 213 214 215 216 217 218 219 220 221 222251 252 253 254 255 256
223 224

Remarque importante : Le bord supérieur gauche de la dalle numéroté 133 (le feu) sur l'image tileset.png est situé à l'abscisse x=((133 modulo 16)-1)*32=128 px et à l'ordonnée y=partie entière (133/16)*32=256 px.
On peut ainsi localiser une dalle sur l'image tileset.png grâce à son numéro.
La fonction modulo notée % donne le reste r de la division de a par b tel que : a=q·b+r
La fonction partie entière s'écrit Math.floor() ;
Nous pouvons écrire le début de game.js

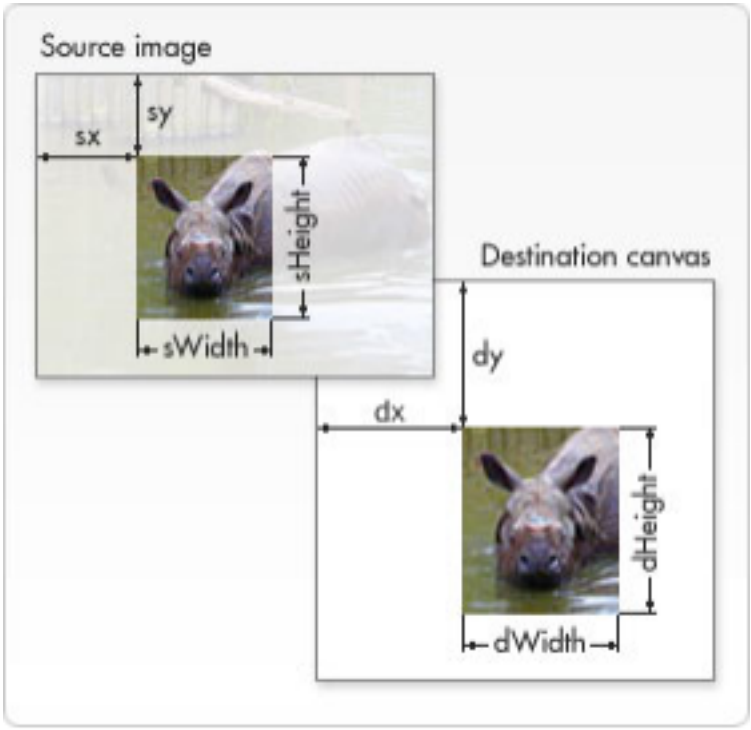
```
var background = [
 [95,95,95,95,95,124,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,95,
95,124,95,95,95,95,95],
 [124,95,95,95,124,124,124,95,124,124,124,124,124,124,95,124,124,124,124,124,124,
124,124,124,124,95,124,95,95,124,124,95],
 [124,95,124,95,95,95,95,95,95,124,95,95,95,95,95,124,95,95,95,95,95,95,95,124,95,
95,124,124,95,95,95,95],
 [124,95,124,124,95,124,95,95,95,124,95,124,124,124,124,124,95,124,124,124,124,124,
124,124,95,95,95,124,124,124,124,95],
 [95,95,124,124,95,124,124,124,124,124,95,95,95,95,95,124,95,95,95,95,124,95,95,124,
95,95,95,124,95,95,95,95],
 [124,124,124,95,95,95,95,95,95,124,124,124,124,95,95,95,95,95,124,95,124,95,95,95,95,
124,124,124,95,95,124,95],
 [95,95,124,95,124,95,95,95,95,95,95,95,124,124,124,124,124,95,124,124,124,95,124,95,
95,95,95,95,95,124,124,95],
 [95,95,95,95,124,95,95,124,124,95,95,95,95,95,95,95,124,95,95,95,95,95,124,95,95,124,
95,95,124,95,95,95],
 [95,95,95,124,124,95,95,95,124,124,124,124,124,124,124,124,124,124,124,124,124,95,95,95,
95,95,124,95,95,124,95,95,95],
 [95,95,95,95,124,124,124,95,124,95,95,95,95,95,95,95,95,95,95,124,95,124,124,124,95,
124,124,124,124,124,124,95],
 [95,124,95,95,95,95,124,124,124,95,95,95,124,95,95,124,95,124,95,124,95,124,124,95,
95,95,95,124,95,95,95,95],
 [124,124,95,124,95,95,124,95,124,95,124,124,124,95,124,124,95,124,124,124,95,124,
124,95,95,124,124,124,95,95,95,95],
 [95,95,95,124,95,95,124,95,95,95,124,95,95,95,95,124,95,95,95,95,95,95,124,124,95,
95,124,95,95,124,124,95],
 [95,124,95,95,95,95,124,95,124,124,124,95,124,124,124,124,124,124,124,124,95,95,95,95,
95,95,124,95,95,124,95,95],
 [95,124,95,124,95,95,95,95,95,95,124,95,124,95,95,95,95,124,95,95,95,95,124,124,95,
124,124,95,95,124,95,95],
 [95,124,95,124,95,124,124,124,95,95,124,95,95,95,95,95,95,124,124,124,124,124,124,
95,95,95,124,95,95,124,95,95],
 [124,124,95,124,124,95,124,95,95,95,95,95,124,95,95,95,95,124,95,124,95,95,124,95,
95,95,124,95,95,95,95,95],
 [95,95,95,95,124,95,95,124,124,124,124,124,124,95,124,95,95,95,95,124,124,124,124
,95,95,124,124,95,95,124,124,95],
 [95,95,124,124,124,95,95,95,95,95,95,95,95,95,124,124,95,95,95,95,95,95,95,95,95,
95,95,95,95,95,95] ];
var tilesetImage = new Image();
tilesetImage.src = 'tileset.png';
tilesetImage.onload = drawBackground;/*Au chargement de l'image, représente le
background */
var bgcanvas = document.getElementById('background');
var bgctx = bgcanvas.getContext('2d');
var spcanvas = document.getElementById('sprite');
```

```
var spctx = spcanvas.getContext('2d');
var tileSize = 32;//La taille d'une dalle(32*32)
var rowTileCount = 20;//Le nombre de dalles par ligne dans le canvas var colTileCount =
32;// Le nombre de dalles par colonne dans le canvas var imageNumTiles = 16; /* Le
nombre de dalles par ligne ou par colonne dans l'image tileset.png*/
function drawBackground () {
}
drawBackground();
```

Pour placer les dalles sur notre background, nous allons utiliser la méthode drawImage qui possède 8 paramètres :

drawImage(image,sx,sy,sWidth,sHeight,dx,dy,dWidth,dHeight)

Les 4 premiers paramètres après le nom de l'image font référence au positionnement sx, sy de la découpe de largeur sWidth et de hauteur sHeight dans l'image source, ici tileset.png, les 4 suivant font référence à son positionnent dx, dy dans le canvas par rapport au bord supérieur gauche de celui-ci, avec un éventuel changement d'échelle si on fait passer la largeur de la découpe à dWidth et sa hauteur à dHeight, ce que l'on ne fera pas ici.



Vérifier que pour placer, au milieu du canvas, la dalle 139 correspondant à l'image du feu, il faut écrire : `bgctx.drawImage(tilesetImage,128,256,32,32,bgcanvas.width/2,bgcanvas.height/2,32,32);`

→ En déduire la fonction complète drawBackground() constituée de deux boucles imbriquées, permettant de représenter le background à partir du tableau background.

```
for (var i=1;i<=rowTileCount;i++) {
```

Pour récupérer la valeur de la dalle de tileset qui sera représentée à la ligne i et à la colonne j sur le canvas, on utilisera :

```
var value=background [i-1] [j-1];
```

On définira les variables sx, sy, dx et dy pour ne pas se tromper et écrire :

```
bgctx.drawImage(tilesetImage,sx,sy,32,32,dx,dy,32,32);
```

II. Création et animation du personnage

Télécharger l'image http://isnangellier.alwaysdata.net/personnagesTiles.png
Il s'agit d'une image de 384 px · 256 px constituée de 12 · 8 dalles de 32 px.
Notre personnage avec la casquette occupe les 3 dernières colonnes avec 6 représentations Down, 6 représentations Up, 6 représentations Right, 6 représentations Left.
La multiplicité de ces représentations va nous permettre d'augmenter le réalisme du personnage lors de son déplacement.
Les orientations de déplacement du personnage seront désignées par la propriété state. Vous trouvez ci-dessous les coordonnées des positions de ces images sur l'image personnagesTiles.

D_1 [9·32=288,0]	L_1 [288,32]	R_1 [288,64]	U_1 [288,96]
D_2 [10·32=320,0]	L_2 [320,32]	R_2 [320,64]	U_2 [320,96]
D_3 [11·32=352,0]	L_3 [352,32]	R_3 [352,64]	U_3 [352,96]
D_4 [9·32=288,4·32=128]	L_4 [288,160]	R_4 [288,192]	U_4 [288,214]
D_5 [10·32=320, 4·32=128]	L_5 [320, 160]	R_5 [320, 192]	U_5 [320, 214]
D_6 [11·32=352, 4·32=128]	L_6 [352, 160]	R_6 [352, 192]	U_6 [352, 214]

→ L'intérêt à nouveau d'un seul fichier image est d'économiser la mémoire et le temps de chargement.

Il est possible de localiser les images de ces 6 représentations en fonction de state à travers un tableau qui prendra la forme :

[[288, state*32], [320, state*32],[352,state*32],[288,128+state*32],[320,128+ state*32],[352, state*32]

] Avec state=0 pour Down, state=1 pour Left, state=2 pour Right et state=3 pour Up

On bouclera sur les 6 images de ce tableau pour simuler le mouvement à l'aide d'une variable appelée numero_image_boy qui augmentera d'une unité à chaque frame.

```
//le dessin du personnage
var personnagesTiles = new Image();
personnagesTiles.src = 'personnagesTiles.png';
var numero_image_boy=0;
var personnage=[[288,0],[320,0],[352,0],[288,128],[320,128],[352,128]];
```

L'interaction avec le clavier se fait de la manière suivante :

```
//interaction clavier
var keysDown = {};
window.addEventListener('keydown', function(e) {
 keysDown[e.keyCode] = true;
});
window.addEventListener('keyup', function(e) {
 delete keysDown[e.keyCode];
});
```

Il faut maintenant tenir compte du background ; le personnage ne peut aller que sur les dalles vertes numérotées 95 du background et ne peut pas sortir du canvas. On crée à cette effet une fonction decoration() qui consiste à repérer la position du personnage sur le canvas puis le numéro de la dalle sur laquelle il se trouve afin d'autoriser ou non le déplacement de celui-ci.

```
function decoration() {
if (mySprite.state==3) {//up
mySprite_row=Math.floor((mySprite.y)/tileSize);
mySprite_col=Math.floor((mySprite.x+mySprite.width/2)/tileSize);
}
if (mySprite.state==2) {//Right
mySprite_row=Math.floor((mySprite.y+mySprite.height/2)/tileSize);
mySprite_col=Math.floor((mySprite.x+mySprite.width)/tileSize);
}
if (mySprite.state==0) {//Down
mySprite_row=Math.floor((mySprite.y+mySprite.height)/tileSize);
mySprite_col=Math.floor((mySprite.x+mySprite.width/2)/tileSize);
}
if (mySprite.state==1) {//Left
mySprite_row=Math.floor((mySprite.y+mySprite.width/2)/tileSize);
mySprite_col=Math.floor((mySprite.x)/tileSize);
}
decor=background [mySprite_row] [mySprite_col];

if (decor!=95 || mySprite.x<4 || mySprite.x>spcanvas.width || mySprite.y<4 || mySprite.y>(spcanvas.height-
4- mySprite.height))
{
 return true;
}
}
```

Placer le fichier bag http://isnangellier.alwaysdata.net/bag.png dans le dossier.

```
var argent = new Image();
argent.src='bag.png';
```

La fonction **update()** pour tenir compte de l'interaction avec le clavier sert à la mise à jour des mouvements et gère l'aspect du personnage.

```
function update() {

if (37 in keysDown) {
mySprite.state = 1; //left
mySprite.x -= 4;
numero_image_boy++;
if (decoration()) {
mySprite.x += 4;
}
}

if (38 in keysDown) {
mySprite.state = 3; //up
mySprite.y -= 4;
numero_image_boy++;
if (decoration()) {
mySprite.y += 4;
}
}

if (39 in keysDown) {
mySprite.state = 2; //right
mySprite.x += 4;
numero_image_boy++;
if (decoration()) {
mySprite.x -= 4;
}
}

if (40 in keysDown) {
mySprite.state = 0; //down
mySprite.y += 4;
numero_image_boy++;
if (decoration()) {
mySprite.y -= 4;
}
}

}
```

Nous allons maintenant créer notre objet mySprite avec 5 **propriétés** et la **méthode** draw.

```
function mySprite(x,y,state) {
this.width= 32;
this.height= 32;
this.x=x;
this.y=y;
this.state=state;
this.draw=function() {
 spctx.drawImage(
 personnagesTiles,
 personnage[numero_image_boy][0],
 personnage[numero_image_boy][1] + this.state*32,
```

```
 this.width,
 this.height,
 this.x,
 this.y,
 this.width,
 this.height
 );
 };
 }
var mySprite=new mySprite(70,10,0);
```

On peut maintenant créer la fonction **render()** qui se charge du rendu à chaque frame.

```
function render() {
spctx.clearRect(0, 0, spcanvas.width, spcanvas.height);
if (numero_image_boy>=5) {numero_image_boy=0};//pour boucler sur les 6
images mySprite.draw();
 }
```

Et celle qui permet de réaliser l'animation grâce à setInterval qui appelle la fonction run() toutes les 10 ms.

```
function run() {
        update();
        render();
}
var myTimer=setInterval(run, 10);
```

On écrira à la dernière ligne du programme :

```
window.onload = init();
```

Avec la fonction init() qui force le positionnement aléatoire du sac d'argent sur une des cases 95 et à dessiner le backgound.

```
function init() {
item = {
 x: Math.random() * spcanvas.width,
 y: Math.random() * spcanvas.height,
 width: 10,
 height: 10
};
item_row=Math.floor((item.y+item.width/2)/tileSize);
item_col=Math.floor((item.x+item.height/2)/tileSize);
argent.onload = spctx.drawImage(argent, item.x, item.y);
decor=background [item_row] [item_col];
if (decor != 95) {
 init();
}
drawBackground();

mySprite.x=70;
mySprite.y=10;

run();

}
```

III. Création et animation des policiers

Nous allons maintenant créer les objets policiers. On leur donne 5 propriétés et 2 méthodes : draw et move. Chaque policier possède un chemin à suivre ; ainsi le policier 1 suivra paths[0] et son repère dans ce chemin est positioninpaths[0] qui augmente d'une unité à chaque frame.

```
var policiers=[[192,0],[224,0],[256,0],[192,128],[224,128],[256,128]];
var positioninpaths=[0,0,0];
var paths=[
[[672,254],[672,164],[770,164],[770,64],[798,64],[798,2],[224,2],[224,64],[132,64],
[132,156],[94,156],[94,218],
[60,218],[60,324],[164,324],[164,448],[222,448],[222,380],[288,380],
[288,294],[516,294],[516,378],[648,378],
[648,254],[672,254]],[[672,254],[638,254],[638,382],[520,382],[520,294],
[288,294],[288,374],[226,374],
[226,444],[168,444],[168,324],[66,324],[66,226],[100,226],[100,154],
[134,154],[134,66],[226,66],[226,2],[452,2],
[452,68],[326,68],[326,122],[422,122],[422,152],[540,152],[540,222],
[642,222],[642,254],[672,254]],
[[130,444],[222,444],[222,384],[286,384],[286,288],[414,288],[414,380],
[354,380],[354,476],[510,476],
[354,476],[354,384],[410,384],[410,288],[514,288],[514,380],[642,380],
[642,412],[770,412],[770,608],[882,608],
[882,544],[990,544],[990,6],[898,6],[898,58],[990,58],[990,130],[898,130],
[898,190],[766,190],[766,258],
[642,258],[642,226],[546,226],[546,158],[418,158],[418,130],[322,130],
[322,66],[446,66],[446,6],[222,6],
[222,66],[130,66],[130,158],[98,158],[98,222],[58,222],[58,418],[158,418],[158,444]],  ];
var numero_image_policier=0;

function ennemi(x,y,i) {
this.width= 32;
this.height= 32;
this.state=0;
this.x=x;
this.y=y;

this.draw=function() {
spctx.drawImage(personnagesTiles,policiers[numero_image_policier][0],
policiers[numero_image_policier][1] + this.state*32,
this.width,this.height,this.x,this.y,this.width,this.height);
};

this.move=function(){
//on cherche à connaître la direction du mouvement le long du chemin
startx=paths[i-1][positioninpaths[i-1]][0];
starty=paths[i-1][positioninpaths[i-1]][1];
endx=paths[i-1][positioninpaths[i-1]+1][0];
endy=paths[i-1][positioninpaths[i-1]+1][1];

if (startx<endx) {this.x +=1;this.state=2;numero_image_policier++}
if (startx>endx) {this.x -=1;this.state=1;numero_image_policier++}
if (starty<endy) {this.y +=1;this.state=0;numero_image_policier++}
if (starty>endy) {this.y -=1;this.state=3;numero_image_policier++}
if (numero_image_policier==5) {numero_image_policier=0};

if (this.x==endx && this.y==endy) {
```

```
positioninpaths[i-1] +=1;
if (positioninpaths[i-1] == paths[i-1].length-1) {positioninpaths[i-1]=0;};
}
}
}
var ennemi1 = new ennemi(paths[0][positioninpaths[0]][0],paths[0][positioninpaths[0]][1],1);
var ennemi2 = new ennemi(paths[1][positioninpaths[1]][0],paths[1][positioninpaths[1]][1],2);
var ennemi3 = new ennemi(paths[2][positioninpaths[2]][0],paths[2][positioninpaths[2]][1],3);
var ennemis=[ennemi1,ennemi2,ennemi3];
```

On ajoute le déplacement des policiers dans le update()

```
for (var i=0;i<=2;i++) {
ennemis[i].move();
}
```

Ainsi que la représentation des policiers dans le render() après l'effacement du canvas

```
for (var i=0;i<=2;i++) {
ennemis[i].draw();
}
```

IV. Le fonctionnement du jeu : détection des collisions

On ajoute, après avoir défini les variables :

```
var itemCounter = 0;
var liveCounter = 3;
```

Dans le render() l'affichage sur le canvas du gain et du nombre de vies restant.

```
spctx.font = '12pt Arial';
spctx.fillStyle = '#fff';
spctx.textBaseline = 'top';
spctx.fillText(itemCounter, 10, 10);
spctx.fillText(liveCounter, 990, 10);
```

On crée la fonction detectCollision() à laquelle on fera appel dans le update() ; cette fonction peut aussi être ajoutée comme méthode à mySprite si l'on veut.

```
function detectCollision() {

for (var i = 0; i <=2; i++) {

if (
 mySprite.x < ennemis[i].x + ennemis[i].width &&
 mySprite.x + mySprite.width > ennemis[i].x &&
 mySprite.y < ennemis[i].y + ennemis[i].height &&
 mySprite.y + mySprite.height > ennemis[i].y
)
{
 liveCounter -=1;
 init();
}
}
}
```

On ajoute dans le update() le cas où il n'y a plus de vies, affichage de « Game over » et on arrête l'animation avec clearInterval.

```
if (liveCounter==0) {
bgctx.font = '40pt Arial';
bgctx.fillStyle = '#ff0000';
bgctx.textBaseline = 'top';
bgctx.fillText("Game Over", 350, bgcanvas.height/2);
clearInterval(myTimer);
}
```

Ajouter dans le update() la prise du sac

```
if (
mySprite.x < item.x + item.width &&
mySprite.x + mySprite.width > item.x &&
mySprite.y < item.y + item.height &&
mySprite.y + mySprite.height > item.y
) {
spctx.clearRect(item.x, item.y, item.width, item.height);
init();
itemCounter ++;
}
```

Et dans le render() le dessin du sac :

```
spctx.drawImage(argent, item.x, item.y);
```

Prolongements pour le jeu :

- Ajouter un policier (une nouvelle instance de l'objet ennemi) et lui assigner un nouveau chemin de ronde ; on peut déplacer le personnage mySprite et utiliser console.log(mySprite.x) et console.log(mySprite.y) pour lire ses positions dans la console ; on crée ainsi un nouveau chemin pour le policier. - Ajouter une « super power ». Si le personnage touche une « super power », il a alors la propriété d'invulnérabilité et il peut passer à travers les policiers sans perdre de vie pendant une certaine durée. Il porte alors un casque blanc puis il porte un casque gris pour prévenir le joueur de la fin prochaine de sa période d'invulnérabilité. On utilisera pour cela les images dans personnageTiles correspondant au personnage casqué. On construira une nouvelle propriété pour mySprite relative à cet état d'invulnérabilité et on établira deux mesures de dates donnés par new Date().getTime() pour gérer les durées.
http://isnangellier.alwaysdata.net/power.png