

three.js

dat.GUI

SAT.js

Université Paris 8

# Moteurs de Jeu<sup>1</sup>

Nicolas JOUANDEAU  
n@up8.edu

septembre 2022

---











1. A la découverte des fonctions proposées par les moteurs et les jeux possibles.

## 2 Sprites

### 2.1 Sprites

---

Dans le domaine du jeu vidéo, un sprite est une image 2D avec un fond transparent, représentant un personnage ou un objet pouvant être déplacé indépendamment du décor ; pour animer un sprite et créer l'illusion d'un mouvement correspondant au déplacement, il est courant d'utiliser une série d'images et de les utiliser itérativement selon des séquences prédéfinies ; Tab. 1 présente des images de l'héroïne Misa du jeu Tuxemon ; l'image `left` présente Misa quand elle regarde vers la gauche ; les images `left_walk` 000 à 003 décompose la marche vers la gauche ; l'image `right` présente Misa quand elle regarde vers la droite ; les images `right_walk` 000 à 003 décompose la marche vers la droite.

 left			
 left_walk.000	 left_walk.001	 left_walk.002	 left_walk.003
 right			
 right_walk.000	 right_walk.001	 right_walk.002	 right_walk.003

TAB. 1 – Images de l'héroïne Misa du jeu Tuxemon.

L'utilisation d'un ensemble d'images peut présenter des redondances ; ici `left` est identique à `left_walk.001` et à `left_walk.003` ; ce qui fait 3 redondances pour `left` et 3 redondances pour `right`.

Pour l'animation d'un personnage joueur, en dehors des questions de position dans la scène, il s'agit de :

- écouter les événements générés par les touches clavier
- mettre à jour l'affichage

Le programme de la page suivante présente l'animation de Misa à la pression de la touche flèche-droite dont le résultat est présenté en Fig. 6 ; on dessine en haut à gauche de la page, les images utilisées pour l'animation ; le personnage animé est dessiné au centre de la fenêtre.

```

1 let cnv = document.getElementById('myCanvas');
2 let ctx = cnv.getContext('2d');
3 ctx.imageSmoothingEnabled= false;
4 let img_names = ["misa_right_walk.000","misa_right_walk.001",
5                 "misa_right_walk.002","misa_right_walk.003"];
6 let all_img = [];
7 for(let i = 0; i < 4; i += 1) {
8     let img = new Image();
9     img.src = "./assets/images/"+img_names[i]+".png";
10    img.onload = function() {
11        ctx.drawImage(img, 16*4*i, 0, 16*4, 32*4);
12    };
13    all_img.push(img);
14 }
15 let img_right_id = 3;
16 let go_right = 1;
17 window.addEventListener('keydown', keydown_fun, false);
18 function keydown_fun(e) {
19     switch(e.code) {
20         case "ArrowRight":
21             go_right += 1;
22             break;
23     }
24 }
25 function update() {
26     if(go_right > 0) {
27         img_right_id += 1;
28         if(img_right_id == 4) { img_right_id = 0; }
29         ctx.beginPath();
30         ctx.fillStyle = "#FFFFFF";
31         ctx.fillRect(cnv.width/2-16*2, cnv.height/2-32*2, 16*4,32*4);
32         ctx.closePath();
33         ctx.drawImage(all_img[img_right_id],
34                     cnv.width/2-16*2, cnv.height/2-32*2, 16*4, 32*4);
35         go_right = 0;
36     }
37 }
38 setInterval(update, 200);

```

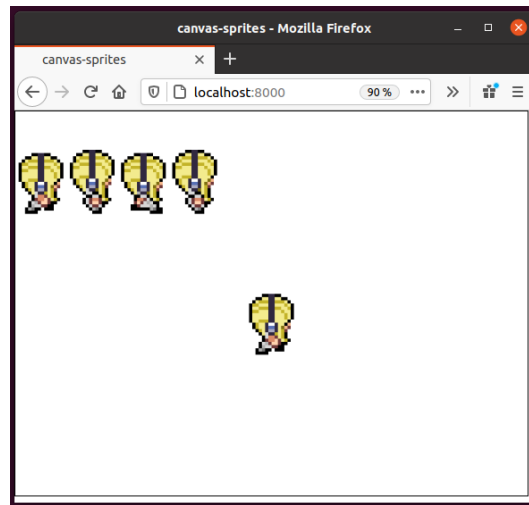


FIG. 5 – Animation de Misa avec un ensemble d'images.

## 2.2 Spritesheet

Un Spritesheet regroupe un ensemble d'images pour créer une simple animation, pour un personnage non joueur ou un élément de décor ; pour une animation de ce type, en dehors des questions de position dans la scène, il s'agit de :

- mettre à jour des variables d'animation
- mettre à jour l'affichage

Le programme de la page suivante présente l'animation de la momie de Metal Slug dont le résultat est présenté en Fig. 7 ; le spritesheet est composé de 18 images de 37 par 45, réparties sur 4 lignes et 5 colonnes.



FIG. 6 – Spritesheet de la momie de Metal Slug.

```

1 let cnv = document.getElementById('myCanvas');
2 let ctx = cnv.getContext('2d');
3 ctx.imageSmoothingEnabled= false;
4 let all_img = [];
5 let img = new Image();
6 let anim_id = -1;
7 img.src = "./assets/spritesheet/metalslug_mummy37x45.png";
8 img.onload = function() {
9     let canvas1 = document.createElement('canvas');
10    canvas1.width = 37*5;
11    canvas1.height = 45*4;
12    let context1 = canvas1.getContext('2d');
13    context1.drawImage(img, 0,0,37*5,45*4);
14    for(let j = 0; j < 4; j += 1) {
15        let imax = 5;
16        if(j == 3) { imax = 3; }
17        for(let i = 0; i < imax; i += 1) {
18            let canvasImageData1 = context1.getImageData(i*37, j*45, 37, 45);
19            let canvas2 = document.createElement('canvas');
20            canvas2.width = 37;
21            canvas2.height = 45;
22            let context2 = canvas2.getContext('2d');
23            context2.putImageData(canvasImageData1, 0,0);
24            all_img.push(canvas2);
25        }
26    }
27    anim_id = 0;
28 };
29 function update() {
30     let zoom = 4;
31     ctx.beginPath();
32     ctx.fillStyle = "#FFFFFF";
33     ctx.fillRect(cnv.width/2-37*zoom/2, cnv.height/2-45*zoom/2,
34                 37*zoom,45*zoom);
35     ctx.closePath();
36     if(anim_id >= 0) {
37         ctx.drawImage(all_img[anim_id], cnv.width/2-37*zoom/2,
38                     cnv.height/2-45*zoom/2, 37*zoom, 45*zoom);
39         anim_id += 1;
40         if(anim_id == all_img.length-1) { anim_id = 0; }
41     }
42 }
43 setInterval(update, 100);

```

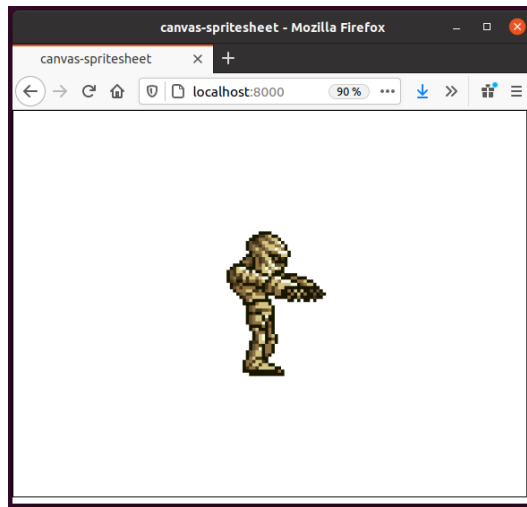


FIG. 7 – Animation de la momie de Metal Slug.

### 2.3 Texture atlas

Un Texture Atlas peut regrouper les images d'un ou plusieurs spritesheet ; l'organisation et les caractéristiques des sprites sont définis dans un fichier PNG et un fichier JSON ; selon l'organisation définie, il est possible de réduire l'espace mémoire nécessaire (en effectuant des rotations pour certaines images ou en utilisant différents type d'encodage des images<sup>4</sup>) ; l'augmentation du nombre de textures dans les jeux vidéos renforce l'importance des atlas.

L'utilitaire TexturePacker permet de réaliser des atlas à partir de série d'images pour différents moteurs de jeu ; la page suivante présenté un aperçu du fichier JSON associé à Fig. 8.



FIG. 8 – Fichier PNG d'un atlas pour les déplacements de Misa.

4. On pourra par exemple pour un résultat visuel équivalent, convertir une image de 16ko composée de sprites en RGBA avec 32bits par pixel en une image de 2ko composée de sprites compressés PVRTC avec 4bit par pixel.

```

1 {"frames": {
2   "misa-back-walk.000":
3   {
4     "frame": {"x":97,"y":1,"w":32,"h":41},
5     "rotated": false,
6     "trimmed": true,
7     "spriteSourceSize": {"x":0,"y":23,"w":32,"h":41},
8     "sourceSize": {"w":32,"h":64}
9   },
10  "misa-back-walk.001":
11  {
12    "frame": {"x":97,"y":44,"w":32,"h":41},
13    "rotated": false,
14    "trimmed": true,
15    "spriteSourceSize": {"x":0,"y":23,"w":32,"h":41},
16    "sourceSize": {"w":32,"h":64}
17  },
18  "misa-back-walk.002":
19  {
20    "frame": {"x":131,"y":1,"w":32,"h":41},
21    "rotated": false,
22    "trimmed": true,
23    "spriteSourceSize": {"x":0,"y":23,"w":32,"h":41},
24    "sourceSize": {"w":32,"h":64}
25  },
26  ...
27
28  "misa-right":
29  {
30    "frame": {"x":163,"y":44,"w":30,"h":41},
31    "rotated": false,
32    "trimmed": true,
33    "spriteSourceSize": {"x":1,"y":23,"w":30,"h":41},
34    "sourceSize": {"w":32,"h":64}
35  }},
36  "meta": {
37    "app": "http://www.codeandweb.com/texturepacker",
38    "version": "1.0",
39    "image": "misa.png",
40    "format": "RGBA8888",
41    "size": {"w":196,"h":86},
42    "scale": "1"
43  }
44 }
45 }

```

Le programme suivant présente le chargement du fichier JSON et du fichier PNG nommé dans le fichier JSON, dont le résultat de l'exécution est présenté en Fig 9.

```
1 let cnv = document.getElementById('myCanvas');
2 let ctx = cnv.getContext('2d');
3 ctx.imageSmoothingEnabled= false;
4 let xobj = new XMLHttpRequest();
5 xobj.onload = onload_atlas;
6 xobj.overrideMimeType("application/json");
7 xobj.open("GET", "./assets/atlas/misa.json", true);
8 xobj.send();
9
10 function onload_atlas () {
11   if(this.status == 200) {
12     let json_infos = JSON.parse(this.responseText);
13     let spritesheet = new Image();
14     spritesheet.src = "./assets/atlas/"+json_infos["meta"]["image"];
15     spritesheet.onload = function() {
16       ctx.drawImage(spritesheet,
17                     cnv.width/2-spritesheet.width,
18                     cnv.height/2-spritesheet.height,
19                     spritesheet.width*2, spritesheet.height*2);
20     }
21   }
22 }
```

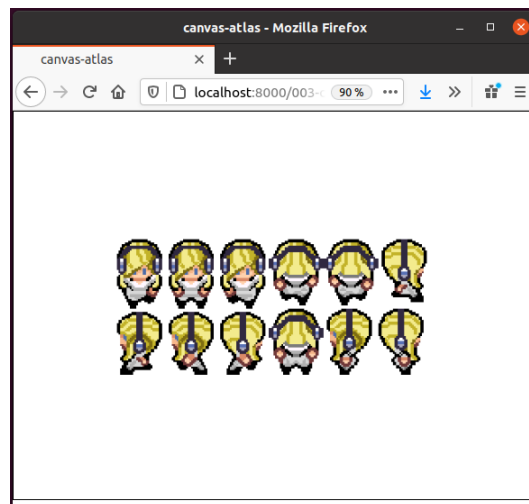


FIG. 9 – Fichier PNG de l'atlas des sprites pour les déplacements de Misa.



Le programme suivant présente l'animation de Misa (comme déjà étudiée en section 2.1) et dont le résultat est présenté en Fig. 10.

```

1  let cnv = document.getElementById('myCanvas');
2  let ctx = cnv.getContext('2d');
3  ctx.imageSmoothingEnabled= false;
4  let all_sprites_event = [];
5  let xobj = new XMLHttpRequest();
6  xobj.onload = onload_atlas;
7  xobj.overrideMimeType("application/json");
8  xobj.open("GET", "../assets/atlas/misa.json", true);
9  xobj.send();
10
11 class SpriteAtlas {
12   constructor(ctx, json) {
13     this.ctx = ctx;
14     this.json = json;
15     this.event_code = 0;
16     this.animestep = 1;
17     this.to_draw = 1;
18     this.animeseq = [];
19   }
20   next_step() {
21     if(this.to_draw == 0) {
22       this.animestep += 1;
23       if(this.animestep >= this.animeseq.length) { this.animestep = 0; }
24       this.to_draw = 1;
25     }
26   }
27   add_anime(prefix, first_id, last_id, event_code) {
28     this.event_code = event_code;
29     for(let i = first_id; i < last_id+1; i += 1) {
30       let filename = prefix;
31       if(i < 10) { filename += ".00"+i; }
32       else if(i < 100) { filename += ".0"+i; }
33       else { filename += "."+i; }
34       let x = this.json["frames"][filename]["frame"]["x"];
35       let y = this.json["frames"][filename]["frame"]["y"];
36       let w = this.json["frames"][filename]["frame"]["w"];
37       let h = this.json["frames"][filename]["frame"]["h"];
38       let canvasImageData1 = this.ctx.getImageData(x, y, w, h);
39       let canvas2 = document.createElement('canvas');
40       canvas2.width = w;
41       canvas2.height = h;
42       let context2 = canvas2.getContext('2d');
43       context2.putImageData(canvasImageData1, 0,0);
44       this.animeseq.push(canvas2);
45     }
46   }
47 }

```

```

1 function onload_atlas () {
2   if(this.status == 200) {
3     let json_infos = JSON.parse(this.responseText);
4     let spritesheet = new Image();
5     spritesheet.src = "./assets/atlas/"+json_infos["meta"]["image"];
6     spritesheet.onload = function() {
7       let canvas1 = document.createElement('canvas');
8       canvas1.width = json_infos["meta"]["size"]["w"];
9       canvas1.height = json_infos["meta"]["size"]["h"];
10      let context1 = canvas1.getContext('2d');
11      context1.drawImage(spritesheet, 0,0,canvas1.width,canvas1.height);
12      let new_sp = new SpriteAtlas(context1, json_infos);
13      new_sp.add_anime("misa-right-walk",0,3,"ArrowRight");
14      all_sprites_event.push(new_sp);
15    }
16  }
17 }
18 window.addEventListener('keydown', keydown_fun, false);
19 function keydown_fun(e) {
20   switch(e.code) {
21     case "ArrowRight":
22       for(let i = 0; i < all_sprites_event.length; i += 1) {
23         if(all_sprites_event[i].event_code == "ArrowRight") {
24           all_sprites_event[i].next_step();
25         }
26       }
27       break;
28     }
29   }
30   function update() {
31     let zoom = 4;
32     for(let i = 0; i < all_sprites_event.length; i += 1) {
33       if(all_sprites_event[i].to_draw == 1) {
34         let step_i = all_sprites_event[i].animestep;
35         let cnv_i = all_sprites_event[i].animeseq[step_i];
36         ctx.beginPath();
37         ctx.fillStyle = "#FFFFFF";
38         ctx.fillRect(cnv_i.width/2-cnv_i.width*zoom/2,
39                     cnv_i.height/2-cnv_i.height*zoom/2,
40                     (cnv_i.width+1)*zoom, (cnv_i.height+1)*zoom);
41         ctx.closePath();
42         ctx.drawImage(cnv_i,cnv_i.width/2-cnv_i.width*zoom/2,
43                     cnv_i.height/2-cnv_i.height*zoom/2,
44                     cnv_i.width*zoom, cnv_i.height*zoom);
45         all_sprites_event[i].to_draw = 0;
46       }
47     }
48   }
49   setInterval(update, 400);

```

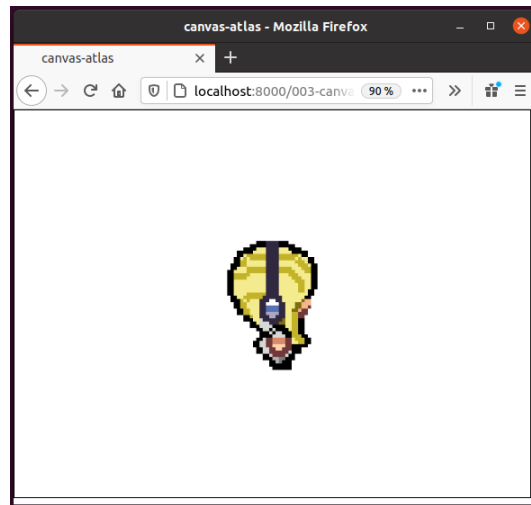


FIG. 10 – Animation de Misa avec un atlas.