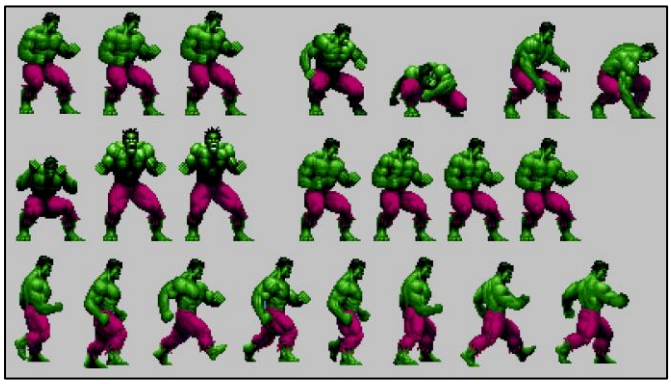
Séance Processing n°5 : Animation II

Lors de la dernière activité, nous avons vu comment faire des animations simples, comme déplacer une raquette ou une balle sur l'écran pour faire un jeu de squash ou de pong. Certains jeux un peu plus évolués nécessitent une animation plus complexe. Par exemple, si on incarne un personnage qui se déplace, il faut décomposer les différentes étapes de la marche pour obtenir un mouvement fluide et réaliste.

I./ Principe de base :

L'idée de base est d'avoir plusieurs sprites différents pour le même personnage, chaque sprite représentant une position particulière du personnage. Il suffit alors d'afficher successivement les différents états de notre personnage de façon suffisamment rapide pour que l'œil de l'utilisateur perçoive une animation fluide. Les différents sprites représentant les différentes positions d'un objet ou d'un personnage sont regroupés dans une grande image appelée « *Sprites Sheet* ».



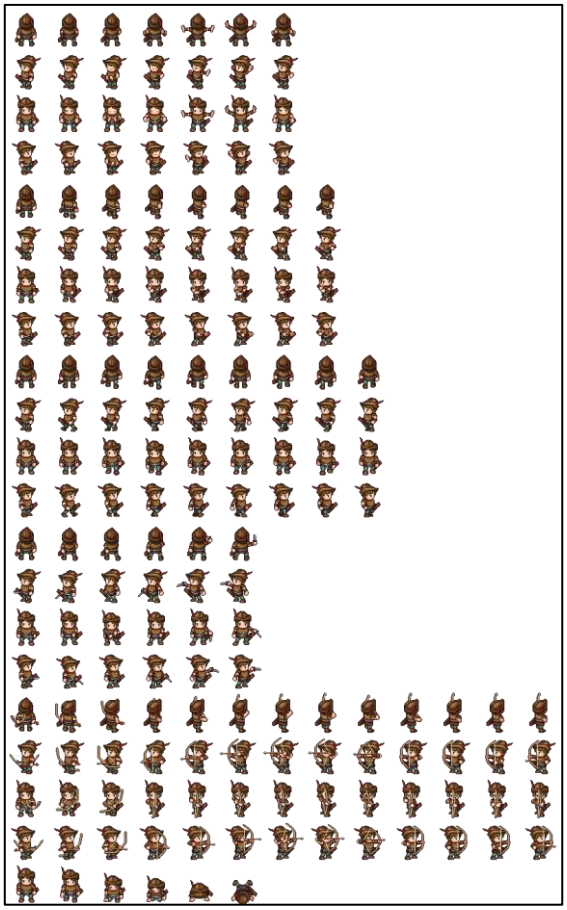
Exemple : extrait de la *sprites sheet* du jeu « The Incredible Hulk » sur SNES (https://en.wikipedia.org/wiki/The_Incredible_Hulk_(1994_video_game)

Vous pourrez trouver d'autres sprites sur le site « Sprite Database » : http://spritedatabase.net/

La réalisation d'une *sprites sheet* complète nécessite des qualités de graphiste ainsi que beaucoup de temps. Nous allons donc utiliser un site qui permet d'en réaliser une assez rapidement :

http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator/#

Un exemple de *sprites sheet* généré par ce site se trouve ci-contre, et correspond au fichier « *sprites_bonhomme.png* » situé dans le répertoire ICN sur le réseau. Lorsque vous ferez la suite de l'activité, rien ne vous empêche de réaliser votre propre personnage et de l'utiliser en lieu et place de celui-ci.



II./ Réalisation :

Charger puis lancer le programme « *Animation_personnage.pde* » situé dans le répertoire ICN sur le réseau. Une fenêtre vide grise s'ouvre, et un petit personnage est affiché au milieu. Ce dernier bouge en s'animant lorsqu'on appuie sur les touches 'a', 'z', 'e' et 'r'.

Ce programme comporte plusieurs procédures dont le but est la gestion des sprites. Leur fonctionnement est un peu compliqué et ne sera décrit que sommairement ici. Si vous voulez aller un peu plus loin dans la compréhension de ces procédures, vous pouvez consulter les sites suivants :

https://fr.flossmanuals.net/processing/introduction/

https://processing.org/reference/

- Procédure « *chargerSprites* » : cette procédure charge une image contenant une sprites sheet et la découpe en petits sprites qu'elle stocke dans un tableau appelé « sprite ».
- Procédure « *choisirDirection* » : cette procédure choisit la direction dans laquelle regarde ou se déplace le personnage. L'argument peut être GAUCHE, DROITE, HAUT ou BAS. (Vous pourrez regarder dans le code source comment sont codés ces noms).
- Procédure « *choisirAction* » : cette procédure choisit l'action que doit effectuer le personnage : ARRET, MARCHE, SORT, DAGUE, ARCHER, LANCE ou MORT. (Là encore, vous pourrez regarder dans le code source comment sont codés ces noms).
- Procédure « *calculerSprite* » : cette procédure détermine quelle ligne du tableau de sprites il faut utiliser en fonction de la direction et de l'action effectuée par le personnage.
- Procédure « *temporisationSprites* » : cette procédure détermine quel sprite il faut afficher en fonction du temps (cela permet de créer l'animation).

Le reste du programme (procédures « *setup* » et « *draw* ») est très classique et peut-être facilement compris avec les leçons précédentes. La seule nouveauté est la structure « *switch … case* » qui est assez intuitive. Vous trouverez de plus amples informations sur cette structure ici : https://processing.org/reference/switch.html.

III./ Améliorations :

- Actuellement, les touches de déplacement ne sont pas très judicieuses. Choisir des touches un peu plus commodes et modifier le code source en conséquence.

- Le personnage peut faire d'autres choses comme lancer un sort, donner un coup de dague, tirer à l'arc… Ajouter dans le code l'activation de ces mouvements en les associant à des touches du clavier (espace, +, –, *, /, … par exemple).

- Notre personnage se déplace dans un grand vide gris. Rajouter un décor dans lequel il puisse se déplacer. Vous trouverez un certain nombre d'images pouvant servir de décor dans le répertoire ICN sur le réseau. Vous pourrez aussi en chercher sur Internet.

- Le personnage peut aller partout dans le décor. Essayer de limiter son mouvement à certaines zones, pour éviter par exemple qu'il puisse voler dans les airs :

IV./ Listing du programme « Animation_personnage.pde »

```
// Activité "Animation de sprites"

int x, y; // coordonnées actuelles du sprite
PImage sprite[][]; // tableau contenant les différents aspects du sprite du personnage
PImage img;
int i, j, // indices de boucle
  nbImageH, nbImageV, // nombre d'images maximun sur la planche de sprites
  tmps, tmpsMax, // gère la temporisation entre 2 images
  iSprite, jSprite, maxSprite, minSprite, // indice de l'image en cours
  direction, // direction du personnage
  action, // action du personnage
  pas; // vitesse

void setup() {
  size(400, 400);
  chargerSprites("sprites_bonhomme.png");
  choisirDirection(BAS);
  choisirAction(ARRET);
}

void draw() {
  background(200);
  temporisationSprites(); // Calcul de l'aspect du sprite à afficher
  image(img, x, y); // Affichage du sprite
  if (keyPressed) { // Si une touche a été pressée...
    switch (key) { // ... on sélectionne l'action accompli par le personnage en fonction de la touche pressée
    case 'a':
      x = x - pas;
      choisirDirection(GAUCHE);
      choisirAction(MARCHE);
      break;
    case 'z':
      x = x + pas;
      choisirDirection(DROITE);
      choisirAction(MARCHE);
      break;
    case 'e':
      y = y - pas;
      choisirDirection(HAUT);
      choisirAction(MARCHE);
      break;
    case 'r':
      y = y + pas;
      choisirDirection(BAS);
      choisirAction(MARCHE);
      break;
    case ' ':
      // à vous de completer avec l'arc (ARCHER)
      break;
    case '*':
      // à vous de completer avec la dague (DAGUE)
      break;
    case '/':
      // à vous de completer avec la lance (LANCE)
      break;
    case '-':
      // à vous de compléter avec la mort (MORT)
      break;
    case '+':
      // à vous de compléter avec incantation (SORT)
      break;
    default:
      choisirAction(ARRET);
      break;
    }
  } else {
      choisirAction(ARRET);
    }
}

// Procédures de gestion des sprites

// Procédure qui charge la planche de sprites, la découpe et place les sprites obtenus dans le tableau "sprite"
void chargerSprites(String chemin) {
  // fichiers images de Wulax, makrohn, jrconway3
  // https://github.com/jrconway3/Universal-LPC-spritesheet CC-BY-SA & GPL 3.0
  //
  PImage imag = loadImage(chemin);//grille de 13*21 images de 64 x 64 pixels
  nbImageH = 13;
  nbImageV = 21;

  sprite = new PImage[nbImageH][nbImageV];
  for (j = 0; j < nbImageV; j = j + 1) {
    for (i = 0; i < nbImageH; i = i + 1) {
      sprite[i][j] = imag.get(i * 64, j * 64, 64, 64);
    }
  }
  tmps = 0;
  tmpsMax = 4;
  frameRate(50);
  iSprite = 0;
  jSprite = 10;
  maxSprite = 8;
  img = sprite[iSprite][jSprite];
  x = width/2 - img.width/2;
  y = height/2 - img.height/2;
  pas = 1;
}
```

```
// Procédure qui en fonction du temps écoulé choisit l'aspect du sprite
void temporisationSprites() {
  tmps = tmps + 1;
  if (tmps > tmpsMax) {
    tmps = 0;
    iSprite = (iSprite + 1);
    if (iSprite >= maxSprite) {
      iSprite = minSprite;
    }
    img = sprite[iSprite][jSprite];
  }
}

// ********** directions
final int BAS = 2;
final int DROITE = 3;
final int HAUT = 0;
final int GAUCHE = 1;

// ********** actions
final int ARRET = -1;
final int SORT = 0;
final int LANCE = 1;
final int ARCHER = 4;
final int MARCHE =2;
final int DAGUE =3;
final int MORT =5;

// On choisit la direction de déplacement du personnage
void choisirDirection(int sens) {
  direction = sens;
  maxSprite = 0;
  minSprite = 0;
  calculerSprite();
}

// On choisit l'action effectuée par le personnage
void choisirAction(int act) {
  if (act == ARRET) {
    maxSprite = 0;
    minSprite = 0;
    iSprite = 0;
  } else {
    action = act;
    switch(action) {
    case SORT :
      maxSprite = 7;
      minSprite = 1;
      break;
    case LANCE :
      maxSprite = 8;
      minSprite = 1;
      break;
    case ARCHER :
      maxSprite = 13;
      minSprite = 1;
      break;
    case MARCHE :
      maxSprite = 9;
      minSprite = 1;
      break;
    case DAGUE :
      maxSprite = 6;
      minSprite = 1;
      break;
    case MORT :
      maxSprite = 6;
      minSprite = 1;
      direction= HAUT;
      break;
    }
  }
  calculerSprite();
}

// En fonction de la direction et de l'action effectuée, on calcule le numéro du sprite à choisir dans le tableau "sprite"
void calculerSprite() {
  jSprite = action * 4 + direction;

}
```