Animation_personnage_correction.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,decor;
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(580, 444);
  decor=loadImage("desert.png");
  chargerSprites("sprites_bonhomme.png");
  choisirDirection(BAS);
  choisirAction(ARRET);
  frameRate(50);
}

void draw() {
  background(200);
  image(decor,0,0);
  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 '4':
      x = x - pas;
      choisirDirection(GAUCHE);
      choisirAction(MARCHE);
      break;
    case '6':
      x = x + pas;
      choisirDirection(DROITE);
      choisirAction(MARCHE);
      break;
    case '8':
      y = y - pas;
      choisirDirection(HAUT);
      choisirAction(MARCHE);
      break;
    case '2':
      y = y + pas;
      choisirDirection(BAS);
      choisirAction(MARCHE);
      break;
    case ' ':
      // à vous de completer avec l'arc (ARCHER)
      choisirAction(ARCHER);
      break;
    case '*':
      // à vous de completer avec la dague (DAGUE)
      choisirAction(DAGUE);
      break;
    case '/':
      // à vous de completer avec la lance (LANCE)
      choisirAction(LANCE);
      break;
    case '-':
      // à vous de compléter avec la mort (MORT)
      choisirAction(MORT);
      break;
    case '+':
      // à vous de compléter avec incantation (SORT)
      choisirAction(SORT);
      break;
    default:
      choisirAction(ARRET);
      break;
    }
  } else {
      choisirAction(ARRET);
    }
}
```

```
// Procédures de gestion des sprites

void chargerSprites(String chemin) { // Procédure qui charge la planche de sprites, la découpe et place les sprites obtenus
dans le tableau "sprite"
  // 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 = 8;
  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;
}

void temporisationSprites() { // Procédure qui en fonction du temps écoulé choisit l'aspect du sprite
  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;

void choisirDirection(int sens) { // On choisit la direction de déplacement du personnage
  direction = sens;
  maxSprite = 0;
  minSprite = 0;
  calculerSprite();
}
void choisirAction(int act) { // On choisit l'action effectuée par le personnage
  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();
}

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