**Atelier Animation**

**Introduction:**

Au moment du déplacement d'une entité (personnage, ennemi, boule de feu ...), l'animation de cette dernière est obligatoire pour rendre les scènes plus dynamiques et proches de la réalité.

A partir des ressources (images) que vous avez réalisées dans les séances de multimédia, nous allons voir dans cet atelier comment les exploiter pour animer vos entités.

1. **Pré-requis:**

Pour bien assimiler les objectifs de cet atelier, vous devez connaître:

* La conception du jeu: Game Loop
* La fonction SDL\_BlitSurface(), les types SDL\_Rect et SDL\_Surface de la bibliothèque SDL 1.2

1. **Définition:**

"Une animation consiste à afficher une succession d'images donnant l' apparence que l'entité animée est en train de se déplacer” [1].

Dépendamment des ressources dont vous disposez, il existe deux manières pour animer une entité 2D: la première à partir d'un sprite sheet et la deuxième à partir d'un ensemble d'images séparées.

1. **Animation à partir de sprite sheet**
2. **Contraintes:**

Un sprite sheet est une image composée de plusieurs "frames" [1]. Pour bien réussir son animation, le sprite sheet doit être régulier c.à.d. que toutes les frames doivent avoir la même longueur et la même largeur. ( voir figure 1 et figure 2)

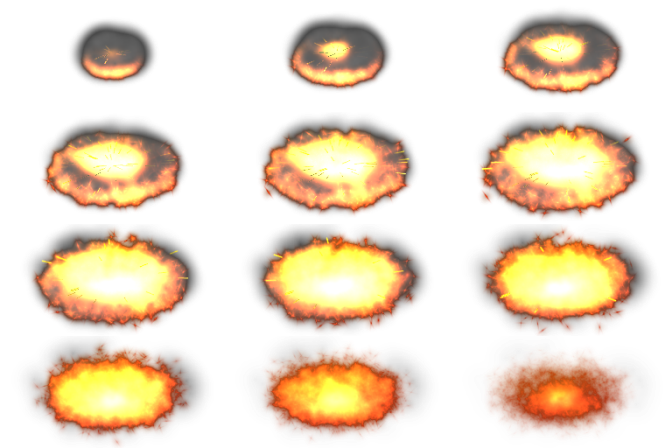
 

Figure 1: sprite sheet irrégulier Figure 2: sprite sheet régulier

1. **Principe**

Le principe de l'animation à partir d'un sprite sheet est de choisir, dans chaque itération de la boucle du jeu, une partie(frame) à afficher. Cette partie qui a une forme rectangulaire doit être exprimée grâce à une variable de type SDL\_Rect en mettant à jour ses 4 champs selon la direction du mouvement :

* x: abscisse dans le repère de l'image
* y: ordonnée dans le repère de l'image
* w: largeur d'un frame (sous image)
* h: hauteur d'un frame

Si on suppose que le sprite peut être divisé en NBL lignes et NBC colonnes, une ligne correspond aux animations d'une direction.



**Remarque: Si le nombre d’images d’animation n’est pas le même dans chaque direction, vous devez stocker, dans un tableau, ce nombre (pour chaque direction).**

1. **Etapes:**

Il s'agit de changer la structure de l’entité à animer, implémenter la fonction qui l'initialise et la fonction qui l'anime.

* Mettre à jour la structure a ce qu'elle contient:

-une image du sprite sheet de type SDL\_Surface \*: sprite

-un champ de type entier, indiquant la direction de déplacement: direction

-Un champ de type SDL\_Rect indiquant la position par rapport à l'écran :postScreen

-Un champ de type SDL\_Rect indiquant la position par rapport à l'image du sprite: posSprite (quelle partie/rectangle du sprite sheet vous souhaitez afficher)

* Dans la fonction **initialiseEntity**, vous devez initialiser les champs suivant comme suit:
* posSprite.x←0
* posSprite.y ←0
* posSprite.w ←largeur image/NBC
* posSprite.h ←Hauteur image/ NBL
* charger l’image du sprite
* Dans la fonction **animateEntity**, vous devez mettre à jour le champ posSprite comme suit:

posSprite.y =direction\* posSprite.h (se positionner dans la ligne correspondant à la direction)

**Si** posSprite.x = largeurSprite - posSprite.w (si dernier frame atteint)

posSprite.x =0 (revenir au premier frame)

**Sinon** posSprite.x = posSprite.x + posSprite.w (avancer au frame suivant)

**Fin Si**

1. **Animation à partir d'images séparées**

**4.1 Tableau d’images**

Comme pour la section précédente, nous devons changer la structure de l’entité à animer, implémenter la fonction qui initialise et la fonction qui l'anime.

* Mettre à jour la structure a ce qu'elle contient:

-un tableau de type SDL\_Surface \* pour stocker les images d'animation : tab

-un champ de type entier, indiquant la direction de déplacement: direction

-un entier indiquant le numéro de l'image à afficher dans une itération de la boucle du jeu: num

* Implémentez la fonction initialiseEntity, en initialisant les champs suivants:
* direction←0
* num←0
* charger toutes les images d’animation
* Implémentez la fonction d'animation:

On suppose que vous disposez de n images d'animation a droite et m images d'animation à gauche. Ainsi la taille de votre tableau tab est n+m.

Animer l'entité, consiste à trouver le numéro de l'image à afficher en suivant ce pseudo code:

**Si** direction =0 (droite)

**Si** num=n-1

num =0

**Sinon** num ++

**Fin Si**

**Sinon**

**Si** num =n+m-1

num=n

**Sinon** num ++

**Fin Si**

**Fin Si**

**sinon si num=n+m+l-1**

**si num =n+m+l-1**

**alors num=n+m**

**sinon num++**

**Fin Si**

**4.2 Matrice d’images**

Vous pouvez utiliser une matrice Mat de SDL Surface \* ou chaque ligne représente l’animation d’une direction donnée.

SI le nombre d’images d’animation est le même le code précédent reste valable :

**Si** num=n-1

num =0

**Sinon** num ++

**Fin Si**

sinon vous devez stocker dans un tableau t le nombre d’images pour chaque direction

Le code précédent devient:

**Si** num=t[direction]-1

num =0

**Sinon** num ++

**Fin Si**

et vous devez MAJ votre fonction d'affichage en affichant Mat[direction][num] au lieu de

1. **Appel dans la boucle du jeu**

L'appel des fonctions d'initialisation, d'affichage et d'animation doit se faire ainsi:

Début 

Initialise\_entity

**Tant que** (continuer)

BlitEntity

**Si** événement = touche droite

direction ← 0

**Sinon**

Si événement = touche gauche

direction ←1

**Fin si**

**Fin si**

**Si** direction dans {0,1}

moveEntity

animateEntity

**Fin si**

**Fin tant que**

**Fin**

1. **Travail à faire :**

Implémentez les fonctions initialiseEntity, animateEntity et blitEntity et appelez les dans la boucle du jeu.

**References:**

[1]De David Conger, Ron Little, Creating Games in C++: A Step-by-step Guide,2006