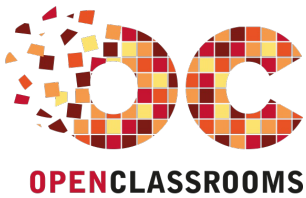
Les animations optimisées avec SDL

Par Arthurus



www.openclassrooms.com

Dernière mise à jour le 24/08/2010

Sommaire

Les animations optimisées avec SDL

Par  Arthurus

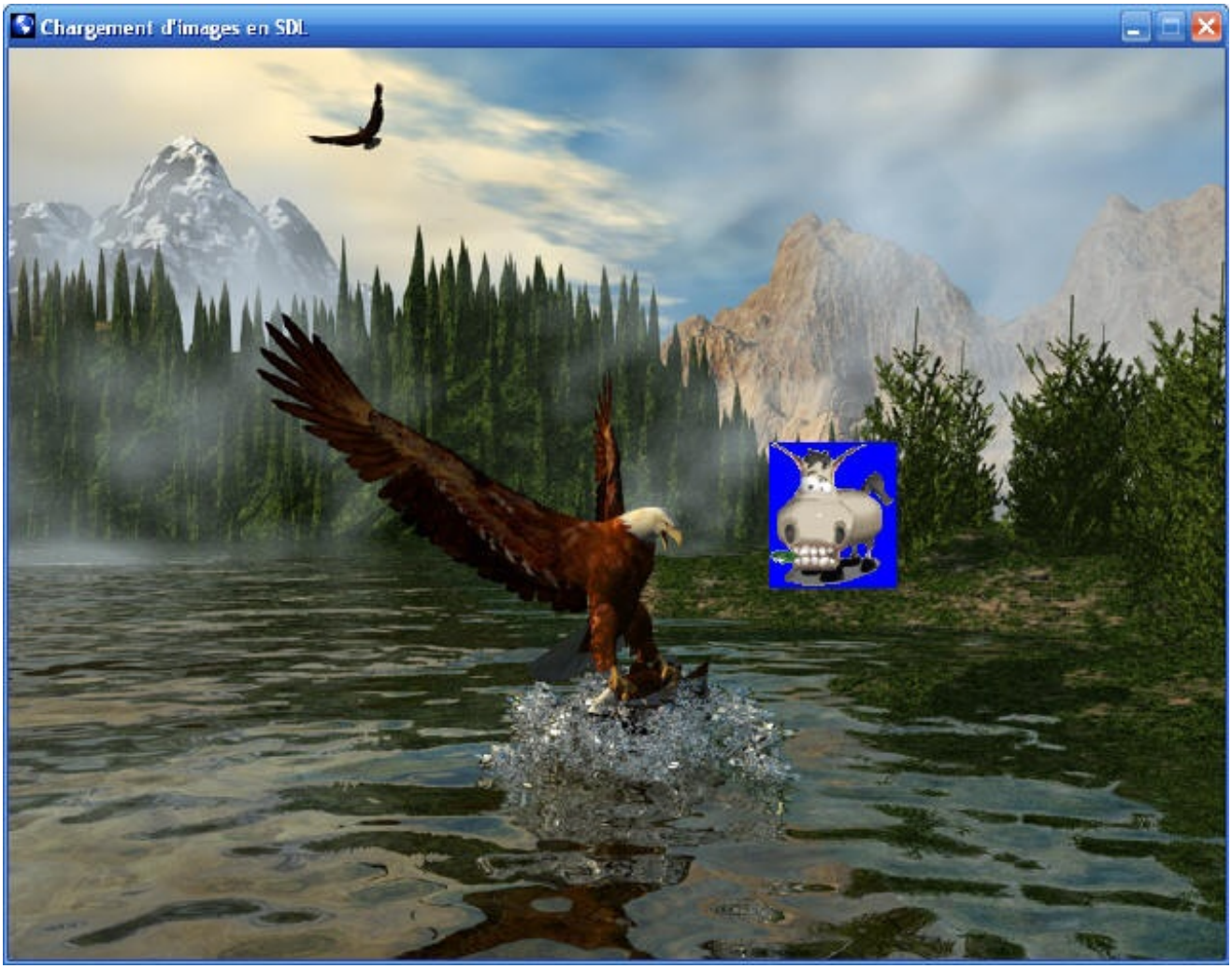
Mise à jour : 24/08/2010
Difficulté : Intermédiaire Durée d'étude : 5 jours

Ce cours s'adresse essentiellement aux personnes qui ont déjà assez bien pratiqué SDL.
Si ce n'est pas le cas, vous pouvez commencer par lire le tuto officiel de M@teo21, puis ensuite vous perfectionner en participant au forum C.
Le but de ce cours est de vous présenter deux méthodes pour optimiser le rafraichissement des images.
Sommaire du tutoriel :

Introduction

Commençons d'abord par une animation dite naïve.
En fait, pour faire une animation, le plus simple est de, à chaque nouveau frame, redessiner tous les objets.

Prenons un exemple :



Pour faire bouger zozor, la méthode intuitive est de redessiner l'image de fond en entier... Ceci aura pour effet d'effacer zozor. Puis ensuite, dessiner zozor à sa nouvelle position.

On peut remarquer qu'en voulant déplacer une petite image (zozor), la méthode intuitive oblige à redessiner tout le décor, même celui qui n'a pas été changé.

Première approche

Méthode naïve

Pour illustrer les méthodes d'optimisation, je vais utiliser tout au long de ce tuto un petit programme qui fait une animation.

Voici une version naïve du programme :

Code : C

```
#include <stdio.h>
#include <SDL/SDL.h>

SDL_Surface *ecran, *imageDeFond, *zozor;
/*
* Fonction pour initialiser SDL
* et les variables globales.
*/
void init(void)
{
 SDL_Init(SDL_INIT_VIDEO);
 ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE);
 imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
 zozor = SDL_LoadBMP("zozor.bmp");
```

```
}

/*
* Fonction pour être à l'écoute
* de l'événement SDL_Quit.
*/
void input_handle(void)
{
 SDL_Event event;

 while (SDL_PollEvent(&event)) {

  switch (event.type) {
  case SDL_QUIT:
   SDL_Quit();
   exit(0);
   break;
  }
 }
}

/*
* La fonction qui fera l'animation
*/
void anime(void)
{
 SDL_Rect positionFond, positionZozor;
 int avanceX = 1, avanceY = 1; // Ces variables diront si zozor doit
avancer ou reculer.

 positionFond.x = 0;
 positionFond.y = 0;

 positionZozor.x = 0;
 positionZozor.y = 0;

 while (1) {

  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond); //
Dessiner le fond
  SDL_BlitSurface(zozor, NULL, ecran, &positionZozor); // Dessiner
zozor

  if (avanceX) {
   positionZozor.x++; // Si avance est à 1 alors on incrémente x
  } else {
   positionZozor.x--; // Sinon on décrémente x
  }

  if (avanceY) {
   positionZozor.y++;
  } else {
   positionZozor.y--;
  }

                /*
* Arrivé à l'une des extrémités, on change la valeur de avance
*/
  if (positionZozor.x == ecran->w - zozor->w - 1)
   avanceX = 0;
  else if (positionZozor.x == 0)
   avanceX = 1;

  if (positionZozor.y == ecran->h - zozor->h - 1)
   avanceY = 0;
  else if (positionZozor.y == 0)
   avanceY = 1;

  SDL_Flip(ecran); // On affiche réellement l'image.
  input handle(); // On appelle le gestionnaire d'évènements.
```

```
  //SDL_Delay(10);
 }
}

int main(void)
{
 init();
 anime();
 return 0;
}
```

Les fichiers bmp sont téléchargeables ici.

Ne vous attardez pas trop pour comprendre tous les détails du programme.
Ce qui nous intéresse le plus est cette partie :

Code : C

```
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);
SDL_Flip(ecran);
```

On voit bien qu'à chaque itération, on redessine tout... Et bien sûr ceci est très couteux.

Le clipping

Préparation

Cette fois nous ferons une approche différente :
Au lieu de redessiner tout le fond, on ne redessinera que la partie du fond qui a été modifiée.
Cette méthode s'appelle le clipping.
C'est-à-dire qu'on définit un clipper (un rectangle), pour dire à SDL : N'applique les blitsurface que sur cette partie de la surface.
Par défaut, le clipper est défini comme étant toute la surface.

Avant d'entamer l'implémentation du clipping, d'abord il nous faut imaginer un moyen pour nous rappeler l'ancienne position de zozor.
Je vous propose de faire comme ceci :

Code : C

```
void anime(void)
{
 SDL_Rect positionFond, positionZozor;
        SDL_Rect oldpositionZozor; // On ajoute une variable qui
mémorisera l'ancienne position de zozor.
 int avanceX = 1, avanceY = 1;

 positionFond.x = 0;
 positionFond.y = 0;

 positionZozor.x = 0;
 positionZozor.y = 0;

 oldpositionZozor.x = 0; // Initialement, cette variable aura la
même valeur que positionZozor.
 oldpositionZozor.y = 0;


 while (1) {

  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
  SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);
```

```
                /*
        * On met à jour oldpositionZozor avant de modifier positionZozor
        */
          oldpositionZozor.x = positionZozor.x;
          oldpositionZozor.y = positionZozor.y;

          if (avanceX) {
           positionZozor.x++;
          } else {
           positionZozor.x--;
          }

          if (avanceY) {
           positionZozor.y++;
          } else {
           positionZozor.y--;
          }

          if (positionZozor.x == ecran->w - zozor->w - 1)
           avanceX = 0;
          else if (positionZozor.x == 0)
           avanceX = 1;

          if (positionZozor.y == ecran->h - zozor->h - 1)
           avanceY = 0;
          else if (positionZozor.y == 0)
           avanceY = 1;

          SDL_Flip(ecran);
          input_handle();
          //SDL_Delay(10);
         }
        }
```

Cette fois ci, oldpositionZozor contient l'ancienne position de zozor.
Maintenant, passons aux choses sérieuses...

Pratique

Comme vous le savez déjà, une image est un rectangle, caractérisé par une position, une hauteur et une largeur.
Donc avec oldpositionZozor, zozor->w et zozor->h, on connait exactement où zozor a été avant.
Il ne reste plus qu'à définir le clipper.

Le clipper sur SDL

SDL offre une fonction qui met un clipper sur une surface.
C'est à dire, comme expliqué plus haut, mettre un rectangle, pour qu'au moment où l'on fait un blit, seul la partie du rectangle sera prise en compte.

Cette fonction s'appelle SDL_SetClipRect, et voici son prototype :

Code : C

```
void SDL_SetClipRect(SDL_Surface *surface, SDL_Rect *rect);
```

Ça prend en paramètre une surface sur laquelle on posera le clipper, et un rectangle qui jouera le rôle du clipper.

Pour ce qui est de SDL_Rect, vous connaissez sûrement cette structure...
Pour mon code, je l'ai utilisée comme ceci :

Code : C

```
SDL_Rect positionFond;
positionFond.x = 0;
positionFond.y = 0;
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
```

En fait ce type offre encore plus de possibilités.
Voici la définition de SDL_Rect :

Code : C

```
typedef struct{
   Sint16 x, y;
   Uint16 w, h;
} SDL_Rect;
```

On voit que ça permet de définir un vrai rectangle, et non seulement une position.
Petit rappel :
w = largeur
h = hauteur

Code

Maintenant, appliquons le clipping sur notre petite animation...
Commençons par définir le clipper :

Code : C

```
SDL_Rect clipper;
/* Comme position, on prend l'ancienne position de zozor */
clipper.x = oldpositionZozor.x;
clipper.y = oldpositionZozor.y;

/* Pour la largeur et la hauteur, nous prendrons celles de zozor */
clipper.h = zozor->h;
clipper.w = zozor->w;
```

Et maintenant, il suffit de mettre ce clipper sur ecran à chaque tour de boucle :

Code : C

```
SDL_SetClipRect(ecran, &clipper);
```

De cette façon, peu importe la surface blitée sur ecran, seule la partie du clipper sera prise en considération.

Maintenant, si on dessine l'image du fond, ceci aura un effet uniquement sur le rectangle de l'ancienne position de zozor... En d'autres mots, dessiner l'image de fond maintenant aura pour effet d'effacer l'ancien zozor, sans pour autant dessiner les pixels qui n'ont pas changé.

Globalement, le code ressemblerait ça :

Code : C

```
clipper.x = oldpositionZozor.x;
clipper.y = oldpositionZozor.y;
clipper.h = zozor->h;
clipper.w = zozor->w;
SDL_SetClipRect(ecran,&clipper);

SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
```

Sauf que là, ça ne va pas marcher.
On ne vous l'a jamais dit, mais SDL_BlitSurface peut changer ce qui est passé comme argument de position quand on manipule les clippers.
Je ne vais pas entrer dans les détails de ce changement dans ce tuto.
Pour dévier ce problème, nous allons créer une copie de cette position, et la passer en paramètre :

Code : C

```
clipper.x = oldpositionZozor.x;
clipper.y = oldpositionZozor.y;
clipper.h = zozor->h;
clipper.w = zozor->w;
SDL_SetClipRect(ecran,&clipper);

/* On met à jour les copies */
positionFond_c.x = positionFond.x;
positionFond_c.y = positionFond.y;
positionZozor_c.x = positionZozor.x;
positionZozor_c.y = positionZozor.y;

/* On passe une copie en paramètre */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond_c);
```

Puis maintenant, pour dessiner zozor, on a le choix, soit on relâche le clipper, en passant comme argument de rectangle à la fonction SDL_SetClipRect() un argument NULL comme ceci :

Code : C

```
SDL_SetClipRect(ecran,NULL);
```

Soit on peut définir un nouveau clipper qui correspondra à la nouvelle position de zozor :

Code : C

```
clipper.x = positionZozor.x;
clipper.y = positionZozor.y;
clipper.h = zozor->h;
clipper.w = zozor->w;
SDL_SetClipRect(ecran,&clipper);

SDL_BlitSurface(zozor, NULL, ecran, &positionZozor_c);
```

Et n'oublions pas que, vu que dans la boucle de la fonction anime on ne dessine qu'une partie de l'image de fond, il faut bien dessiner l'image en entier dans l'initialisation.

Code : C

```
void init(void)
{
 SDL_Init(SDL_INIT_VIDEO);
 ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE);
 imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
 zozor = SDL_LoadBMP("zozor.bmp");

 SDL_Rect positionFond;
 positionFond.x = 0;
 positionFond.y = 0;

        /* On dessine entièrement l'image de fond */
 SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
```

```
}
```

Code complet :

Secret (cliquez pour afficher)

Code : C

```c
#include <stdio.h>
#include <SDL/SDL.h>

SDL_Surface *ecran, *imageDeFond, *zozor;

void init(void)
{
 SDL_Init(SDL_INIT_VIDEO);
 ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE);
 imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
 zozor = SDL_LoadBMP("zozor.bmp");

 SDL_Rect positionFond;
 positionFond.x = 0;
 positionFond.y = 0;

 SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
}

void input_handle(void)
{
 SDL_Event event;

 while (SDL_PollEvent(&event)) {

  switch (event.type) {
  case SDL_QUIT:
   SDL_Quit();
   exit(0);
   break;
  }
 }
}

void anime(void)
{
 SDL_Rect positionFond, positionZozor, oldpositionZozor;
 SDL_Rect positionFond_c, positionZozor_c;
 SDL_Rect clipper;
 int avanceX = 1, avanceY = 1;

 positionFond.x = 0;
 positionFond.y = 0;

 positionZozor.x = 0;
 positionZozor.y = 0;

 oldpositionZozor.x = 0;
 oldpositionZozor.y = 0;


 while (1) {

  clipper.x = oldpositionZozor.x;
  clipper.y = oldpositionZozor.y;
  clipper.h = zozor->h;
  clipper.w = zozor->w;
  SDL_SetClipRect(ecran,&clipper);

  positionFond_c.x = positionFond.x;
```

```
    positionFond_c.y = positionFond.y;

    positionZozor_c.x = positionZozor.x;
    positionZozor_c.y = positionZozor.y;


    SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond_c);

                    SDL_SetClipRect(ecran,NULL); // J'ai choisi cette
solution
    SDL_BlitSurface(zozor, NULL, ecran, &positionZozor_c);

    oldpositionZozor.x = positionZozor.x;
    oldpositionZozor.y = positionZozor.y;

    if (avanceX) {
     positionZozor.x++;
    } else {
     positionZozor.x--;
    }

    if (avanceY) {
     positionZozor.y++;
    } else {
     positionZozor.y--;
    }

    if (positionZozor.x == ecran->w - zozor->w - 1)
     avanceX = 0;
    else if (positionZozor.x == 0)
     avanceX = 1;

    if (positionZozor.y == ecran->h - zozor->h - 1)
     avanceY = 0;
    else if (positionZozor.y == 0)
     avanceY = 1;

    SDL_Flip(ecran);
    input_handle();
    //SDL_Delay(10);
   }
  }

  int main(void)
  {
   init();
   anime();
   return 0;
  }
```

Je vous laisse admirer la vitesse de zozor avec la méthode de clipping.

Alternative

Je vous rappelle le code naïf qu'on avait avant :

Code : C

```
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);
```

Une autre façon d'optimiser l'affichage, est d'utiliser un paramètre de SDL_BlitSurface que vous n'avez pas beaucoup l'habitude d'utiliser.
Le prototype de cette fonction étant :

Code : C

```
int SDL_BlitSurface(SDL_Surface *src, SDL_Rect *srcrect, SDL_Surface
*dst, SDL_Rect *dstrect);
```

Nous allons utiliser le paramètre srcrect.

Comme son nom peut l'indiquer, il précise le rectangle dans lequel le blit sera effectué.
Avec ceci, le code devient très facile, car il suffit de faire directement :

Code : C

```
/*
* De la même manière, on positionne le clipper
* sur l'ancienne position de zozor
*/
clipper.x = oldpositionZozor.x;
clipper.y = oldpositionZozor.y;
clipper.h = zozor->h;
clipper.w = zozor->w;

/* Maintenant, on dessine la partie de l'image de fond à l'ancienne
position de zozor */
SDL_BlitSurface(imageDeFond, &clipper, ecran, &oldpositionZozor);

/* On dessine le nouveau zozor */
SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);
```

Cela veut dire qu'on dessine une partie de l'image du fond (définie par le clipper) à l'ancienne position de zozor.

Le code complet :

Secret (cliquez pour afficher)

Code : C

```
#include <stdio.h>
#include <SDL/SDL.h>

SDL_Surface *ecran, *imageDeFond, *zozor;

void init(void)
{
 SDL_Init(SDL_INIT_VIDEO);
 ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE);
 imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
 zozor = SDL_LoadBMP("zozor.bmp");

 SDL_Rect positionFond;
 positionFond.x = 0;
 positionFond.y = 0;

 SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
}

void input_handle(void)
{
 SDL_Event event;

 while (SDL_PollEvent(&event)) {

  switch (event.type) {
  case SDL_QUIT:
   SDL_Quit();
   exit(0);
   break;
```

```
    }
   }
  }

  void anime(void)
  {
   SDL_Rect positionFond, positionZozor, oldpositionZozor;
   SDL_Rect clipper;
   int avanceX = 1, avanceY = 1;

   //SDL_SetClipRect(ecran,&clipper);

   positionFond.x = 0;
   positionFond.y = 0;

   positionZozor.x = 0;
   positionZozor.y = 0;

   oldpositionZozor.x = 0;
   oldpositionZozor.y = 0;


   while (1) {

    clipper.x = positionZozor.x;
    clipper.y = positionZozor.y;
    clipper.h = zozor->h;
    clipper.w = zozor->w;
    SDL_BlitSurface(imageDeFond, &clipper, ecran,
  &oldpositionZozor);
    SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);
    oldpositionZozor.x = positionZozor.x;
    oldpositionZozor.y = positionZozor.y;

    if (avanceX) {
     positionZozor.x++;
    } else {
     positionZozor.x--;
    }

    if (avanceY) {
     positionZozor.y++;
    } else {
     positionZozor.y--;
    }

    if (positionZozor.x == ecran->w - zozor->w - 1)
     avanceX = 0;
    else if (positionZozor.x == 0)
     avanceX = 1;

    if (positionZozor.y == ecran->h - zozor->h - 1)
     avanceY = 0;
    else if (positionZozor.y == 0)
     avanceY = 1;

    SDL_Flip(ecran);
    input_handle();
    //SDL_Delay(10);
   }
  }

  int main(void)
  {
   init();
   anime();
   return 0;
  }
```

Encore une fois, le résultat est surprenant.

Extension

Nous parlerons dans cette partie de comment utiliser le clipping pour gérer plusieurs objets animés.

Pour illustrer un exemple, je prendrai zozor et une planète qui se baladeront sur l'ecran.
Je vous propose ce code naïf :

Code : C

```
#include <stdio.h>
#include <SDL/SDL.h>

SDL_Surface *ecran, *imageDeFond, *zozor, *icone;

void init(void)
{
 SDL_Init(SDL_INIT_VIDEO);
 ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE);
 imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
 zozor = SDL_LoadBMP("zozor.bmp");
 icone = SDL_LoadBMP("sdl_icone.bmp"); // La 2eme image animée.

 SDL_Rect positionFond;
 positionFond.x = 0;
 positionFond.y = 0;
}

void input_handle(void)
{
 SDL_Event event;

 while (SDL_PollEvent(&event)) {

  switch (event.type) {
  case SDL_QUIT:
   SDL_Quit();
   exit(0);
   break;
  }
 }
}

void anime(void)
{
 SDL_Rect positionFond, positionZozor, positionIcone;
 int avanceX = 1, avanceY = 1;
 int avanceXI = 0, avanceYI = 1;

 positionFond.x = 0;
 positionFond.y = 0;

 positionZozor.x = 0;
 positionZozor.y = 0;

 positionIcone.x = ecran->w - icone->w; // La nouvelle image sera
initialement en haut à droite
 positionIcone.y = 0;

 while (1) {
  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);

  SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);

  SDL_BlitSurface(icone, NULL, ecran, &positionIcone);

  if (avanceX) {
```

```
        positionZozor.x++;
       } else {
        positionZozor.x--;
       }

       if (avanceY) {
        positionZozor.y++;
       } else {
        positionZozor.y--;
       }

       if (positionZozor.x == ecran->w - zozor->w - 1)
        avanceX = 0;
       else if (positionZozor.x == 0)
        avanceX = 1;

       if (positionZozor.y == ecran->h - zozor->h - 1)
        avanceY = 0;
       else if (positionZozor.y == 0)
        avanceY = 1;

                    /* Les calculs qui suivent sont analogues aux
    précédents */
       if (avanceXI) {
        positionIcone.x++;
       } else {
        positionIcone.x--;
       }
       if (avanceYI) {
        positionIcone.y++;
       } else {
        positionIcone.y--;
       }

       if (positionIcone.x == ecran->w - icone->w - 1)
        avanceXI = 0;
       else if (positionIcone.x == 0)
        avanceXI = 1;

       if (positionIcone.y == ecran->h - icone->h - 1)
        avanceYI = 0;
       else if (positionIcone.y == 0)
        avanceYI = 1;

       SDL_Flip(ecran);
       input_handle();
       //SDL_Delay(10);
      }
     }

     int main(void)
     {
      init();
      anime();
      return 0;
     }
```

Application du clipping

Nous avons initialement ce code :

Code : C

```
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);

SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);

SDL_BlitSurface(icone, NULL, ecran, &positionIcone);
```

De façon très analogue à ce qui a été vu plus haut, nous utiliserons deux variables pour nous souvenir des anciennes positions de zozor et de la planète.

Code : C

```
SDL_Rect oldpositionZozor, oldpositionIcone;
 oldpositionZozor.x = 0;
 oldpositionZozor.y = 0;

 oldpositionIcone.x = ecran->w - icone->w;
 oldpositionIcone.y = 0;

 while (1) {
  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);

  SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);

  SDL_BlitSurface(icone, NULL, ecran, &positionIcone);

                /* Sauvegarde de l'ancienne position de zozor */
  oldpositionZozor.x = positionZozor.x;
  oldpositionZozor.y = positionZozor.y;

                /* Sauvegarde de l'ancienne position de la planète
*/
  oldpositionIcone.x = positionIcone.x;
  oldpositionIcone.y = positionIcone.y;
```

Maintenant, il suffit d'appliquer les mêmes notions vues auparavant... Le schéma est le suivant :

- Mettre le clipper sur le rectangle de zozor
- Dessiner le fond pour effacer zozor.
- Mettre le clipper sur le rectangle de la planète
- Dessiner le fond pour effacer la planète.
- Enlever le clipper.
- Dessiner les nouveaux zozor et planète.

En code, ça donnerait ceci :

Code : C

```
//Code sans copie
         clipper.x = oldpositionZozor.x;
  clipper.y = oldpositionZozor.y;
  clipper.h = zozor->h;
  clipper.w = zozor->w;
  SDL_SetClipRect(ecran,&clipper); // Clipper sur zozor

  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond); //
Effacer zozor

  clipper.x = oldpositionIcone.x;
  clipper.y = oldpositionIcone.y;
  clipper.h = icone->h;
  clipper.w = icone->w;
  SDL_SetClipRect(ecran,&clipper); // Clipper sur la planète

  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond); //
Effacer la planète.

  SDL_SetClipRect(ecran,NULL); // Enlever le clipper

  SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);
```

```
SDL_BlitSurface(icone, NULL, ecran, &positionIcone);
```

Pour que ça marche, il ne reste plus qu'à rajouter des copies (rappelez vous, blitsurface peut changer le paramètre de position).

Code : C

```
//Code avec copie.
  clipper.x = oldpositionZozor.x;
  clipper.y = oldpositionZozor.y;
  clipper.h = zozor->h;
  clipper.w = zozor->w;
  SDL_SetClipRect(ecran,&clipper); // Clipper sur zozor

  positionFond_c.x = positionFond.x;
  positionFond_c.y = positionFond.y;

  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond_c); //
Effacer zozor

  clipper.x = oldpositionIcone.x;
  clipper.y = oldpositionIcone.y;
  clipper.h = icone->h;
  clipper.w = icone->w;
  SDL_SetClipRect(ecran,&clipper); // Clipper sur la planète

  positionFond_c.x = positionFond.x;
  positionFond_c.y = positionFond.y;

  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond_c); //
Effacer la planète.

  SDL_SetClipRect(ecran,NULL); // Enlever le clipper

                /* Pas besoin de copie ici, car le clipper est
désactivé.*/

  SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);

  SDL_BlitSurface(icone, NULL, ecran, &positionIcone);

  oldpositionZozor.x = positionZozor.x;
  oldpositionZozor.y = positionZozor.y;

  oldpositionIcone.x = positionIcone.x;
  oldpositionIcone.y = positionIcone.y;
```

Voici le code complet :

Secret (cliquez pour afficher)

Code : C

```
#include <stdio.h>
#include <SDL/SDL.h>

SDL_Surface *ecran, *imageDeFond, *zozor, *icone;

void init(void)
{
 SDL_Init(SDL_INIT_VIDEO);
 ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE);
 imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
 zozor = SDL_LoadBMP("zozor.bmp");
 icone = SDL_LoadBMP("sdl_icone.bmp");
```

```
 SDL_Rect positionFond;
 positionFond.x = 0;
 positionFond.y = 0;

 SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
}

void input_handle(void)
{
 SDL_Event event;

 while (SDL_PollEvent(&event)) {

  switch (event.type) {
  case SDL_QUIT:
   SDL_Quit();
   exit(0);
   break;
  }
 }
}

void anime(void)
{
 SDL_Rect positionFond, positionZozor, positionIcone;
 SDL_Rect positionFond_c;
 SDL_Rect oldpositionZozor, oldpositionIcone;
 SDL_Rect clipper;

 int avanceX = 1, avanceY = 1;
 int avanceXI = 0, avanceYI = 1;

 positionFond.x = 0;
 positionFond.y = 0;

 positionZozor.x = 0;
 positionZozor.y = 0;

 positionIcone.x = ecran->w - icone->w;
 positionIcone.y = 0;

 oldpositionZozor.x = 0;
 oldpositionZozor.y = 0;

 oldpositionIcone.x = ecran->w - icone->w;
 oldpositionIcone.y = 0;

 while (1) {

  clipper.x = oldpositionZozor.x;
  clipper.y = oldpositionZozor.y;
  clipper.h = zozor->h;
  clipper.w = zozor->w;
  SDL_SetClipRect(ecran,&clipper); // Clipper sur zozor

  positionFond_c.x = positionFond.x;
  positionFond_c.y = positionFond.y;

  SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond_c); //
Effacer zozor

  clipper.x = oldpositionIcone.x;
  clipper.y = oldpositionIcone.y;
  clipper.h = icone->h;
  clipper.w = icone->w;
  SDL_SetClipRect(ecran,&clipper); // Clipper sur la planète

  positionFond_c.x = positionFond.x;
  positionFond_c.y = positionFond.y;
```

```
    SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond_c); //
Effacer la planète.

    SDL_SetClipRect(ecran,NULL); // Enlever le clipper

    SDL_BlitSurface(zozor, NULL, ecran, &positionZozor);

    SDL_BlitSurface(icone, NULL, ecran, &positionIcone);

    oldpositionZozor.x = positionZozor.x;
    oldpositionZozor.y = positionZozor.y;

    oldpositionIcone.x = positionIcone.x;
    oldpositionIcone.y = positionIcone.y;


    if (avanceX) {
     positionZozor.x++;
    } else {
     positionZozor.x--;
    }

    if (avanceY) {
     positionZozor.y++;
    } else {
     positionZozor.y--;
    }

    if (positionZozor.x == ecran->w - zozor->w - 1)
     avanceX = 0;
    else if (positionZozor.x == 0)
     avanceX = 1;

    if (positionZozor.y == ecran->h - zozor->h - 1)
     avanceY = 0;
    else if (positionZozor.y == 0)
     avanceY = 1;

    if (avanceXI) {
     positionIcone.x++;
    } else {
     positionIcone.x--;
    }
    if (avanceYI) {
     positionIcone.y++;
    } else {
     positionIcone.y--;
    }

    if (positionIcone.x == ecran->w - icone->w - 1)
     avanceXI = 0;
    else if (positionIcone.x == 0)
     avanceXI = 1;

    if (positionIcone.y == ecran->h - icone->h - 1)
     avanceYI = 0;
    else if (positionIcone.y == 0)
     avanceYI = 1;

    SDL_Flip(ecran);
    input_handle();
    //SDL_Delay(10);
   }
  }

  int main(void)
  {
   init();
   anime();
```

```
 return 0;
}
```

Ou encore un code (donné par Mircko) pour gérer l'affichage de plusieurs zozor :

Secret (cliquez pour afficher)

Code : C

```
#include <stdio.h>
#include <time.h>
#include <SDL/SDL.h>
#define LARGEUR 800
#define HAUTEUR 600

SDL_Surface *ecran, *imageDeFond, *zozor;

typedef struct{
    int vx;
    int vy;
    SDL_Rect pos;
    SDL_Rect oldPos;
} Objet ;

void init(void)
{
 SDL_Init(SDL_INIT_VIDEO);
        srand(time(NULL));
 ecran = SDL_SetVideoMode(LARGEUR, HAUTEUR, 32, SDL_HWSURFACE);
 imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
 zozor = SDL_LoadBMP("zozor.bmp");

 SDL_Rect positionFond;
 positionFond.x = 0;
 positionFond.y = 0;

 SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
}

void input_handle(void)
{
 SDL_Event event;

 while (SDL_PollEvent(&event)) {

  switch (event.type) {
  case SDL_QUIT:
   SDL_Quit();
   exit(0);
   break;
  }
 }
}

void init_objets(Objet objs[], int nbObjs)
{
    int i;

    for(i = 0; i < nbObjs; i++)
    {
        objs[i].vx = rand() % 4 + 1;
        objs[i].vy = rand() % 4 + 1;
        objs[i].pos.x = rand() % (LARGEUR - zozor->w);
        objs[i].pos.y = rand() % (HAUTEUR - zozor->h);
        objs[i].oldPos.x = objs[i].pos.x;
        objs[i].oldPos.y = objs[i].pos.y;
    }
```

```
}

void anime(int nbZozors)
{
 int i;
    Objet objs[nbZozors];
    SDL_Rect positionFond = {0, 0, 0, 0},
             positionFond_c = {0, 0, 0, 0},
           clipper;

    init_objets(objs, nbZozors);

 while (1) {
        for(i = 0; i < nbZozors; i++)
        {
            clipper.x = objs[i].oldPos.x;
            clipper.y = objs[i].oldPos.y;
            clipper.h = zozor->h;
            clipper.w = zozor->w;

            SDL_SetClipRect(ecran, &clipper); // Clipper sur
l'objet[i]

            positionFond_c.x = positionFond.x;
            positionFond_c.y = positionFond.y;

            SDL_BlitSurface(imageDeFond, NULL, ecran,
&positionFond_c); // Effacer l'objet[i]
        }

  SDL_SetClipRect(ecran,NULL); // Enlever le clipper
        for(i = 0; i < nbZozors; i++)
        {
            SDL_BlitSurface(zozor, NULL, ecran, &objs[i].pos);
            objs[i].oldPos.x = objs[i].pos.x;
            objs[i].oldPos.y = objs[i].pos.y;
        }

        for(i = 0; i < nbZozors; i++)
        {
            objs[i].pos.x += objs[i].vx;
            objs[i].pos.y += objs[i].vy;
            if(objs[i].pos.x >= ecran->w - zozor->w - 1)
            {
                objs[i].pos.x = ecran->w - zozor->w - 1;
                objs[i].vx = -objs[i].vx;
            }
            else if(objs[i].pos.x <= 0)
            {
                objs[i].pos.x = 0;
                objs[i].vx = -objs[i].vx;
            }
            if(objs[i].pos.y >= ecran->h - zozor->h - 1)
            {
                objs[i].pos.y = ecran->h - zozor->h - 1;
                objs[i].vy = -objs[i].vy;
            }
            else if(objs[i].pos.y <= 0)
            {
                objs[i].pos.y = 0;
                objs[i].vy = -objs[i].vy;
            }
        }

  SDL_Flip(ecran);
  input_handle();
  //SDL_Delay(10);
 }
}
```

```
int main(int argc, char *argv[])
{
 init();
 anime(8);
 return 0;
}
```

Le tuto est fini !!
Avec ces deux techniques, les animations deviennent beaucoup plus fluides.
Si vous avez d'autres idées d'optimisation, vous pouvez toujours les présenter dans les commentaires.