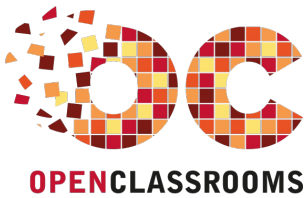
Faire des animations avec SDLP_Anim

Par pOpOp99O0



www.openclassrooms.com

Dernière mise à jour le 17/11/2012

Sommaire

Faire des animations avec SDLP_Anim

Par pOpOp99O0

Mise à jour : 17/11/2012
Difficulté : Facile Durée d'étude : 20 minutes

Bienvenue dans mon tutoriel !

Ce dernier a pour but de vous apprendre à gérer des animations simplement grâce à un outil dont je suis l'auteur : `SDLP_Anim`.
Les utilisateurs de Dev-C++ auront une petite surprise dans ce tutoriel (ils ont quand même de la chance ceux-là) ! Les autres, eh bien, tant pis pour eux, na !
Pour utiliser cet outil, vous aurez besoin d'avoir lu tout le tutoriel de M@teo21 sur le C/C++ (y compris la partie sur SDL).
Sommaire du tutoriel :

Téléchargement

La première chose à faire est de télécharger le fichier ZIP qui contient le *header* pour que nos fonctions fonctionnent.
C'est ici : SDLP_Anim.zip !
Version actuelle de SDLP_Anim.h : **2.0 (2012)**.

Il se peut qu'une nouvelle version sorte et que vous ne le remarquiez pas. Dans ce cas, de nouvelles fonctions ajoutées dans ce tutoriel ne fonctionneront pas ! Il faudra alors remplacer l'ancien fichier par le nouveau qui est en téléchargement à côté de SDLP_Anim.zip.

Dès la version 2.0, la méthode de la bibliothèque change et nous passerons désormais avec la méthode standard dites "Static Library" soit l'installation d'un fichier .a et .h

Windows

Code::Blocks

Ouvrez le fichier compressé que vous avez téléchargé à l'aide de WinZip ou WinRar selon ce que vous avez.
Commencez par prendre en compte le fichier de license (GNU GPL). Je sais que vous ne le lirez pas mais par principe, il est là.
Nous pouvons voir également 3 images différentes, conservez les car nous les utiliserons toutes pendant ce tutoriel.
Finalement il ne nous reste plus que les fichiers qui vous permettront de faire des animations !

Plaçons d'abord le fichier d'en tête `SDLP_Anim.h` !
Vous allez placer ce fichier comme suit :
`..\mingw32\include\SDLP_Anim.h`
Ensuite, prenons notre deuxième fichier d'extension .a pour le mettre comme suit :
`..\mingw32\lib\libSDLP_Anim.a`

Et c'est tout !

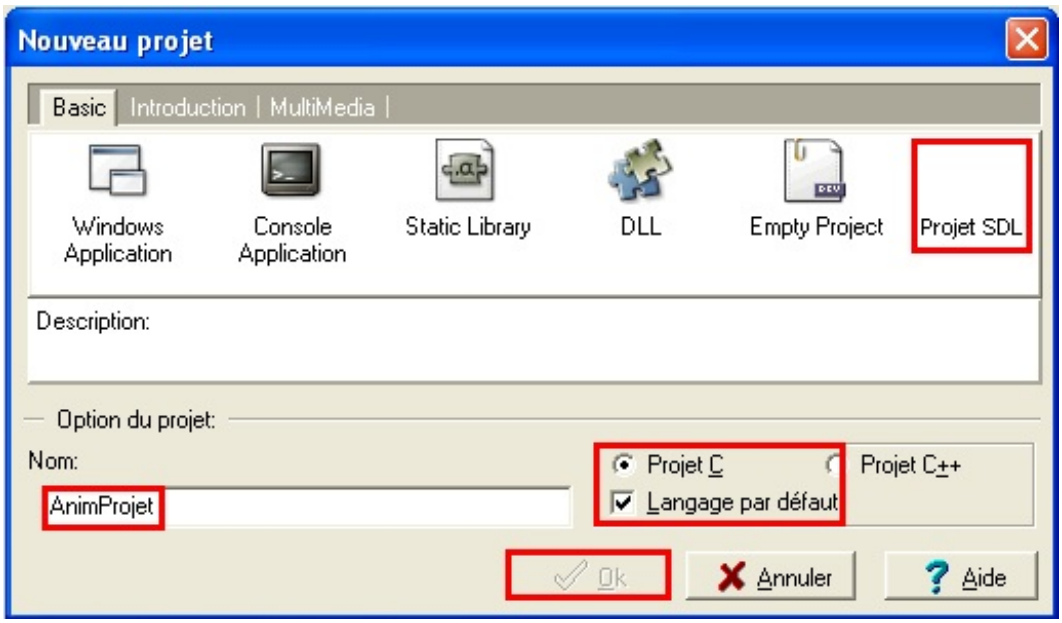
Dev-C++

La manipulation d'installation avec Dev-C++ est la même que celle de Code::Block. Vous devez placer vos deux fichiers dans le dossier mingw32 comme indiqué précédemment.

J'ai créé un template compatible avec Dev-C++ qui vous permettra de créer des projets SDL rapidement presque sans configuration !
Le fichier ZIP est disponible ici.
Décompressez le fichier et insérez le tout dans le dossier `Templates`.
Pour actionner le bouton, cliquez sur `Nouveau projet`. Puis, vous aurez le choix entre plusieurs types de programme comme la console. Cherchez `Projet SDL` puis cliquez dessus. Un code en SDL typique vient d'apparaître. Et ce n'est pas tout ! Avant, vous deviez mettre dans l'éditeur de lien -lmingw32, -lSDL, etc. Eh bien là, c'est déjà fait !

L'éditeur de liens contient ces fichiers: -lmingw32 -lSDLmain -lSDL -lSDL_image -lFMOD -lSDL_ttf ../../mingw32/lib/sdlgfx.lib.
Il se peut que vous n'ayez pas tout installé ; retirez les fichiers que vous n'avez pas installés.

Après l'installation de l'astuce, voilà ce que ça donne :



Bon certes, il n'y a pas d'icône mais c'est très pratique car on n'a pas besoin d'inclure à la main -lmingw32 -lSDLmain et surtout d'en rappeler.

Visual C++ 2010 Express

Il n'y a pas grand-chose qui change si ce n'est que nous n'utilisons plus *mingw32* mais *VC*.
Mettez le *header* dans : `..\Microsoft Visual Studio 10\VC\include\SDLP_Anim.h`
Et le fichier *.a* dans `..\Microsoft Visual Studio 10\VC\lib\libSDLP_Anim.a`

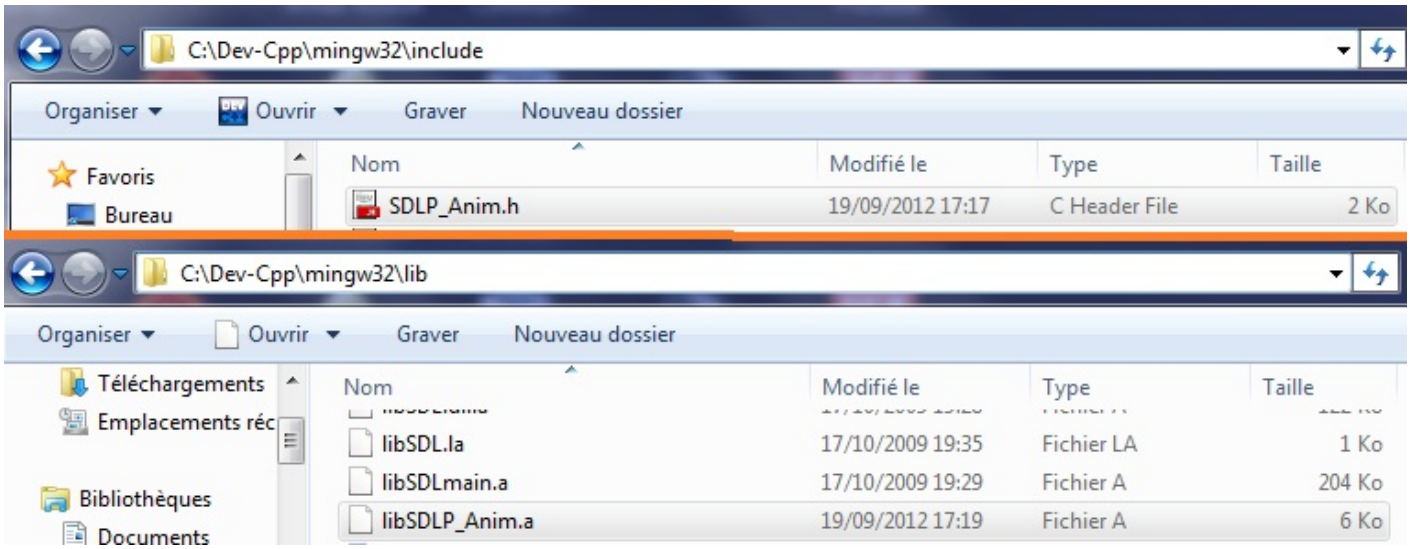
Je ne connais pas bien cet IDE. Si un problème survient chez un bon nombre de Zéros, merci de me contacter par **MP**.

En général

Maintenant, nous allons configurer notre IDE favori.
Pour se faire, il suffit de créer un projet, et comme avec M@théo21, de "linker" notre fichier `.a` !
Allez dans les configurations de votre projet, et à l'endroit correspondant où l'on rentre les fichiers .lib et .a.
Vous pouvez maintenant ajouter le fichier .a dans cette liste.
Ou copiez-collez cette ligne permettant de configurer les bibliothèques pour la SDL, SDL_image et SDLP_Anim :
`-lmingw32 -lSDLP_Anim -lSDLmain -lSDL -lSDL_image`
Enregistrez et le tour est joué !

Il est important de placer `-lSDLP_Anim` avant `-lSDLmain` et le reste

Vous aurez ceci après vos manipulations (j'ai utilisé Dev-C++ mais ça devrait être presque pareil pour les autres) :



Exemple de placement des fichiers .a et .h pour mingw32

Mac OS (Xcode) & Linux

Je ne sais pas comment vous aider si vous êtes dans ce cas.

Animation normale

Nous allons maintenant commencer les choses sérieuses !

Je vous demande de créer un nouveau projet et de le compléter tout au long du chapitre. Normalement, votre fichier `.c` est vide en ce moment.

Accrochez-vous, ça va secouer !

Les fonctions, nous en raffolons

Prenons un code SDL de base (ayant inclus SDL_image). Recopiez-le dans votre fichier `.c`.

Code : C

```c
#include <stdlib.h>
#include <stdio.h>
#include <SDL.h>
#include <SDL_image.h>

int main(int argc, char *argv[])
{
```

```
    SDL_Surface *ecran = NULL;
    SDL_Event event;
    int continuer = 1;

    SDL_Init(SDL_INIT_VIDEO);

    ecran = SDL_SetVideoMode(200, 200, 32, SDL_HWSURFACE);
    SDL_WM_SetCaption("Animations en SDL !", NULL);

    while (continuer)
    {
        SDL_WaitEvent(&event);
        switch(event.type)
        {
            case SDL_QUIT:
                continuer = 0;
            break;
        }

    SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));


    SDL_Flip(ecran);
    }

    SDL_Quit();

    return EXIT_SUCCESS;
}
```

On va partir du tout début pour que vous n'oubliiez rien. Complétez votre fichier `.c` à l'aide de toutes les fonctions présentes.

Code : C

```
ecran = SDL_SetVideoMode(200, 200, 32, SDL_HWSURFACE);
```

Vous avez tous remarqué que si, avec ce code, on *blitte* une image, elle va scintiller (si vous doutez, essayez) !
On va régler tout ça en mettant en place le double *buffering*.

Code : C

```
ecran = SDL_SetVideoMode(200, 200, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);
```

Code : C

```
SDL_WaitEvent(&event);
```

Nous allons mettre SDL_PollEvent car une animation n'a pas besoin d'événement pour se mettre à fonctionner.

Code : C

```
SDL_PollEvent(&event);
```

Heu... Tu n'aurais pas oublié le *header* par hasard ?

Bien vu.
Dans l'exemple du tutoriel, on trouve ceci dans l'en-tête du fichier `.c`.

Code : C

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDLP_Anim.h>
```

Vous n'allez devoir retenir qu'**une** seule structure. Voici un exemple de son utilisation.

Code : C

```
SDLP_Anim *Anim;
```

On a créé notre surface `Anim`. Ça serait bien de libérer sa mémoire, non ?
Pour cela, on va utiliser `SDLP_FreeAnim` !
Placez-le juste avant le `SDL_Quit();`.

Code : C

```
SDLP_FreeAnim(Anim);
```

Et le tour est joué !

Maintenant, nous allons charger l'animation. Reprenez les images qui étaient dans le dossier compressé. Nous allons utiliser `Feu.bmp`. Je vais vous présenter le prototype de la fonction de chargement d'animation.

Mettez l'image dans le dossier de l'exécutable. Sinon, elle ne pourra pas être chargée avec l'exemple qui va suivre.

Code : C

```
SDLP_Anim* SDLP_LoadAnim(char *SurfaceAnim, int nbPicture, int
coter, long temps, int Loop, int Actif, int Util);
```

Vous allez définir **toutes** les options de l'animation :

- `char *SurfaceAnim` : entrez ici le chemin de l'exécutable à l'image ;
- `int nbPicture` : nombre d'images à séparer dans celle qui a été déclarée plus haut (l'image sera coupé en parts égales) ;
- `int coter` : définir ici le sens de défilement. Utilisez les *defines* `SDLP_GAUCHE`, `SDLP_DROITE`, `SDLP_HAUT` ou `SDLP_BAS`. Si je prends SDLP_HAUT par exemple, l'animation sera coupée et animée de l'image tout en bas vers l'image tout en haut (on monte) ;
- `long temps` : définir ici le temps entre chaque image (en millisecondes) ;
- `int Loop` : choisir ici le nombre de fois où l'image va être animée. Utilisez `SDLP_INFINI` ou un nombre supérieur ou égal à 1. ;
- `int Actif` : définir ici l'état de l'animation en utilisant `SDLP_ARRETER`, `SDLP_PAUSE` ou `SDLP_LANCER` ;
- `int Util` : stipulez ici si l'animation doit se jouer seule ou à l'appui sur une touche. Utilisez les *defines* `SDLP_NORMAL`

ou SDLP_TOUCHE.
- N'oubliez pas que cette fonction retourne vos configurations vers une variable de type SDLP_Anim.

Nous verrons comment utiliser l'animation par événement plus tard.

Voici les détails des différents états dont peuvent être assignés l'animation :

SDLP_ARRETER	Votre animation n'apparait pas/disparait. Celle-ci s'arrête et reviens à la 1ère image découpée. Si le nombre de boucles n'est pas infini, celui-ci revient au début.
SDLP_PAUSE	Votre animation reste présente mais se fige sur l'instant où elle était. Il n'y a pas d'écarts de temps et quand l'animation reprend, elle reprend là où elle s'était arrêtée. Le nombre de boucles est conservée.
SDLP_LANCER	Permet de lancer l'animation depuis n'importe quelle état.

Code : C

```
Anim=SDLP_LoadAnim("Feu.bmp", 4, SDLP_DROITE, 1000, SDLP_INFINI,
SDLP_LANCER, SDLP_NORMAL);
```

Ça va ? Vous suivez ? N'hésitez pas à relire si cela s'avère nécessaire !
Ça devient intéressant, non ?

Maintenant, il faut définir la position de l'animation dans la fenêtre.
Déclarez une variable du type SDL_Rect ! Faites comme moi, appelez-la pAnim.

Code : C

```
SDL_Rect pAnim;
/* Chargement de l'animation. */
pAnim.x=0;
pAnim.y=0;
```

Nous avons chargé notre animation. Mais ce n'est que dans la « tête » de l'ordinateur ça. Nous, on veut que ça s'affiche, non ?
NON ?
Nous allons utiliser la fonction de *blittage* la plus simple qui n'ait jamais existé.
Son prototype l'est aussi :

Code : C

```
void SDLP_BlitAnim(SDLP_Anim *str, SDL_Surface *ecran, SDL_Rect
pos);
```

- SDLP_Anim *str : entrez ici votre variable d'animation ;
- SDL_Surface *ecran : surface qui gère la fenêtre, souvent appelée *ecran* ou *screen* ;
- SDL_Rect pos : inscrivez ici votre variable de position.

Mais si je charge une animation avec comme état SDLP_ARRETER, l'animation n'apparaîtra jamais ?!

Qu'ouïe-je ? Effectivement, il faut bien une fonction pour changer l'état de notre animation !
Là voici avec son prototype :

Code : C

```
void SDLP_SetEtatAnim(SDLP_Anim *str, int Etat);
```

- `SDLP_Anim *str` : entrez ici votre variable d'animation ;
- `int Etat` : Entrez votre variable d'état (voir ci-dessus, il y en a trois seulement).

Il serait utile de savoir dans quel état est l'animation, voici la fonction qui permet de vous renseigner sur cela :

Code : C

```
int SDLP_GetEtatAnim(SDLP_Anim *str);
```

- `SDLP_Anim *str` : entrez ici votre variable d'animation ;
- Renvoi une valeur de type INT. Comparez les comme ceci :

  Code : C

  ```
  if (SDLP_GetEtatAnim(Anim)==SDLP_LANCER)
  ```

Voilà ! Nous avons nos fonctions maintenant.
Maintenant que vous savez faire des animations, nous allons mettre vos talents en pratique !

Fonctions annexes

Ne grillez pas cette partie ! Plusieurs fonctions vous seront bien utiles (indispensables) pour gérer vos animations correctement.
Imaginons-nous un problème. Vous savez comme moi que la largeur/hauteur d'une surface en SDL s'écrit de cette facon :

Code : C

```
SDL_Surface *Surface = NULL;
Surface->w // Largeur
Surface->h // Hauteur
```

Cependant, nous ne connaissons pas la largeur/hauteur de notre image d'animation.
Il existe trois fonctions différentes pour répondre à ce problème :

Code : C

```
void SDLP_GetWHSurfaceAnim(SDLP_Anim *str, int *w, int *h);
```

- `SDLP_Anim *str` : Entrez ici votre variable d'animation ;
- `int *w` : Entrez votre pointeur qui retournera la largeur ;
- `int *h` : Entrez votre pointeur qui retournera la hauteur.

Cette fonction est utile si vous devez capter ces variables pour les traiter.

Code : C

```
int SDLP_GetWSurfaceAnim(SDLP_Anim *str);
int SDLP_GetHSurfaceAnim(SDLP_Anim *str);
```

- `SDLP_Anim *str` : Entrez ici votre variable d'animation.

Ces fonctions sont très utiles puisqu'elles permettent de récupérer la largeur/hauteur.
Voici un exemple de leur utilisation :

Code : C

```
SDL_Rect p;
SDLP_Anim *Anim;
p.x = SDLP_GetWSurfaceAnim(Anim)/2;
p.y = SDLP_GetHSurfaceAnim(Anim)/2;
```

Ces fonctions retournent la largeur/hauteur non pas de l'image initiale mais de **l'image découpée**

Imaginons-nous un autre problème à présent : Nous voudrions qu'au feu vert (pour reprendre l'exemple précédent), des voitures quelconques puissent passer, et pas au rouge, ni au orange. Il faudrait pouvoir savoir à quelle instance en est l'animation. Les fonctions suivantes vont répondre à notre problème :

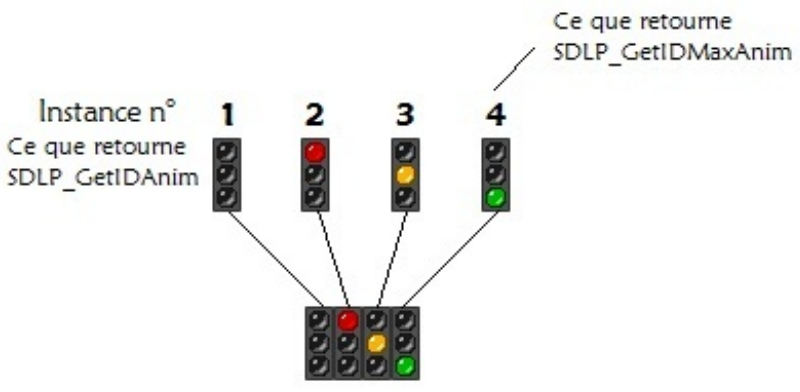
Code : C

```
int SDLP_GetIDAnim(SDLP_Anim *str);
int SDLP_GetIDMaxAnim(SDLP_Anim *str);
```

- `SDLP_Anim *str` : Entrez ici votre variable d'animation.

`SDLP_GetIDAnim` Retourne l'instance à laquelle est l'animation, alors que `SDLP_GetIDMaxAnim` retourne la valeur de la dernière instance existante (qui est égale au nombre de découpage).

Voici une image pour schématiser tout ça, à partir de notre feu habituel :



La valeur de la première instance est **1** et non 0

J'espère que ca sera plus clair pour vous avec cette image.

Encore et toujours des problèmes ! Je voudrais bien que mon animation subisse la gestion de la transparence ! Tout d'abord, n'oubliez pas qu'il faut que votre image soit du format **.bmp**.
Voici la fonction qui va vous sauver. N'oubliez pas de l'appeler après la déclaration de l'animation.

Code : C

```
void SDLP_SetColorKeyAnim(SDLP_Anim *str, int R, int V, int B);
```

- `SDLP_Anim *str` : Entrez ici votre variable d'animation ;
- `int R` : Entrez ici la valeur de la couleur rouge sur laquelle il faudra pratiquer la transparence ;
- `int V` : Entrez ici la valeur de la couleur verte sur laquelle il faudra pratiquer la transparence ;
- `int B` : Entrez ici la valeur de la couleur bleue sur laquelle il faudra pratiquer la transparence.

C'est l'accouplement des trois couleurs qui formera la couleur finale sur laquelle la transparence doit s'effectuer.

Dernière fonction qui peut être utile. Lorsque vous chargez une animation, vous indiquez le temps de défilement entre chaque image. Le programme va vérifier si il doit blitter l'image correspondante. Pour éviter une surcharge du CPU, la fonction `SDL_Delay` est utilisée et est de valeur par défaut à **5ms**. Comme tout le monde possède un ordinateur à peu près potable de notre temps, 5ms suffisent largement à ne pas tout faire ramer. Il existe cependant une fonction pour modifier ce temps de "Delay" :

Code : C

```
void SDLP_SetWaitAnim(SDLP_Anim *str, int t);
```

- `SDLP_Anim *str` : Entrez ici votre variable d'animation ;
- `int t` : Entrez ici le temps en millisecondes d'attente entre chaque boucle.

Exercice

Maintenant, au boulot !
Je vous demande, à partir de `Feu.bmp`, de me faire un joli feu tricolore qui s'anime toutes les secondes, de sens DROITE et de durée infinie. Le feu tricolore doit être au centre de la fenêtre. De plus, je veux que quand on appuie sur la touche "entrée", l'animation se mette en pause et reprenne selon l'état.
Vous êtes fins prêts, c'est parti !

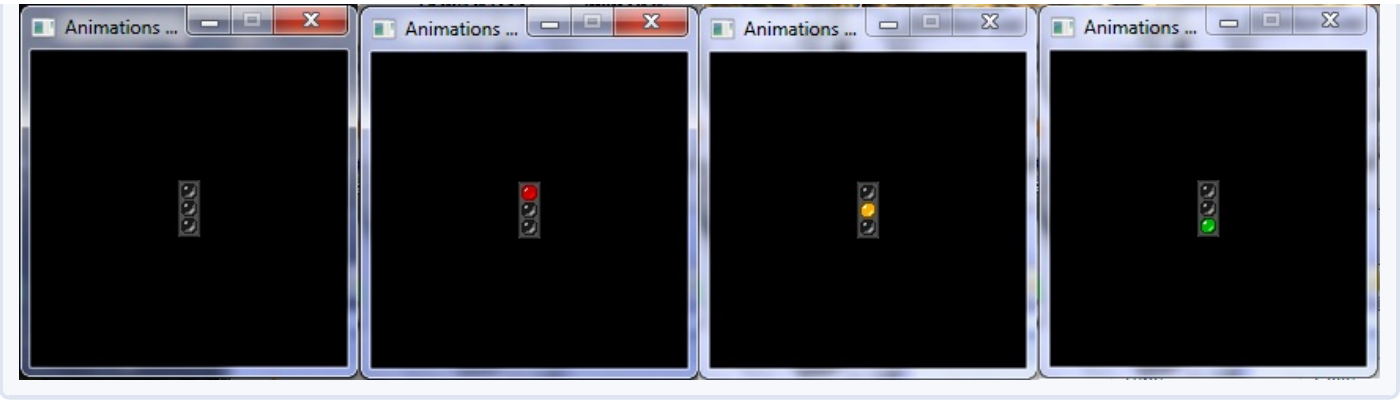
Correction

Secret (cliquez pour afficher)

Code : C

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDLP_Anim.h>
```

```c
/* SDLP_ANIM.C
Créé par Syndrome5 (POPOP9900)
Site du Zér0
*/

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;
    SDL_Rect pAnim;
    SDLP_Anim *Anim = NULL;
    SDL_Event event;
    int continuer = 1, NoRepeat=0;

    SDL_Init(SDL_INIT_VIDEO);

    ecran = SDL_SetVideoMode(200, 200, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);
    SDL_WM_SetCaption("Animations en SDL !", NULL);

    Anim=SDLP_LoadAnim("Feu.bmp", 4, SDLP_DROITE, 1000,
SDLP_INFINI, SDLP_LANCER, SDLP_NORMAL);

    pAnim.x=ecran->w / 2 - SDLP_GetWSurfaceAnim(Anim) / 2;
    pAnim.y=ecran->h / 2 - SDLP_GetHSurfaceAnim(Anim) / 2;

    while (continuer)
    {
        SDL_PollEvent(&event);
        switch(event.type)
        {
        case SDL_QUIT:
            continuer = 0;
            break;
        case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
            case SDLK_RETURN:
                if (NoRepeat == 0)
                {
                    if (SDLP_GetEtatAnim(Anim)==SDLP_ARRETER ||
SDLP_GetEtatAnim(Anim)==SDLP_PAUSE)
                        SDLP_SetEtatAnim(Anim, SDLP_LANCER);
                    else
                        SDLP_SetEtatAnim(Anim, SDLP_PAUSE);
                    NoRepeat=1;
                }
                break;
            }
            break;
        case SDL_KEYUP:
            NoRepeat=0;
            break;
        }

        SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 0, 0,
0));
        SDLP_BlitAnim(Anim, ecran, pAnim);
        SDL_Flip(ecran);
    }

    SDLP_FreeAnim(Anim);

    SDL_Quit();

    return EXIT_SUCCESS;
}
```



Télécharger le projet.

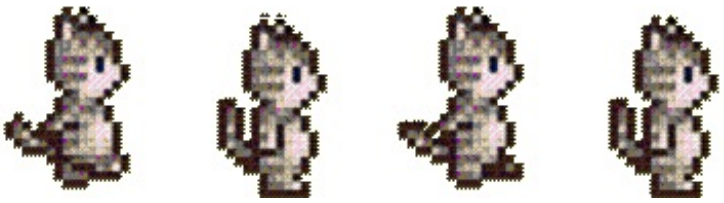
Notez que pour éviter la répétition de l'action de la touche en appuyant qu'une seule fois, j'ai fais ce petit système de boolean avec la variable `NoRepeat`.

Ce chapitre se termine ! Nous avons vu une forme d'animation automatisée par le temps.
Dans le prochain chapitre, nous verrons un nouveau type d'animation : l'animation par évènement !

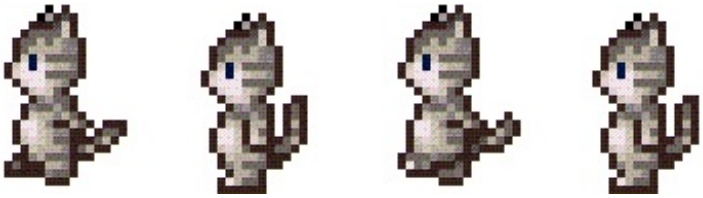
Animation par événement

Faisons courir le chaton

Nous allons à présent utiliser les deux images qui restaient dans `SDLP_Anim.zip` :



et



On va animer ce chat quand on appuiera **sur une touche**.
C'est-à-dire que, quand on appuiera sur une touche, l'image s'animera. Je prends l'exemple de Zozor qui se déplace dans le tutoriel sur les événements de M@teo21. Il est statique quand il se déplace, l'image pourrait bouger et cela donnerait « vie » à l'image.
C'est ce que j'appelle une **animation par événement**.

Mais j'y pense, on peut combiner plusieurs `SDLP_SetEtatAnim` pour gérer nous-même ces évènements !

Bravo si vous y avez pensé tout seul. Il est en effet possible par le biais de cette fonction de gérer des animations avec les évènements. Mais imaginons que vous ayez besoin ne serait-ce que cinq animations de ce type, imaginez le nombre de ligne de code pour ne gérer que l'affichage de ce dernier !

Pour pallier à ce problème, nous avons intégrer un système d'animation par évènement très simple d'utilisation et prenant peu d'espace (dans l'éditeur de texte, mais aussi dans votre mémoire).

Vous vous souvenez dans le chapitre précédent, nous déclarions nos animations de cette façon :

Code : C

```
Anim=SDLP_LoadAnim("Feu.bmp", 4, SDLP_DROITE, 1000, SDLP_INFINI,
SDLP_LANCER, SDLP_NORMAL);
```

Vous connaissez chaque paramètre de la fonction, mais nous sommes passé très rapidement sur le dernier sans le détailler. Sachez que c'est d'abord ici que vous allez dire à votre programme "`Je veux une animation par évènement`".

Pour se faire, c'est très simple. Nous allons remplacer `SDLP_NORMAL` par `SDLP_TOUCHE`. Ceci fait, vous deviez avoir cette ligne :

Code : C

```
Anim=SDLP_LoadAnim("Feu.bmp", 4, SDLP_DROITE, 1000, SDLP_INFINI,
SDLP_LANCER, SDLP_TOUCHE);
```

Si vous compilez maintenant votre code, à votre grande surprise, l'image s'affiche à l'instance 0 (première image) mais ne défile plus. Rien de plus normal ! L'animation attend maintenant un évènement pour se lancer.

Revenons à présent sur notre exemple du chat souhaitant courir. 
Pour nous simplifier la vie, nous ne prendrons pour le moment que l'image du chat qui regarde vers la droite. De plus, on modifie la déclaration de l'animation pour indiquer que l'on va utiliser la gestion par évènement.

Code : C

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDLP_Anim.h>

/* SDLP_ANIM.C
Créé par Syndrome5 (POPOP9900)
Site du Zér0
*/

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;
    SDL_Rect pAnim;
    SDLP_Anim *Anim = NULL;
    SDL_Event event;
    int continuer = 1;

    SDL_Init(SDL_INIT_VIDEO);

    ecran = SDL_SetVideoMode(400, 400, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);
    SDL_WM_SetCaption("Animations en SDL !", NULL);

    Anim=SDLP_LoadAnim("CatD.png", 4, SDLP_GAUCHE, 200, SDLP_INFINI,
SDLP_LANCER, SDLP_TOUCHE);

    pAnim.x=0;
    pAnim.y=ecran->h - SDLP_GetHSurfaceAnim(Anim);

    while (continuer)
```

```
        {
            SDL_PollEvent(&event);
            switch(event.type)
            {
                case SDL_QUIT:
                    continuer = 0;
                break;
                case SDL_KEYDOWN:
                switch (event.key.keysym.sym)
                {
                }
                break;
            }

            SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,
    255, 255));
            SDLP_BlitAnim(Anim, ecran, pAnim);
            SDL_Flip(ecran);
        }

        SDLP_FreeAnim(Anim);

        SDL_Quit();

        return EXIT_SUCCESS;
    }
```

Vous pouvez également compiler ce code, qui ne donnera rien qu'une image fixe !
Nous allons par contre reprendre cet exemple tout au long du chapitre. Je vous conseille donc de copier/coller ce bout de code.

Bon, comment on l'anime cette image finalement ?!

Je vois que vous commencez à vous vous impatienter. 
Alors on est parti !

À l'aide d'une fonction appelée `SDLP_Touch`, vous allez pouvoir gérer l'événement sans rien faire ; la fonction fait tout. 

Code : C

```
    void SDLP_Touch(SDL_Event event, SDLP_Anim *str);
```

- `SDL_Event event` : variable de type `SDL_Event` qui gère tous les événements ;
- `SDLP_Anim *str` : variable de type `SDLP_Anim`.

Voilà, il ne reste plus qu'à placer la fonction et le tour est joué !
Puisque cette fonction agit en fonction des évènements, vous devrez la placer **après** le `switch event.type` comme ceci :

Code : C

```
    switch (event.type)
       {
          /* Événements */
       }
      SDLP_Touch(event, Anim);
```

Résultat : Lorsqu'on reste appuyé sur n'importe quelle touche, le chat s'anime bien !
Faisons maintenant en sorte que lorsqu'on appuie sur les touches directionnelles gauche et droite, il se déplace dans le sens correspondant à la touche !

Je suis gentil, alors je donne ce petit bout de code misérable.

Code : C

```
case SDL_KEYDOWN:
switch (event.key.keysym.sym)
{
    case SDLK_RIGHT:
        pAnim.x+=2;
    break;
    case SDLK_LEFT:
        pAnim.x-=2;
    break;
    default:
    break;
}
break;
```

Ajoutez ceci à votre code. Vous pouvez maintenant observer que notre chat se déplace bien en effectuant son animation et, quand on relâche la touche, l'animation revient au départ. On peut même voir que notre chat fait du moonwalk ! 

 C'est bien gentil tout ça, mais je vois qu'il y a un problème, quand j'appuie sur une autre touche que gauche ou droite, l'animation se met en route quand même !

Vous avez de l'œil dites moi aujourd'hui ! Heureusement, une fonction va vous sauver :

Code : C

```
void SDLP_BlockTouch(SDLP_Anim *str);
```

- `SDLP_Anim *str` : variable de type SDLP_Anim.

 Mais je la place où cette fonction moi ?

A chaque endroit où vous ne voulez pas que l'animation se produise.
En l'occurrence, ce sera souvent dans votre `default:` puisque l'on choisis la réalisation de l'évènement sur quelques touches uniquement. On exclut donc toutes les touches sauf celles précédés du `default:` !

Finalement, on obtient ceci :

Code : C

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDLP_Anim.h>

/* SDLP_ANIM.C
Créé par Syndrome5 (POPOP9900)
Site du Zér0
*/
```

```
int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;
    SDL_Rect pAnim;
    SDLP_Anim *Anim = NULL;
    SDL_Event event;
    int continuer = 1;

    SDL_Init(SDL_INIT_VIDEO);

    ecran = SDL_SetVideoMode(400, 400, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);
    SDL_WM_SetCaption("Animations en SDL !", NULL);

    Anim=SDLP_LoadAnim("CatD.png", 4, SDLP_GAUCHE, 150, SDLP_INFINI,
SDLP_LANCER, SDLP_TOUCHE);

    pAnim.x=0;
    pAnim.y=ecran->h - SDLP_GetHSurfaceAnim(Anim);

    while (continuer)
    {
        SDL_PollEvent(&event);
        switch(event.type)
        {
            case SDL_QUIT:
                continuer = 0;
            break;
            case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
                case SDLK_RIGHT:
                    pAnim.x+=2;
                break;
                case SDLK_LEFT:
                    pAnim.x-=2;
                break;
                default:
                    SDLP_BlockTouch(Anim);
                break;
            }
            break;
        }
        SDLP_Touch(event, Anim);

        SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,
255, 255));
        SDLP_BlitAnim(Anim, ecran, pAnim);
        SDL_Flip(ecran);
    }

    SDLP_FreeAnim(Anim);

    SDL_Quit();

    return EXIT_SUCCESS;
}
```

Bon c'est bien beau tout ça, mais est-ce vraiment réaliste que notre chat fasse le moonwalk ?
Ceux qui me réponde "oui", je leur conseille d'arrêter le cannabis tout de suite. 
Et comme moi j'ai assez travaillé pour aujourd'hui, c'est vous qui allez me faire ce chat qui marche normalement.

Exercice 1 : terminer le programme

Il faut penser à utiliser la deuxième image à votre disposition.

Vous pouvez changer le code comme vous voulez et même tout recommencer ; c'est ce que je vous conseille d'ailleurs ! À vous de voir. Au moins, vous aurez le mérite d'avoir tout fait tout seul.
A vos curseurs ! Vous avez tout pour y arriver, alors faites-moi ce chat, et plus vite que ça !

Correction

Voilà la correction de mon casse-tête.
Vous n'étiez pas obligés de faire exactement la même chose.
Si vous ne comprenez pas mon code, relisez le chapitre avec plus de concentration.

Secret (cliquez pour afficher)

Code : C

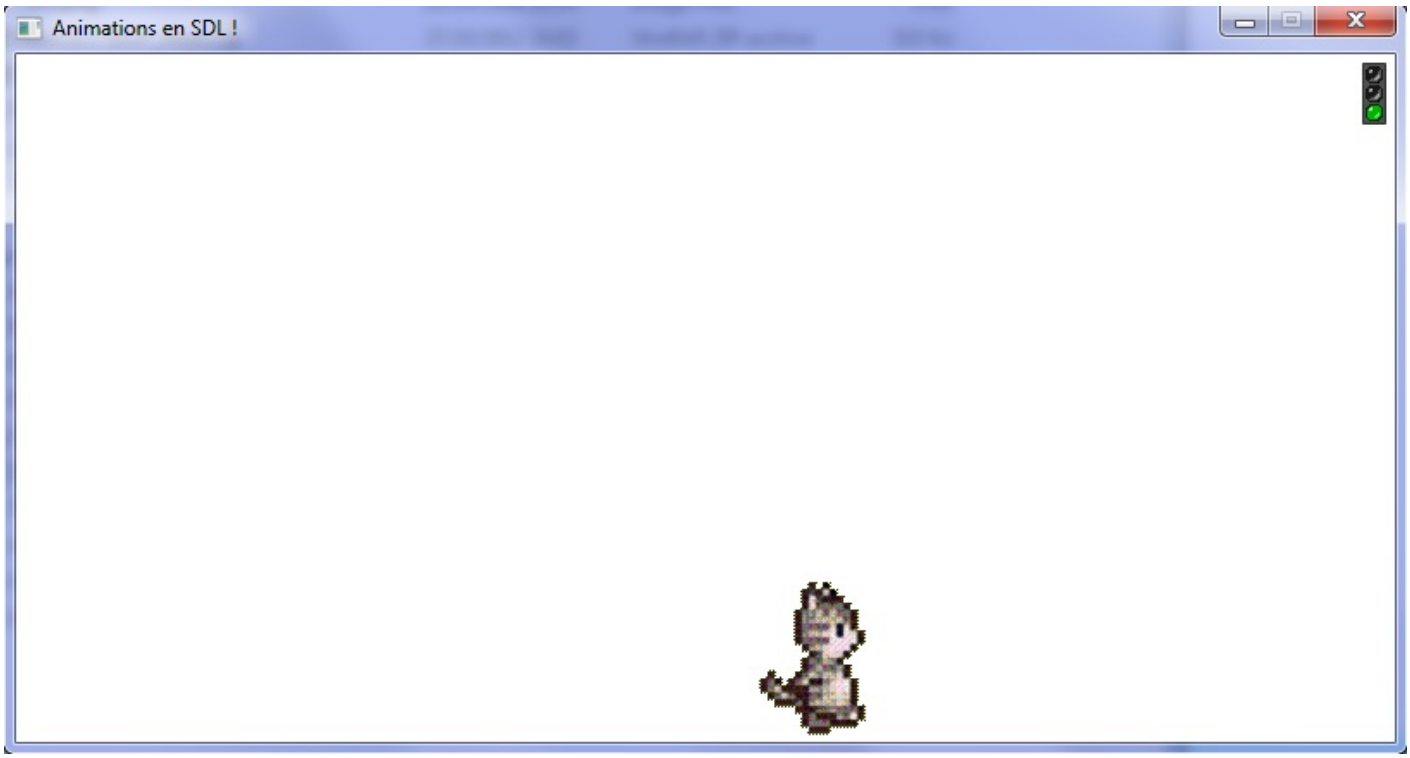
```c
#include <stdlib.h>
#include <stdio.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDLP_Anim.h>

/* SDLP_ANIM.C
Créé par Syndrome5 (POPOP9900)
Site du Zér0
*/

enum {DROITE,GAUCHE};

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;
    SDL_Rect pCat;
    SDLP_Anim *Cat[2] = {NULL};
    SDL_Event event;
    int continuer = 1, Regarde = DROITE;

    SDL_Init(SDL_INIT_VIDEO);

    ecran = SDL_SetVideoMode(400, 400, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);
    SDL_WM_SetCaption("Animations en SDL !", NULL);

    Cat[DROITE]=SDLP_LoadAnim("CatD.png", 4, SDLP_GAUCHE, 150,
SDLP_INFINI, SDLP_LANCER, SDLP_TOUCHE);
    Cat[GAUCHE]=SDLP_LoadAnim("CatG.png", 4, SDLP_GAUCHE, 150,
SDLP_INFINI, SDLP_LANCER, SDLP_TOUCHE);

    pCat.x=ecran->w/2 - SDLP_GetWSurfaceAnim(Cat[DROITE])/2;
    pCat.y=ecran->h - SDLP_GetHSurfaceAnim(Cat[DROITE]);

    while (continuer)
    {
        SDL_PollEvent(&event);
        switch(event.type)
        {
            case SDL_QUIT:
                continuer = 0;
            break;
            case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
                case SDLK_RIGHT:
                    pCat.x+=2;
                    Regarde = DROITE;
                break;
                case SDLK_LEFT:
                    pCat.x-=2;
```

```
                    Regarde = GAUCHE;
                break;
                default:
                    SDLP_BlockTouch(Cat[DROITE]);
                    SDLP_BlockTouch(Cat[GAUCHE]);
                break;
            }
            break;
        }
        SDLP_Touch(event, Cat[DROITE]);
        SDLP_Touch(event, Cat[GAUCHE]);

        SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,
255, 255));

        switch (Regarde)
        {
            case DROITE:
                SDLP_BlitAnim(Cat[DROITE], ecran, pCat);
            break;
            case GAUCHE:
                SDLP_BlitAnim(Cat[GAUCHE], ecran, pCat);
            break;
            default:
            break;
        }
        SDL_Flip(ecran);
    }

    SDLP_FreeAnim(Cat[DROITE]);
    SDLP_FreeAnim(Cat[GAUCHE]);

    SDL_Quit();

    return EXIT_SUCCESS;
}
```

Télécharger le projet.

Exercice 2 : Mêlons les 2 types d'animation

Vous avez réussi l'exercice 1 sans problème ? Alors nous allons maintenant mêler nos connaissances.
L'idée est la suivante : Notre chat va faire une course seul ( ). Il démarre à gauche de la fenêtre et son but est d'arriver à droite. Lorsqu'il arrive, le feu disparait. Cependant, celui-ci ne doit partir qu'au feu vert. Sinon il est bloqué.
Là aussi vous avez les capacités pour y arriver sans aide supplémentaire ! Au travail !
Conseil : Basez vous à partir du code de l'exercice 1.
Petit screen du résultat possible :



Correction

Je relève les copies. 
Pour ma part, voici mon code :

Secret (cliquez pour afficher)

Code : C

```c
#include <stdlib.h>
#include <stdio.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDLP_Anim.h>

/* SDLP_ANIM.C
Créé par Syndrome5 (POPOP9900)
Site du Zér0
*/

enum {DROITE,GAUCHE};

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;
    SDL_Rect pCat, pFeu;
    SDLP_Anim *Cat[2] = {NULL}, *Feu = NULL;
    SDL_Event event;
    int continuer = 1, Regarde = DROITE;

    SDL_Init(SDL_INIT_VIDEO);

    ecran = SDL_SetVideoMode(800, 400, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);
    SDL_WM_SetCaption("Animations en SDL !", NULL);

    Cat[DROITE]=SDLP_LoadAnim("CatD.png", 4, SDLP_GAUCHE, 150,
SDLP_INFINI, SDLP_LANCER, SDLP_TOUCHE);
    Cat[GAUCHE]=SDLP_LoadAnim("CatG.png", 4, SDLP_GAUCHE, 150,
SDLP_INFINI, SDLP_LANCER, SDLP_TOUCHE);
```

```
    Feu=SDLP_LoadAnim("Feu.bmp", 4, SDLP_DROITE, 2000, 1,
SDLP_LANCER, SDLP_NORMAL);

    pCat.x=0;
    pCat.y=ecran->h - SDLP_GetHSurfaceAnim(Cat[DROITE]);

    pFeu.x=ecran->w - SDLP_GetWSurfaceAnim(Feu) - 5;
    pFeu.y=5;

    while (continuer)
    {
        SDL_PollEvent(&event);

        if (SDLP_GetIDAnim(Feu)==SDLP_GetIDMaxAnim(Feu))
        {
            SDLP_SetEtatAnim(Feu, SDLP_PAUSE);
            SDLP_SetEtatAnim(Cat[DROITE], SDLP_LANCER);
            SDLP_SetEtatAnim(Cat[GAUCHE], SDLP_LANCER);
        }

        if (pCat.x >= ecran-
>w-SDLP_GetWSurfaceAnim(Cat[DROITE])-10)
        {
            SDLP_SetEtatAnim(Feu, SDLP_ARRETER);
            SDLP_SetEtatAnim(Cat[DROITE], SDLP_PAUSE);
            SDLP_SetEtatAnim(Cat[GAUCHE], SDLP_PAUSE);
        }

        switch(event.type)
        {
            case SDL_QUIT:
                continuer = 0;
            break;
            case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
                case SDLK_RIGHT:
                    if (SDLP_GetEtatAnim(Feu)==SDLP_PAUSE)
                    {
                        pCat.x+=2;
                    }
                    Regarde = DROITE;
                break;
                case SDLK_LEFT:
                    if (SDLP_GetEtatAnim(Feu)==SDLP_PAUSE)
                    {
                        pCat.x-=2;
                    }
                    Regarde = GAUCHE;
                break;
                default:
                    SDLP_BlockTouch(Cat[DROITE]);
                    SDLP_BlockTouch(Cat[GAUCHE]);
                break;
            }
            break;
        }
        SDLP_Touch(event, Cat[DROITE]);
        SDLP_Touch(event, Cat[GAUCHE]);

        SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,
255, 255));

        SDLP_BlitAnim(Feu, ecran, pFeu);
        switch (Regarde)
        {
            case DROITE:
                SDLP_BlitAnim(Cat[DROITE], ecran, pCat);
            break;
            case GAUCHE:
```

```
                    SDLP_BlitAnim(Cat[GAUCHE], ecran, pCat);
                break;
                default:
                break;
            }
            SDL_Flip(ecran);
        }

        SDLP_FreeAnim(Feu);
        SDLP_FreeAnim(Cat[DROITE]);
        SDLP_FreeAnim(Cat[GAUCHE]);

        SDL_Quit();

        return EXIT_SUCCESS;
    }
```

Télécharger le projet.

Rien de très bien compliqué, il suffit juste de bien connaître ses fonctions !

Q.C.M.

Le premier QCM de ce cours vous est offert en libre accès.
Pour accéder aux suivants

Connectez-vous Inscrivez-vous

Quel est le sens de l'animation si je passe `SDLP_GAUCHE` à la fonction `LoadAnim` ?

- ◯ De droite à gauche.
- ◯ De gauche à droite.
- ◯ Il est aléatoire.

Quelle est la fonction qui bloque l'animation même si on appuie sur une touche ?

- ◯ SDLP_BlitAnim.
- ◯ SDL_AnimBlock.
- ◯ SDLP_BlockTouch.
- ◯ SDLP_BlockAnim.
- ◯ SDLP_TouchBlock.

Code : C

```
SDLP_BlitAnim(&Anim, ecran);
```

Que pourrait bien faire ce code à la compilation ?
Sachant que le reste du code est correct et qu'une variable de type `SDLP_Anim` a été déclarée.

- ◯ Il compilera avec succès !
- ◯ Il plantera lamentablement !
- ◯ Je n'en ai vraiment aucune idée !

Pour qu'une animation soit autonome, on utilise :

- ◯ SDL_WaitEvent.
- ◯ SDL_PollEvent.
- ◯ SDLP_PosInit.

Je veux créer une variable de type animation. Que dois-je utiliser ?

- ◯ SDLP_Anim *rouleauPQ = NULL;
- ◯ SDLP_Anim Detritus;
- ◯ SDLP_Anim &Poubelle;

Correction !

Statistiques de réponses au QCM

J'espère que vous avez tout retenu (mais je vous laisse revenir, au cas où). Il y a pas mal de fonctions ; je ne vous oblige pas à toutes les retenir !

C'est ici que nous nous quittons. J'espère vous avoir aidé !
À bientôt.