Cours de C n°15 (intermédiaire)

Trimestre 2

Développement avec la bibliothèque SDL2 partie 2

Rappel

Initialiser SDL

Initialiser SDL :
int SDL_Init(int flags)

- SDL_Init(...) permet d'initialiser SDL
- retourne 0 si tout est ok
- retourne une valeur négative si une erreur

Quelques exemples :

- SDL_Init(SDL_INIT_VIDEO)
- SDL_Init(SDL_INIT_VIDEO|SDL_INIT_AUDIO)
- SDL_Init(SDL_INIT_EVERYTHING)

=> retourne 0 si tout est ok

La fenêtre (window)

Créer une fenêtre

- SDL_Window* SDL_CreateWindow(const char* titre,
  int x,
  int y,
  int largeur,
  int hauteur,
  Uint32 flags)

- titre : titre de la fenêtre
- x, y : position de la fenêtre
- largeur, hauteur : taille de la fenêtre
- flags : les options qu'on veut utiliser

Détruire une fenêtre

- void SDL_DestroyWindow(SDL_Window* window)

- window → la fenêtre qu'on veut détruire

Afficher une fenêtre pendant 5 secondes (exemple)

```c
// creer une fenetre : SDL_CreateWindow(...)

#include "SDL.h"
#include <stdio.h>

int main(int argc, char** argv)
{
        SDL_Window* window = NULL;

        // Initialiser SDL (on utilise seulement la vidéo)
        if (SDL_Init(SDL_INIT_VIDEO) != 0)
        {
                printf("Erreur pour initialiser SDL (%s)\n", SDL_GetError());
                return -1;
        }

        // Creer la fenetre
        window = SDL_CreateWindow("01 - Ma premiere fenetre SDL2",
                SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
                640, 480,
                SDL_WINDOW_SHOWN);

        if(window == NULL)
        {
                printf("Erreur pour creer la fenetre (%s)\n", SDL_GetError());
                return -1;
        }

        // attendre 5 secondes
        SDL_Delay(5000);

        // fin du programme
        SDL_DestroyWindow(window);
        SDL_Quit();
        return 0;
}
```

Afficher une fenêtre pendant 5 secondes

Attendre une durée fixe

Attendre N millisecondes

void SDL_Delay(Uint32 ms)

- ms : durée en millisecondes

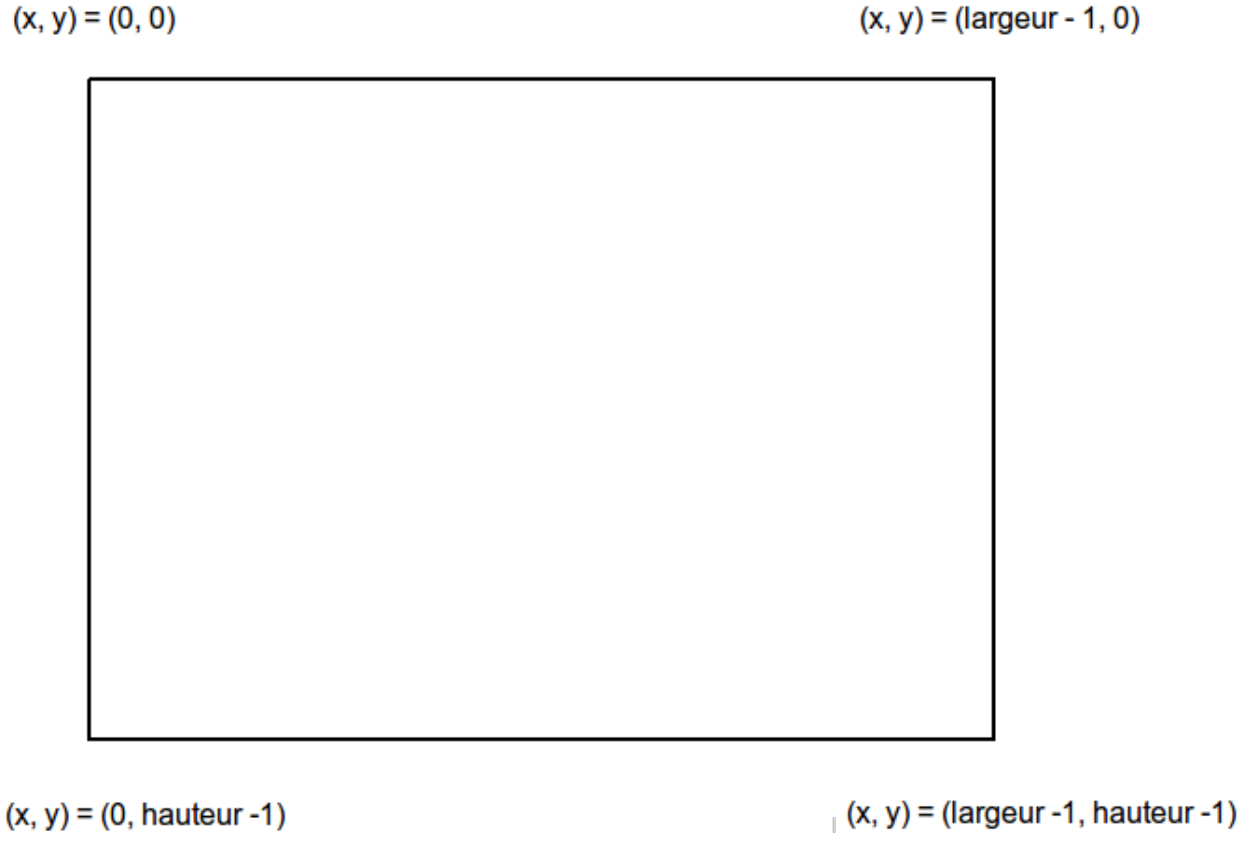
Attendre N millisecondes (exemples)

```
// attendre 1 seconde
SDL_Delay(1000);

// attendre 5 secondes
SDL_Delay(5000);

// attendre 1/4 seconde
SDL_Delay(250);
```

Dessiner avec SDL

Position écran

Le point (0,0) est toujours en haut à gauche pour dessiner



Quelques fonctions graphiques - 1/2

- créer le renderer : SDL_Renderer* SDL_CreateRenderer(SDL_Window* window, int index, Uint32 flags)
- détruire le renderer : void SDL_DestroyRenderer(SDL_Renderer* renderer)
- changer la couleur de dessinage ou remplissage : SDL_SetRenderDrawColor(SDL_Renderer* renderer, int r, int g, int b, int a)
- effacer tout : SDL_RenderClear(SDL_Renderer* renderer)
- mettre à jour la fenêtre : SDL_RenderPresent(SDL_Renderer* renderer)

Quelques fonctions graphiques - 2/2

- dessiner un point : SDL_RenderDrawPoint(SDL_Renderer* renderer, int x, int y)
- dessiner N points : SDL_RenderDrawPoints(SDL_Renderer* renderer, const SDL_Point* points, int N)
- dessiner une ligne : SDL_RenderDrawLineSDL_Renderer* renderer, int x1, int y1, int x2,int y2)
- dessiner N lignes : SDL_RenderDrawLinesSDL_RenderDrawLines(SDL_Renderer* renderer, const SDL_Point* points, int N)
- dessiner un rectangle vide : SDL_RenderDrawRect(SDL_Renderer* renderer, const SDL_Rect* rect)
- dessiner N rectangles vides : SDL_RenderDrawRects(SDL_Renderer* renderer, const SDL_Rect* rects, int N)
- dessiner un rectangle rempli : SDL_FillRect(SDL_Renderer* renderer, const SDL_Rect* rect)
- dessiner N rectangles remplis : SDL_FillRects(SDL_Renderer* renderer, const SDL_Rect* rects, int N)

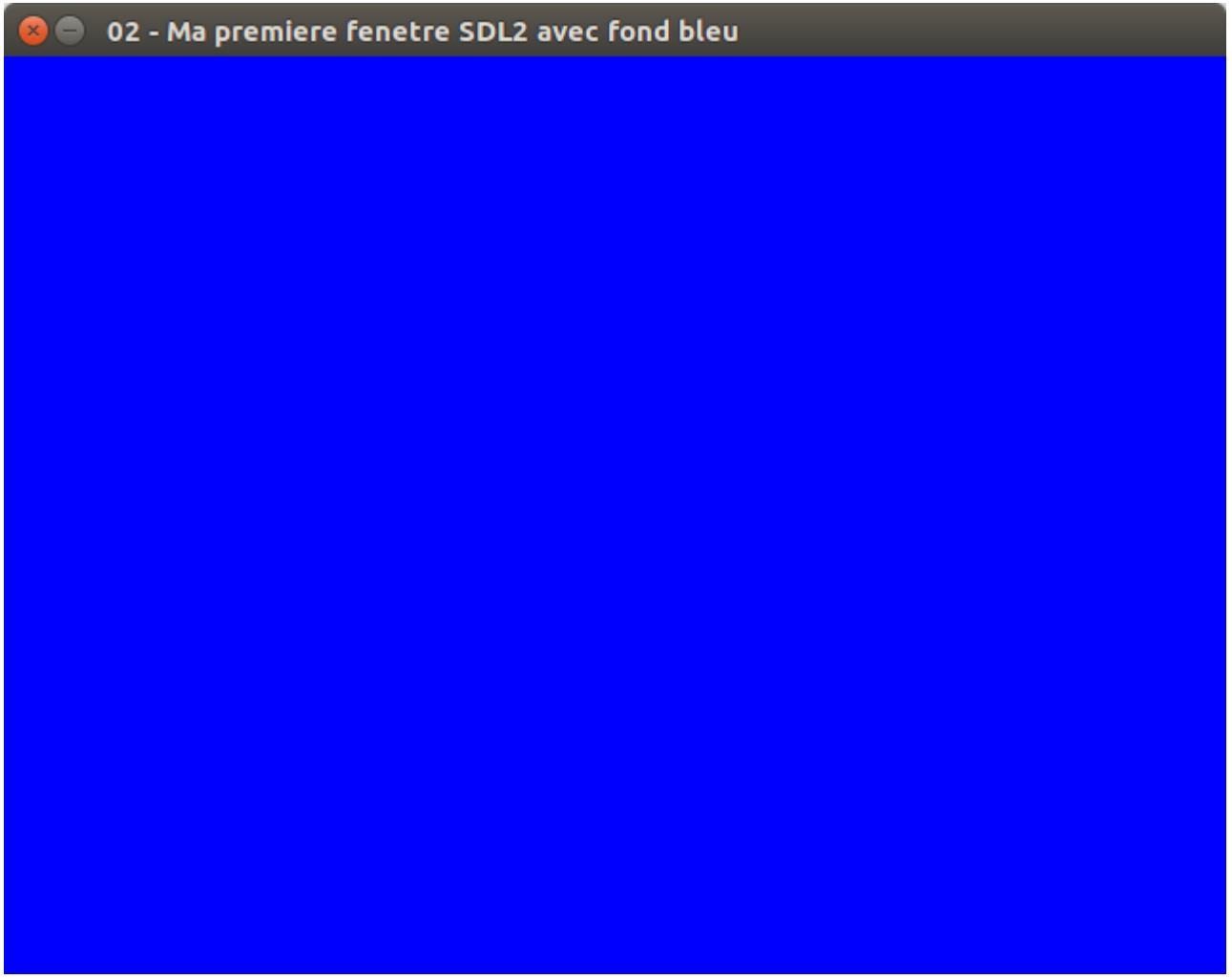
Afficher un arrière-plan bleu - 1/2

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

int main(int argc, char** argv)
{
        SDL_Window* window = NULL;
        SDL_Renderer* renderer = NULL;

        // Initialiser SDL (on utilise seulement la vidéo)
        if (SDL_Init(SDL_INIT_VIDEO) != 0)
        {
                printf("Erreur pour initialiser SDL (%s)\n", SDL_GetError());
                return -1;
        }

        // Creer la fenetre
        window = SDL_CreateWindow("02 - Ma premiere fenetre SDL2 avec fond blue",
                SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
                640, 480,
                SDL_WINDOW_SHOWN);
        if(window == NULL)
        {
                printf("Erreur pour creer la fenetre (%s)\n", SDL_GetError());
                return -1;
        }
```

Afficher un arrière-plan bleu - 2/2

```
        // creer le renderer
        renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
        if(renderer != NULL)
        {
                // effacer le contenu de la fenetre avec du blue
                SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255);
                SDL_RenderClear(renderer);

                // mettre à jour l'ecran
                SDL_RenderPresent(renderer);
        }

        // attendre 5 secondes
        SDL_Delay(5000);

        // fin du programme
        SDL_DestroyWindow(window);
        SDL_Quit();
        return 0;
}
```

Afficher un arrière plan bleu



Charger une image et libérer une image

```
SDL_Surface* image = NULL;

// charger une image
image = SDL_LoadBMP("image.bmp");

// libérer une image
SDL_FreeSurface(image);
```

Charger + Convertir une image en texture pour l'afficher

```
SDL_Surface* image = NULL;
SDL_Texture* texture = NULL;

// charger une image
image = SDL_LoadBMP("image.bmp");

// convertir l'image en texture
texture = SDL_CreateTextureFromSurface(renderer, image);

// libérer une image
SDL_FreeSurface(image);

// effacer la texture
SDL_DestroyTexture(texture);
```

Afficher une image (texture)

```
// rectangle de l'image, il contient la position X, Y et la taille (largueur et hauteur)
SDL_Rect rectangle = { /*x*/100, /*y*/80, /*largeur*/200, /*hauteur*/150};

// afficher la texture/image à l'ecran
SDL_RenderCopy(renderer, texture, NULL, &rectangle);
```

Afficher une image - 1/3 (exemple complet)

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

int main(int argc, char** argv)
{
        int ret = 0;
        SDL_Window* window = NULL;
        SDL_Renderer* renderer = NULL;
        SDL_Surface* image = NULL;
        SDL_Texture* texture = NULL;

        // Initialiser SDL (on utilise seulement la vidéo)
        if (SDL_Init(SDL_INIT_VIDEO) != 0)
        {
                printf("Erreur pour initialiser SDL (%s)\n", SDL_GetError());
                return -1;
        }

        // Creer la fenetre
        window = SDL_CreateWindow("03 - Afficher une image",
                SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
                640, 480,
                SDL_WINDOW_SHOWN);
        if(window == NULL)
        {
                printf("Erreur pour creer la fenetre (%s)\n", SDL_GetError());
                return -1;
        }
```

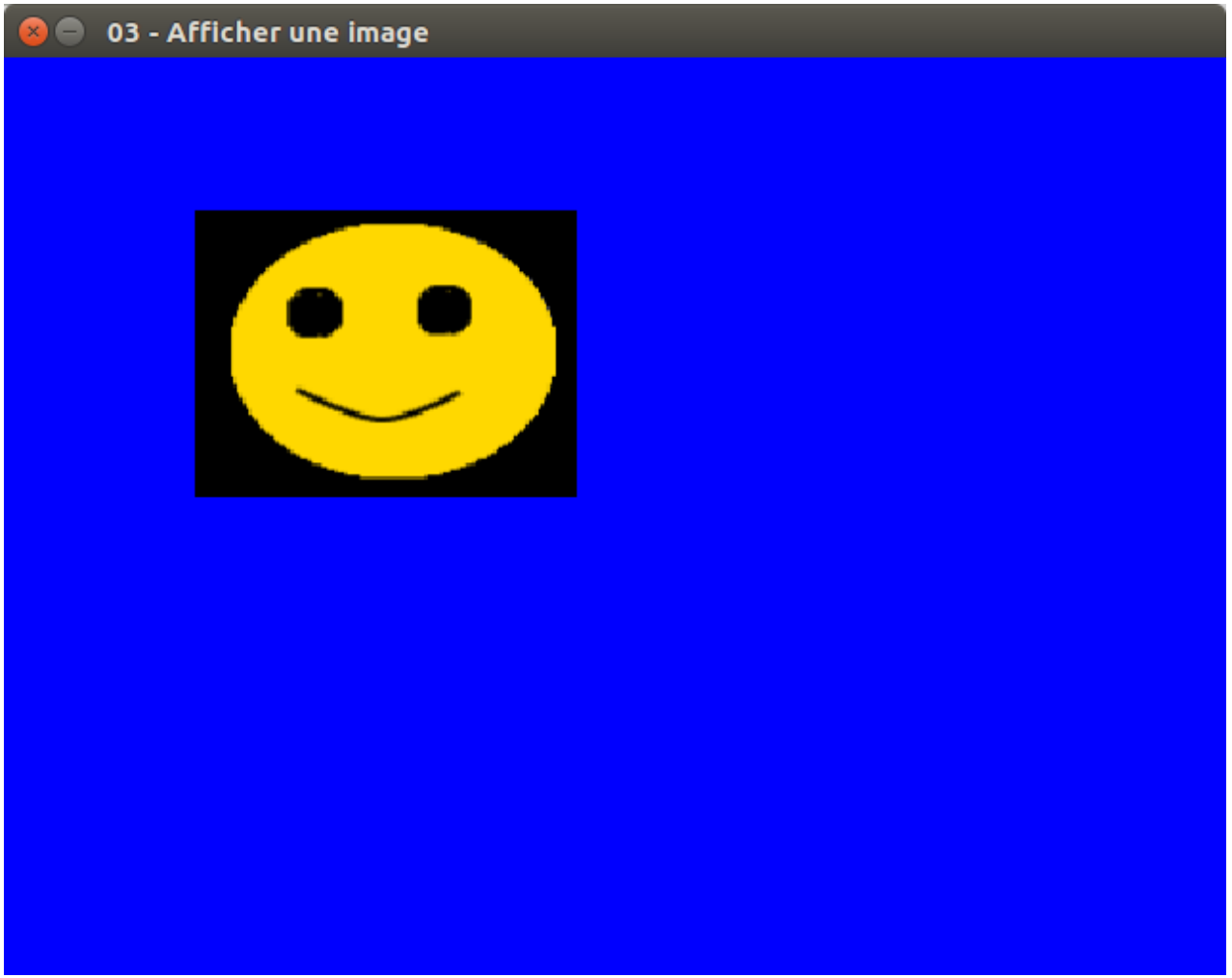
Afficher une image - 2/3 (exemple complet)

```
// creer le renderer
renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
if(renderer != NULL)
{
        // effacer le contenu de la fenetre avec du blue
        SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255);
        SDL_RenderClear(renderer);

        // charger l'image
        image = SDL_LoadBMP("image.bmp");
        if(image != NULL)
        {
                // rectangle contient la position X, Y et la taille (largueur et hauteur)
                SDL_Rect rectangle = { /*x*/100, /*y*/80, /*largeur*/200, /*hauteur*/150};

                // convertir l'image en texture
                texture = SDL_CreateTextureFromSurface(renderer, image);

                // effacer l'image car on n'en a plus besoin car la texture contient l'image
                SDL_FreeSurface(image);

                // afficher la texture/image à l'ecran
                SDL_RenderCopy(renderer, texture, NULL, &rectangle);

                // effacer la texture car on en a plus besoin une fois affichée
                SDL_DestroyTexture(texture);
        }
        else
        {
                ret = -1;
                printf("Erreur de chargement de l'image (%s)\n", SDL_GetError());
        }


        // mettre à jour l'ecran
        SDL_RenderPresent(renderer);
}
```

Afficher une image - 3/3 (exemple complet)

```
        else
        {
                ret = -1;
                printf("Erreur pour creer le renderer (%s)\n", SDL_GetError());
        }

        // attendre 5 secondes
        SDL_Delay(5000);

        // fin du programme
        SDL_DestroyWindow(window);
        SDL_Quit();
        return ret;
}
```

Afficher une image (screenshot)



Les événements (events)

Qu'est-ce qu'un événement (event) ?

- Un événement est une action utilisateur ou de l'OS
- Ils sont stockés dans une file/queue : liste qui conserve l'ordre d'arrivé (FIFO : First In First Out = 1er arrivé 1er servi)
- Quelques exemples d'évenement : Touche du clavier préssée, Click de la souris, redimensionnement de la fenêtre, ...

Quelques types d'événements en SDL :

- SDL_QUIT : L'utilisateur veut fermer l'application SDL
- SDL_KEYDOWN : Une touche du clavier est préssée
- SDL_KEYUP : Une touche du clavier est relachée
- SDL_MOUSEBUTTONDOWN : Un bouton de la souris est préssé
- SDL_MOUSEBUTTONUP : Un bouton de la souris est relaché

Lire tous les événements

```c
int quit = 0;
SDL_Event event;

// lire tous les evenements
while (SDL_PollEvent(&event))
{
        // Test le type de l'évenement : SDL_QUIT, SDL_KEYDOWN, SDL_KEYUP, SDL_MOUSEBUTTONDOWN, SDL_MOUSEBUTTONUP
        switch (event.type)
        {
                case SDL_QUIT:
                {
                        printf("reception de l evenement SDL_QUIT\n");
                        quit = 1;
                        break;
                }
                case SDL_KEYDOWN:
                {
                        printf("touche %d du clavier préssée\n", event.key.keysym.sym);
                        break;
                }
                case SDL_MOUSEBUTTONDOWN:
                {
                        printf("bouton %d de souris préssée\n", event.button.button);
                        break;
                }
        }
}
```

Gérer l'événement "Quitter"

```
// L'évenement SDL_QUIT est envoyé quand on clique sur le "X" de la fenetre

int quit = 0;
SDL_Event e;

// boucle principale
while(!quit)
{
        // lire tous les evenements
        while (SDL_PollEvent(&e))
        {
                // stopper la boucle principale si évenememement est SDL_QUIT
                if (e.type == SDL_QUIT)
                {
                        printf("reception de l evenement SDL_QUIT\n");
                        quit = 1;
                }
        }

        // Mettre ici votre code d'affichage SDL
        // ...
}
```

Utiliser le clavier

Il faut gérer les événements de type SDL_KEYDOWN (touche préssée) ou SDL_KEYUP (touche relachée)

Utiliser le clavier

```
SDL_Event evenement;

while(!quit)
{
        // lire tous les evenements
        while (SDL_PollEvent(&evenement))
        {
                // Test le type de l'évenement : SDL_QUIT, SDL_KEYDOWN, SDL_KEYUP, SDL_MOUSEBUTTONDOWN, SDL_MOUSEBUTTONUP
                switch (evenement.type)
                {
                        case SDL_KEYDOWN:
                        {
                                printf("touche %d du clavier préssée\n", evenement.key.keysym.sym);
                                if (evenement.key.keysym.sym == SDLK_a)
                                {
                                        // gérer la touche A
                                }
                                else if (evenement.key.keysym.sym == SDLK_b)
                                {
                                        // gérer la touche B
                                }
                                else if (evenement.key.keysym.sym == SDLK_2)
                                {
                                        // gérer la touche 2
                                }
                                break;
                        }
                }
        }
}
```

Utiliser la souris

Il faut gérer les événements de type SDL_MOUSEBUTTONDOWN (bouton de la souris préssé) ou SDL_MOUSEBUTTONUP (bouton de la souris relaché)

Info supplémentaire : evenement.motion.x et evenement.motion.y donne la position de la souris

Utiliser la souris (exemple)

```
SDL_Event evenement;

// lire tous les evenements
while (SDL_PollEvent(&evenement))
{
        // Test le type de l'évenement : SDL_QUIT, SDL_KEYDOWN, SDL_KEYUP, SDL_MOUSEBUTTONDOWN, SDL_MOUSEBUTTONUP
        switch (evenement.type)
        {
                case SDL_MOUSEBUTTONDOWN:
                {
                        printf("bouton %d de souris préssée\n", evenement.button.button);
                        printf("position de la souris (%d, %d)\n", evenement.motion.x, evenement.motion.y);
                        break;
                }
        }
}
```

La boucle principale ou la gameloop

La boucle principale ou la gameloop en 4 étapes:

- (1) Event - boucle pour lire/gérer tous les évenements via SDL_PollEvent()
- (2) Update - Faire nos calculs
- (3) Draw - Afficher l' "écran" de la fenetre + mettre à jour à la fin via SDL_RenderPresent()"
- (4) Wait - Attendre une petite durée

Gérer un framerate constant (version simplifiée)

- Par exemple, pour synchroniser l'affichage de l'application SDL avec un framerate fixe de 20 FPS (20 images par secondes), il suffit d’attendre : 1 secondes / 20

  1 secondes / 20 = 1000 millisecondes / 20 = 50 ms

- Remarque : on considère que les autres étapes prennent 0 millisecondes (pas toujours vrais)

La gameloop avec un framerate constant de 20 FPS (version simplifiée)

```
while (!quit)
{
        HandleEvents();          // (1) - Gérer les évenements
        Update();                // (2) - Faire nos calculs
        Draw();                  // (3) - Afficher


        #define FPS 20
        SDL_Delay(1000 / FPS);   // (4) - Attendre
}
```

Gérer un framerate constant (meilleur version)

- Synchroniser l'affichage de l'application SDL avec un framerate fixe de 20 FPS (20 images par secondes)
- Remarque : On veut que l'application tourne toujours à 20 FPS quelque soit la durée des 3 autres étapes.

Gérer un framerate constant (meilleur version)

```
while (!quit)
{
        int tempsDebut;         // en millisecondes
        int tempsFin;           // en millisecondes
        int dureeTotale;        // en millisecondes
        int dureeAttente;       // en millisecondes

        tempsDebut = SDL_GetTicks();
        HandleEvents();         // (1) - Gérer les évenements
        Update();               // (2) - Faire nos calculs
        Draw();                 // (3) - Afficher
        tempsFin = SDL_GetTicks();

        dureeTotale = tempsFin - tempsDebut;

        // (4) - Attendre
        #define FPS 20
        dureeAttente = 1000 / FPS - dureeTotal;
        if (dureeAttente > 0)
        {
                SDL_Delay(dureeAttente);
        }
}
```

Gérer une gameloop avec un framerate indépendant (autre version)

- Synchroniser l'affichage de l'application SDL avec un framerate indépendant (N images par secondes), c’est à dire quelque soit la vitesse du programme

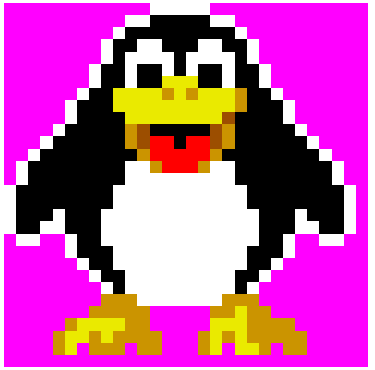
Gérer une gameloop avec un framerate indépendant (autre version)

```
int tempsDebut; // en millisecondes
int tempsFin;   // en millisecondes
int tempsDelta = 0; // en millisecondes

while (!quit)
{
        tempsDebut = SDL_GetTicks();
        HandleEvents();         // (1) - Gérer les évenements
        Update(tempsDelta);     // (2) - Faire nos calculs en utilisant la variable tempsDelta
        Draw();                 // (3) - Afficher
        tempsFin = SDL_GetTicks();

        tempsDelta = tempsFin - tempsDebut; // calcul du temps écoulé

        // (4) - plus besoin d'attendre
        // SDL_Delay(0);
}
```

La transparence

La transparence

- Objectif : on veut afficher une image bmp contenant de la transparence



Exemple : sans transparence – 1/3

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

int main(int argc, char** argv)
{
	int ret = 0;
	SDL_Window* window = NULL;
	SDL_Renderer* renderer = NULL;

	// Initialiser SDL (on utilise seulement la vidéo)
	if (SDL_Init(SDL_INIT_VIDEO) != 0)
	{
		printf("Erreur pour initialiser SDL (%s)\n", SDL_GetError());
		return -1;
	}

	// Creer la fenetre
	window = SDL_CreateWindow("08 - afficher une image sans activer la transparence",
		SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
		640, 480,
		SDL_WINDOW_SHOWN);
	if(window == NULL)
	{
		printf("Erreur pour creer la fenetre (%s)\n", SDL_GetError());
		return -1;
	}

	// creer le renderer
	renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
	if(renderer != NULL)
	{
		int quit = 0;
		SDL_Rect pingouin_rectangle = { /*x*/100, /*y*/80, /*largeur*/64, /*hauteur*/64};

		SDL_Surface* pingouin_image = NULL;
		SDL_Texture* pingouin_texture = NULL;
```

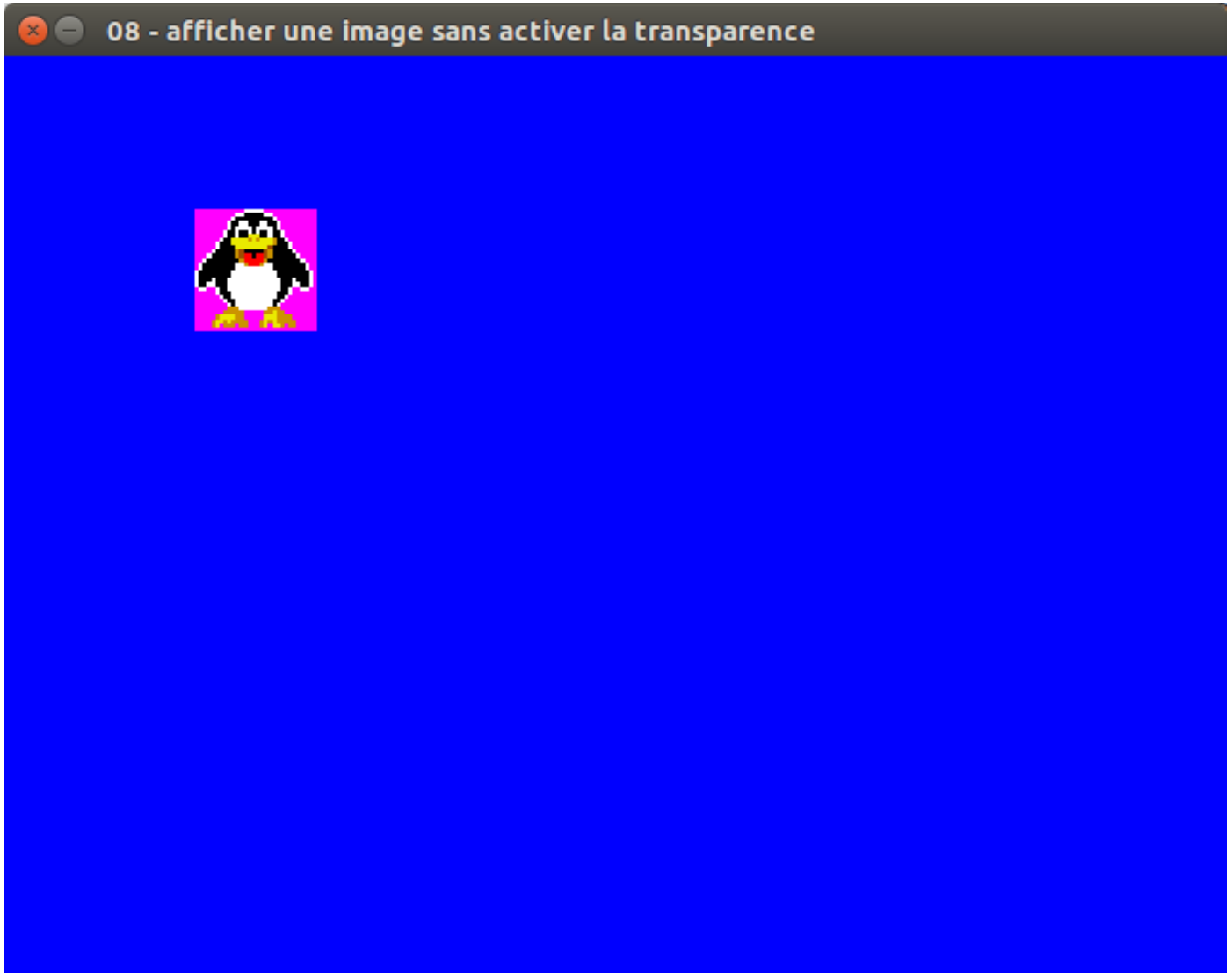
Exemple : sans transparence – 2/3

```
SDL_Event evenemement;

// charger l'image sans activer la transparence
pingouin_image = SDL_LoadBMP("pingouin.bmp");
if(pingouin_image != NULL)
{
        // convertir l'image en texture
        pingouin_texture = SDL_CreateTextureFromSurface(renderer, pingouin_image);

        // effacer l'image car on n'en a plus besoin car la texture contient l'image
        SDL_FreeSurface(pingouin_image);
}
else
{
        ret = -1;
        printf("Erreur de chargement de l'image (%s)\n", SDL_GetError());
}

while(!quit)
{
        // lire tous les evenememements
        while (SDL_PollEvent(&evenemement))
        {
                // Test le type de l'évenement : SDL_QUIT, SDL_KEYDOWN, SDL_KEYUP, SDL_MOUSEBUTTONDOWN, SDL_MOUSEBUTTONUP
                switch (evenemement.type)
                {
                        case SDL_QUIT:
                        {
                                printf("reception de l evenement SDL_QUIT\n");
                                quit = 1;
                                break;
                        }
                }
        }
```

Exemple : sans transparence – 3/3

```
                // effacer le contenu de la fenetre avec du blue (Rouge=0, Vert=0, Bleu=255, Aplha=255)
                SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255);
                SDL_RenderClear(renderer);

                // afficher la texture/image à l'ecran
                SDL_RenderCopy(renderer, pingouin_texture, NULL, &pingouin_rectangle);

                // mettre à jour l'ecran
                SDL_RenderPresent(renderer);

                // attendre 50 millisecondes
                SDL_Delay(50);
        }

        if (pingouin_texture)
        {
                SDL_DestroyTexture(pingouin_texture);
        }
        SDL_DestroyRenderer(renderer);
    }

    // fin du programme
    SDL_DestroyWindow(window);
    SDL_Quit();
    return 0;
}
```

Exemple : sans transparence



Exemple : avec transparence

- Juste après l'appel à SDL_LoadBMP(), il suffit d'ajouter cette ligne de code pour choisir la couleur rose comme couleur transparente :

```
SDL_SetColorKey(image, SDL_TRUE, SDL_MapRGB(pingouin_image->format, 255, 0, 255));
```

- Ou pour n'importe quelle couleur :

```
SDL_SetColorKey(image, SDL_TRUE, SDL_MapRGB(pingouin_image->format, r, v, b));
```

Exemple : avec transparence – 1/3

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

int main(int argc, char** argv)
{
        int ret = 0;
        SDL_Window* window = NULL;
        SDL_Renderer* renderer = NULL;

        // Initialiser SDL (on utilise seulement la vidéo)
        if (SDL_Init(SDL_INIT_VIDEO) != 0)
        {
                printf("Erreur pour initialiser SDL (%s)\n", SDL_GetError());
                return -1;
        }

        // Creer la fenetre
        window = SDL_CreateWindow("09 - afficher une image avec la transparence activée",
                SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
                640, 480,
                SDL_WINDOW_SHOWN);
        if(window == NULL)
        {
                printf("Erreur pour creer la fenetre (%s)\n", SDL_GetError());
                return -1;
        }

        // creer le renderer
        renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
        if(renderer != NULL)
        {
                int quit = 0;
                SDL_Rect pingouin_rectangle = { /*x*/100, /*y*/80, /*largeur*/64, /*hauteur*/64};

                SDL_Surface* pingouin_image = NULL;
                SDL_Texture* pingouin_texture = NULL;
```

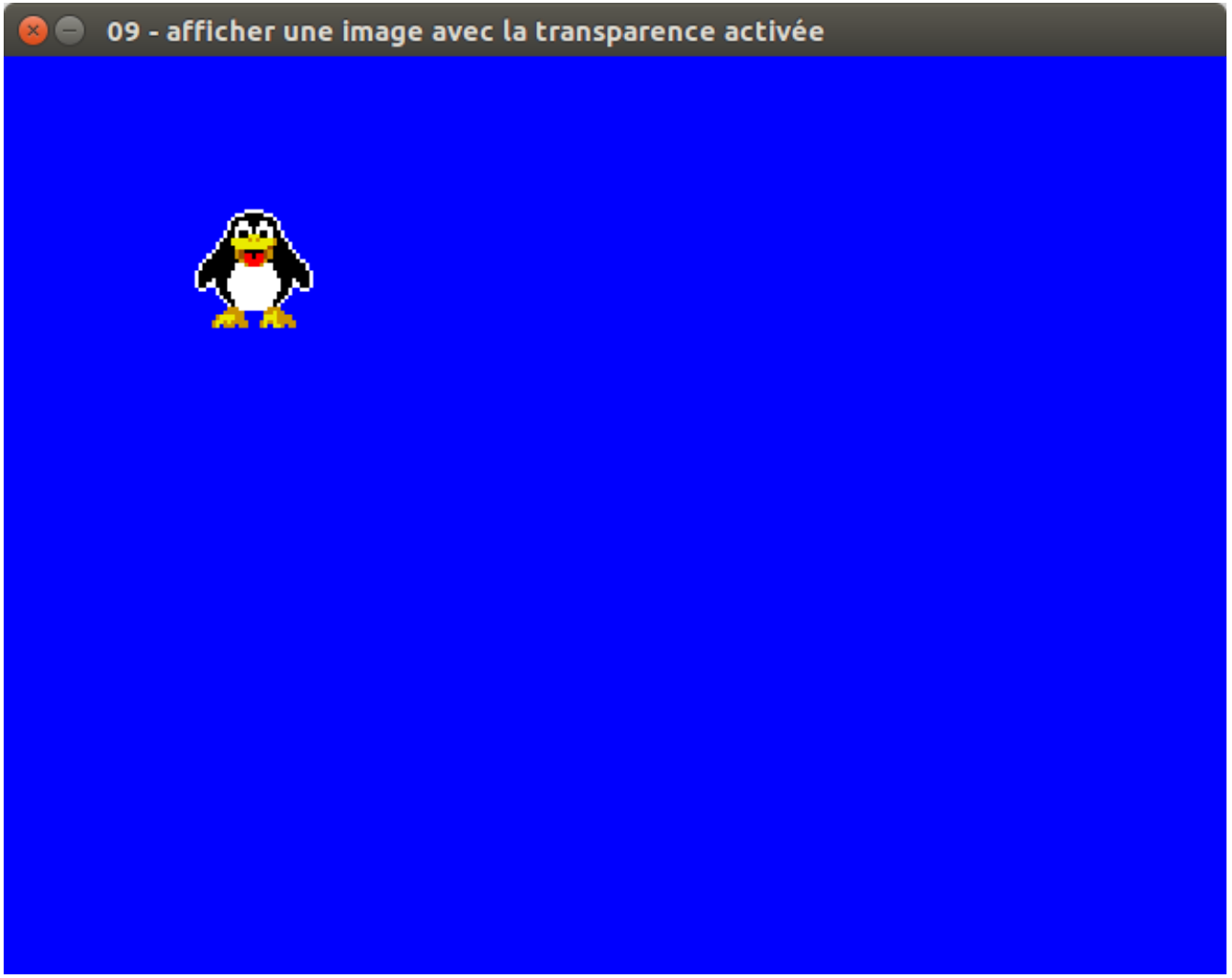
Exemple : avec transparence – 2/3

```
SDL_Event evenemement;

// charger l'image
pingouin_image = SDL_LoadBMP("pingouin.bmp");

// ajouter/activer la transparence pour la couleur rose (Rouge=255, Vert=0, Bleu=255)
SDL_SetColorKey(pingouin_image, SDL_TRUE, SDL_MapRGB(pingouin_image->format, 255, 0, 255));

if(pingouin_image != NULL)
{
        // convertir l'image en texture
        pingouin_texture = SDL_CreateTextureFromSurface(renderer, pingouin_image);

        // effacer l'image car on n'en a plus besoin car la texture contient l'image
        SDL_FreeSurface(pingouin_image);
}
else
{
        ret = -1;
        printf("Erreur de chargement de l'image (%s)\n", SDL_GetError());
}

while(!quit)
{
        // lire tous les evenememements
        while (SDL_PollEvent(&evenemement))
        {
                // Test le type de l'évenement : SDL_QUIT, SDL_KEYDOWN, SDL_KEYUP, SDL_MOUSEBUTTONDOWN, SDL_MOUSEBUTTONUP
                switch (evenemement.type)
                {
                        case SDL_QUIT:
                        {
                                printf("reception de l evenement SDL_QUIT\n");
                                quit = 1;
                                break;
                        }
```

Exemple : avec transparence – 3/3

```
                                }
                        }
                }

                // effacer le contenu de la fenetre avec du blue (Rouge=0, Vert=0, Bleu=255, Aplha=255)
                SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255);
                SDL_RenderClear(renderer);

                // afficher la texture/image à l'ecran
                SDL_RenderCopy(renderer, pingouin_texture, NULL, &pingouin_rectangle);

                // mettre à jour l'ecran
                SDL_RenderPresent(renderer);

                // attendre 50 millisecondes
                SDL_Delay(50);
        }

        if (pingouin_texture)
        {
                SDL_DestroyTexture(pingouin_texture);
        }
        SDL_DestroyRenderer(renderer);
}

// fin du programme
SDL_DestroyWindow(window);
SDL_Quit();
return 0;
}
```

Exemple : avec transparence



Créer une animation

Qu'est-ce qu'un sprite ? Qu'est-ce qu'un tile (une tuile) ?

- Un sprite et un tile (une tuile) sont tous les deux une petite image.
- Un sprite est souvent utilisé pour un personnange ou un objet alors qu'un tile (une tuile) sert pour le décors.
- « La différence entre les deux est qu'un sprite contient de la transparence » (pas toujours vrai car on peut aussi avoir plusieurs plans contenant des tiles avec de la transparence dans le but de les superposer).

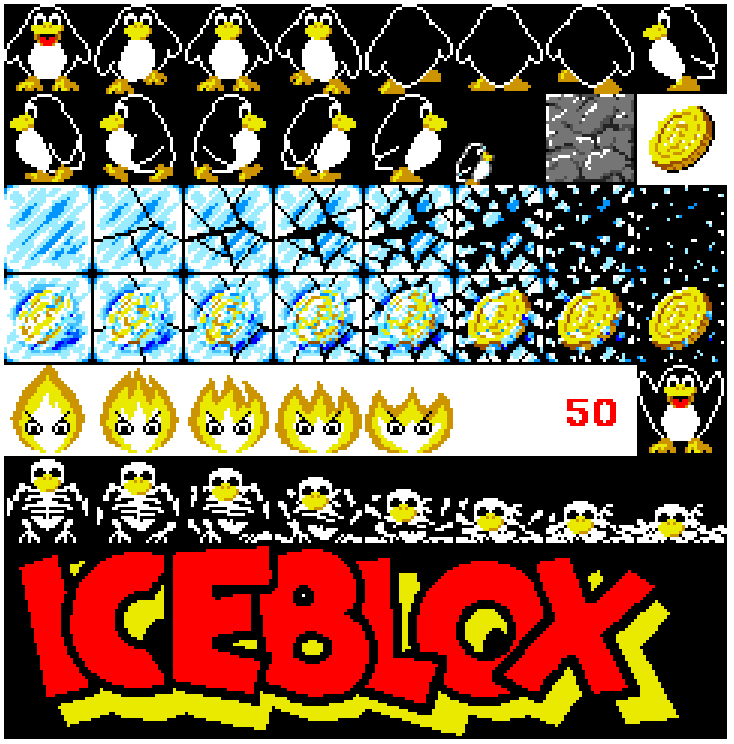
Créer une animation venant d'un "sprite sheet"

Qu'est-ce qu'un "sprite sheet" ?

- Une sprite est une (petite) image contenant un personnage ou un objet.
- Un sprite sheet est une image contenant plusieurs sprites.

Créer une animation venant d'un "sprite sheet"

Qu'est-ce qu'un "sprite sheet" ?



Un exemple de fichier contenant une animation

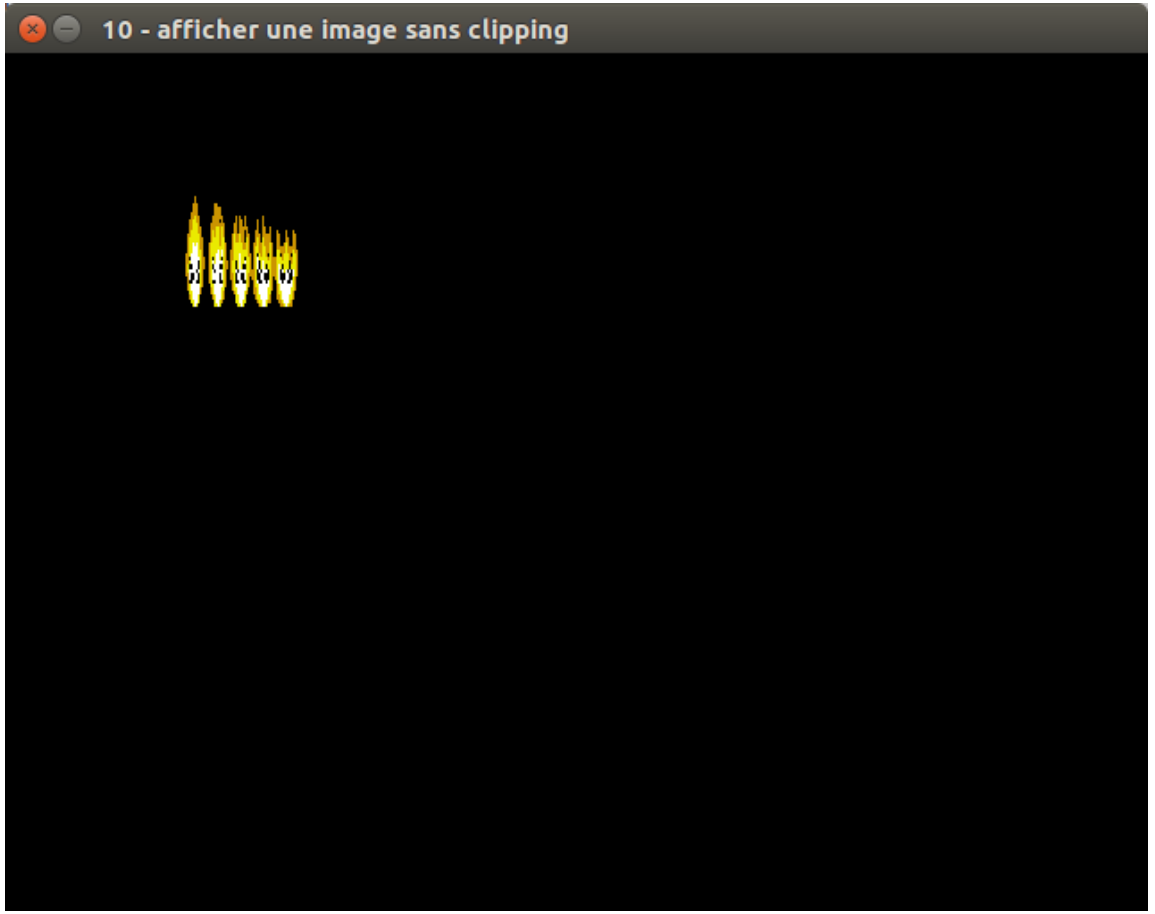
Créer une animation en SDL

- Pour créer une animation à partir d'un sprite sheet, on va faire un "clipping" puis changer la zone du clipping en fonction du temps T. On va modifier le paramètre "srcrect" (source rectangle) de la fonction SDL_RenderCopy(...).

```
int SDL_RenderCopy(SDL_Renderer*   renderer,
                   SDL_Texture*    texture,
                   const SDL_Rect* srcrect,
                   const SDL_Rect* dstrect)
```

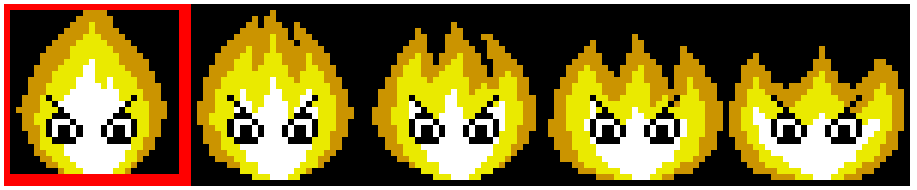
Afficher une image sans clipping

```
SDL_RenderCopy(renderer, texture, NULL, &rectangle);
```

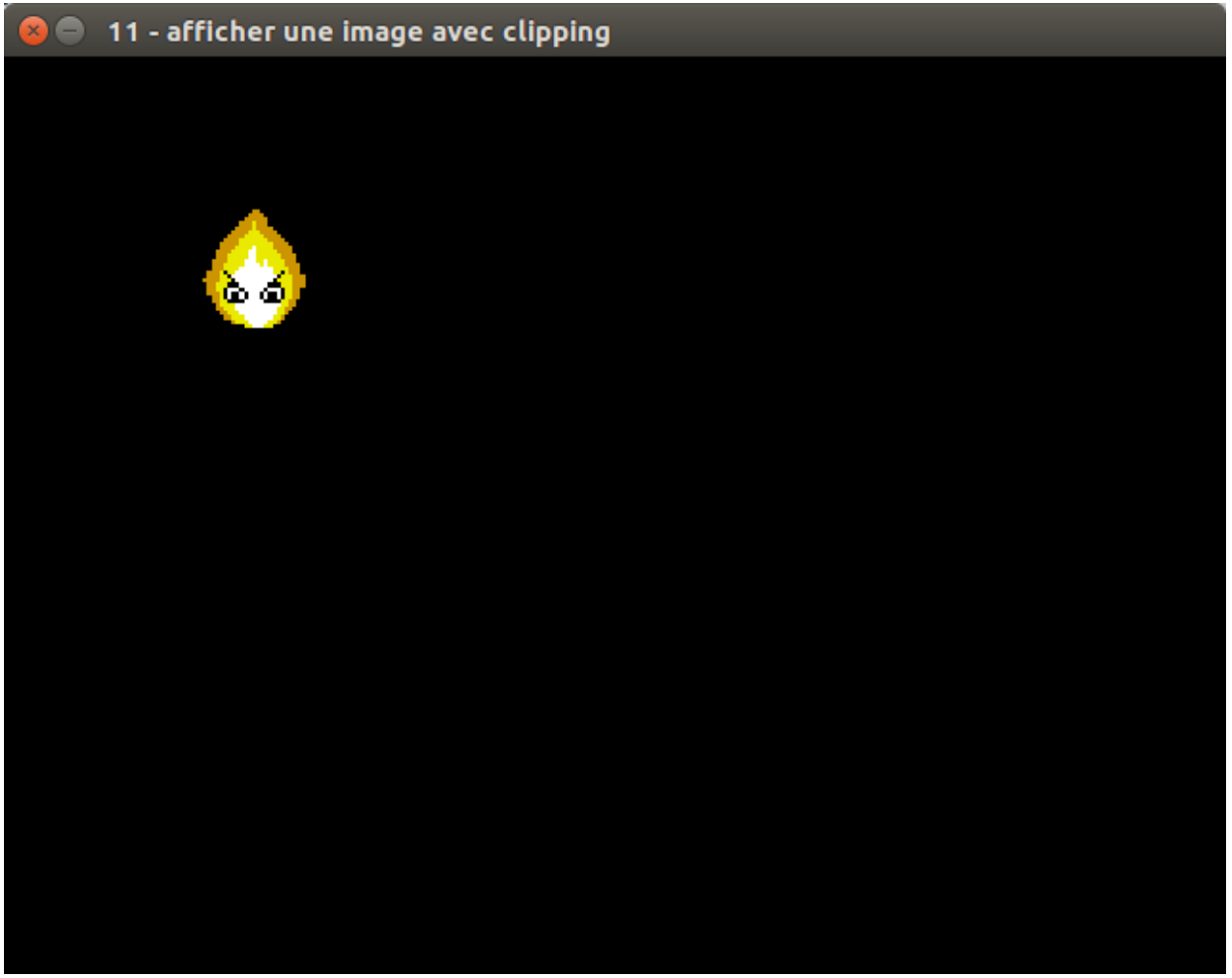


Afficher une image avec clipping sur le 1er sprite

```
SDL_Rect clipping_rectangle;

clipping_rectangle.x = 0;
clipping_rectangle.y = 0;
clipping_rectangle.w = SPRITE_LARGEUR;
clipping_rectangle.h = SPRITE_HAUTEUR;

SDL_RenderCopy(renderer, texture, &clipping_rectangle, &rectangle);
```

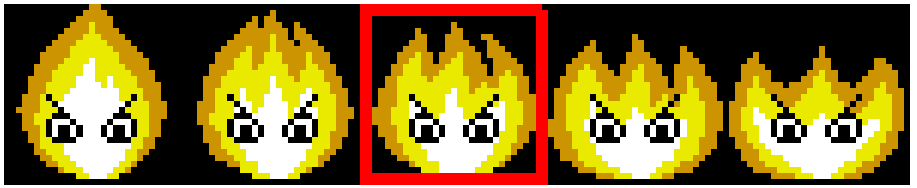


Afficher une image avec clipping sur le 1er sprite

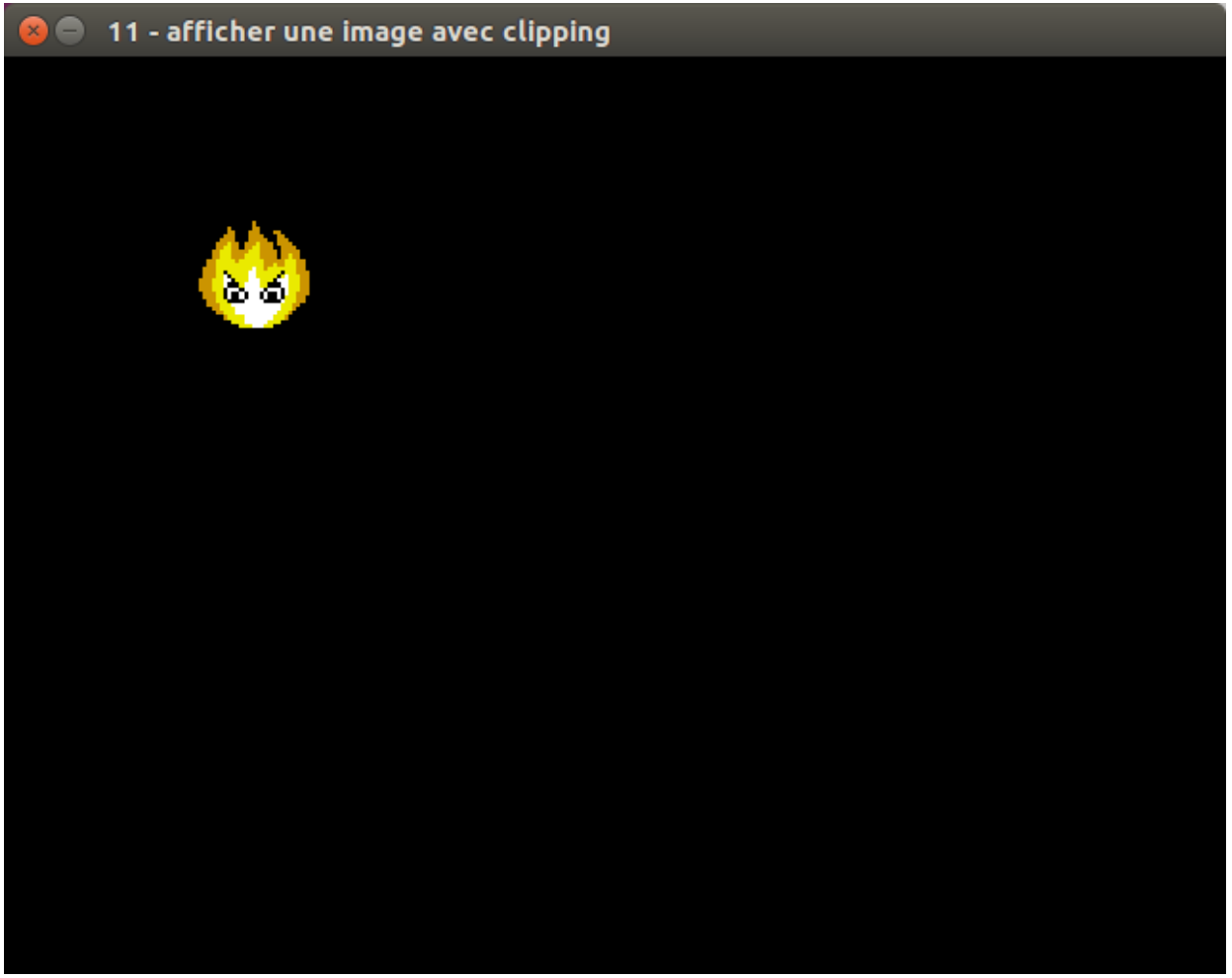


Afficher une image avec clipping sur le 3ième sprite (numéro 2 en C)

```
SDL_Rect clipping_rectangle;

clipping_rectangle.x = 2 * SPRITE_LARGEUR;
clipping_rectangle.y = 0;
clipping_rectangle.w = SPRITE_LARGEUR;
clipping_rectangle.h = SPRITE_HAUTEUR;

SDL_RenderCopy(renderer, texture, &clipping_rectangle, &rectangle);
```



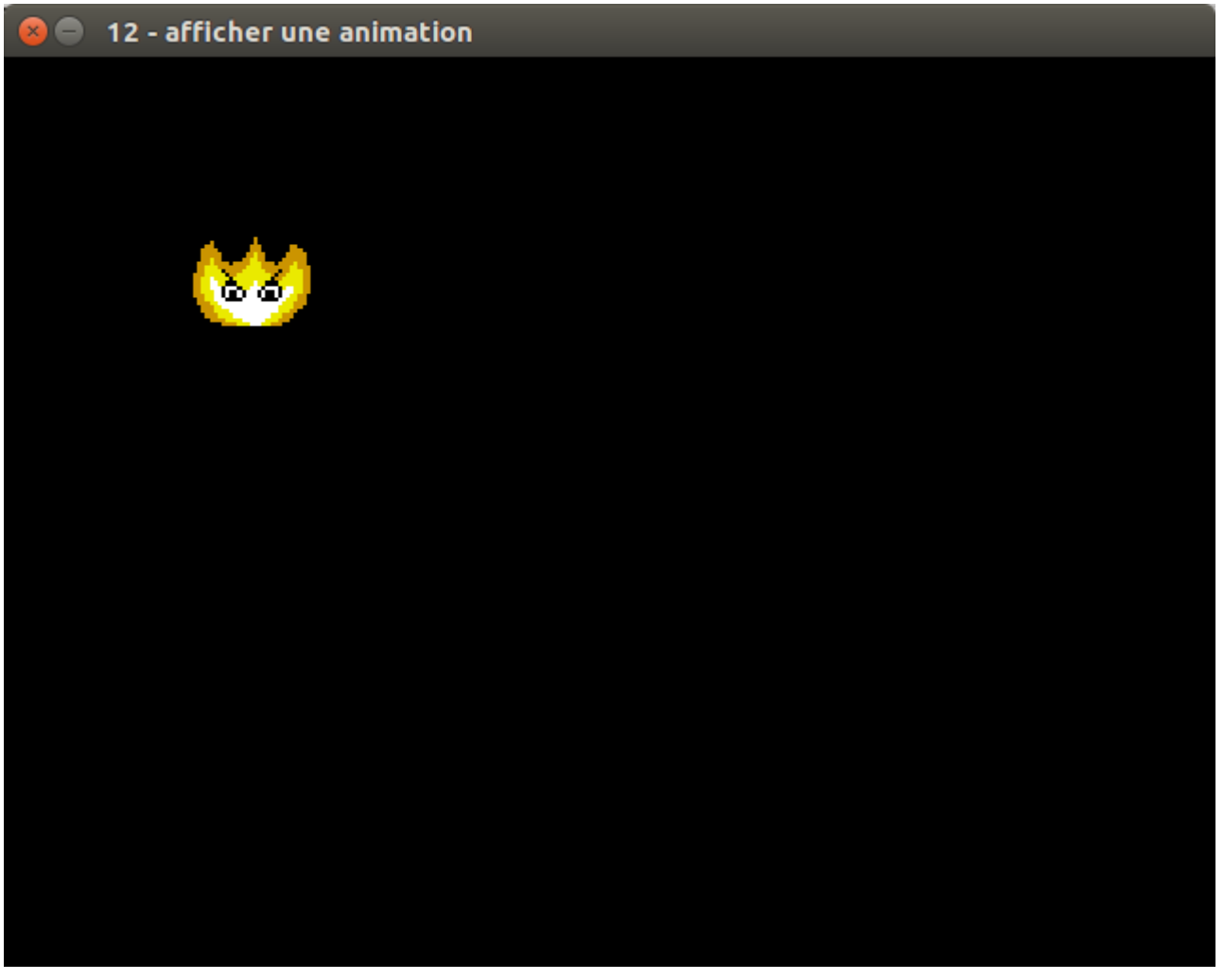
Afficher une image avec clipping sur le 3ième sprite (numéro 2 en C)



Afficher une animation en fonction à un instant T

```
int i;
SDL_Rect clipping_rectangle;

i = (SDL_GetTicks() / DUREE_PAR_ANIMATION) % NB_SPRITES_TOTAL;
clipping_rectangle.x = i * SPRITE_LARGEUR;
clipping_rectangle.y = 0;
clipping_rectangle.w = SPRITE_LARGEUR;
clipping_rectangle.h = SPRITE_HAUTEUR;

SDL_RenderCopy(renderer, texture, &clipping_rectangle, &rectangle);
```

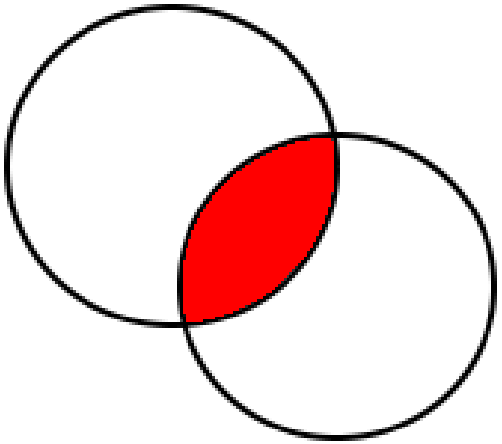
Afficher une animation en fonction à un instant T (le sprite est animé)



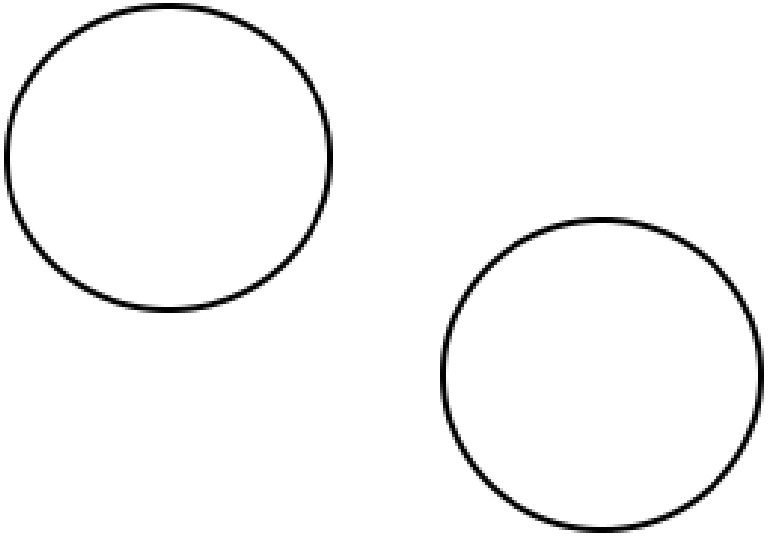
Les collisions 2d
La théorie des collisions

Les collisions

Collision = VRAI



Collision = FAUX



Exemple : La fonction TestCercleCercle

```
int TestCercleCercle(
        float  cercle1X, float  cercle1Y, float  cercle1Rayon,
        float  cercle2X, float  cercle2Y, float  cercle2Rayon)
{
        // ...
}
```

Les collisions

- Nécessaire dans les jeux vidéos 2d ou 3d
- Par exemple : dans un jeu de foot quand le ballon tombe en touchant le sol, il rebondit car il y a une collision avec la sol
- Fonctions mathématiques qui retourne VRAI(1) ou FAUX (0)

Exemple : la fonction TestPointCercle

```
int TestPointCercle(
        float pointX, float pointY,
        float cercleX, float cercleY, float cercleRayon)
{
        // ...
}
```

Exemple : la fonction TestPointRectangle

```
int TestPointRectangle(
        float pointX, float  pointY,
        float rectangleX, float rectangleY, float rectangleLargeur, float rectangleHauteur)
{
        // ...
}
```

Exemple : la fonction TestRectangleRectangle

```
int TestRectangleRectangle(
        float rectangle1X, float rectangle1Y, float rectangle1Largeur, float rectangle1Hauteur
        float rectangle2X, float rectangle2Y, float rectangle2Largeur, float rectangle2Hauteur)
{
        // ...
}
```