

The background is a light blue sky with a large yellow sun in the top right corner. There are three white, fluffy clouds. The bottom of the image shows a green rolling landscape with two stylized green trees with brown trunks. Small pink flowers are scattered on the grass.

# Projet 1A

Ben Mahmoud Landolsi Houssem

2020



# Objectives

Gérer un projet  
Présentation projet



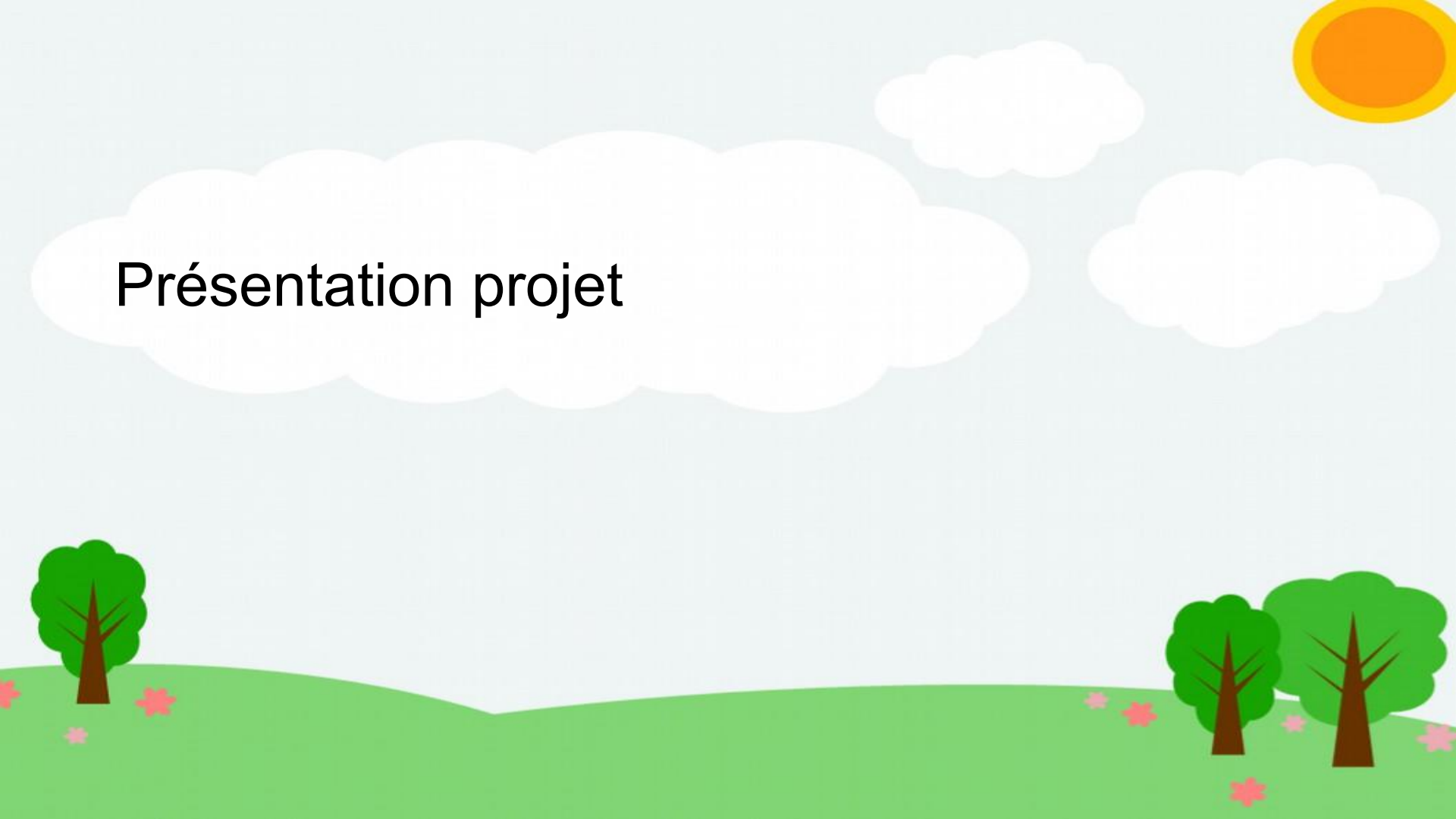
# Gérer un projet

Citation et organization des tâches

Manipulations des outils de travaux

Manipulation du temps


# Présentation projet





# Agenda :

1. De quoi s'agit-il ?
2. Conception
3. Description
4. Outils de travail
5. Code et Prototype du jeu
7. Démonstration



# 1. De quoi s'agit-il ?

Création d'un jeu vidéo

## 2. Conception

Nom : Mario World

Type : Jeu 2D



### 3. Description

Personnages : Mario / Luigi

Environnement: Scène sombre de bataille

Musique: Musique du Fond(début) /externe(Fin)







## 4. Outils de travail

Système d'exploitation Linux(Ubuntu)

Programmation en C

Terminal Ubuntu+outils de développements

Bibliothèque SDL



# Programmation en C

Implémentation du code du jeu avec la programmation en C

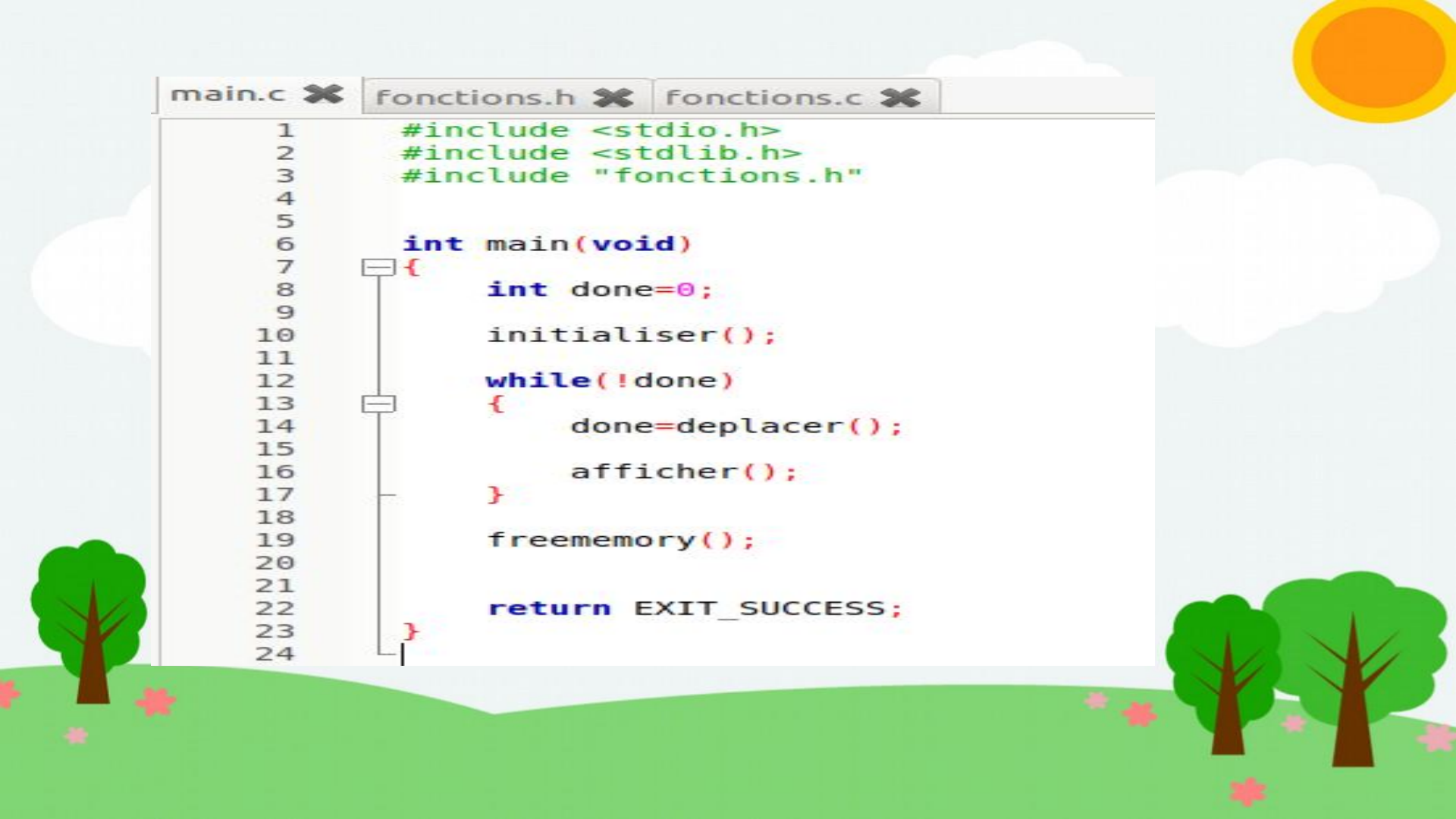
Le code va contenir trois fichiers importants: Main.c,  
Fonctions.h et Fonctions.c



# Main.c

Contient la déclaration des variables

L'appel des fonctions de la boucle du jeu



The image shows a code editor window with three tabs: 'main.c', 'fonctions.h', and 'fonctions.c'. The 'main.c' tab is active, displaying a C program. The code includes headers for `<stdio.h>`, `<stdlib.h>`, and `"fonctions.h"`. The `main` function is defined with a `void` parameter. Inside, it declares an `int done=0;`, calls `initialiser();`, enters a `while(!done)` loop, and inside the loop, it calls `done=deplacer();` and `afficher();`. After the loop, it calls `freememory();` and returns `EXIT_SUCCESS`. The code is formatted with line numbers on the left and a vertical line with markers indicating the structure of the code blocks.

```
1      #include <stdio.h>
2      #include <stdlib.h>
3      #include "fonctions.h"
4
5
6      int main(void)
7      {
8          int done=0;
9
10         initialiser();
11
12         while(!done)
13         {
14             done=deplacer();
15
16             afficher();
17         }
18
19         freememory();
20
21
22         return EXIT_SUCCESS;
23     }
24
```



# Fonctions.h

- Contient : les structures et les entêtes des fonctions

main.c ✕ fonctions.h ✕ fonctions.c ✕

```
1  #ifndef FONCTIONS_H_INCLUDED
2  #define FONCTIONS_H_INCLUDED
3  #include <SDL/SDL.h>
4
5  int initialiser();
6  void afficher();
7  int deplacer();
8  /*
9  | signature de la fonction collision.
10 | */
11 int collision(SDL_Rect p1, SDL_Rect p2);
12
13 void freememory();
14
15
16
17 #endif // FONCTIONS_H_INCLUDED
18
```



# Fonctions.c

Contient l'implémentation des fonctions

```

1  #include "SDL/SDL.h"
2  #include <SDL/SDL_mixer.h>
3  #include <SDL/SDL_ttf.h>
4  #include "fonctions.h"
5
6  SDL_Surface *screen=NULL; //reference the backbuffer
7  Mix_Music *music; //Construct Mix_Music pointer
8  Mix_Music *musicgameover;
9  SDL_Surface *mario=NULL; //reference our image
10 SDL_Surface *luigi=NULL;
11 SDL_Rect positionmario;
12 SDL_Rect positionluigi; //rect to describe the source destination region of our blit
13 SDL_Rect positiontext;
14 SDL_Rect positiontextgameover;
15 SDL_Event event;
16 TTF_Font *police = NULL;
17 TTF_Font *policegameover=NULL;
18 SDL_Surface *text;
19 SDL_Surface *textgameover;
20 int gameover=0;
21
22
23 int initialiser()
24 {
25
26     if(SDL_Init(SDL_INIT_VIDEO|SDL_INIT_AUDIO)!=0)
27     {
28         printf("unable to initialize SDL: %s\n",SDL_GetError());
29         return 1;
30     }

```

```

31
32     if(TTF_Init()== -1)
33     {
34         printf("Erreur d'initialisation de TTF_Init : %s\n", TTF_GetError());
35         return 1;
36     }
37
38
39     police = TTF_OpenFont("signature.ttf", 65);
40
41     policegameover= TTF_OpenFont("gameover.ttf",50);
42
43     SDL_Color couleurrouge = {250, 0, 0};
44
45     text = TTF_RenderText_Blended(police,"Mario World",couleurrouge);
46
47     textgameover = TTF_RenderText_Blended(policegameover,"Game Over",couleurrouge);
48
49     //API Mixer Initialization
50     if(Mix_OpenAudio(44100,MIX_DEFAULT_FORMAT,MIX_DEFAULT_CHANNELS,1024)==-1)
51     {
52         printf("%s",Mix_GetError());
53     }
54
55
56     music = Mix_LoadMUS("smbss_intro-cartoon.mp3");//load the music
57     Mix_PlayMusic(music,-1);//play music forever("-1")
58
59     musicgameover = Mix_LoadMUS("smb_gameover.wav");//load the music
60

```



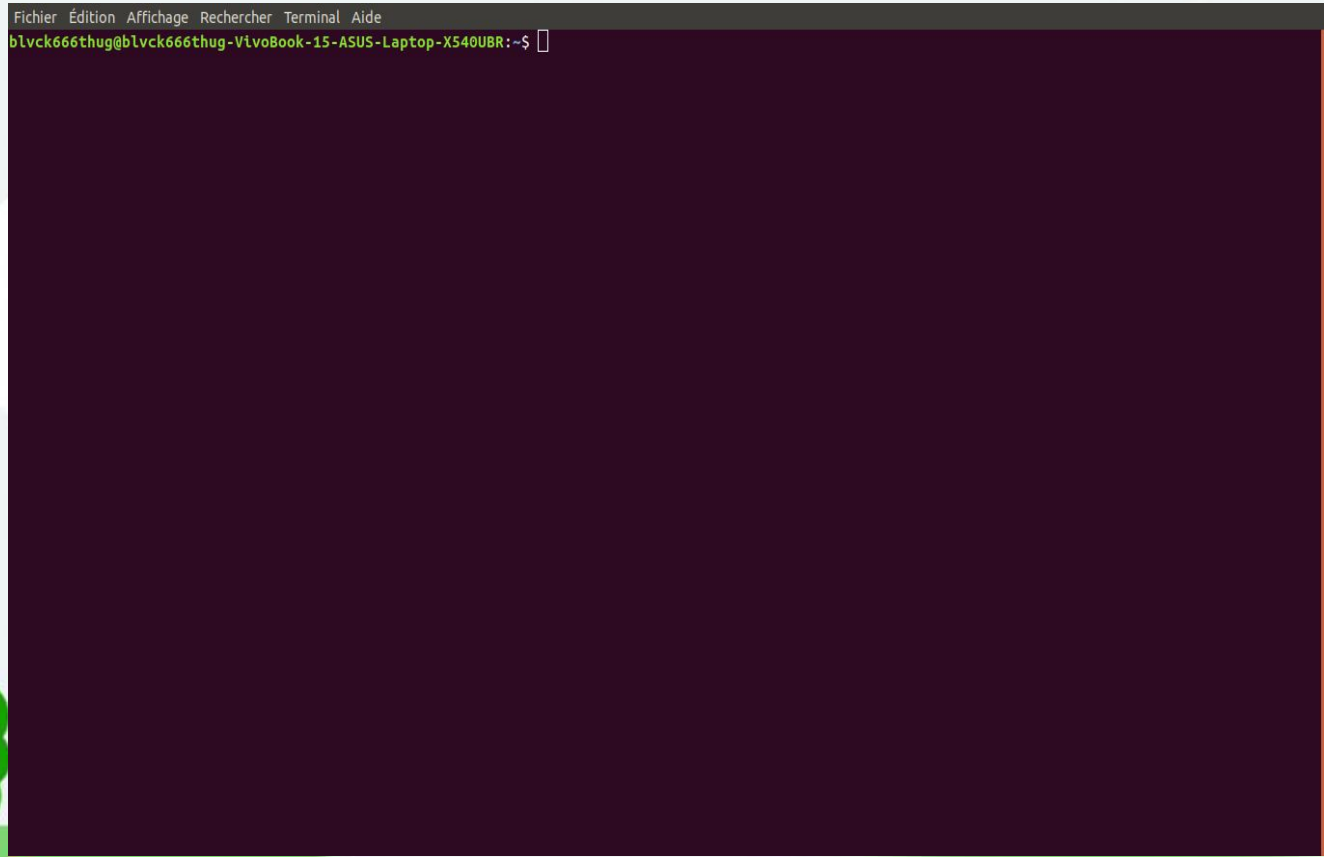
The background of the slide is a stylized landscape. It features a light blue sky with three white, fluffy clouds. In the top right corner, there is a bright yellow sun with an orange center. The ground is represented by green rolling hills. On the left hill, there is a single green tree with a brown trunk. On the right hill, there are two green trees with brown trunks. Small pink flowers are scattered across the green hills.

# Terminal Ubuntu

Un Programme qui émule une console dans une interface graphique  
Il permet de lancer des commandes pour manipuler notre ordinateur

Fichier Édition Affichage Rechercher Terminal Aide

blvck666thug@blvck666thug-VivoBook-15-ASUS-Laptop-X540UBR:~\$





# Outils de développement Terminal

GCC (GNU Compiler Collection)

Make (Gestionnaire de Compilation Make)

GDB (Gnu DeBugger)

Git-Hub



# GCC (GNU Compiler Collection)

Une suite de logiciels libres de compilation utilisé dans le monde Linux dès qu'on veut transcrire du code source en langage machine




# Make (Gestionnaire de Compilation Make)

Automatiser la phase de compilation.

Automatiser l'installation.

Mise en paquetage .

L'élimination de fichiers temporaires créés par votre éditeur de texte et le compilateur.

The background is a light blue sky with a large yellow sun in the top right corner. There are three white, fluffy clouds. The ground is a green hill with several green trees and small pink flowers. The text 'Git-Hub' is written in black on the left cloud.

# Git-Hub

## Définition

### Info P1A sur Git



# Définition

Un service web d'hébergement et de gestion de développement de logiciels



Info P1A sur Git

Code de P1A sur Git

Documentation de P1A sur Git



The background is a light blue sky with a large yellow sun in the top right corner. There are three white, fluffy clouds. The bottom of the image shows a green rolling landscape with two stylized green trees on the left and two on the right, with small pink flowers scattered on the grass.

Code de P1A sur Git

<https://github.com/BLVCK666Thug/MarioWorld>

The background is a light blue sky with a large yellow sun in the top right corner. There are three white, fluffy clouds. The bottom of the image shows a green rolling landscape with two stylized green trees on the left and two on the right, with small pink flowers scattered on the grass.

Documentation de P1A sur Git

<https://github.com/BLVCK666Thug/MarioWorld/wiki/Documentation-du-code>



# Bibliothèque SDL

Simple Direct Media Layer, bibliothèque de développement multi-plateformes qui fournit un accès bas niveau au matériel audio, clavier, souris, manette de jeu, etc

De ce fait, elle permet de développer des programmes gérant le son, la vidéo, le clavier, la souris et le lecteur CD [1][2]



# Création d'une fenêtre graphique

La surface graphique ou le jeu va s'afficher pour  
le joueur créer avec la bibliothèque SDL

# Constantes de chargement SDL

Liste des constantes de chargement de la SDL	
Constante	Description
SDL_INIT_VIDEO	Charge le système d'affichage (vidéo). C'est la partie que nous chargerons le plus souvent.
SDL_INIT_AUDIO	Charge le système de son. Vous permettra donc par exemple de jouer de la musique.
SDL_INIT_CDROM	Charge le système de CD-ROM. Vous permettra de manipuler votre lecteur de CD-ROM
SDL_INIT_JOYSTICK	Charge le système de gestion du joystick.
SDL_INIT EVERYTHING	Charge tous les systèmes listés ci-dessus à la fois.

- Et pour quitter on utilise `SDL_QUIT()`

# Initialisation et chargement

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL/SDL.h>
int main(int argc, char *argv[])
{
    SDL_Init(SDL_INIT_VIDEO); // Démarrage de la SDL (ici : chargement du système
    vidéo)
    /*
    La SDL est chargée.
    Vous pouvez mettre ici le contenu de votre programme
    */
    SDL_Quit(); // Arrêt de la SDL (libération de la mémoire).
    return 0;
}
```

# Ouverture fenêtre

```
int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;

    SDL_Init(SDL_INIT_VIDEO);

    ecran = SDL_SetVideoMode(640, 480, 32, SDL_HWSURFACE);
    SDL_WM_SetCaption("Ma super fenêtre SDL !", NULL);

    // Coloration de la surface ecran en bleu-vert
    SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 17, 206, 112));

    SDL_Flip(ecran); /* Mise à jour de l'écran avec sa nouvelle couleur */
    pause();

    SDL_Quit();

    return EXIT_SUCCESS;
}
```

```
void pause()
{
    int continuer = 1;
    SDL_Event event;
    while (continuer)
    {
        SDL_WaitEvent(&event);
        switch(event.type)
        {
            case SDL_QUIT:
                continuer = 0;
        }
    }
}
```



# Affichage d'image

Avec SDL on peut charger et afficher une image dans une plateforme graphique créer par SDL



# Affichage d'image

```
int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL, *imageDeFond = NULL;
    SDL_Rect positionFond;
    positionFond.x = 0;
    positionFond.y = 0;
    SDL_Init(SDL_INIT_VIDEO);
    ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE);
    SDL_WM_SetCaption("Chargement d'images en SDL", NULL);
    /* Chargement d'une image Bitmap dans une surface */
    imageDeFond = SDL_LoadBMP("lac_en_montagne.bmp");
    /* On blitte par-dessus l'écran */
    SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
    SDL_Flip(ecran);
    pause();
    SDL_FreeSurface(imageDeFond); /* On libère la surface */
    SDL_Quit();
    return EXIT_SUCCESS;
}
```



# Gestion des évènements

SDL nous permet de gestionner les évènements qui se déroulent dans notre application SDL

# Gestion des évènements

```
int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;
    SDL_Event event; /* La variable contenant l'événement */
    int continuer = 1; /* Notre booléen pour la boucle */
    SDL_Init(SDL_INIT_VIDEO);
    écran = SDL_SetVideoMode(640, 480, 32, SDL_HWSURFACE);
    SDL_WM_SetCaption("Gestion des événements en SDL", NULL);
    while (continuer) /* TANT QUE la variable ne vaut pas 0 */
    {
        SDL_WaitEvent(&event); /* On attend un événement qu'on récupère dans event */
        switch(event.type) /* On teste le type d'événement */
        {
            case SDL_QUIT: /* Si c'est un événement QUITTER */
                continuer = 0; /* On met le booléen à 0, donc la boucle va s'arrêter */
                break;
        }
    }
    SDL_Quit();
    return EXIT_SUCCESS;
}
```



## 5. Code et prototype du jeu

Initialisation du jeu

Affichage du jeu(scene+texte+personnage).

Déplacement de 2 entités(Mario/luigi)

Collision bounding box



# Initialisation du jeu

Déclaration des variables et pointeur et  
bibliothèque de travail

Déclaration de leurs types

Implementation dans une fonction appelée :

« int initialiser() »

# Initialisation du jeu

```
1  #include "SDL/SDL.h"
2  #include <SDL/SDL_mixer.h>
3  #include <SDL/SDL_ttf.h>
4  #include "fonctions.h"
5
6  SDL_Surface *screen=NULL; //reference the backbuffer
7  Mix_Music *music;//Construct Mix_Music pointer
8  Mix_Music *musicgameover;
9  SDL_Surface *mario=NULL; //reference our image
10 SDL_Surface *luigi=NULL;
11 SDL_Rect positionmario;
12 SDL_Rect positionluigi;//rect to describe the source destination region of our blit
13 SDL_Rect positiontext;
14 SDL_Rect positiontextgameover;
15 SDL_Event event;
16 TTF_Font *police = NULL;
17 TTF_Font *policegameover=NULL;
18 SDL_Surface *text;
19 SDL_Surface *textgameover;
20 int gameover=0;
21
```

# Implementation dans « int initialiser() »

```
int initialiser()
{
    if(SDL_Init(SDL_INIT_VIDEO|SDL_INIT_AUDIO)!=0)
    {
        printf("unable to initialize SDL: %s\n",SDL_GetError());
        return 1;
    }

    if(TTF_Init()== -1)
    {
        printf("Erreur d'initialisation de TTF_Init : %s\n", TTF_GetError());
        return 1;
    }

    police = TTF_OpenFont("signature.ttf", 65);
    policegameover= TTF_OpenFont("gameover.ttf",50);
    SDL_Color couleurrouge = {250, 0, 0};
    text = TTF_RenderText_Blended(police,"Mario World",couleurrouge);
    textgameover = TTF_RenderText_Blended(policegameover,"Game Over",couleurrouge);
}
```



# Implementation dans « int initialiser() »

```
//API Mixer Initialization
if(Mix_OpenAudio(44100,MIX_DEFAULT_FORMAT,MIX_DEFAULT_CHANNELS,1024)==-1)
{
    printf("%s",Mix_GetError());
}

music = Mix_LoadMUS("smbss_intro-cartoon.mp3");//load the music
Mix_PlayMusic(music,-1);//play music forever("-1")

musicgameover = Mix_LoadMUS("smb_gameover.wav");//load the music

screen = SDL_SetVideoMode( 600, 400, 16, SDL_HWSURFACE|SDL_DOUBLEBUF);
SDL_WM_SetCaption("Gestion des évènements en SDL", NULL);

if (screen==NULL)
{
    printf("Unable to set video mode : %s\n",SDL_GetError());
    return 1;
}
```



# Implementation dans « int initialiser() »

```
//Load the bitmap into the image surface, and check for success
mario=SDL_LoadBMP("Mario.bmp"); //you can use IMG_load
```

```
if (mario==NULL)
{
    printf("Unable to load bitmap:%s\n",SDL_GetError());
    return 1;
}
```

```
luigi=SDL_LoadBMP("luigi.bmp"); //you can use IMG_load
```

```
if (luigi==NULL)
{
    printf("Unable to load bitmap:%s\n",SDL_GetError());
    return 1;
}
```

```
//Construct the source rectangle for our blit
```

```
positionmario.x=0;
positionmario.y=200;
positionmario.w=mario->w;
positionmario.h=mario->h;
```

```
positionluigi.x=screen->w-luigi->w;
positionluigi.y=200;
positionluigi.w=luigi->w;
positionluigi.h=luigi->h;
```

```
positiontextgameover.x=150;
positiontextgameover.y=150;
```

```
SDL_EnableKeyRepeat(10, 10); /* Activation de la répétition des touches */
```

# Affichage du jeu

Implémentation dans une fonction appelée :

« void afficher() »



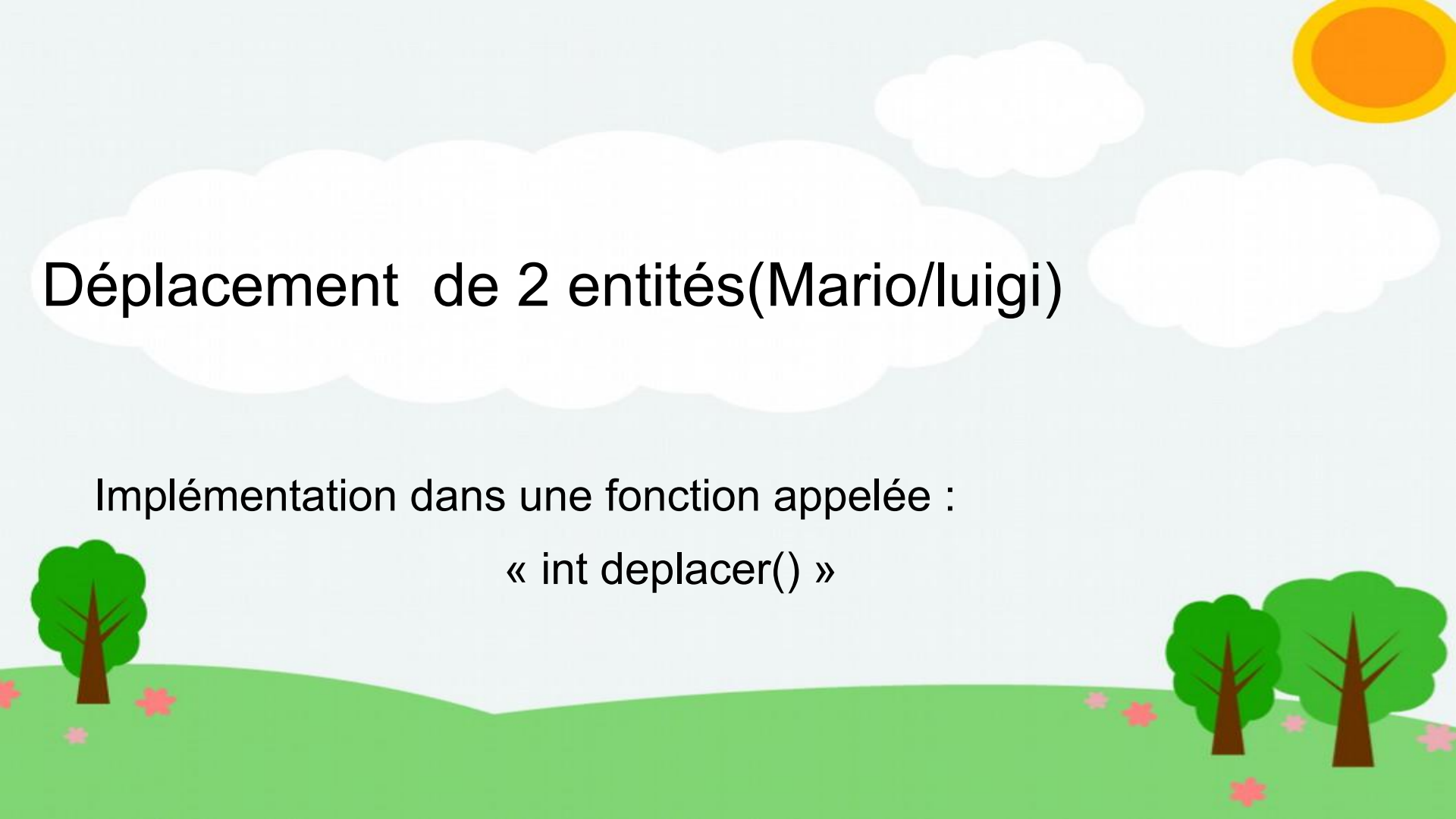
# Affichage du jeu

```
void afficher()
{
    SDL_FillRect(screen, NULL, SDL_MapRGB(screen->format, 0, 0, 0));

    SDL_BlitSurface(text, NULL, screen, &positiontext);

    //Blit the image to the backbuffer|
    SDL_BlitSurface(mario, NULL, screen, &positionmario);

    SDL_BlitSurface(luigi, NULL, screen, &positionluigi);
    if(gameover==1)
    {
        SDL_BlitSurface(textgameover, NULL, screen, &positiontextgameover);
    }
    //flip the backbuffer to the primary Hardware Video Memory
    SDL_Flip(screen);
}
```



# Déplacement de 2 entités(Mario/Luigi)

Implémentation dans une fonction appelée :

« int deplacer() »

# Déplacement de 2 entités(Mario/luigi)

```
int deplacer()
{
    int done=0;
    while (SDL_PollEvent(&event))
    {
        //check for messages
        switch(event.type)
        {
            case SDL_QUIT:
                done = 1;
                printf("Event SDLquit!\n");
                break;
            //check for keypresses
            case SDL_KEYDOWN:
                printf("Event SDL KeyDown!\n");
                switch(event.key.keysym.sym)
                {
                    case SDLK_ESCAPE:
                        done=1;
                        printf("Event SDLESCAPE!\n");
                        break;
                    case SDLK_UP:
                        if(gameover==1)
                        {
                            break;
                        }
                        positionmario.y--;
                        printf("Event SDLK UP !\n");
                        break;
                }
            }
        }
    }
}
```

# Déplacement de 2 entités(Mario/luigi)

```
case SDLK_DOWN:
    if(gameover==1)
    {
        break;
    }
    positionmario.y++;
    printf("Event SDLK DOWN!\n");
    break;
case SDLK_RIGHT:
    if(gameover==1)
    {
        break;
    }
    positionmario.x++;
    printf("Event SDLK RIGHT!\n");
    break;
case SDLK_LEFT:
    if(gameover==1)
    {
        break;
    }
    positionmario.x--;
    printf("Event SDLK LEFT!\n");
    break;
```

# Déplacement de 2 entités(Mario/luigi)

```
//Déplacement luigi.  
case SDLK_z:  
    if(gameover==1)  
    {  
        break;  
    }  
    positionluigi.y--;  
    printf("Event SDLK UP !\n");  
    break;  
case SDLK_s:  
    if(gameover==1)  
    {  
        break;  
    }  
    positionluigi.y++;  
    printf("Event SDLK DOWN!\n");  
    break;  
case SDLK_d:  
    if(gameover==1)  
    {  
        break;  
    }  
    positionluigi.x++;  
    printf("Event SDLK RIGHT!\n");  
    break;  
case SDLK_q:  
    if(gameover==1)  
    {  
        break;  
    }
```

# Déplacement de 2 entités(Mario/luigi)

```
        positionluigi.x--,
        printf("Event SDLK LEFT!\n");
        break;
    }
    break;
}
if(collision(positionmario,positionluigi)==0)
{
    printf("Il n y a pas de collision entre mario et luigi");
}
else
{
    printf("I y a collision entre mario et luigi");
    gameover=1;
    Mix_HaltMusic();
    Mix_PlayMusic(musicgameover,0);
}
return done;
}
```





# Collision bounding box

Détection des collisions

Repérer des déplacements interdits

(choc direct entre deux objets)

# Collision bounding box

```
/*  
 * Implementation de la fonction collision .  
 */  
int collision(SDL_Rect p1, SDL_Rect p2)  
{  
    if(p2.y+p2.h<p1.y || p1.y+p1.h<p2.y || p1.x+p1.w<p2.x || p2.x+p2.w<p1.x)  
    {  
        return 0; //pas de collision.  
    }  
    else  
    {  
        return 1; //il y a collision.  
    }  
};
```

## 7. Démonstration du jeu

Début du jeu

Déplacement de Mario et Luigi

Fin du jeu

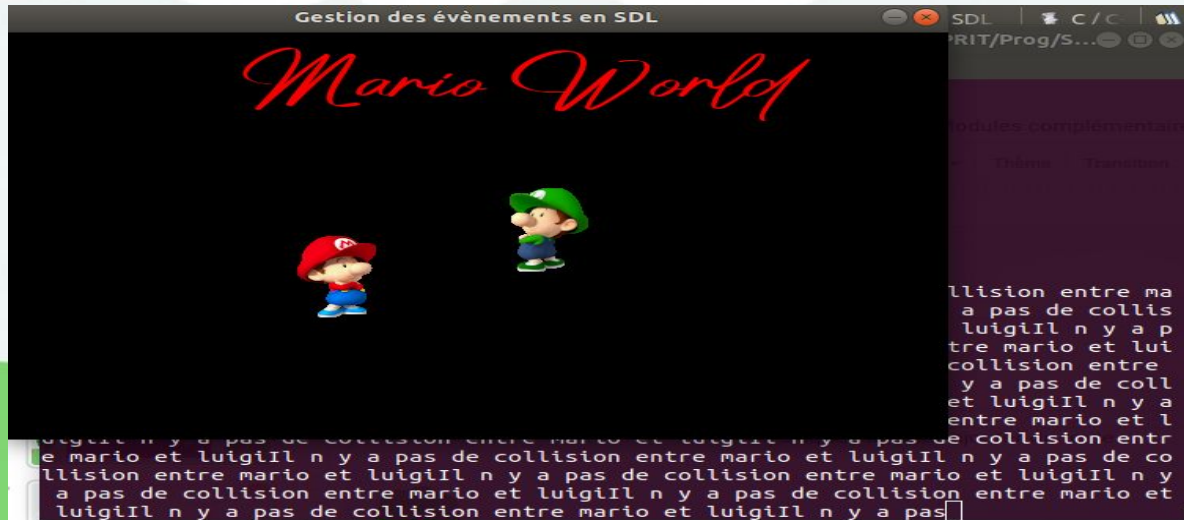
Video



# Début du jeu



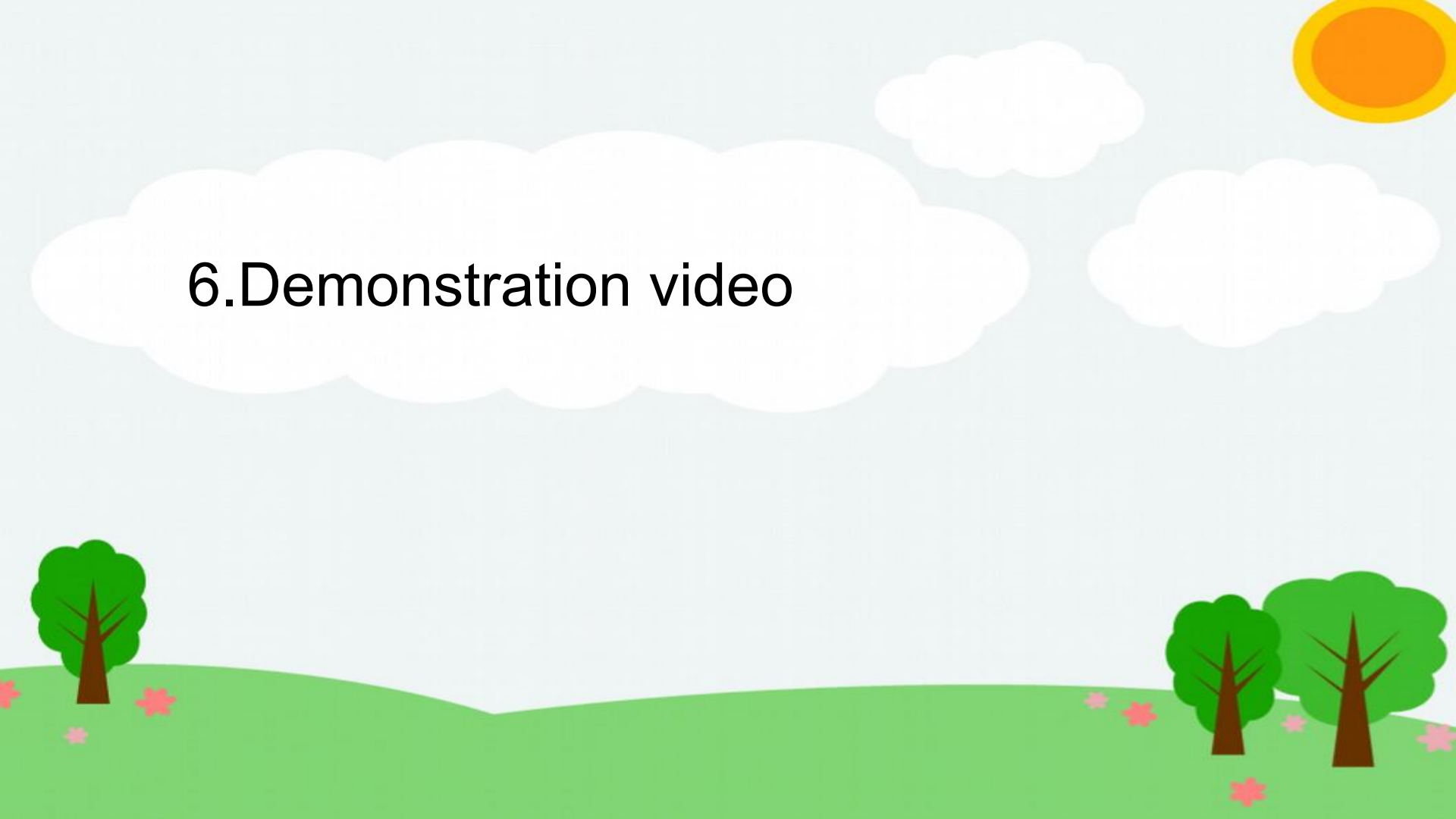
# Déplacement de Mario et Luigi



Fin du jeu



## 6.Demonstration video





Demonstration live demo





# Perspectives

Background

Obstacles

Animation

Déplacement avec accélération et Saut



Merci pour votre attention