**SDL 2**

Partie 1

Question 1 :

La SDL 2 (ou Simple DirectMedia Layer) est une bibliothèque logicielle permettant de gérer l’affichage vidéo et l’audio.

Question 2 : Que peut faire la SDL 2 ?

La SDL 2 permet d’afficher un rendu graphique 2D, utile à la création de jeux vidéo ou d’interfaces graphiques. La SDL 2, contrairement à la SDL 1, utilise les ressources de la carte graphique. Elle permet également de gérer plusieurs fenêtres, leurs tailles, des boites de dialogues par exemple.

Question 3 : Sur quelles plateformes fonctionne la SDL 2 ?

La SDL 2 fonctionne officiellement sur Windows, Mac OS X, Linux, iOS, et Android.

Question 4 : Comment obtenir et installer la SDL 2 sur un EDI ? Donnez les étapes de la configuration d’un projet SDL2 avec CodeBlocks (gcc).

Tout d’abord, nous devons récupérer l’archive contenant SDL 2 sur le siteUne image contenant capture d’écran

Description générée automatiquement

A partir d’ici, nous téléchargerons l’archive SDL2-devel-2.0.10-mingw.tar.gz. On extrait le fichier, que l’on place ensuite dans un dossier prévu pour accueillir cette librairie (par exemple dans un dossier SDL\_2 dans programmes files, à la racine du fichier CodeBlocks)

Puis nous ouvrons CodeBlocks, dans ‘Settings’ puis ‘Compiler’, on sélectionne ‘Linker Settings’ et on y ajoute les chemins de notre librairie comme tel :

* C:\Program Files (x86)\CodeBlocks\SDL2-2.0.10\i686-w64-mingw32\lib\libSDL2.a
* C:\Program Files (x86)\CodeBlocks\SDL2-2.0.10\i686-w64-mingw32\lib\libSDL2.dll.a

Attention à bien sélectionner le fichier correspondant à votre ordinateur : i686 pour les processeurs 32 bit, ou x86-64 pour les processeurs 64 bits.

Un fois ceci fait, on appuie sur ok et on va sur l’onglet ‘Search directories’. Dans l’onglet ‘Compiler’ on entre les chemins pour indiquer au compilateur où se situe le dossier include et le dossier lib pour la SDL2.

* C:\Program Files (x86)\CodeBlocks\SDL2-2.0.10\i686-w64-mingw32\include
* C:\Program Files (x86)\CodeBlocks\SDL2-2.0.10\i686-w64-mingw32\lib

Question 5 : Quelles sont les différences entre SDL 1 et SDL 2 ?

La plus grosse différence entre SDL et SDL2 est que la SDL utilise le processeur d’un PC là où la SDL2 utilise la carte graphique de celui-ci.

Question 6 : Pour utiliser la SDL 2 dans votre projet, vous devez initialiser l’instance SDL 2, quelles sont les étapes d’initialisation à réaliser pour utiliser la lib SDL 2, donnez le code correspondant et expliquez précisément chaque instruction.

#include <SDL2/SDL.h>

int main(int argc, char \*argv[])

{

if(SDL\_Init(SDL\_INIT\_VIDEO) < 0)//initialisation de la SDL

{

printf("Erreur d'initialisation de la SDL : %s",SDL\_GetError()); //on affiche l'erreur

return EXIT\_FAILURE; //on sort du programme pour éviter de plus gros problèmes

}

SDL\_Quit(); //on quitte la SDL

return EXIT\_SUCESS;

}

Question 7 : Comment créer une fenêtre avec SDL 2, donnez et expliquez le code correspondant à la création d’une fenêtre avec SDL 2.

window = SDL\_CreateWindow("SDL2", SDL\_WINDOWPOS\_CENTERED, SDL\_WINDOWPOS\_CENTERED, 640, 480, SDL\_WINDOW\_SHOWN);

if(NULL == window)

{

fprintf(stderr, "Erreur SDL\_CreateWindow : %s", SDL\_GetError());

return EXIT\_FAILURE;

}

SDL\_DestroyWindow(window);

Question 8 :

SDL\_CreateWindow("SDL\_Application",SDL\_WINDOWPOS\_CENTERED, SDL\_WINDOWPOS\_CENTERED, 1024, 780, SDL\_WINDOW\_SHOWN);

SDL\_CreateRenderer(pWindow,-1,SDL\_RENDERER\_PRESENTVSYNC);

SDL\_RenderPresent(pRenderer);

Question 9 :

renderer = SDL\_CreateRenderer(window, -1, SDL\_RENDERER\_ACCELERATED);

if(NULL == renderer)

{

fprintf(stderr, "Erreur SDL\_CreateRenderer : %s", SDL\_GetError());

return EXIT\_FAILURE;

}

if(0 != SDL\_SetRenderDrawColor(renderer, 125, 125, 125, 255))

{

fprintf(stderr, "Erreur SDL\_SetRenderDrawColor : %s", SDL\_GetError());

return EXIT\_FAILURE;

}

if(0 != SDL\_RenderClear(renderer))

{

fprintf(stderr, "Erreur SDL\_SetRenderDrawColor : %s", SDL\_GetError());

return EXIT\_FAILURE;

}

Question 10 :

SDL\_SetRenderDrawColor(renderer, 0, 0, 255, 255);

SDL\_RenderDrawPoint(renderer, 150, 150);

SDL\_RenderPresent(renderer);

SDL\_Rect rect = {100, 100, 100, 100};

SDL\_RenderFillRect(renderer, &rect);

SDL\_RenderPresent(renderer);

Question 11 :

On gère la couleur grâce à la structure SDL\_Color qui prends en paramètre la valeur ‘rgba’ d’une couleur, allant de 0 à 255.

SDL\_Color orange = {255, 127, 40, 255};

Question 12 :

if(SDL\_SetRenderDrawColor(renderer, color.r, color.g, color.b, color.a) < 0)

return EXIT\_FAILURE;

if(SDL\_RenderClear(renderer) < 0)

return EXIT\_FAILURE;

Question 13 :

texture = SDL\_CreateTexture(renderer, SDL\_PIXELFORMAT\_RGBA8888,

SDL\_TEXTUREACCES\_TARGET, 200, 200);

SDL\_Rect rect = {50, 50, 100, 100};

SDL\_SetRenderDrawColor(renderer, 150, 0, 150, 255);

SDL\_SetRenderTarget(renderer, texture);

SDL\_RenderFillRect(renderer, &rect); /\*à enlever si on ne veut pas remplir le carré\*/

SDL\_SetRenderTarget(renderer, NULL);

Question 14 :

int SDL\_RenderDrawPoint(SDL\_Renderer\* renderer, int x, int y)

et

int SDL\_RenderDrawLine(SDL\_Renderer\* renderer, int x1, int y1, int x2, int y2)

Question 15 :

SDL\_RenderPresent permet de mettre à jour l’affichage, tandis que SDL\_RenderClear permet de ‘nettoyer’ l’écran.

Question 16 :

SDL\_Delay permet d’attendre un certain nombre de milliseconde avant de continuer le programme.

Question 17 :

Les surfaces sont l’équivalent des textures, mais bien moins performantes que ces dernières. Cependant, les surfaces peuvent êtres modifiées pixel par pixel plus facilement que sur les textures.

Question 18 :

SDL\_Surface \*surface = NULL;

surface = SDL\_CreateRGBSurface(0, 300, 200, 32, 0, 0, 0, 0);

if(NULL == surface)

{

fprintf(stderr, "Erreur SDL\_CreateRGBSurface : %s", SDL\_GetError());

return EXIT\_FAILURE;

}

SDL\_FreeSurface(surface);

Question 19 :

int SDL\_FillRect(SDL\_Surface\* dst, const SDL\_Rect\* rect, Uint32 color)

Question 20 : SDL\_BlitSurface(..) : donnez le code pour tester cette méthode.

Question 21 :

Une texture est un paquet de pixels affichant une image.

SDL\_Texture\* SDL\_CreateTexture(SDL\_Renderer\* renderer, Uint32 format, int access, int w, int h)