SDL 2

Partie 1

Question 1 :

La SDL 2, Simple DirectMedia Layer, est la deuxième version d'une bibliothèque logicielle permettant de gérer l'affichage vidéo et les systèmes audio.

Question 2 :

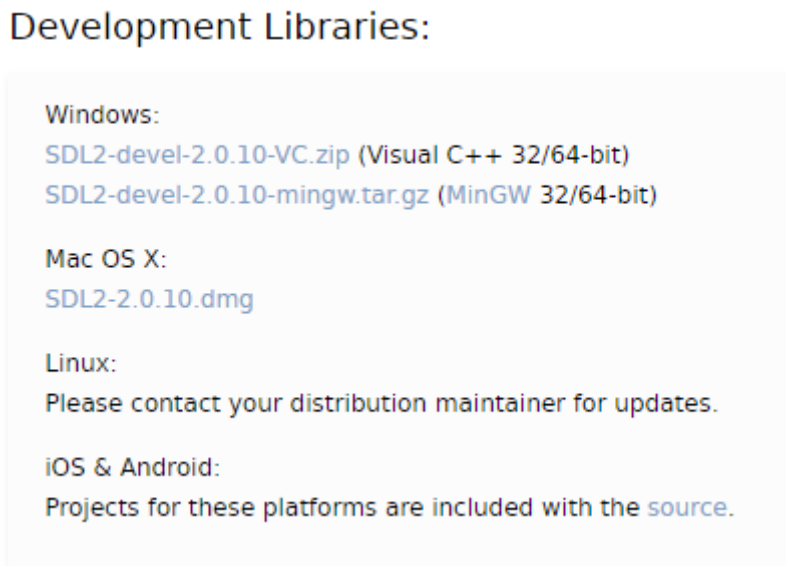
La SDL 2 permet d'afficher un rendu graphique 2D, permettant de créer des interfaces graphiques ou des jeux vidéo.

Question 3 :

La SDL 2 fonctionne sur les systèmes d'exploitation 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 site https://www.libsdl.org/download-2.0.php



Nous téléchargerons l'archive SDL2-devel-2.0.10-mingw.tar.gz. On extrait le fichier, que l'on place ensuite dans le dossier d'installation CodeBlocks.

Afin de configurer la SDL, il faut suivre, CodeBlocks -> 'Settings' -> 'Compiler'-> 'Linker Settings' et on y ajoute les chemins de la librairies :

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.

Ensuite, dans l'onglet 'Search directories'-> 'Compiler' on entre les chemins des dossier include et lib de la SDL2 afin de configurer le compilateur.

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 :

La SDL 2, contrairement à la SDL 1, utilise les ressources de la carte graphique. Elle permet également de gérer plusieurs fenêtres, leur taille, des boites de dialogues par exemple.

Question 6 :

```
#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 :

```
window = SDL_CreateWindow("SDL2", SDL_WINDOWPOS_CENTERED,
SDL_WINDOWPOS_CENTERED, 640, 480, SDL_WINDOW_SHOWN);
//création d'une fenêtre par SDL2, les deux première valeur (WINDOWPOS) permet
d'indiquer la position de la fenêtre sur l'écran, 640 et 480 indique la taille de la fenêtre crée,
et la dernière commande demande au programme d'afficher la fenêtre.
if(NULL == window)
{
        fprintf(stderr, "Erreur SDL_CreateWindow : %s", SDL_GetError());
        return EXIT_FAILURE;
}
SDL_DestroyWindow(window);          //Destruction de la fenêtre
```

Question 8 :.

SDL_CreateWindow() : Crée une fenêtre
SDL_CreateWindowFrom() : Crée une fenêtre SDL à partir d'une fenêtre déjà existante
SDL_DestroyWindow() : Détruit une fenêtre
SDL_GetWindowData() : Récupère les données utilisateur associées à la fenêtre
SDL_GetWindowFlags() : Récupère les options actuelles de la fenêtre
SDL_GetWindowGrab() : Détermine si la fenêtre a le focus clavier
SDL_GetWindowPosition() : Récupère la position actuelle de la fenêtre S
DL_GetWindowSize() : Récupère la taille actuelle de la fenêtre
SDL_GetWindowTitle() : Récupère le titre actuel de la fenêtre
SDL_HideWindow() : Cache la fenêtre
SDL_MaximizeWindow() : Agrandit la fenêtre
SDL_MinimizeWindow() : Réduit la fenêtre dans la barre des tâches
SDL_RaiseWindow() : Place la fenêtre devant les autres
SDL_RestoreWindow() : Restaure la taille et la position d'une fenêtre minimisée ou maximisée
SDL_SetWindowData() : Définit les données utilisateur de la fenêtre
SDL_SetWindowFullscreen() : Passe la fenêtre en plein écran
SDL_SetWindowGrab() : Donne le focus clavier à cette fenêtre
SDL_SetWindowIcon() : Définit l'icône de la fenêtre
SDL_SetWindowPosition() : Définit la position de la fenêtre
SDL_SetWindowSize() : Définit la taille de la fenêtre
SDL_SetWindowBordered() : Définit l'affichage des bordures de la fenêtre
SDL_SetWindowTitle() : Définit le titre de la fenêtre
SDL_ShowWindow() : Affiche la fenêtre

Question 9 :

Le Renderer est le « rendu » de la fenêtre, concrètement ce qui va apparaitre dans la fenêtre précédemment créer, la fenêtre et son contenu pouvant être considérer comme entités différentes.

```
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))
                        //Les 3 premières valeurs gèrent les valeurs RGB et la dernière gère
                        l'opacité.
{
        fprintf(stderr, "Erreur SDL_SetRenderDrawColor : %s", SDL_GetError());
        return EXIT_FAILURE;
}
if(0 != SDL_RenderClear(renderer))      //Fonction permettant de « vider » le 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);
        //Dessine un point aux coordonnées indiquées
SDL_RenderPresent(renderer);
        //Affichage du Renderer
SDL_Rect rect = {100, 100, 100, 100};
        //Les deux premières valeurs sont les positions x et y du carré, les deux suivantes
        indiquent la hauteur et la largeur
SDL_RenderFillRect(renderer, &rect);
        //Remplissage du rectangle
SDL_RenderPresent(renderer);
```

Question 11 :

On gère la couleur grâce à la structure SDL_Color qui prends en paramètre la valeur 'rgb' d'une couleur, allant de 0 à 255. La dernière valeur gère l'opacité.

```
SDL_Color orange = {255, 127, 40, 255};
```

Question 12 :

```
SDL_SetRenderDrawColor(pRenderer, 255, 0, 0, SDL_ALPHA_OPAQUE);
SDL_RenderClear(pRenderer);
SDL_RenderPresent(pRenderer);
```

Question 13 :

Cf. code programme

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 réinitialiser l'écran.

Question 16 :

SDL_Delay permet d'établir un temps d'attente en 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(surface, &source_rect, temp_surface, null) ;

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)

Question 22 :

Cf. code programme

Question 23 :

SDL_RenderCopy permet de copier une partie désirer d'une texture à la cible de rendue.

Cf. code programme

Question 24 :

Requête permettant d'accéder aux valeurs d'une texture.

Question 25 :

Question 26/27 :

```
//Extrait de code du programme rendus
SDL_SetRenderTarget(pRenderer, NULL); //redéfinitions de la zone de travail
SDL_Surface* pBmp = SDL_LoadBMP("staline.bmp");
SDL_Texture* Image = SDL_CreateTextureFromSurface(pRenderer,pBmp);

SDL_FreeSurface(pBmp);

if (Image == NULL)
 printf("erreur");

SDL_Rect myRect;
myRect.x = 0;
myRect.y = 0;
SDL_QueryTexture(Image, NULL, NULL, &myRect.w, &myRect.h);//Récupère le format de
l'image

SDL_RenderCopy(pRenderer, Image, NULL, &myRect);
SDL_RenderPresent(pRenderer);
```

Question 28 :

Il faut :

SDL_DestroyWindow

SDL_DestroyRenderer

SDL_DestroyTexture

SDL_FreeSurface