**1) Qu’est-ce que la SDL 2 ? (10 points)**

=> La SDL2 est une librairie de développement multi-plateformes écrite en C.

**2) Que peut faire la SDL 2 ? (10 points)**

=> La bibliothèque met à disposition un ensemble de fonctions permettant à l'utilisateur de gérer de manière très basique l'audio, les input (clavier/souris) et afficher des images.

**3) Sur quelles plateformes fonctionne la SDL 2 ? (10 points)**

=> La SDL2 étant une librairie multi-plateformes, elle est supportée sur Windows, Mac OS X, Linux, iOS et Android.

**4) Comment obtenir et installer la SDL 2 sur un EDI ? Donnez les étapes de la configuration d’un projet SDL2 avec CodeBlocks (gcc). (10 points)**

=> Il suffit de télécharger la librairie sur le site et de lier la bibliothèque à son compilateur.   
Pour CodeBlocks, il faut aller dans "Project" => "Build options" => "Debug" => "Linker settings", cliquer sur "Add" et chercher le dossier i686-w64-mingw23/lib et sélectionner le fichier libSDL2.dll.a  
Il faut ajouter également le fichier libSDL2main.a.  
Il faut également rajouter « -lmingw32 » dans la case « Other linker options ».  
Enfin, il faut également aller dans « Search directories », et dans « Compiler » inclure le dossier i686-w64-mingw23/include, et dans l’onglet « Linker » il faut inclure le dossier i686-w64-mingw23/lib.  
Pour finir, il faut mettre le fichier SDL2.dll à la racine du dossier où est contenu le main.c

**5) Quelles sont les différences entre SDL 1 et SDL 2 ? (5 points)**

Une des plus grosses fonctionnalités de la SDL 2 est le support de la 3D, elle est plus performante que la SDL 1, la licence de distribution est zlib et non plus LGPL, permettant une utilisation gratuite et libre de la bibliothèque.

**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. (15 points)**

Pour initialiser la SDL 2, il faut utiliser la fonction SDL\_Init() qui peut prendre divers paramètres, et va initialiser le sous-système correspondant.

|  |  |
| --- | --- |
| **SDL\_INIT\_TIMER** | timer subsystem |
| **SDL\_INIT\_AUDIO** | audio subsystem |
| **SDL\_INIT\_VIDEO** | video subsystem; automatically initializes the events subsystem |
| **SDL\_INIT\_JOYSTICK** | joystick subsystem; automatically initializes the events subsystem |
| **SDL\_INIT\_HAPTIC** | haptic (force feedback) subsystem |
| **SDL\_INIT\_GAMECONTROLLER** | controller subsystem; automatically initializes the joystick subsystem |
| **SDL\_INIT\_EVENTS** | events subsystem |
| **SDL\_INIT\_EVERYTHING** | all of the above subsystems |
| **SDL\_INIT\_NOPARACHUTE** | compatibility; this flag is ignored |

Ci-dessus les flags disponibles selon le wiki officiel de SDL.   
  
  
Dans notre cas on utilise SDL\_INIT\_EVERYTHING, qui va initialiser tous les sous-systèmes directement.  
La fonction retourne 0 en cas de succès, ou bien une valeur négative en cas d’erreur.  
Du coup pour vérifier que l’initialisation s’est bien passée, on va donc faire :  


**7) Comment créer une fenêtre avec SDL 2, donnez et expliquez le code correspondant à la création d’une fenêtre avec SDL 2. (10 points)**

Pour créer une fenêtre en SDL 2, il faut d’abord commencer par créer un pointeur de type SDL\_Window possédant la référence d’une window(Win32) ainsi :



Ensuite, il faut utiliser la fonction SDL\_CreateWindow() qui prend ces paramètres suivants :

const char\* title,

int x,

int y,

int w,

int h,

Uint32 flags)

Title = le titre de la fenêtre qui va être créer  
x = position x de la fenêtre  
y = position y de la fenêtre  
w = largeur de la fenêtre  
h = hauteur de la fenêtre  
  
Les flags peuvent prendre la valeur suivante :

|  |  |
| --- | --- |
| SDL\_WINDOW\_FULLSCREEN | fullscreen window |
| SDL\_WINDOW\_FULLSCREEN\_DESKTOP | fullscreen window at the current desktop resolution |
| SDL\_WINDOW\_OPENGL | window usable with OpenGL context |
| SDL\_WINDOW\_VULKAN | window usable with a Vulkan instance |
| SDL\_WINDOW\_HIDDEN | window is not visible |
| SDL\_WINDOW\_BORDERLESS | no window decoration |
| SDL\_WINDOW\_RESIZABLE | window can be resized |
| SDL\_WINDOW\_MINIMIZED | window is minimized |
| SDL\_WINDOW\_MAXIMIZED | window is maximized |
| SDL\_WINDOW\_INPUT\_GRABBED | window has grabbed input focus |
| SDL\_WINDOW\_ALLOW\_HIGHDPI | window should be created in high-DPI mode if supported (>= SDL 2.0.1) |

**8) Gestion d’une fenêtre SDL 2, donnez les primitives associées à la fenêtre SDL 2. (10 points)**

**Créer une fenêtre (déjà vu)** :

SDL\_CreateWindow(**const** char\* title, int x, int y, int w, int h, Uint32 flags);

**Détruire une fenêtre :**

void SDL\_DestroyWindow(SDL\_Window\* window);

On passe en argument la fenêtre à détruire.



**Donner un titre à la fenêtre :**

void SDL\_SetWindowTitle(SDL\_Window\* window , **const** char\* title);

****

**Récupérer le titre de la fenêtre :**

**const** char\* SDL\_GetWindowTitle(SDL\_Window\* window);

****

**Changer la position de la fenêtre :**

void SDL\_SetWindowPosition(SDL\_Window\* window , int x, int y);

****

**Changer la taille de la fenêtre :**

|  |
| --- |
| void SDL\_SetWindowSize(SDL\_Window\* window , int w , int h); |
|  |

****

Il existe pour ces deux fonctions l’équivalent en Get, on peut également agrandir, réduire, restaurer, passer en mode plein écran ou en mode normal, définir la luminosité de la fenêtre ( ou la récupérer ), ou pareil avec l’ID de la fenêtre.