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

La SDL 2 est une bibliothèque bas niveau, gratuite et open source, cross platefrom, développée en C. Elle permet un accès aux entrées du clavier, de la souris, des contrôleurs (manettes, joysticks), et aussi aux sorties audios et graphiques en utilisant OpenGL et Direct3D.  
  
Cette bibliothèque est principalement utilisée pour le développement de jeux vidéo, en C, C++, C# et Python. (Mais aussi d’autres langages)

1. Que peut faire la SDL 2 ? (10 points)

La SDL2 permet principalement de créer des jeux 2D.  
Elle contient toutes les fonctions bas niveau nécessaires à l’affichage 2D, a l’utilisation de bus audios, et a l’accès aux inputs de l’utilisateur, que ce soit via le clavier, la souris ou un autre contrôleur.  
Elle gère aussi les évènements, et peut être utilisé pour faire du multi-thread.

1. Sur quelles plateformes fonctionne la SDL2 ? (10 points)

La SDL2 fonctionne sur Windows, Linux, Mac OS, iOS et Android.

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

* Téléchargement des « Developpements Librairies »
* Décompresser l’archive « .tar.gz » et extraire vers une destination quelconque sur le disque dur.
* Créer un projet de type « SDL project » dans CodeBlocks en précisant l’emplacement ou la bibliothèque a été enregistrée sur le disque dur.
* Copier le « .ddl » de la SDL dans le dossier du projet CodeBlocks

1. Quelles sont les différences entre SDL1 et SDL2 ? (5 points)

La SDL 1.2 disposait d’un support pour C, C++, Java, Lua et Python principalement.  
La SDL 2 a apportée le support pour Ada, C#, D, GoLang, OCaml, Pascal, Rust

La SDL 2 a aussi apporté un gros gain de performance notamment pour la 3D, L’utilisation de OpenGL 3, l’utilisation de multiples bus audios, une API plus accessible et ajoute un support pour les écrans tactiles … etc.

1. Pour utiliser la SDL 2 dans votre projet, vous devez initialiser l’instance SDL2, quelles sont les étapes d’initialisation à réaliser pour utiliser la lib SDL2, donnez le code correspondant et expliquez précisément chaque instruction. (15 points)

(Voir le main.c, au commit « question 6 »)

1. Comment créer une fenêtre avec SDL2, donnez et expliquez le code correspondant à la création d’une fenêtre avec SDL2. (10 points)

(Voir le main.c, au commit « question 7 » )

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

On peut utiliser les fonctions pour obtenir et modifier le titre de la fenêtre :

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

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

Pour obtenir et modifier la position de la fenêtre :

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

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

Pour obtenir et modifier la taille de la fenêtre :

void SDL\_GetWindowSize(SDL\_Window\* window, int\* w, int\* h)

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

Pour obtenir les flags de la fenêtre :

Uint32 SDL\_GetWindowFlags(SDL\_Window\* window)

On peut utiliser les fonctions suivantes pour réduire, agrandir et restaurer la fenêtre :

void SDL\_MaximizeWindow(SDL\_Window\* window)

void SDL\_MinimizeWindow(SDL\_Window\* window)

void SDL\_RestoreWindow(SDL\_Window\* window)

On peut aussi utiliser les fonctions suivantes pour cacher/rendre visible la fenêtre :

void SDL\_HideWindow(SDL\_Window\* window)

void SDL\_ShowWindow(SDL\_Window\* window)

Pour mettre la fenêtre en plein écran :

int SDL\_SetWindowFullscreen(SDL\_Window\* window, Uint32 flags)

Enfin, on peut mettre en avant plan la fenêtre (devant toutes les autres), avec la fonction :

void SDL\_RaiseWindow(SDL\_Window\* window)

1. Gérer un rendu : qu’est-ce qu’un renderer, comment est-il associé à la fenêtre SDL2 ? Donnez et expliquez le code permettant de gérer un rendu, création, affichage, effacement... (15 points)

Le renderer est le moteur de rendu de la fenêtre, qui crée le contexte d’affichage de notre fenêtre.  
Il est associé a la fenêtre lors de sa création avec la fonction suivante a laquelle on passe un pointeur vers notre fenêtre en argument.

SDL\_Renderer\* SDL\_CreateRenderer(SDL\_Window\* window,

int index,

Uint32 flags)

On peut aussi utiliser :

int SDL\_CreateWindowAndRenderer(int width,

int height,

Uint32 window\_flags,

SDL\_Window\*\* window,

SDL\_Renderer\*\* renderer)

Et ainsi générer la fenêtre et le renderer d’un même temps.

(Voir le main.c au commit « Question 9 » pour l’exemple de code)

1. SDL\_Point et SDL\_Rect: Donnez et expliquez le code de la déclaration d’un SDL\_Point et SDL\_Rect (5 points)

(Voir le main.c au commit « Question 10 »)

1. Les couleurs avec la SDL2 : donnez les instructions permettant de gérer la couleur avec la SDL2

Avec la SDL2 on utilise le type structuré SDL\_Color pour déclarer une couleur.

On utilise ensuite ses propriété r, g, b et a pour déterminer la couleur à définir.

R, G et B correspondent aux trois couleurs Rouge, Vert et Bleu et le A correspond à Alpha, c’est-à-dire le niveau d’opacité de la couleur.

(Voir le main.c au commit « Question 11 » pour un exemple)

1. Donnez le code permettant d’afficher un fond rouge dans le rendu.

(Voir le main.c au commit « Question 12 »)

1. Dessiner dans le rendu : donnez le code des fonctions permettant de dessiner dans le rendu les formes suivantes :

-Carré vide

-Carré plein

-Cercle vide

-Cercle plein

(Voir le main.c au commit « Question 13 »)

1. Quelles sont les fonctions permettant de dessiner des points et des lignes ? (5 points)

Pour dessiner un point :

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

On a simplement besoin de lui passer le renderer et un point en argument

Pour dessiner une ligne :

int SDL\_RenderDrawLine(SDL\_Renderer\* renderer, int x1, int y1,

int x2, int y2)

On lui passe le renderer, un point d’origine et un point de destination.

1. A quoi servent les fonctions SDL\_RenderClear et SDL\_RenderPresent (5 points)

La fonction SDL\_RenderClear permet de vider la fenêtre, afin d’y afficher ce qui est près dans le renderer par la suite, a l’aide de la fonction SDL\_RenderPresent

1. Expliquez SDL\_Delay (5 points)

La fonction SDL\_Delay, permet d’attendre le nombre de millisecondes passées en arguments avant de retourner de la fonction et ainsi passer a l’instruction suivante.

1. Qu’est-ce qu’une surface en SDL2 (5 points)

Une surface est une zone de la fenêtre sur dans laquelle on peut dessiner des couleurs ou des textures. Elle n’utilise pas l’accélération matérielle, a contrario du type SDL\_Texture, mais permet d’accéder facilement aux pixels qui la compose.

1. Donnez le code permettant de créer une surface (10 points)

(voir le main.c au commit « Question 17 et 18 »)

1. Dessiner dans une surface : Donnez le code de la fonction SDL\_FillRect (10 points)

(voir le main.c au commit « Question 19 »)

1. SDL\_BlitSurface(): Donnez le code pour tester cette méthode.(5 points)

(voir le main.c au commit « Question 20 »)

1. Les textures : Qu’est-ce qu’une texture ? Donnez le code permettant de créer une texture. (10 points)

(voir le main.c aux commits « Question 19 » et « Question 20 »)

1. Dessinez dans une texture : SDL\_SetRenderTarget, donnez le code pour tester cette méthode. (10 points)

(voir le main.c au commit « Question 22»)