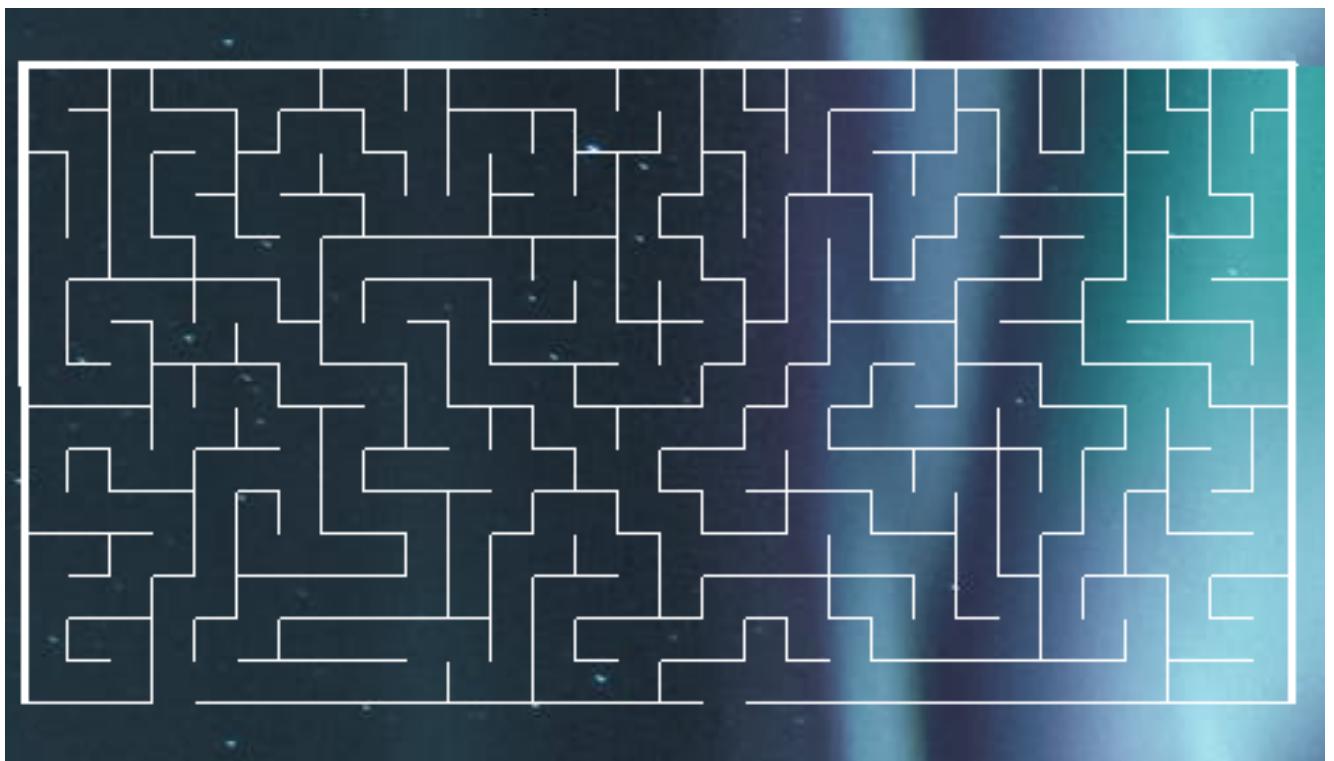


ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES SYSTÈMES - RABAT

Rapport de Projet de programmation : Le labyrinthe



Réalisé par :

Alaoui Ismaili MOHAMMED AMINE
Bourhim IKRAM

Année académique 2020/2021

Encadré par :

Pr. Mohamed RADOUANE



Remerciements :

C'est parce que nous avons beaucoup estimé tous ceux qui nous ont écoutés, conseillés, critiqués et encadrés que nous tenons à leur faire part de toutes nos gratitude, et plus particulièrement, nous tenons à remercier à travers ces courtes lignes : Monsieur Mohamed Radouane professeur à l'ENSIAS, qui a accepté d'examiner et d'encadrer ce travail pour le valoriser . Nos remerciements vont également à nos professeurs des modules de « Structures de données » : Monsieur Abdellatif EL FAKER pour son entière disponibilité, « Algorithmique » : Monsieur Ahmed ETTALBI pour le savoir qu'il nous a transmis et de « Techniques de programmation » : Monsieur Hatim Guermah. Enfin, nos remerciements s'adressent aux professeurs et au corps administratif de l'Ecole Nationale Supérieure de l'Informatique et de l'Analyse des Systèmes.



Table des matières

1	Introduction	1
1.1	Présentation du projet	3
1.1.1	Sujet	3
1.1.2	Étymologie de labyrinthe	4
1.1.3	Principe du jeu :	4
2	Analyse Théorique et conception	5
2.1	Cahier des charges	5
2.2	Le principe :	6
2.3	Génération aléatoire d'un labyrinthe parfait	7
2.3.1	Fusion aléatoire de chemins	7
2.3.2	Recursive Backtracking	7
2.4	Aspects et fonctionnement du jeu	9
3	La réalisation	10
3.1	Introduction	10
3.2	Difficultés rencontrées	10
3.2.1	La bibliothèque SDL	10
3.2.2	Programmation des fonctions	10
3.2.3	Organisation du code	10
3.3	Les outils et techniques de développement	-
3.4	La bibliothèque SDL	-
3.5	Organisation	-
3.5.1	Les directives des préprocesseurs	11
3.5.2	Des fichiers Headers (.h)	-
4	Interface graphique sur SDL	14
4.1	Premier Affichage	14
4.2	Les options du menu :	15
4.2.1	play	15
4.2.2	Ranking	15
4.2.3	Instruction	16
4.2.4	About :	16
4.2.5	Settings :	17
4.2.6	Quit :	17
4.2.7	Exemple de labyrinthe généré aléatoirement par l'algorithme et résolu en temps réel	18
5	Conclusion :	20
6	WEBOGRAPHIE	21

Chapitre 1

Introduction

L'étude de cheminement est importante pour différents domaines de travail, notamment : la logistique, la programmation de jeux, la gestion des opérations, l'analyse et la conception du système, la gestion de projet, le réseau et la ligne de production. L'étude du chemin le plus court a développé une capacité à penser de cause à effet, à apprendre et à penser comme l'humain dans l'IA. La recherche de chemins dans les jeux informatiques est devenue un domaine étudié depuis de nombreuses années. C'est à peu près le problème d'intelligence artificielle (IA) de jeu le plus populaire mais très difficile dans l'industrie du jeu. Le problème de la recherche de chemin dans les jeux informatiques commerciaux doit être résolu en temps réel, généralement sous des exigences de mémoire et de ressources CPU limitées. Le travail de calcul est censé trouver un chemin en utilisant un algorithme de recherche. La recherche de chemin peut être utilisée pour répondre à la question «Comment puis-je aller de la source à la destination ?».

C'est pour cela après avoir étudié plusieurs matières du langage C durant le premier semestre, et afin de nous permettre de mettre en pratique les différentes notions acquises durant les cours, les TDs et les TPs, nous étions invités à développer une application en C qui génère un labyrinthe et cherche un chemin de l'entrée vers la sortie du labyrinthe.

Le présent rapport résume les différentes étapes suivies pour atteindre cet objectif et afin de respecter les délais de rendu des différents livrables tout au long du projet.

PROJET C s'agit de produire un programme d'environ 500 lignes de code afin de valider les compétences descours : « Algorithmique », « Technique de programmation » et « Structures de données ».

Le programme correspond à 20 heures de travail effectives en langage C.

Les étudiants travaillent en binôme et bénéficient des conseils d'un professeur encadrant¹

1. Notice du projet de programmation 2020/2021

1.1 Présentation du projet

1.1.1 Sujet

Mathématiquement, un labyrinthe est représenté par un graphe connexe : simple, ou comportant des anneaux ou îlots. Le mathématicien Leonhard Euler a été l'un des premiers à analyser mathématiquement les labyrinthes plans et, ce faisant, a apporté les premières contributions significatives à la branche des mathématiques connue sous le nom de topologie .

Deux représentations sont possibles : labyrinthes dits « parfaits » où chaque cellule est reliée à toutes les autres et, ce, de manière unique, et labyrinthes dits « imparfaits » qui sont tous les labyrinthes qui ne sont pas parfaits.

Or nous, dans ce travail nous adopterons la représentation « parfaite ».

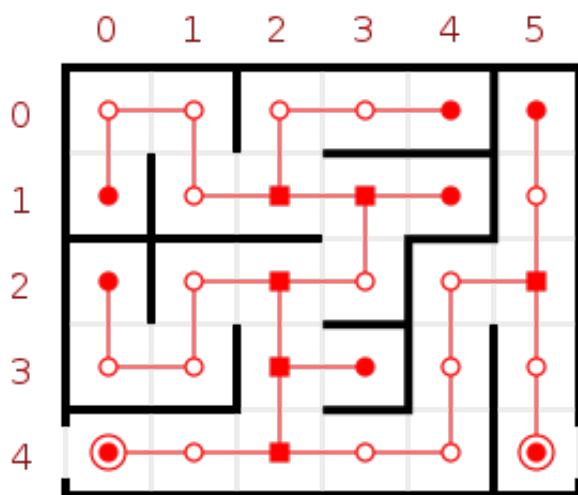


FIGURE 1.1 – Wikipédia-Modélisation mathématique de labyrinthe

Le jeu de résolution Labyrinthe consiste en la recherche d'un chemin vers la sortie.

Il existe de nombreuses approches différentes pour générer des labyrinthes, avec divers algorithmes de génération de labyrinthes pour les construire, soit à la main, soit automatiquement par ordinateur et suite aux exigences de ce projet nous utiliserons un algorithme de génération aléatoire ainsi que pour la résolution nous adopteront le même algorithme.

Pour parcourir le labyrinthe , on posera un objet de suivi qui suit le parcours de notre algorithme ,comme une boule .

1.1.2 Étymologie de labyrinthe

1- (Antiquité) Édifice composé d'un grand nombre de chambres et de galeries dont la disposition était telle, que ceux qui rentrent parvenaient difficilement à en trouver l'issue.

- Le labyrinthe de Crète fut construit par Dédale pour le roi Minos.
- Le plus célèbre labyrinthe était celui d'Égypte que nous a décrit Hérodote.

2- Un labyrinthe (labúrinthos en grec ancien), est un tracé sinueux, muni ou non d'embranchements, d'impasses et de fausses pistes, destiné à perdre ou à ralentir celui qui cherche à s'y déplacer. Ce motif, apparu dès la préhistoire, se retrouve dans de très nombreuses civilisations sous des formes diverses. Le mot désigne dans la mythologie grecque une série complexe de galeries construites par Dédale pour enfermer le Minotaure.

1.1.3 Principe du jeu :

C'est de parcourir les différents chemins possibles afin de tomber sur la porte de sortie mise sur le labyrinthe. En effet, un labyrinthe est un quadrillage composé de cases noires et traits blancs, les traits blancs correspondant aux murs et les cases noires aux chemins que peut emprunter le robot ou boule qui ne suit que les instructions que l'on lui donne, à travers un algorithme. De plus, ce robot a une vision limitée : il est seulement capable de voir ses cases environnantes.

Chapitre 2

Analyse Théorique et conception

Ce chapitre nous permet de faire une analyse théorique du projet, de présenter la problématique et identifier toutes les fonctionnalités de notre application.

2.1 Cahier des charges

La réalisation du projet 'Le labyrinthe' nécessite la mise en place d'un labyrinthe (utilisation de la bibliothèque SDL) avec une seule entrée d'un objet (à dessiner : cercle, boule, etc) et deux sorties. La première fois l'objet va sortie avec la sortie 1 et la deuxième fois par la sortie 2.

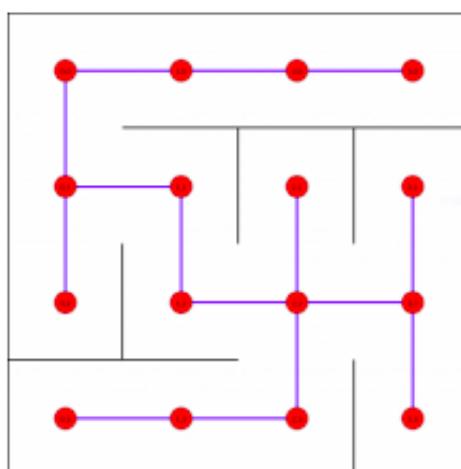


FIGURE 2.1 – Cahier de charge

2.2 Le principe :

Un labyrinthe représenté en deux dimensions sera construit aléatoirement à l'aide de l'un des algorithmes de génération. Les cellules seront mises en places sur une interface graphique qui rendra le jeu facile en plus des traits blancs qui représenteront les murs ainsi que l'entrée qui est le point de départ du jeu et deux sorties. Le caractère aléatoire provient du fait que la disposition des murs dans l'espace à deux dimensions est aléatoire et les entrées sorties seront aussi choisies au hasard parmi les pixels possibles . La génération de labyrinthes aléatoires est complexe or ce projet est majoritairement basé sur la logique et l'algorithme alors cela doit nécessiter des compétences mathématiques (théorie des graphes).

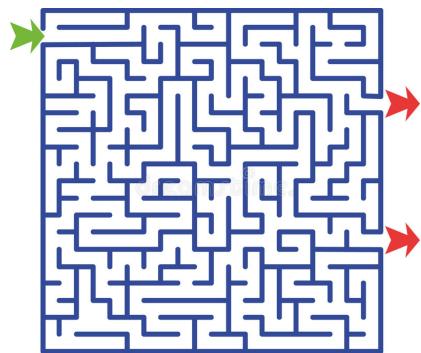


FIGURE 2.2 – Vector Illustration Maze Two Exits

2.3 Génération aléatoire d'un labyrinthe parfait

2.3.1 Fusion aléatoire de chemins

Nommé aussi en anglais "Randomized Kruskal's algorithm", fusionne progressivement des chemins depuis la simple cellule jusqu'à l'obtention d'un chemin unique. Il associe une valeur numérique unique à une cellule en partant d'une grille de cellules ,choisit de façon aléatoire les portes à ouvrir entre deux cellules adjacentes ayant un identifiant différent , et relie ces deux cellules créant ainsi un chemin ,ces deux cellules auront la même valeur par la suite , et continuer cette démarche jusqu'à obtenir un chemin unique si le nombre de portes ouvertes est $m \times n - 1$.

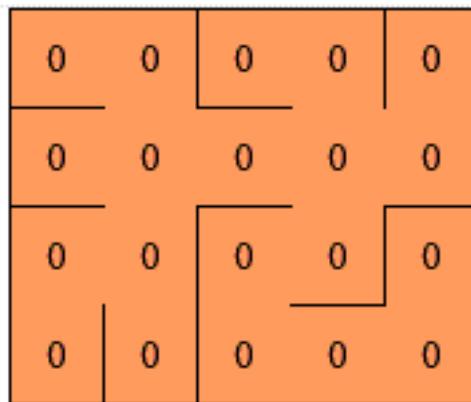


FIGURE 2.3 – Randomized Kruskal's algorithm

2.3.2 Recursive Backtracking

C'est l'algorithme que nous adopterons pour générer et résoudre notre labyrinthe de façon aléatoire, vu qu'il est facile à implémenter et qu'on peut le comprendre parcequ'il utilise des structures simples ; Cet algorithme consiste à revenir en arrière sur des décisions prises afin de sortir d'un blocage et ne donne pas beaucoup de "Dead Ends". On obtient la sortie d'un labyrinthe en testant toutes les combinaisons de sortie, si aucun blocage n'apparaît alors le programme continue sur les possibilités suivantes, si un blocage apparaît alors le principe du Backtracking apparaît et on revient au dernier cas examiné où il n'y avait pas de blocage et on continue le programme à partir de là (l'historique).

Enfin, parmi ces algorithmes ,on ne peut pas dire qu'un algorithme est meilleur qu'un autre.

Ils existent d'autres algorithmes de recherche du chemin et de génération dans le site que nous avons cité dans la Webographie mais aussi d'autres qui sont très rapides et cherchent le plus court chemin.

L'algorithme de recherche en profondeur de la génération de labyrinthe est fré-

quemment mis en œuvre à l'aide du retour en arrière. Cela peut être décrit avec l'algorithme récursif suivant :

Algorithm 1 Depth-first-Search

1. Étant donné une cellule courante comme paramètre ;
 2. Marquer la cellule actuelle comme visitée ;
 3. Alors que la cellule actuelle contient des cellules voisines non visitées ;
 - (a) Choisissez l'un des voisins non visités ;
 - (b) Supprimer le mur entre la cellule actuelle et la cellule choisie ;
 - (c) Invoquer la routine de manière récursive pour une cellule choisie qui est appelée une fois pour toute cellule initiale de la zone ;
-

Cela fonctionne sous une forme de stack où elle stock (empile) les branchements effectués et les met à jour (dépile) lors d'un blocage. Dans L'image ci-dessus de notre application , l'algorithme de backtracking lorsqu'il tombe sur le cas ou la cellule courante n'a pas de cellules adjacentes non visitées alors il est obligé de dépiler la pile en laissant des "Dead Ends" représentées en gris.

L'implémentation récursives se fait de telle façon : durant l'exploration, on marque les sommets afin d'éviter de re-parcourir des sommets parcourus.

Initialement, aucun sommet n'est marqué. De la même façon que les autres algorithmes de parcours de graphe, il trouve l'ensemble des sommets accessibles depuis un sommet donné s , donc ceux vers lesquels il existe un chemin partant de s .

Ces sommets sont marqués par l'algorithme comme étant visités.

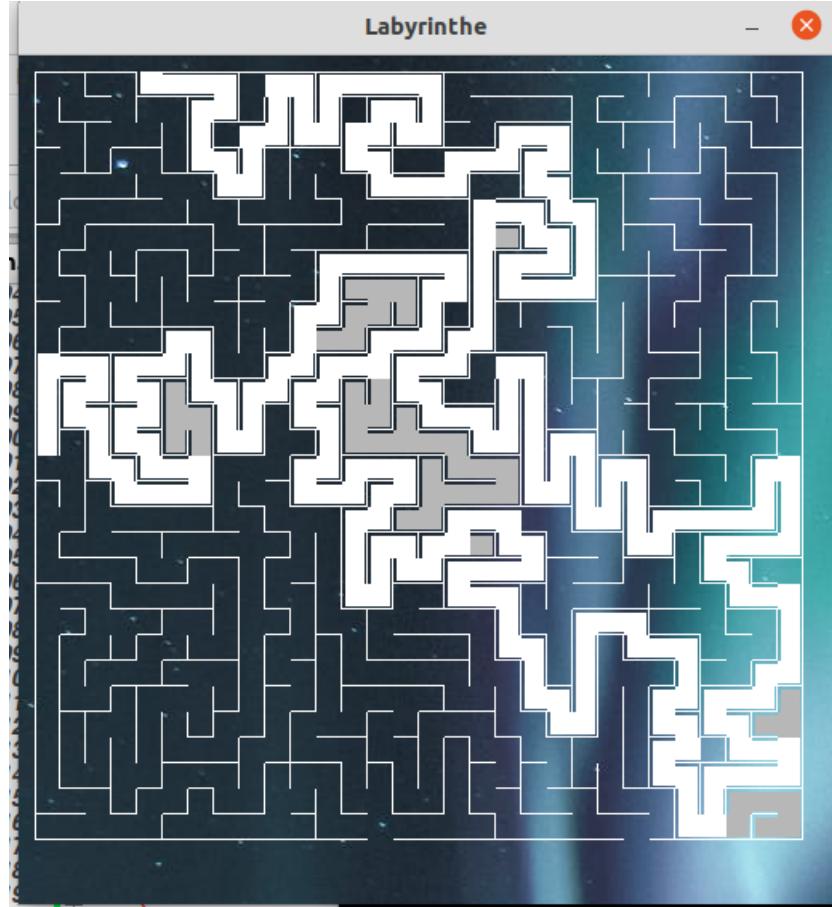


FIGURE 2.4 – Affichage du backtrack sur le labyrinthe

2.4 Aspects et fonctionnement du jeu

Dans le labyrinthe, l'objet qui fait le parcours ne peut se déplacer que de cases en cases dans notre monde dans 4 directions (Nord, Sud, Est, Ouest) et doit respecter l'existence ou pas des murs entre les cases. Nous constatons que même si l'objet en mouvement choisit aléatoirement chacun de ses mouvements il finira certainement par trouver la sortie vu le caractère déterministe du matériel ,l'hypothèse du labyrinthe "parfait" et la finitude du labyrinthe.

Chapitre 3

La réalisation

3.1 Introduction

Dans ce chapitre nous présenterons les environnements utilisés pour mettre en place notre projet ainsi que le langage exigé par le cahier de charge , les fonctions codés et les différents outils informatiques utilisés .

3.2 Difficultés rencontrées

3.2.1 La bibliothèque SDL

Certes SDL est un outil très bien adapté au sujet de programmation donnée et contient un nombre réduit de fonctions environ 600 , mais nous avons trouvé un difficulté dans l'organisastion la bibliothèque pour pouvoir l'utiliser de facon appropriée , et il y'avait une ambiguïté entre la version SDL1.2 et SDL2 à cause des chagements des fonctions dans version SDL2 donc nous avons opté pour SDL1.2.

3.2.2 Programmation des fonctions

C'est la difficulté principale trouvée ,lors de la compilation on trouvait des erreurs de compilations dans le code source ce qui nous a pris beaucoup de temps pour la révision de la logique du programme et comment les variables sont reliées entre elles. La programmation de la fonctions de génération de labyrinthe était en elle même une étape difficile surtout dans la partie de génération de deux sorties et leurs reconnaissance lors de résolution.Ainsi que la logique du menu principale et comment les differentes fonctionnalités sont reliées à travers le code.

3.2.3 Organisation du code

Nous avons pu divisé le code en deux fichiers " .c " et des fichiers " .h " après avoir régler les erreurs de reconnaissance des fonctions de la SDL.

3.3 Les outils et techniques de développement : Code : :Blocks :

Code : :Blocks est un environnement de développement intégré libre , gratuit et multiplate-forme qui est orienté C et C++ et qui est utilisé pour développer plusieurs jeux notamment en C et facile à manipuler avec les bibliothèques standards du langage C et de SDL. Nous avons choisi cet environnement puisqu'il est disponible pour tout système d'exploitation et vu les difficultés rencontrés lors de l'intégration de SDL dans les autres éditeurs.

3.4 La bibliothèque SDL :

Pour programmer notre jeu de plateformes, nous allons avoir besoin de bibliothèques tierces (libraries en anglais) nous permettant de gérer les entrées (clavier, joystick), la vidéo, le texte, le son, etc.



FIGURE 3.1 – Simple DirectMedia Layer

3.5 Organisation :Description des fonctions et structures

3.5.1 Les directives des préprocesseurs

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include "constantes.h"
5 #include <SDL/SDL_mixer.h>
6 #include <SDL/SDL.h>
7 #include "SDL_ttf.h"
8 #include <SDL_image.h>
9 #include "SDL_mixer.h"
```

Ces directives ont comme rôle l'importation les bibliothèques et les fichiers header contenant les fonctions à utiliser dans le main.

3.5.2 Des fichiers Headers (.h) :

Fichier Constantes.h

```

1 #ifndef DEF_CONSTANTES
2 #define DEF_CONSTANTES
3 #define MAKE_POS(x, y) ((y << 16) | (x))
4 #define BLOC_W 16
5 #define BLOC_H 16
6 /* definition des directions */
7 enum
8 {
9     LABF_NORD = 0x01,
10    LABF_EST = 0x02,
11    LABF_SUD = 0x04,
12    LABF_OUEST = 0x08,
13    LABF_NSOE = 0x0f,
14    LABF_VISIT = 0x10
15 };
16 static int opposite[] = {0, LABF_SUD, LABF_OUEST, 0, LABF_NORD, 0, 0, 0, LABF_EST};
17 #endif

```

Definition des constantes : Ce fichier definit les dimensions de la fentre contenant le labyrinthe ainsi que les directions des cellules et l'état d'une cellule.

Fichier fonctions.h Ce fichier contient les prototypes de toutes les fonctions et la déclaration des structures utilisées.

```
1 laby_t alloc_laby(int w, int h);
```

*Fonction qui creer un tableau ou grille à 1D qui représente le labyrinthe dans la mémoire

```
1 void SDL_DrawLine(SDL_Surface * ecran, uint32_t color, int x, int y, int x2, int y2);
```

*Fonction qui dessine les lignes des bords et des murs du labyrinthe

```
1 Uint32 SDL_Ticks(Uint32 interval, void * unused);
```

*Fonction de callback (sera appelée toutes les 100 ms)

```
1 void Music(char * music);
```

*Fonction permettant de jouer du son avec *SDLMixer* en spécifiant le nom de la musique

```
1 void jouer_auto();
```

*Fonction qui cree un labyrinthe aléatoirement et l'affiche sur la surface graphique puis donne la résolution

```
1
2 void play();
```

*Fonction d'appel pour commencer le jeu à travers la commande play du menu du jeu.

```
1 void menu();
```

*Fonction qui englobe l'ensemble des opérations exécutées dans l'application : Affiche le menu :

- Play
- Ranking
- About
- Help
- Quit

```
1 void ranking();
```

*Fonction qui donne le meilleur timing fait pour résoudre le labyrinthe à chaque lancement de l'application.

```
1 void help();
```

*Fonction qui affiche à chaque clique du boutton instruction ,les instructions de fonctionnement du jeu , c'est à dire le fonctionnement général.

```
1 void about();
```

*Fonction qui affiche à chaque clique du boutton About , les informations relatives au cadre de réalisation de l'application et des réalisateurs de celle-ci.

```
1 void settings();
```

*Fonction qui permet de régler le son ; le mettre en pause ou résumer la lecture de la musique, selon le choix de l'utilisateur.

```
1 void pause();
```

*Fonction qui met en pause l'écran sur lequel est dessiné le labyrinthe.

Chapitre 4

Interface graphique sur SDL

Dans ce chapitre, nous allons valider les attentes du projet à travers une interface graphique en utilisant la bibliothèque SDL.

4.1 Premier Affichage

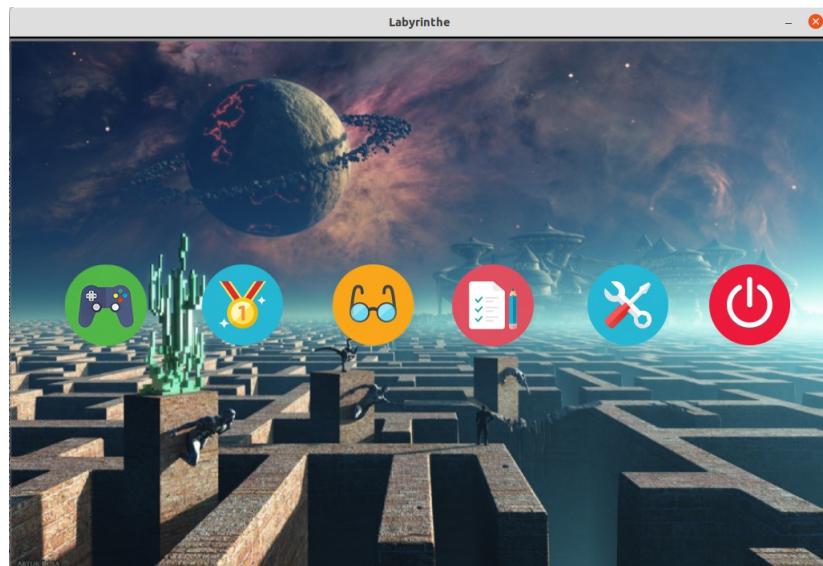


FIGURE 4.1 – Premier affichage lors de l'execution de l'interface graphique

Afin de générer aléatoirement nos labyrinthes, et aussi, de tester nos stratégies, nous avons créé un programme qui se présente sous cette forme :

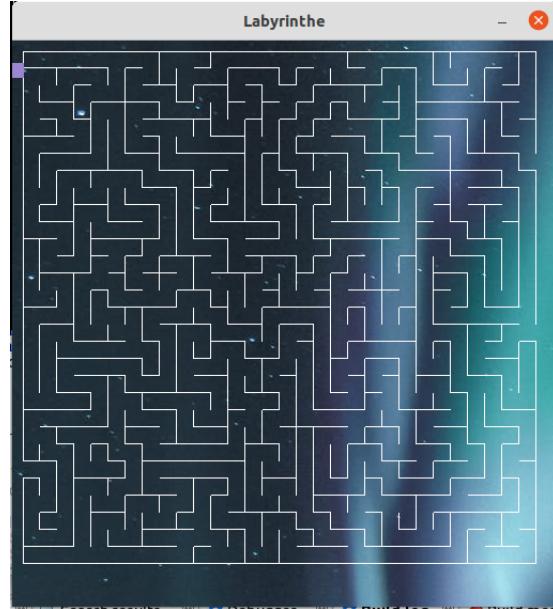


FIGURE 4.2 – Labyrinthe non encore parcouru

4.2 Les options du menu :

4.2.1 play

Permet de lancer le labyrinthe et la recherche de sorties.

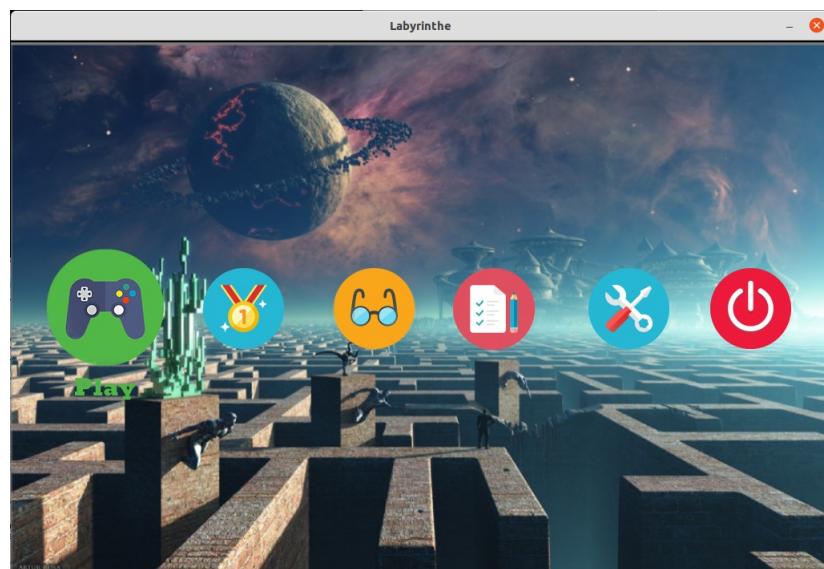


FIGURE 4.3 – Le menu du labyrinthe

4.2.2 Ranking

Représente le meilleur dernier timing de résolution fait.

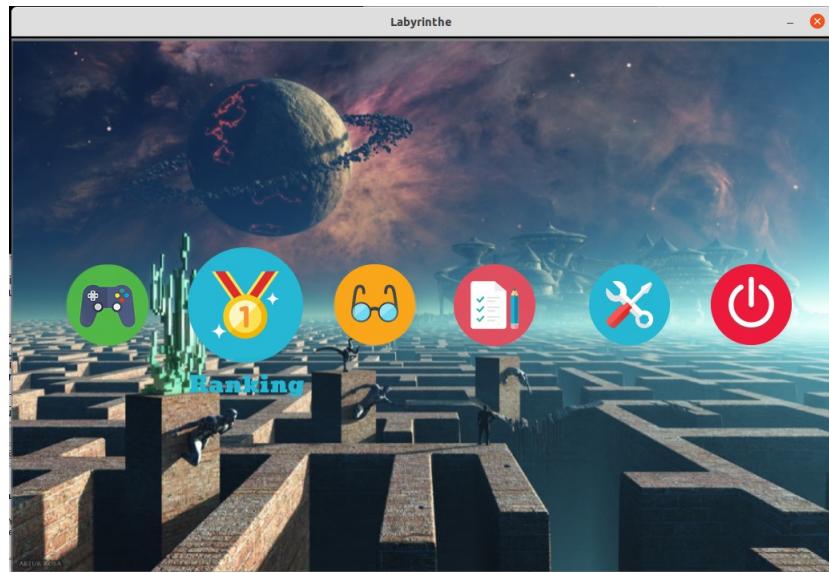


FIGURE 4.4 – Le boutton Ranking

4.2.3 Instruction

Les instructions de fonctionnement du jeu, contient des informations sur le cahier de charge.

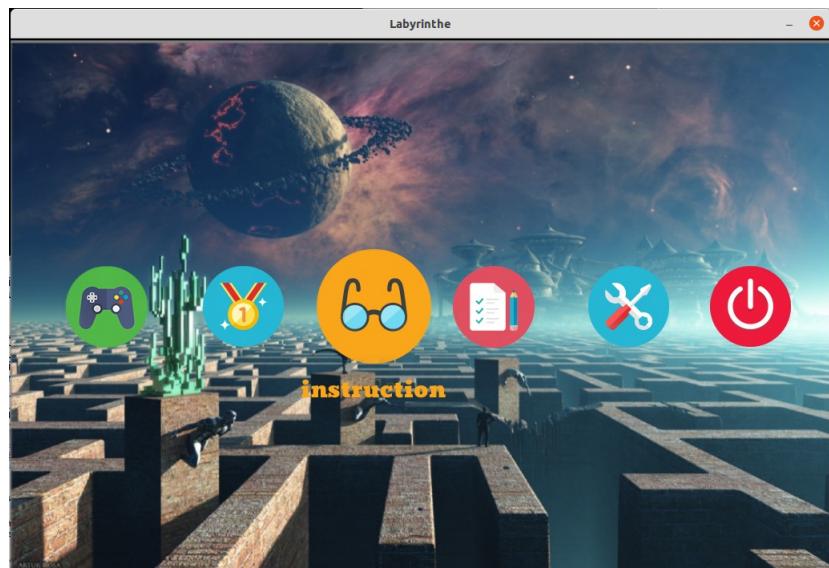


FIGURE 4.5 – Le boutton instructions

4.2.4 About :

Informations générales sur le projet

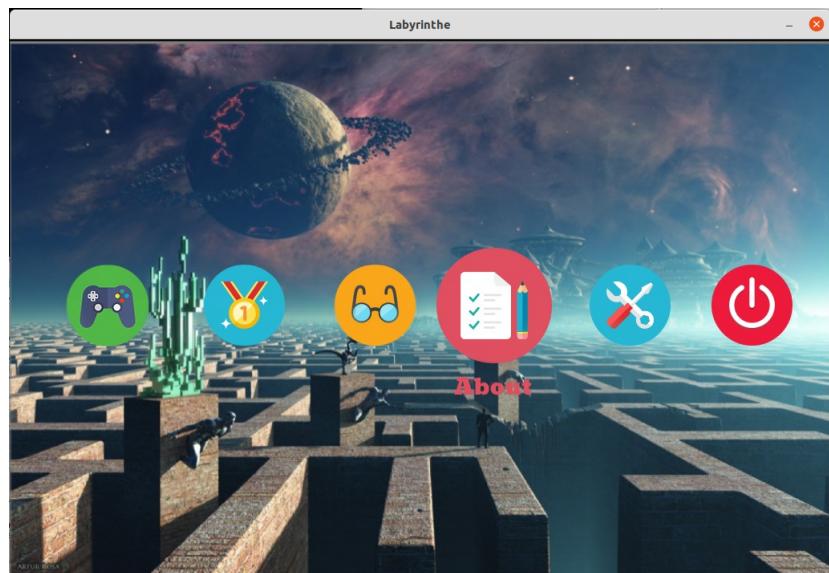


FIGURE 4.6 – Le boutton About

4.2.5 Settings :

pour activer ou désactiver le son du jeu

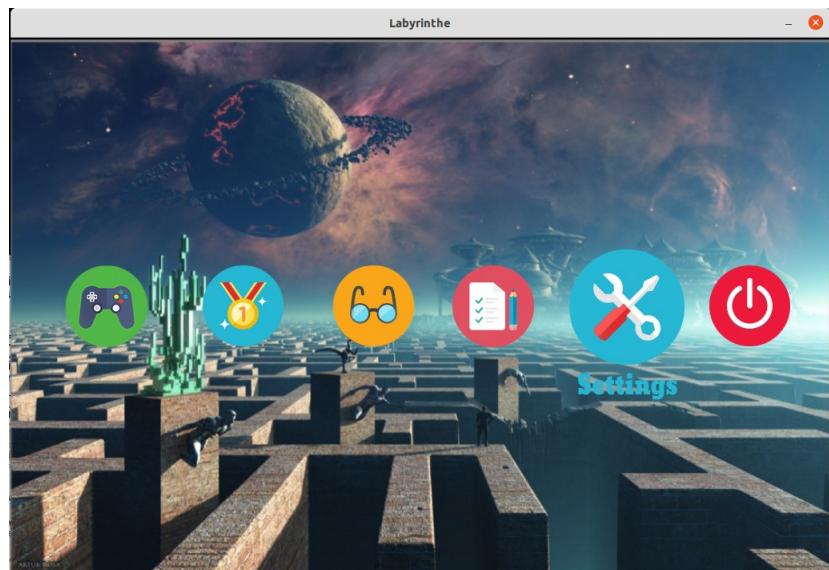


FIGURE 4.7 – Le boutton Settings

4.2.6 Quit :

Boutton pour quitter le jeu ,autre que le boutton de la fenêtre

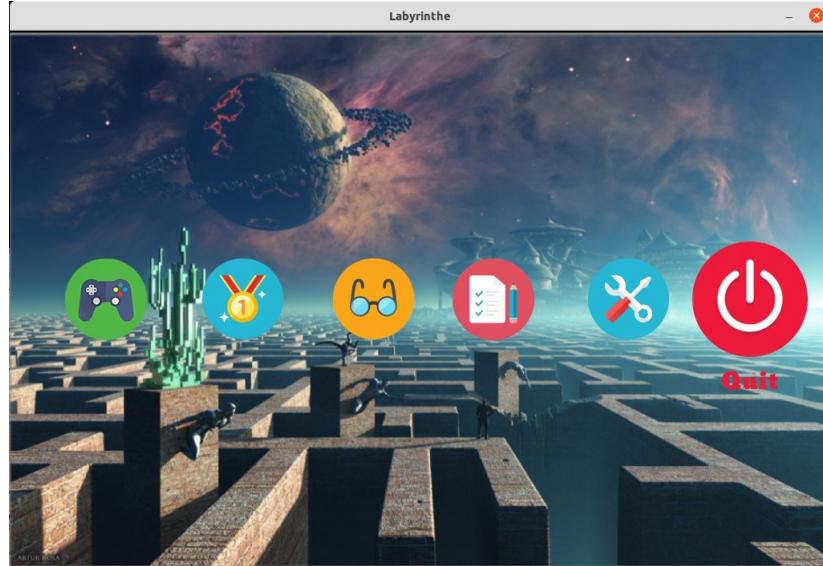


FIGURE 4.8 – Le boutton Quitter

You win : si on arrive à une certaine sortie ; cette fenêtre s'affiche pour féliciter l'utilisateur .

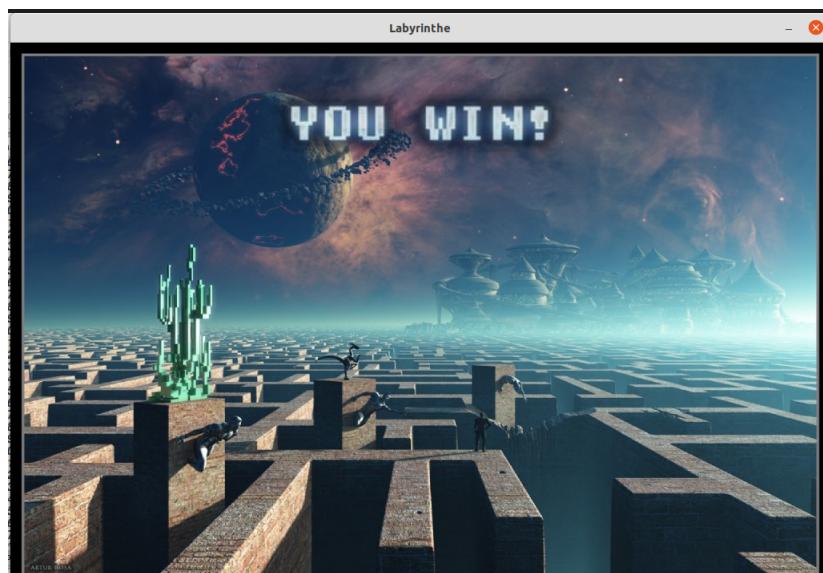
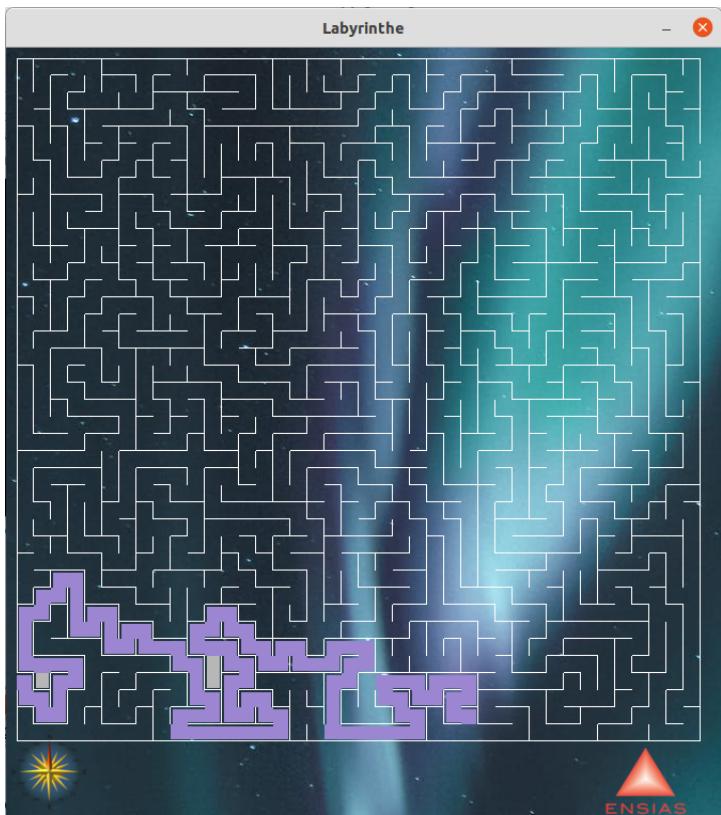
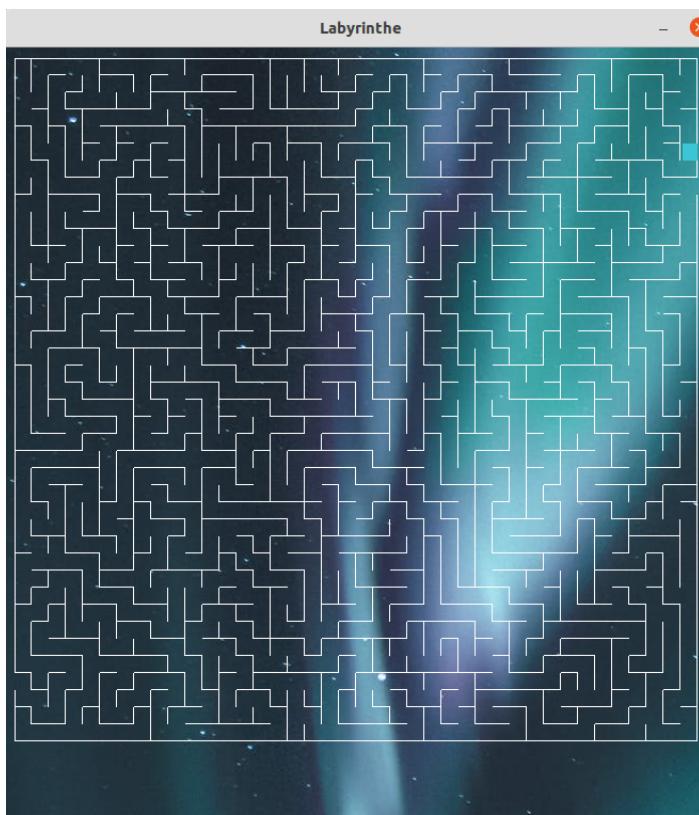


FIGURE 4.9 – L'affichage de "You win"

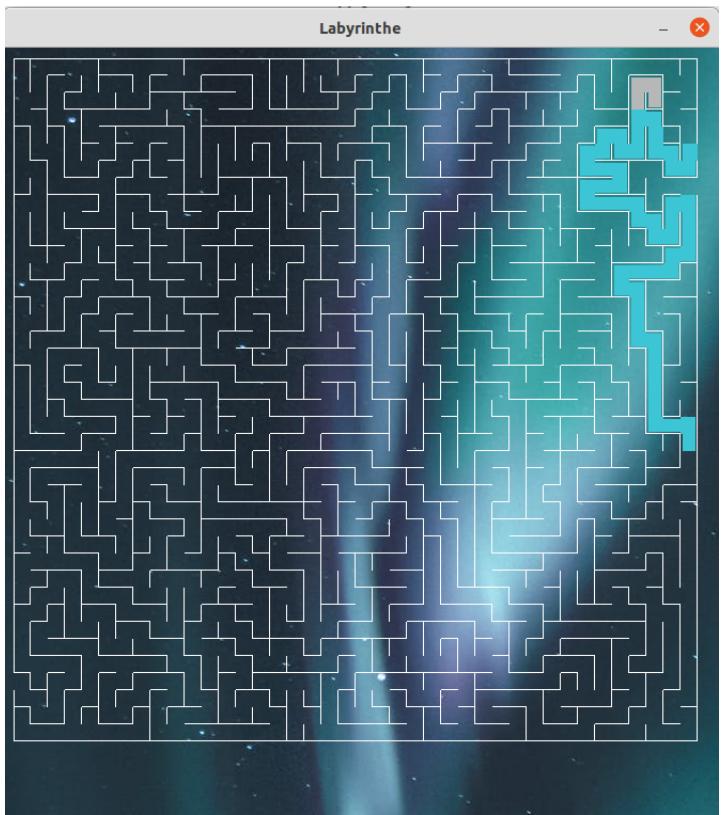
4.2.7 Exemple de labyrinthe généré aléatoirement par l'algorithme et résolu en temps réel



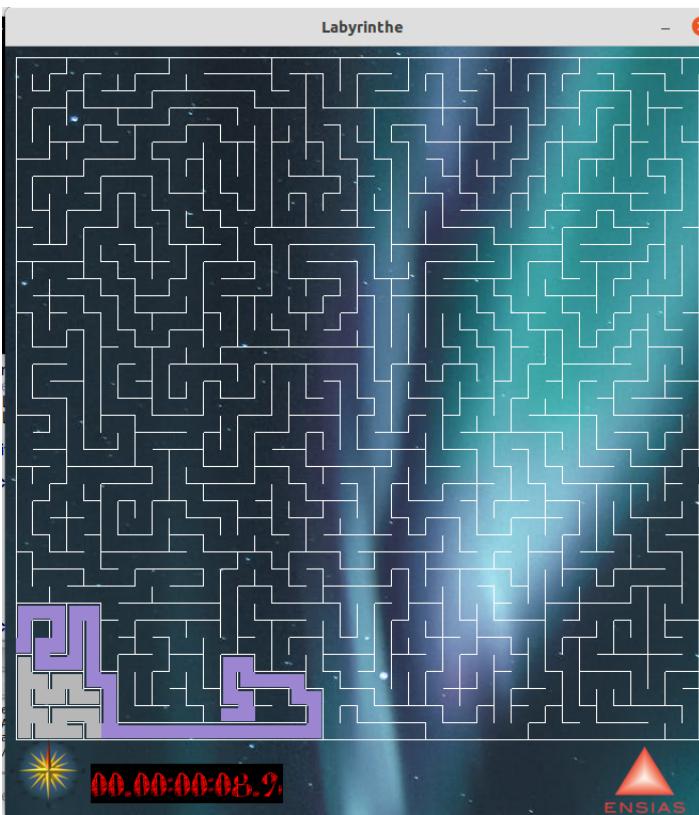
(a) Recherche de la première sortie



(b) On a retrouvé la première sortie ,on commence à chercher la 2-ème



(c) Recherche de la deuxième sortie



(d) Ajout du chrono.

FIGURE 4.10 – Exemple de résolution de labyrinthe.

Chapitre 5

Conclusion :

Notre projet a consisté à la réalisation d'un jeu qui sert à générer un labyrinthe avec une seule entrée choisi aléatoirement et plusieurs sorties et de le résoudre automatiquement. Ce projet nous a permis d'approfondir nos connaissances théoriques, acquises tous le long de notre formation, par la pratique des nouvelles technologies. Cette expérience nous a permis de mettre en pratique nos connaissances en langage C, de maîtriser la bibliothèque utilisé qu'est SDL et de se familiariser avec le travail en groupe sur des projets d'ingénierie.

Chapitre 6

WEBOGRAPHIE

[`https://www.meruvia.fr/index.php/programmation/24-big-tuto/280-big-tuto-Tutoriel SDL1`](https://www.meruvia.fr/index.php/programmation/24-big-tuto/280-big-tuto-Tutoriel(SDL1)) Dernière visite :le 30/01/2021

[`https://fr.wikipedia.org/wiki/Mod%C3%A9lisation_math%C3%A9matique_de_labyrinthe#R%C3%A9solution`](https://fr.wikipedia.org/wiki/Mod%C3%A9lisation_math%C3%A9matique_de_labyrinthe#R%C3%A9solution)
Dernière visite :le 7/02/2021

[`https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/16421-lire-et-ecrire-dans-des-fichiers`](https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/16421-lire-et-ecrire-dans-des-fichiers)
Dernière visite :le 11/02/2021

[`https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/17117-installation-de-la-sdl`](https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/17117-installation-de-la-sdl)
Dernière visite :le 7/02/2021

[`https://wiki.libsdl.org/`](https://wiki.libsdl.org/) Le site de la bibliothèque SDL qui contient des tutoriels et la documentation SDL1.2 [`https://en.wikipedia.org/wiki/Stack_\(abstract_data_type\)`](https://en.wikipedia.org/wiki/Stack_(abstract_data_type)) [`http://www.astrolog.org/labyrnth/algrithm.htm#perfect`](http://www.astrolog.org/labyrnth/algrithm.htm#perfect) : un site intéressant qui liste tous les Algorithmes de génération de labyrinthe et leurs propriétés et conditions.
Dernière visite :le 7/02/2021

[`https://www.youtube.com/watch?v=yFLa3ln16w0&ab_channel=CS50`](https://www.youtube.com/watch?v=yFLa3ln16w0&ab_channel=CS50) : video qui explique les fonctions de la bibliothèque SDL.

[`https://www.xm1math.net/doculatex/integrale.html`](https://www.xm1math.net/doculatex/integrale.html) :
Nous avons utilisé ce site qui est très riche pour écrire ce mémoire pour apprendre à manipuler les fonctions et compiler le texte Latex.
Dernière visite :le 7/02/2021

Bibliographie

- [1] Mr.El Faker. Cours de structures de données, 2020.
- [2] Hatim Guermah. Cours techniques de programmation, 2019.
- [3] Mathieu Nebra. Apprenez à programmer en c!
. <http://user.oc-static.com/pdf/14189-apprenez-a-programmer-en-c.pdf>, 2020.
Livre du tutoriel SDL1 par Openclassroom,TP Mario SOKOBAN.