

# Ethnirybal

## Jeu de Labyrinthe

### < Sommaire >

1. Introduction
2. Manuel Utilisateur
3. Documentation Technique
4. Conclusion

### < Introduction >

Dans le cadre du projet de fin du premier semestre de la Licence Mathématique-Informatique, nous avons réalisé un programme permettant de se déplacer dans les couloirs d'un labyrinthe en vue subjective. Le labyrinthe, généré aléatoirement est stocké dans un fichier texte permettant sa lecture par le programme. De plus, quelques améliorations ont été ajoutées au projet de base.

### < Manuel d'Utilisation >

#### Comment Jouer ?

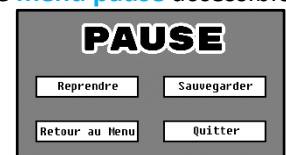
Pour démarrer le jeu, lancer le fichier « **Ethnirybal** » et attendez quelques instants (Vous pouvez passer l'affichage « **ProjectX** » au démarrage via un clic ou l'appuie sur n'importe quelle touche).



Une fois sur le **menu principal**, utilisez la souris pour sélectionner l'option désirée :

- **Continuer** : Reprendre votre partie sauvegardée (si une partie a été sauvegardée)
- **Nouvelle Partie** : Commencer une nouvelle partie depuis le début
- **Quitter** : Fermer le jeu

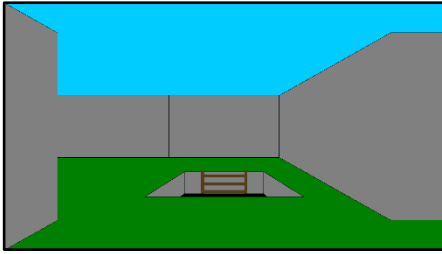
Vous pouvez utiliser le clic de votre souris pour interagir avec tous les menus du jeu comme le **menu pause** accessible en jeu depuis le bouton :



Lorsque vous êtes dans le labyrinthe, vous pouvez vous déplacer avec les boutons suivant :

- pour faire des **déplacements** dans les 4 directions suivantes :
- pour faire une **rotation** de 90° à gauche (**a**) ou à droite (**e**)

## Présentation du Jeu



Dans ce jeu, le joueur se retrouve dans un labyrinthe en vue subjective dans lequel il peut se déplacer afin d'atteindre la sortie afin de passer au labyrinthe suivant. Cependant, il y a un défi : le temps pour finir le labyrinthe est limité. En effet, quand le temps restant total arrive à 0, le joueur perd la partie. De plus, chaque labyrinthe est chronométré individuellement en fonction de la difficulté choisie au début (facile, normal, difficile, ...) et si ce chronomètre passe dans le négatif, le temps restant total diminuera plus vite.

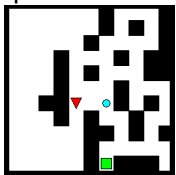
Mais pas de panique ! Lorsque le joueur arrive à la sortie représentée par une échelle, il augmente son score et son temps restant total est lui aussi augmenté afin de continuer à jouer.

Temps Restant Total
01:08

Temps Restant du Labyrinthe En Cours
00:35

Score
000171

Pour se repérer plus facilement, il dispose également d'une boussole indiquant la direction dans laquelle il regarde ainsi qu'une mini carte qui se dessine avec la progression du joueur dans le labyrinthe.



**Triangle Rouge** : Joueur  
**Rond Bleu** : Point de Départ  
**Carré Vert** : Sortie  
**Carré Noir** : Mur



Enfin, sur le menu pause, le joueur peut sauvegarder sa partie afin de la reprendre plus tard.

## < Documentation Technique >

Afin de réaliser ce projet, nous avons développé diverses fonctionnalités : les déplacements, l'affichage, les menus, la carte, la sauvegarde, etc... Nous allons présenter ici les fonctionnalités les plus techniques.

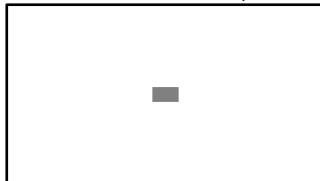
## Affichage du Labyrinthe

L'affichage d'un labyrinthe en vue subjective pose un principal problème : le grand nombre de possibilités de choses à afficher. Or nous ne pouvons pas dessiner chaque image pour chaque cas possible. Nous avons donc décidé de créer un système d'affichage avec des textures « modulables » permettant d'afficher indépendamment chaque partie du labyrinthe (un mur à droite, un sol loin devant, un mur au fond à gauche, etc...). Voici quelques exemples de ces textures :

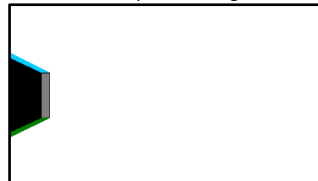
Le sol et ciel de la case du joueur



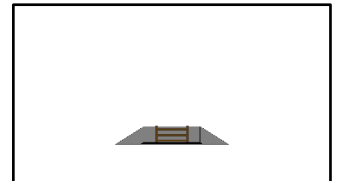
Le mur de face deux case plus loin



L'absence de mur à gauche une case en face puis une à gauche

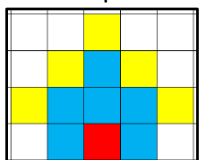


La sortie une case en face

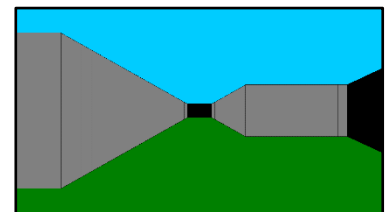


Pour choisir quelle image afficher, le programme va d'abord vérifier l'état de la case en face du joueur. Si c'est un mur, on affiche un mur, sinon le programme va vérifier les cases autour de celle-ci. Et le programme refait cela jusqu'à ce qu'il ait affiché tout le champs visuel du joueur. Cela permet de ne pas avoir à afficher les images de la case tout au fond si c'est en dehors du champs visuel à cause d'un mur se trouvant en face du joueur.

Le champs visuel du joueur est limité suivant ce schéma :



**■** Joueur regardant vers le haut  
**■** Case où l'affichage du sol et des mur peut être fait  
**■** Case où uniquement l'affichage du mur adjacent à **■** peut être fait



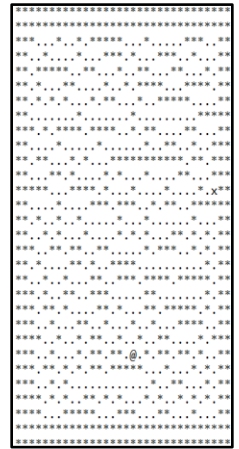
## Sauvegarde

Etant donné la potentielle longueur d'une partie, une option de sauvegarde a été implémentée afin d'accorder des pauses au joueur.

Lorsque celui-ci appuie sur le bouton « Sauvegarder », sa position dans le labyrinthe, sa direction, mais aussi son score et les temps sont sauvegardés dans un fichier « joueur.txt ». Le labyrinthe et la carte, quant à eux, sont sauvegardés chacun dans un fichier unique pour faciliter leur lecture.

```
17 23;1 0;
1
1
1;1;959;0;270;269
joueur.txt
```



Les données ne sont pas sauvegardées dans le fichier sous forme de listes ou de nombres mais elles sont converties en chaînes de caractères au moment de la sauvegarde puis reconverties dans leur type d'origine lors du chargement.



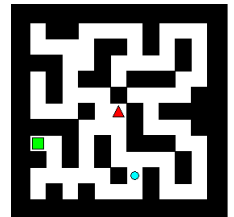
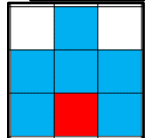
labyrinthe.txt

## Carte

Dans un labyrinthe en vue subjective, il est facile de se perdre. Nous avons donc ajouté une carte qui se dessine au fur et à mesure de la progression du joueur afin de l'aider sans lui dévoiler tout le labyrinthe. Voici les différentes étapes que nous avons réalisé pour créer une carte :

- **Création d'une carte vide** : On crée d'abord une liste de listes de la même taille que le labyrinthe contenant que des . représentant les chemins de notre labyrinthe puis on les entoure avec deux lignes de \* car tous les labyrinthes que l'on crée ont deux couches de murs afin de faciliter l'affichage. (module jeu fonction init\_carte)
- **Remplissage de la carte** : A chaque itération de la boucle principale du jeu, on actualise la carte en ajoutant dans la liste de listes les éléments du labyrinthe proches du joueur, c'est-à-dire ceux se trouvant aux emplacements des  du schéma (le joueur  regarde vers le haut). On permet de « voir à travers les murs » via la carte. (fonction actualise\_carte)
- **Affichage de la carte** : Pour afficher la carte, nous avons choisis de donner un champs de vision de rayon 6 autour du joueur. On redimensionne donc la carte à une taille de 13x13 avec le joueur au centre. On fait attention quand le joueur est à moins de 6 cases du bord du labyrinthe. En effet, pour éviter les erreurs, on décide que toutes les cases de la carte en dehors du labyrinthe deviennent des murs. (fonction redimension\_carte)  
Puis on affiche simplement la carte avec les fonctions de fltk.

```
*****
*****
** ..... **
** ..... **
** ..... **
** ..... **
*****
*****
```



## Score / Temps

Pour ajouter une difficulté au jeu nous avons implémenté un système de temps. Cela permet de créer un enjeu lors de la partie pour donner plus d'intérêt au jeu.

Puisque la taille des labyrinthes est choisie aléatoirement, cela fait un bon élément de référence pour le score.

Pour le temps on pourra se fier à la surface du labyrinthe : plus la surface d'un labyrinthe est grande, plus le temps nécessaire pour le parcourir est grand et inversement, plus la surface d'un labyrinthe est petite, plus le temps nécessaire pour le parcourir est petit. Ces valeurs sont déterminées par une fonction issue de divers tests. Par exemple, un labyrinthe de 25x25 aura un temps de parcours de 3 minutes dans la difficulté facile.

Cependant pour éviter les frustrations liées à des générations de labyrinthe peu avantageux nous avons décidé d'ajouter un deuxième compteur : le temps total autorisé. Celui-ci augmente (en même temps que le score) à chaque fois que le joueur réussit à finir un labyrinthe en dessous du temps de parcours et c'est ce compteur qui détermine la fin d'une partie.

Temps de Jeu	Temps Restant Total	Temps Restant Du Labyrinthe En Cours	Score
00:05	03:40	01:06	000000

## Génération du Labyrinthe

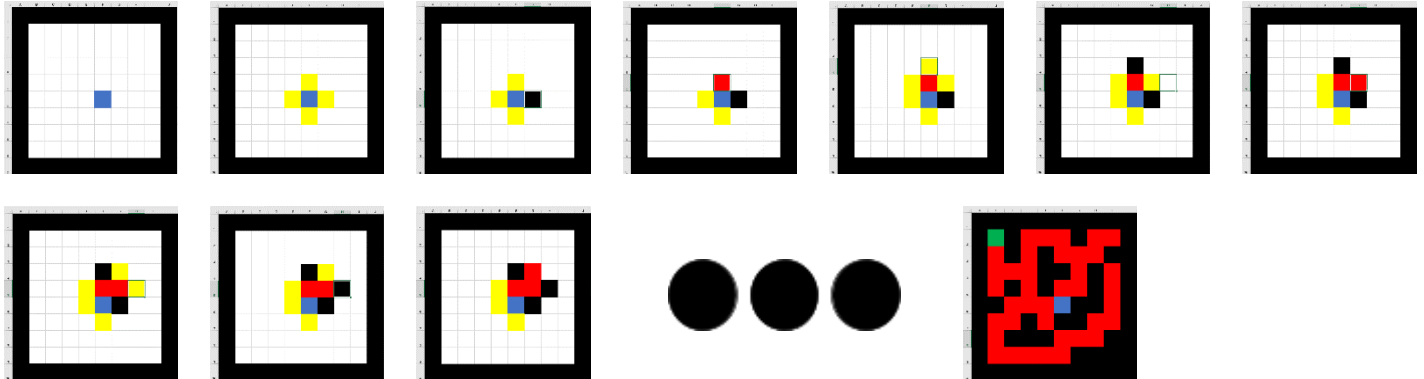
### Génération :

Pour pouvoir générer un labyrinthe, nous devons trouver un algorithme qui permettrait de produire des schémas variés de couloirs et de murs. Nous avons donc créé un algorithme simple basé sur l'aléatoire dont le fonctionnement est le suivant :

- Créer un labyrinthe vide d'une taille générée aléatoirement
- On ajoute les murs de 2 d'épaisseur autour
- On pose un curseur à une position aléatoire (Emplacement de départ du labyrinthe)
- Boucle While
  - Si au moins deux cases adjacentes au curseur sont vides : On met un mur sur l'une d'entre elles
  - On avance dans une direction aléatoire
  - Si il reste des cases vides autour de la position précédente du curseur, on les enregistre en tant qu'intersection
  - Si le curseur est bloqué (si il ne peut plus se déplacer) : Il se « téléporte » à une intersection aléatoire
  - Si il n'y a plus d'intersection : Fin de la génération (Le dernier emplacement du curseur représente la sortie du labyrinthe)

Cet algorithme nous permet de générer des labyrinthes variés avec des patrons plutôt intéressants.

Voici un exemple d'exécution :



### Légendes :



### Affichage :

Pour afficher l'écran de chargement du labyrinthe, on prend les éléments autour du curseur à chaque répétition de l'algorithme et on regarde leur état (chemin, mur, intersection, etc...). On affiche alors à l'endroit correspondant un carré de couleur correspondant à cet état.

Cependant, avec cette méthode, on remarque de temps en temps une superposition du bord des carrés. Pour corriger ces bavures, on a décidé de réinitialiser le rendu de temps en temps en effaçant l'affichage pour le redessiner depuis les éléments connus du labyrinthe et ainsi lisser le résultat.

Dès la fin de l'affichage (qui peut être passé avec un clic de souris ou l'appui sur une touche de clavier), le jeu reprend et le joueur peut se déplacer dans le labyrinthe qui vient d'être généré.

Cependant, nous aurions aimé implémenter des dictionnaires pour limiter notamment le nombre de paramètres de certaines fonctions, ce qui aurait fait grandement gagner en lisibilité. Un autre point sur lequel nous aurions aimé plus insister est la rédaction du rapport. En effet, le contenu d'un rapport ne nous paraît pas encore assez clair et pose des difficultés.

[illegible]