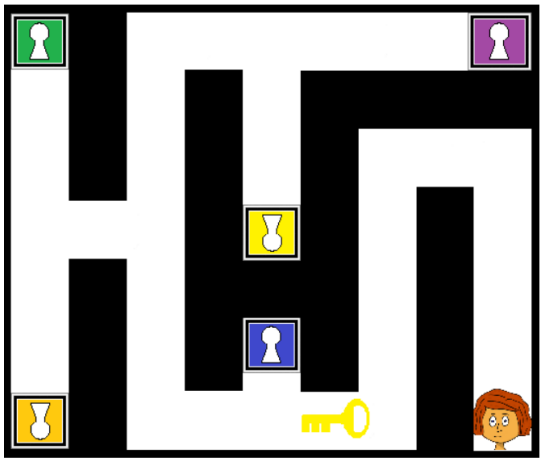
• Fonctionnement du programme:

Le programme attend la pression d'une touche de déplacement afin de lancer une fonction qui va déplacer le personnage dans la direction souhaitée.
En revanche, avant chaque déplacement, le programme vérifie le contenu la case dans laquelle le personnage s'apprête à se déplacer :

- Si cette dernière contient un caractère représentant l'un des murs du labyrinthe, il empêche le déplacement du personnage.
- Si elle contient un caractère avec lequel on doit interagir (porte, clé…), le programme va exécuter la fonction associée à ce caractère.

À chaque déplacement, le programme met à jour une variable stockant le nombre de déplacement du joueur ; Il y ajoute 1 à chaque fois
Cette variable est affichée à la fin de la partie et permet au joueur d'avoir un retour de sa performance.

•Gestion de l’affichage du jeu:
Pour l’affichage du jeu la documentation du projet nous a proposé l’image ci-contre comme exemple.

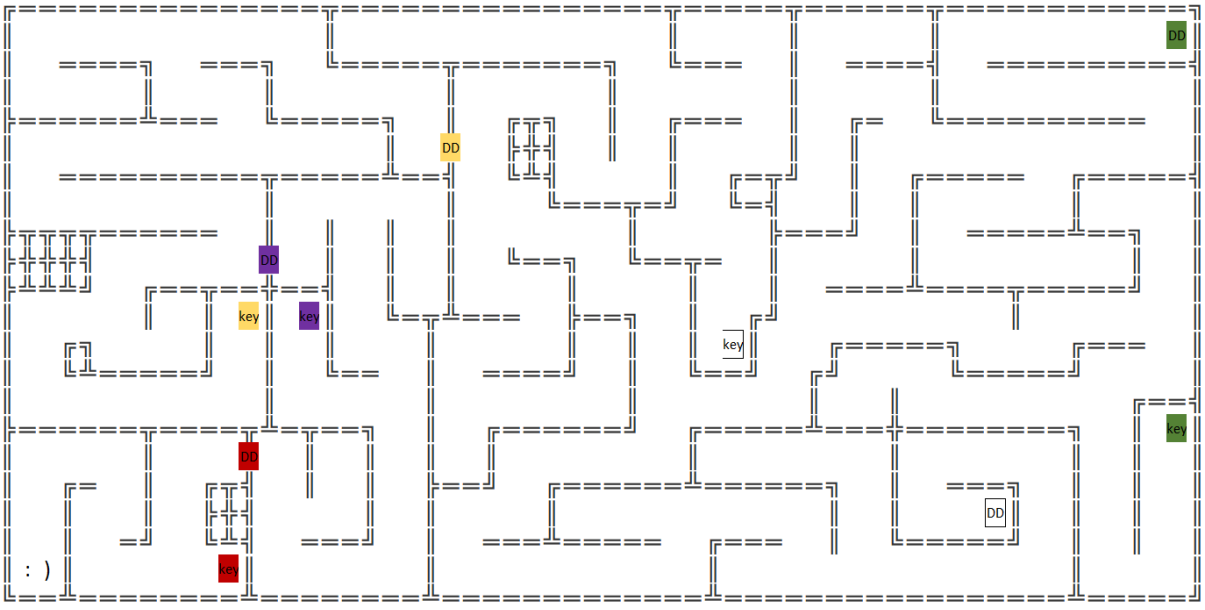


Nous avons alors commencé par faire les objets de notre jeu:

Puis nous avons constaté qu’il était trop ambitieux de vouloir ajouter des images dans notre jeu en assembleur 8086.

Mais loin de baisser les bras nous avons choisi de modéliser notre labyrinthe en caractères ascii afin de faciliter son affichage. Étant toujours aussi ambitieux nous avons choisi de faire un labyrinthe de dimension 60x22 contrairement à l’exemple de la documentation du projet ou le labyrinthe semble faire 9x8 de dimension.

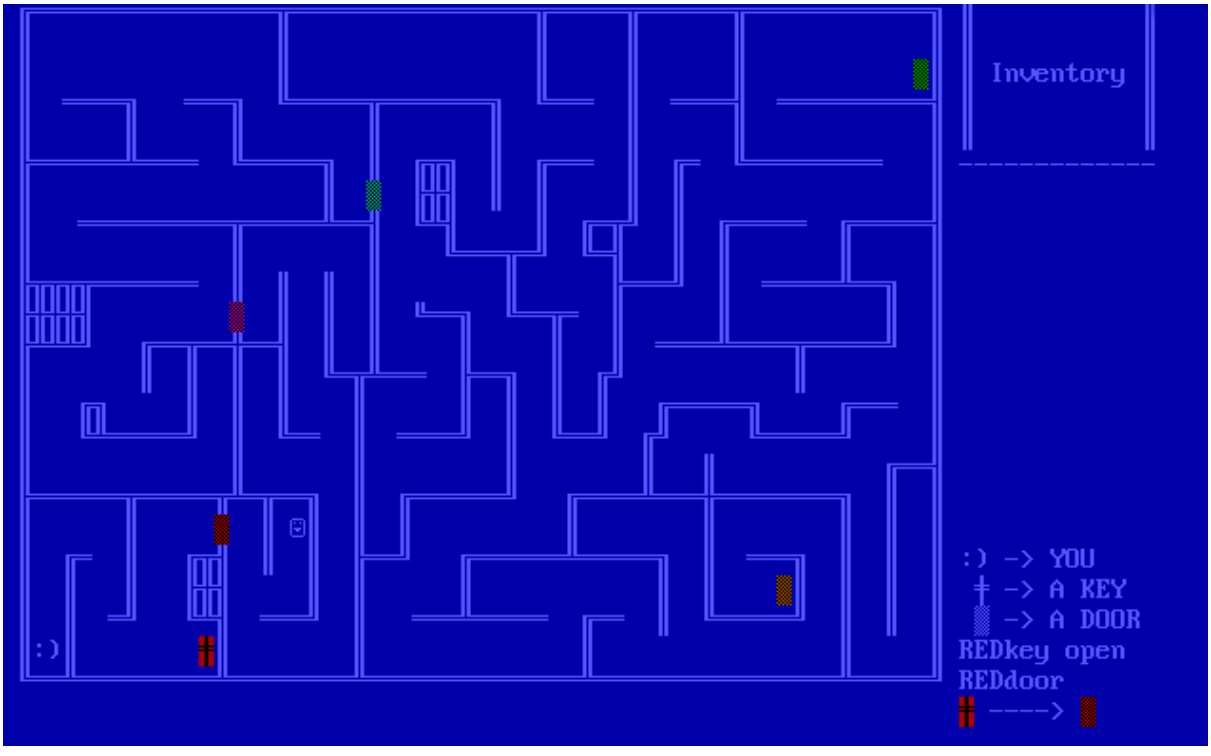
Avant de commencer à créer le labyrinthe dans le jeu nous en avons fait un plan sur excel aux dimensions de notre écran.



Nous avons utilisé les caractères ASCII suivants :

‖	═	╝	╗	╔	╚	╠	╣	╩	╦	╬	ï	▒
186	205	188	187	201	200	204	185	202	203	206	216	177

Le résultat en jeu est satisfaisant



Nous avons généré les couleurs via le code en assembleur

Difficultés rencontrées:

Étant donné que la création du labyrinthe monopolisait énormément de lignes de code dans notre fichier principal (environ 2000 lignes), et pour des raisons d’optimisation, la création du labyrinthe a son propre fichier asm qui est utilisé dans le programme principal.

Nous avons même dû séparer le fichier de code assembleur servant à dessiner le labyrinthe en deux parties, car il était ignoré à la compilation (trop volumineux). Actuellement, compiler l’ensemble du code de notre jeu prend approximativement une heure.