1 Présentation du projet

Le jeu sur lequel j'ai travaillé est un jeu de labyrinthe en vue du dessus, avec comme particularité que les actions de notre personnage sont limitées : il ne peut aller que vers le haut ou la droite, l'empêchant ainsi de revenir en arrière. Mon binôme a abandonné la licence quelques semaines après le début du projet, j'ai donc écrit moi-même l'entièreté du code.

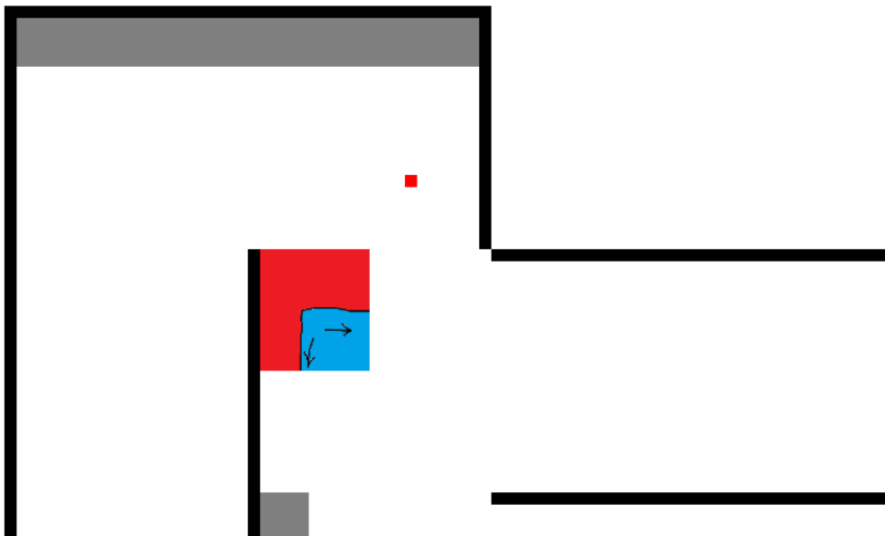

Figure 1.1: Un exemple d'un niveau du jeu

1.1 Structure du projet

Le projet a été fait sous Ocaml, avec le modèle OCS comme prévu, et le jeu est bien de type AABB. Mes composants principaux sont les murs, lae joueureuse, la caméra et les "zones", qui sont les différents endroits où lae joueureuse peut recevoir des pouvoirs, interargir avec l'environnement ou créer des effets. On va donc voir chacun de ces objets un à un, avec les problèmes et contraintes que j'ai rencontré.

2 Objets, composants et systèmes

2.1 Les murs

Les murs sont simplement des rectangles de couleur noire, le plus difficile a faire fut de créer les collisions entre le personnage et ces murs. Etant donné que je ne voulais pas faire un jeu avec un modèle physique poussé, je me suis cantonné au code fourni en TD, qui fonctionne globalement très bien après quelques réajustements.

2.2 Lae joueureuse

Le personnage est un carré rouge qui se déplace. J'ai préféré utiliser des couleurs et graphismes extrêmement simples, la quasi-totalité des objets du jeu sont donc juste des couleurs unies. Notre personnage apparait au début de chaque niveau aux mêmes coordonnées, et avec les mêmes directions acceptables : le haut et la droite. C'est un des objets qui a été le plus compliqué a créer, car à chaque fois qu'un nouvelle fonctionnalité était ajoutée, il fallait recréer de toutes pièces des composants. Pour faciliter le debug, j'ai rajouter 2 touches qui autorisent ou non les mouvements interdits : e pour les autoriser, a pour les interdire, ce qui m'a permi de beaucoup plus facilement voir si j'avais correctement créé les niveaux.

2.3 la caméra

La caméra est un objet à part entière, qui me permet de suivre mon personnage dans ses déplacements dans les niveaux. Elle existe en deux mode : freecam true et false. Le mode false fait que quand la caméra est sur le bord inférieur gauche du niveau, elle arrête de suivre lae joueureuse, afin de bien montrer que le niveau ne s'étend pas plus bas. Malheureusement, lorsque j'ai plus tard réalisé le créateur de niveau, le mode freecam false causait des bugs, notamment sur les niveaux où le personnage passait en dessous de son point de départ, j'ai donc du le passer à true. J'ai aussi fait attention à ce que les seuls objets affichés soient ceux qui intersectent avec la caméra, afin de ne pas afficher des objets que l'on ne peut pas voir.

2.4 les zones

C'est de loin l'objet avec le plus de code associé. Il regroupe toutes les zones de jeu qui vont avoir un effet sur lae joueureuse : Changement de mouvements autorisés, zone de mort, zone de victoire, téléportation. J'ai voulu créer une zone de glace, mais j'ai au final décidé de ne pas l'implémenter, il en reste en revanche un squelette non fonctionnel dans le code. Afin de

faciliter la création de niveau, j'ai décidé de faire une fonction de création de zone en général, ainsi qu'une spécifiquement pour certains types de zone.

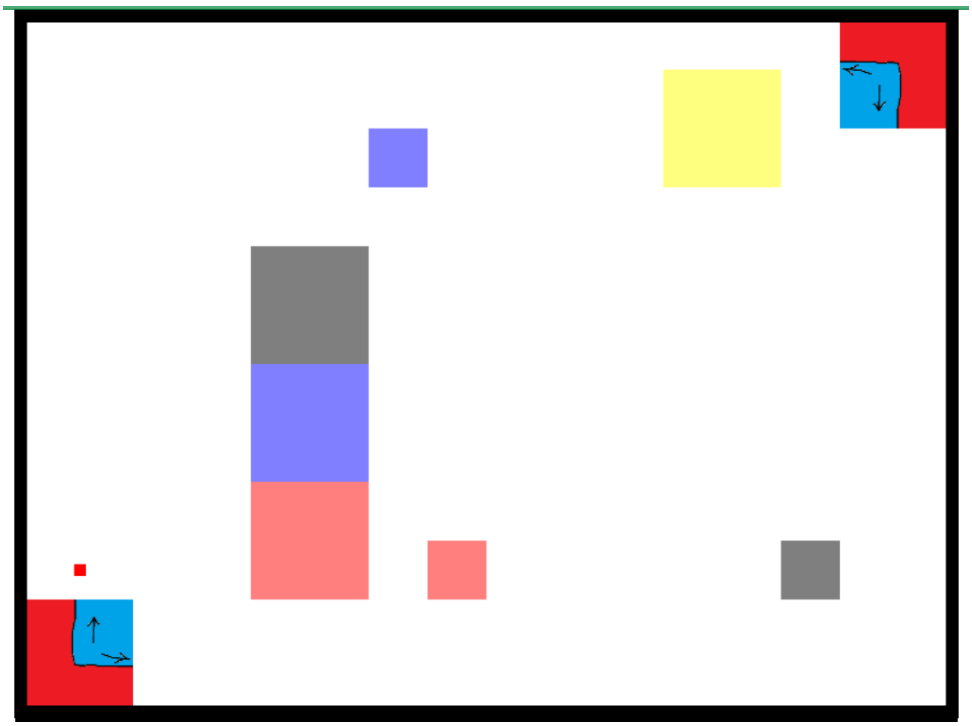

Figure 2.1: Ma zone de test, où toutes les différentes zones sont exposées

2.4.1 Zones de changement de mouvement

Ce sont les seuls objets avec une image, bien que très rudimentaire. Afin de les afficher, je charge d'abord toutes les images (au nombre de 16) en mémoire afin de n'avoir aucun problème ensuite. Je n'ai eu que très peu de problèmes en les codant. D'un point de vue ludique, c'est la brique principale du jeu, puisque le changement des options de mouvement permet de créer plusieurs situations intéressantes. Ces zones permettent aussi de s'en affranchir, comme dans le niveau 7, grâce à la zone "all moov" qui permet l'accès à toutes les directions et donc proposer d'autres défis. Elle sert aussi de "Power Up" dans certains niveaux comme le 9, où l'acquisition de cette zone permet de finir le niveau facilement.

2.4.2 Zone de mort

Ces zones sont grises et servent à "réinitialiser" le niveau quand on est bloqué. Le défi était de replacer tout le niveau à son stade initial, notamment le personnage, et j'ai d'abord pensé à supprimer le personnage et le recréer au bon endroit, mais cela revenait à faire la même chose pour la caméra et était assez compliqué, j'ai donc juste décidé de changer les valeurs des objets directement. Un problème assez récurrent que j'ai découvert en jouant était que dès que l'on

entrait dans une zone de mort, on réapparaissait et se remettait à bouger directement, ce qui ne rendait pas le jeu agréable. J'ai donc incorporé un timer qui empêche tout mouvement pendant 30 frames, afin de laisser à lae joueureuse le temps de lâcher les touches et de réfléchir à ce qu'iel veut faire.

2.4.3 Zone de téléportation

Il y a en réalité 2 types de zone de téléportation : les entrées et les sorties. On ne peut pas savoir quelle entrée (en rouge) est reliée à quelle sortie (en bleu) avant de les avoir empruntées, afin de pouvoir créer des niveaux où l'on doit mourir pour savoir quel est le bon chemin. Pour relier une entrée à une sortie, j'ai utilisé leur id, où la zone d'entrée a un paramètre qui contient l'identifiant de la zone de sortie. J'ai eu quelques problèmes quant au placement de la caméra, mais ceux-ci se sont résolus tout seuls dès que j'ai passé tout les niveaux en mode "free cam".

2.4.4 Zone de victoire

Ce sont les carrés jaunes qui indiquent la fin du niveau. Afin de donner l'indication au jeu de passer au niveau suivant, j'ai utilisé le personnage comme véhicule d'un booléen "niveau fini". Dans la création de niveau, elles sont générées avec 50 pixels d'écart afin de les mettre au milieu du chemin.

2.4.5 Zone de glace

Ce sont les seules zones qui n'apparaissent pas dans le jeu final. Le but était d'avoir des zones où dès que l'on arrivait dessus, on perdait le contrôle de notre personnage qui allait glisser jusqu'à la fin de la zone. Cela a posé de nombreux problèmes sur où stocker les mouvements autorisés, comment le faire bouger, que se passe-t-il si le personnage rencontre un mur pendant qu'il glisse... J'ai donc décidé de ne pas faire faire aboutir cela afin de me concentrer plus sur le coeur de ce projet, la création de niveau.

3 Création de niveau

La création de niveau a été la partie la plus intéressante de ce projet. Il a fallut réfléchir à la manière dont seraient stockés les niveaux, comment les lire, quelles contraintes cela créerait pour le game design... Pour se faire, j'ai d'abord dessiné sur papier des niveaux assez simplistes pour réfléchir sur les spécificités dont mon système avait besoin.

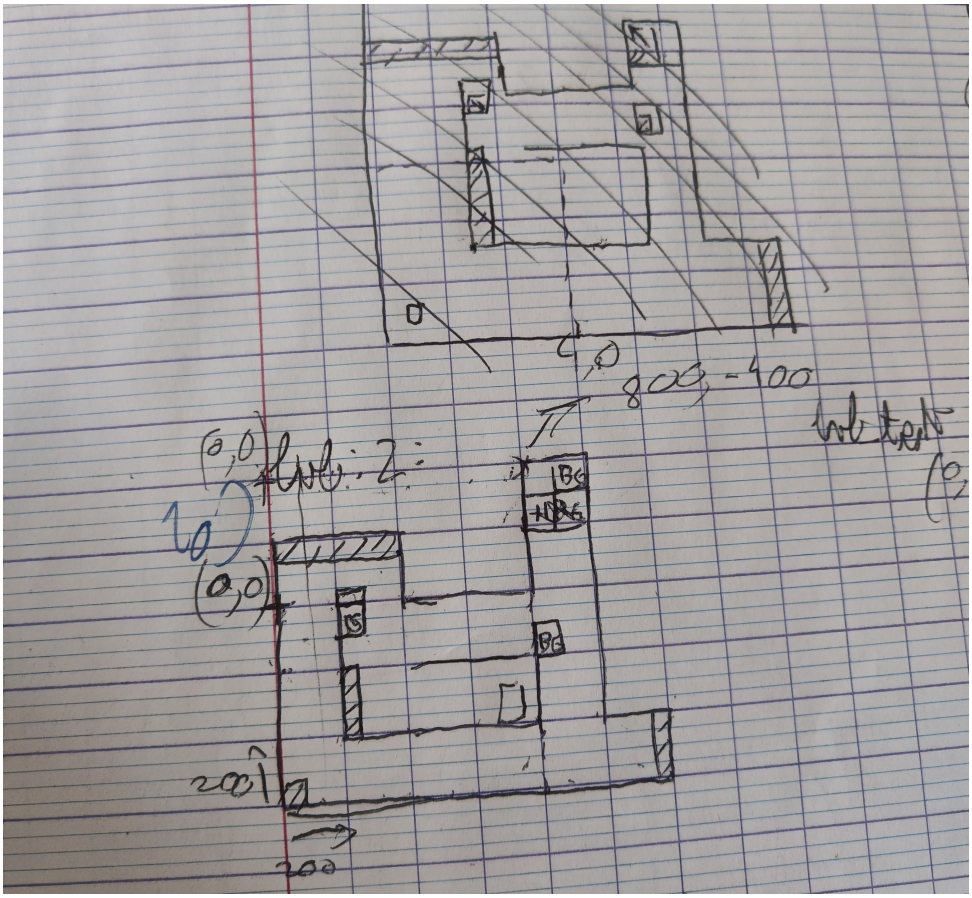

Figure 3.1: Les premiers dessins du niveau 10

3.1 Comment s'écrit un niveau

J'ai d'abord pensé à créer des murs de 100 pixels de large, afin de créer des "cases" de 100x100 où l'on pourrait mettre des zones et des murs. Mais quand j'ai calculé, pour le niveau 10, le nombre de murs et de zones créées par rapport au même niveau créé "à la main" (donc directement codé, dans les fonctions init wall et init zone), je me suis aperçu que l'on passait de 14 murs et 7 zones dans la version "à la main" à 61 murs et 12 zones dans la version avec des cases de 100x100, ce qui causerait un facteur 4 ou plus sur le nombre de mur, ce qui est bien trop haut.

J'ai donc décidé de coder séparément les murs et les zones : les murs seraient créés sur les côtés des cases de taille 200x200, et les zones seraient des cases de taille 100x100. On passe ainsi, pour le niveau 10, à 32 murs et 12 zones, ce qui est déjà bien plus raisonnable si l'on veut créer des niveaux de grande envergure.

Mais je me suis encore heurté à un autre problème : les zones de taille 100x100 étaient cette fois assez souvent trop grandes ! J'ai donc ajouté une possibilité de découper une case de zone de 100x100 en 4 cases de 50x50, permettant plus de flexibilité sur le positionnement des zones. Toujours avec le niveau 10, cela augmente le nombre de zones de 12 à 20, mais cela est beaucoup moins grave puisque l'on crée bien moins de zones que de murs.

Afin de garder à l'écrit la manière exacte d'écrire un niveau, j'ai enfin créé un fichier "How to create.txt" qui explique tout en détail.

3.2 Codage

La totalité du code de création de niveau a été écrit dans le fichier "load lvl". Le premier problème fut de savoir comment charger correctement un fichier. Etant donné le peu de niveaux, j'ai opté pour le fait de charger la totalité des niveaux en mémoire dès le lancement du jeu, afin de n'avoir aucun temps de chargement. Tout ceci se fait avec le fichier "level manager". Lors de la lecture du fichier, je me suis aperçu que la décision d'où serait le coin supérieur gauche du labyrinthe serait crucial : il permet de placer bien plus facilement tout les morceaux du niveau ainsi que le personnage facilement. Il a été décidé à posteriori d'avoir toujours les mêmes coordonnées de départ et les mêmes directions disponibles au début de chaque niveau, par mesure de simplicité et car leur inclusion n'était pas indispensable suite au passage de tout les niveaux en free cam. Il en reste néanmoins des traces dans la création de niveau avec la première ligne, mais celle-ci n'est juste pas lue.

La création des murs utilise grandement la récursivité et l'aspect fonctionnel d'Ocaml, afin de ne pas avoir à créer un nombre fixe de case de mur par ligne. Le code s'est révélé assez simple pour cette partie, malgrès des bugs tel que l'affichage de la moitié des murs dans le mauvais sens, mais qui étaient dû à des fautes de frappe.

La création des zones s'est elle aussi faite avec beaucoup de récursif, notamment dans la partie des zones de mouvement. Contrairement aux murs, on ne fait pas ça case par case, mais on indique directement les coordonnées des cases où l'on veut des zones, car il y en a peu par rapport au nombre de cases possibles. La partie "Split" a subi plusieurs changements mais

fonctionne au final assez bien.

3.3 Ecriture des niveaux

Pour chaque niveau, j'ai d'abord dessiné ce à quoi il ressemblait sur une feuille. J'ai surtout créé des niveaux simples mettant en avant les différentes possibilités données par le créateur de niveaux, mais quelques uns tels que les niveaux 8, 9 et 10 sont des réels niveaux mettant à l'épreuve certaines idées de game design.

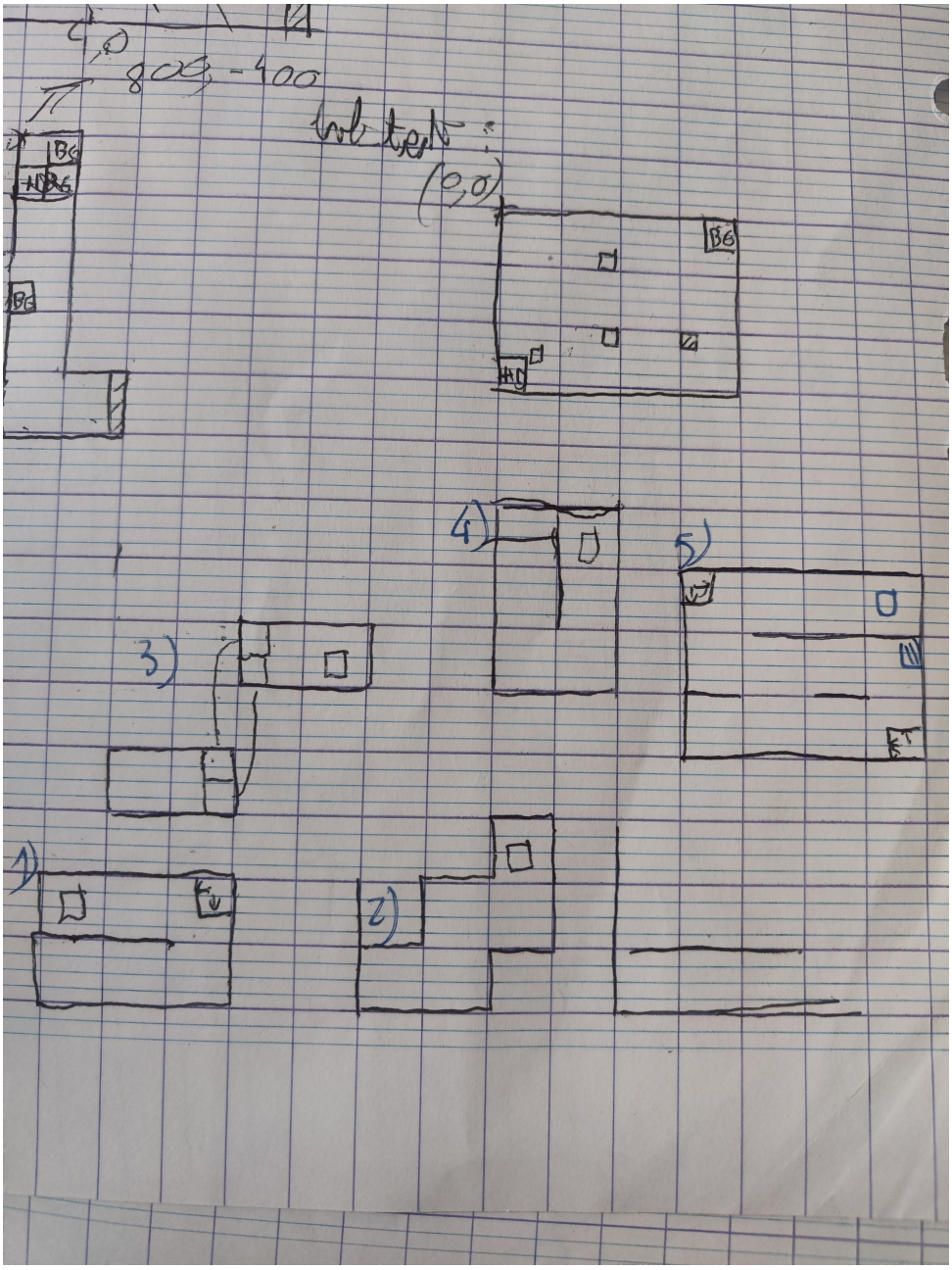

Figure 3.2: Les dessins des niveaux 1 à 5, qui ont été ensuite retravaillés sur ordinateur

Un problème auquel j'ai été très vite confronté fût le fait que les murs étaient créés sur les bords intérieurs des cases, ce qui crée parfois des trous au niveau des intersections, comme dans le niveau 10. Pour résoudre cela, j'ai décidé d'alterner les murs sur les côtés "intérieurs" et "extérieurs", ce qui s'est avéré efficace mais donne l'impression que certaines cases ne sont pas carrées. Le niveau 10 est en double afin de montrer la différence entre les versions avec et sans trou.

L'écriture des zones peut se faire dans n'importe quel ordre, mais j'ai décidé de toujours les trier d'abord par colonne puis par ligne, ce qui s'est révélé bien plus pratique pour debugger par la suite.

3.3.1 Niveau test

Ce niveau est inaccessible normalement, mais on y accède en tapant 0 dans la variable level, à la ligne 10 du fichier "game.ml". C'est le niveau dans lequel tout les tests de zone étaient fait.

3.3.2 Niveaux 1 à 5

Ces niveaux présentent rapidement les différentes mécaniques de jeu :
Le niveau 1 montre comment se déplacer.
Le niveau 2 introduit les zones de mouvement.
Le niveau 3 introduit les téléporteurs.
Le niveau 4 introduit les zones de mort ainsi que la necessité parfois de devoir recommencer le niveau avec plus d'informations sur le chemin à prendre.
Le niveau 5 a un trou qui renforce le fait de ne pas pouvoir revenir en arrière si on le prend.

3.3.3 Niveaux 6 à 10

Ces niveaux sont des ébauches de ce que l'on pourrait retrouver dans un jeu fini :
Le niveau 6 demande de choisir parmis 2 téléporteurs, l'un offrant une mort certaine, l'autre garantissant la fin du niveau. C'est un des niveaux qui aurait bénéficié d'une caméra statique.
Le niveau 7 est aussi un niveau à choix, où il faut savoir si l'on veut aller en haut ou en bas pour passer.
Le niveau 8 consiste à suivre le chemin donné par les directions autorisées, afin de réussir à atteindre la fin.
Le niveau 9 tourne autour du fait de gagner le "All moov" et de ne pas le perdre ensuite.
Le niveau 10 (et 10 bis) fut le premier niveau créé et le seul entièrement abouti.

4 Conclusion

4.1 Problèmes non résolus

Certaines fonctionnalités n'ont au final pas été implémentés, pour différentes raisons : Il était prévu que lae joueureuse laisse une trace des endroits où iel était passé.e quand iel meurt, sous forme d'une ligne rouge qui s'éclaircirait avec le nombre de mort. J'ai décidé de ne pas l'implémenter car cela n'avait au final pas grand interêt étant donné la durée des niveaux proposés.
La glace n'a pas pu être implémentée, comme expliqué plus haut.
La génération procédurale de niveau aurait pu être codée, mais c'était mon binôme qui voulait s'en charger, et j'ai finalement décidé de ne pas m'en occuper.
Un projet de zone de mort active seulement une seconde sur deux a été proposée, mais non implémentée par manque de temps et d'utilité.
Le mode free cam false aurait pu être utile, ainsi qu'un mode où la caméra serait statique.

4.2 Récapitulatif

J'ai donc créé un jeu avec un personnage ayant des directions autorisées et d'autre interdites, des zones donnant différents pouvoirs ou déclenchant différentes actions, certaines ayant des graphismes rudimentaires. J'ai aussi codé un créateur de niveau fonctionnel, et ai créé 10 niveaux pour présenter les possibilités de game design offertes. Ce projet m'a appris les contraintes et forces de faire un projet entièrement seul, même si cela n'était pas prévu au début. J'espère que ce jeu vous plaira.

5 Addendum

Le choix du niveau de départ se fait dans la variable level à la ligne 10 du fichier "game.ml"
Le choix de la freecam se fait dans la variable free cam à la ligne 6 du fichier "move.ml"
Le choix de l'activation ou non des touches de debuggage se fait dans la variable debug mode à la ligne 7 du fichier "key.ml"