1) Présentation du projet dans l’ensemble :

Pour ce projet maths-info, nous avons choisi de créer un flipper informatique sous langage c. Le but de ce jeu est de renvoyer une balle à l’aide de clapets commandés par les flèches du clavier pour ne pas la faire tomber tout en marquant un maximum de points. Ce jeu possède trois niveaux de difficulté qui utilisent tous trois une trajectoire aléatoire de départ. Il est tout de même possible de choisir son vecteur de direction de départ si besoin est.

Partie de chacun :

Nous avons travaillé ce projet toujours ensemble car pour nous il était difficile de créer une partie sans faire appel à une autre, ce qui signifie que nous devions créer et tester chaque code à deux. Nous avions donc besoin à chaque instant des capacités de chacun.

2) Présentation générale du programme :

Le programme peut se décomposer en trois grandes parties : La gestion des obstacles, la gestion du mouvement de la bille et l’interface utilisateur. Pour cela, le code est réparti dans plusieurs fichiers source en .c accompagnés de leurs headers en .h .

Le programme se compile sans erreur/warning via la commande : gcc -o flipper flipper.c outils.c obstacle.c mouvement.c -lm -lX11 -Wall

A) Explication des différentes parties du code :

Le fichier source flipper.c :

Ce fichier s’occupe de la gestion de la fenêtre de jeu, il permet à l’utilisateur de jouer au jeu en appelant les fonctions des autres fichiers source. Dans cette partie, on récupère les choix de l’utilisateur quand à la difficulté du jeu ou aux conditions de départ de la bille. Elle permet également en fonction du choix en cas d’échec du joueur de recommencer la partie tant qu’il le souhaite sans se revoir proposer l’explication du jeu à chaque partie.

Le fichier source mouvement.c :

Dans ce fichier, on s’occupe du mouvement de la bille, celui-ci est en permanence calculé et adapté en fonction des conditions appliquées à la bille. C’est également dans ce fichier que les actions de l’utilisateur sont gérées avec le mouvement des clapets.

Le fichier source obstacle.c :

Dans ce fichier, on gère les diverses fonction de créations des obstacles ainsi que l’affichage du flipper.

Le fichier source outils.c :

Ce fichier contient divers outils de calcul utilisés par le programme ne nécessitant pas l’usage de variables globales de mouvement.c ce qui permet d’alléger le fichier mouvement.c .

B) Explications des fonctions/procédures principales :

Pour le fichier source mouvement.c :

bool mouvementbille(bool succes,int affichage,float difficultee) :

Cette fonction récursive est le cœur du programme, elle gère entièrement le déplacement de la bille, elle calcule la trajectoire de la bille, l’affiche, s’occupe des animations, de la gestion du mouvements des clapets, etc …

En fonctionnement normal, celle fonction est appelé 120 fois par secondes, c’est pourquoi il est nécessaire de ne pas afficher tout le temps le flipper en entier. L’écran n’est généralement pas fait pour supporter une telle fréquence, on réduit donc à 40 le nombre d’affichage par seconde ceci à également pour effet de réduire le scintillement. Pour augmenter la précision des calculs, les tests des collisions sont eux même scindés en 10, ainsi on effectue jusqu'à 1200 tests de collision par seconde. Lorsque la bille tombe, cela met fin à cette fonction.

void creerobstacles(void) :

Cette procédure initialise les obstacles utiles pour la gestion des collisions dans la variable globale. Ceux-ci ne sont pas affichés. Ils servent à la création des diverses droites nécessaires aux calculs d’impacts.

void collision\*\*\*\*\*(void) :

Ces procédures s’occupent de modifier la bille en fonction de la collision qui se produit. Elles appellent les fonctions du type bool collision\*\*\*\*\*qui l’informent sur la collision ou non.

void gererlagravite(float difficultee) :

Cette procédure applique une modification à la vitesse créant l’illusion d’une inclinaison sur l’axe y.

void jouerauflipper(float difficultee,dXPoint vecteur) :

Cette procédure correspond au lancement du jeu après choix de l’utilisateur, elle comporte une petite animation pour le lancement de la bille.

void animation\*\*\*\*\* :

Ces procédures réalisent une fonction esthétique donnant l’illusion d’un choc sur les obstacles augmentant le score. Un son est aussi émit.

void debogageflipper(void) :

Cette procédure a pour but de corriger des bugs éventuels concernant le flipper, par exemple, elle vérifie que le calcul effectué par l’ordinateur n’est pas faux, sinon le calcul est réalisé à nouveau.

void chocreel(void) :

Cette procédure permet de donner l’illusion que les chocs sont inélastiques, on augmente la vitesse sur les obstacles qui font rebondir et on la diminue sur les autres.

int gameover(void) :

Cette fonction gère la fin de partie en cas de chute de la bille. Elle permet à l’utilisateur de recommencer ou d’abandonner ce qui met fin au jeu.

Pour le fichier source obstacle.c :

void afficher\*\*\*\*\*(int k) :

Ces procédures affichent les obstacles dont l’indice est passé en paramètre ou la bille.

void afficherflipperdepart(void) :

Cette procédure initialise la variable globale permettant d’afficher les obstacles.

void afficherflipper(void) :

Cette procédure affiche le flipper en dessinant dans un ordre précis les obstacles.

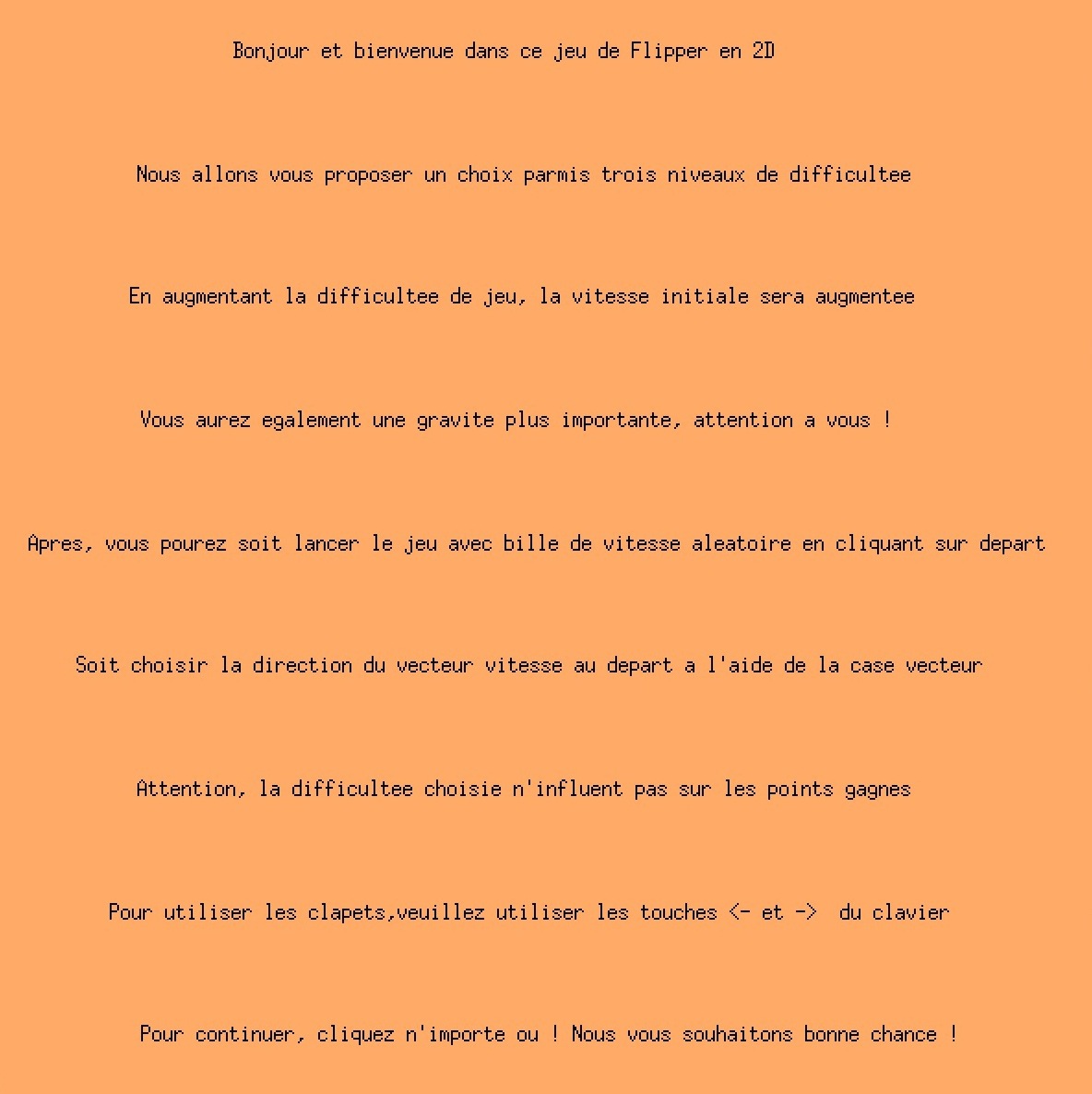
XPoint \*dXPtoXP(dXPoint \*tab,int nbpoints) :

Cette fonction permet de convertir un tableau de dXPoint en un tableau de XPoint .

3) Les trois phases de fonctionnement du programme :

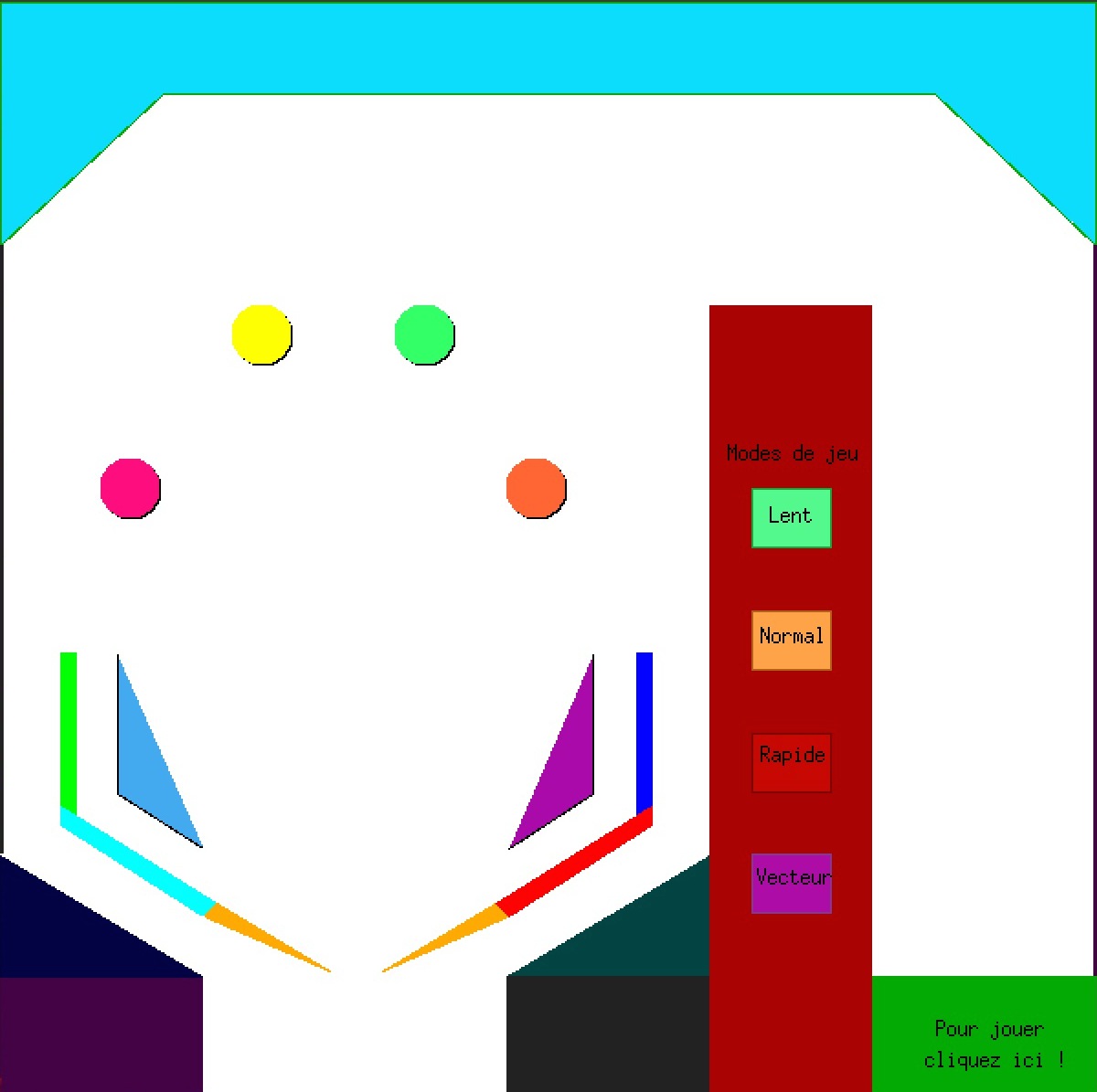
1. Présentation du programme :

On utilise cette première page de jeu comme notice de jeu pour l’utilisateur. Cette page apparaît que lors du démarrage du jeu et non pas à chaque fois que l’on perd. Nous rappelons qu’il faut choisir son niveau de jeu avant de cliquer sur « jouer ».

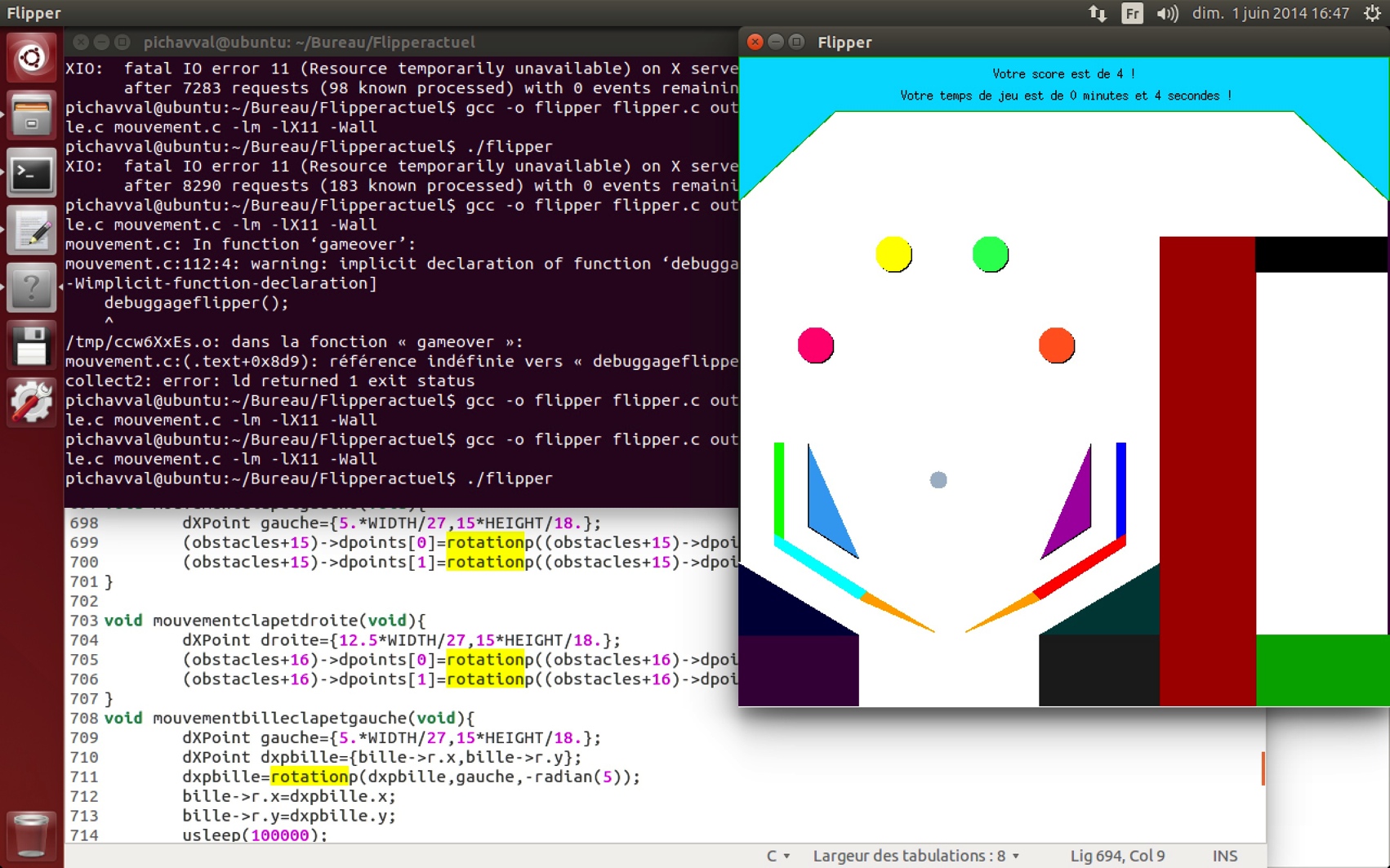


1. Interface de jeu :

1ère image : On observe ici correctement les différentes cases pour le démarrage de jeu (lent, rapide etc..). On remarque les obstacles et les clapets sont bien place.

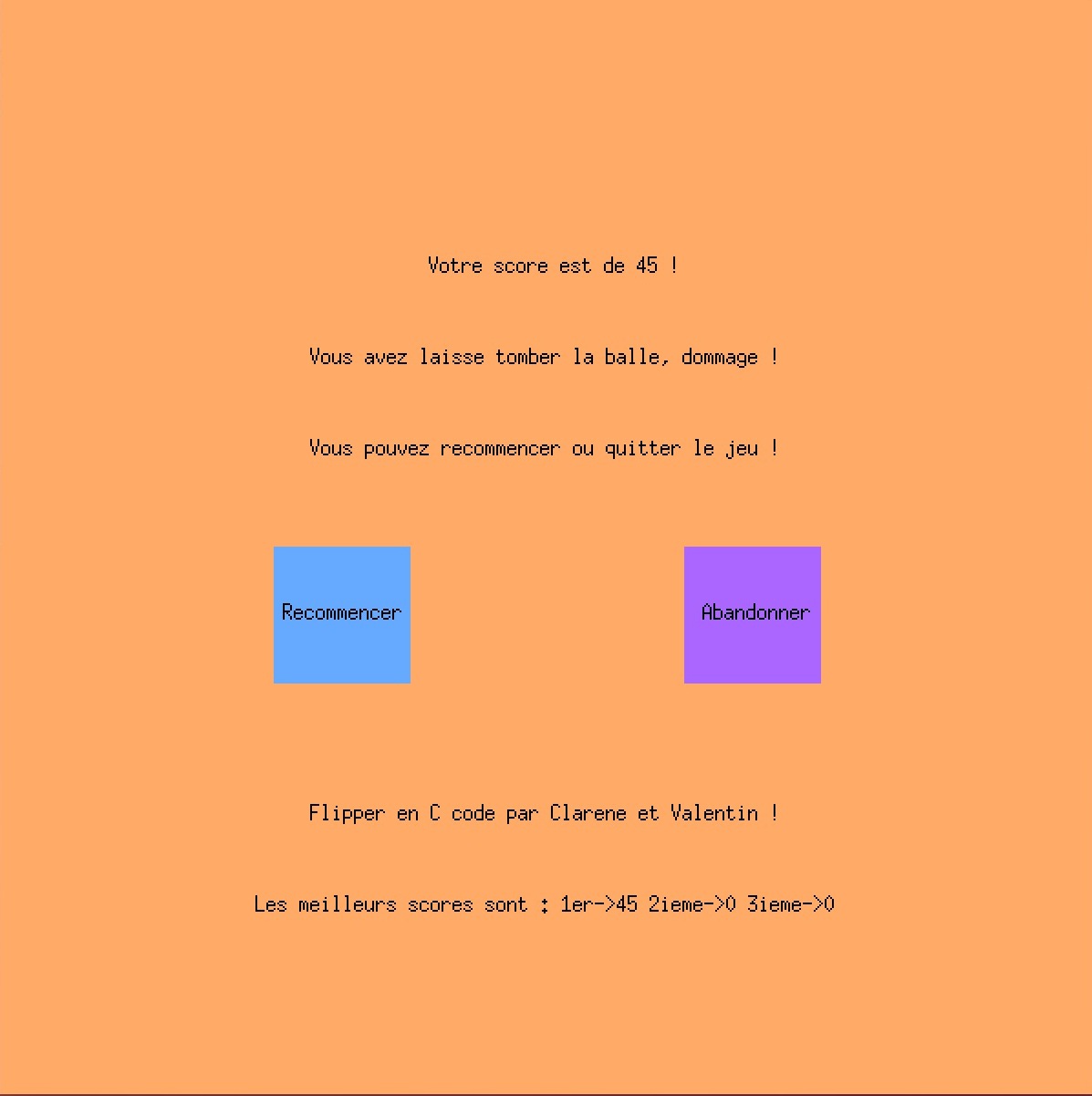


2ème image : On observe en haut de la fenêtre le temps qui s’écoule au fur et à mesure que l’on joue. Le score y est aussi affiché.



1. Menu de fin de jeu :

Cette page de fin de jeu permet de laisser le choix à l’utilisateur de recommencer ou quitter le jeu. Il est également affiché les 3 meilleurs scores depuis le premier lancement du jeu.



4) Formules de calculs mathématiques utilisés par le programme :

Dans ce projet, nous n’avons rencontré que très peu de problèmes mathématiques, les seuls réellement utiles sont les suivants :

Pour le calcul de la distance du centre de la bille à une droite d’équation ax+by+c=0 :

Pour la rotation d’un point A par rapport à un point B pour créer un point M d’un angle z :

M.x= (A.x-B.x)\*cos(z)-(A.y-B.y)\*sin(z)+B.x

M.y= (A.x-B.x)\*sin(z)+(A.y-B.y)\*cos(z)+B.y

Pour la translation d’un point A d’un vecteur v pour créer un point M :

M.x=A.x+v.x

M.y=A.y+v.y

Note 1 :

Les bruits normalement émit par les printf(« \a») ; ne semblent pas émit sur les mac.

ANNEXE :

Figure détaillée de la position des obstacles :

