TELECOM NANCY

Rapport C 2015

Conception d'un jeu vidéo « TRON »

VOYAT Thomas & NORMAND Maxime

Table des matières

Introduction

Ce rapport est une restitution du travail effectué en binôme pour le Projet du moduleC-Shell. L'objectif de ce projet était la programmation d'un jeu vidéo de type TRON où deux motos s'affrontent dans un rectangle, laissant une traînée derrière elles, et essayent de piéger l'autre pour qu'elle entre en collision avec l'une de ces traînées. Nous avons réalisé ce projet au moyen de la bibliothèque graphique SDL.

Outre la conception d'une stratégie de jeu dont la complexité peut être poussée aussi loin que l'on veut, il convient dans les spécifications de respecter la taille de la grille de jeu (34x20).

Pour le binôme, deux formes d'études se sont dégagées : la conception du graphisme du jeu grâce aux outils SDL et la conception des fonctions permettant un bon déroulement du jeu (détection de défaite, arrêt du jeu). Chaque membre du binôme a été désigné comme responsable de l'une de ces études, bien que la progression se soit déroulée avec des concertations régulières. Une batterie de test a due être mise en place au fil de la progression dans le travail.

Le travail a finalement abouti à un exécutable tron généré à partir d'un main tron.c accompagné de son architecture de fonction Graphiques. Celui-ci est capable de faire s'affronter deux joueurs sur la même machine.

Analyse

1) *Travail de conception et algorithmes*

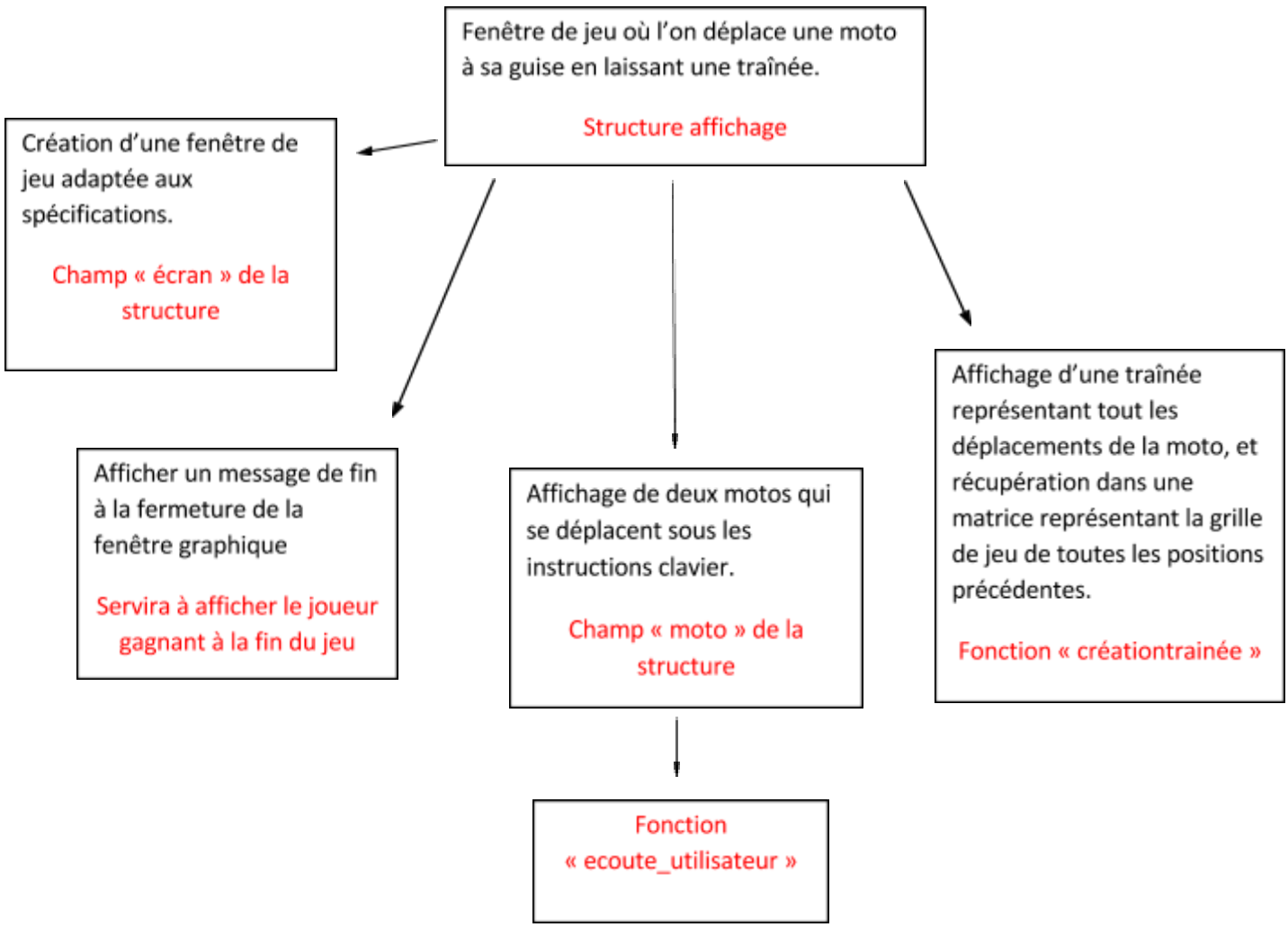
L'organisation du travail de conception a reposé sur une division de l'étude en deux tâches principales :

- **Conception du graphisme du jeu**

Ce sont les outils SDL qui ont été exploités.

Pour commencer, il a fallu créer une fenêtre de jeu sans, pour l'heure, l'inclure dans le code du joueur. En somme l'objectif fut de créer une fenêtre de jeu avec deux motos se déplaçant et laissant derrière elles une traînée, gardant dans une matrice représentant la grille leurs positions antérieures. Il semble important de noter que nous nous sommes inspirés de l'exemple SDL 1 fourni sur LMS et de tutoriels SDL. Le problème a été découpé en 3 grands sous-problèmes de complexité inégale. Toutes les caractéristiques de graphisme ont été regroupées dans une structure « affichage » qui sera détaillée plus tard.

Figure : Conception descendante de la fenêtre de jeu



Il s'agit maintenant de présenter les différents algorithmes mis en jeu dans ce petit programme d'essai.

Algorithme du programme affichant la moto, la trainée, et gardant en mémoire les positions précédentes : (fichier testtrainee.c)

Déclaration de la structure affichage (détaillée plus loin) :

-écran (fenêtre de jeu)

-moto (image chargée par SDL comportant la moto)

-position moto (via un SDL_Rect c'est-à-dire un rectangle repéré par sa taille et la position du coin supérieur gauche)

-un son et une police ont été accessoirement ajoutés, ce n'est pas notre propos.

Programme en lui-même :

Définition : (via #DEFINE) LARGEUR=1020, HAUTEUR=600, PAS=30

Ce sont les dimensions de l'écran et le pas de déplacement de la surface moto en pixels. Cette définition permet un jeu en écran assez large convenant à tout type de PC et respectant le fait d'avoir une grille de jeu de 34x20 cases. Ces paramètres sont exploités dans les fonctions (voir code).

AFFICHAGE affich

Tableau d'entier (34x20) matrice /*représentant la grille de jeu, une case vaut 1 si la moto est déjà passée dessus, 0 sinon*/

Entier quitter=0

Initialisation SDL /*l'écran et la moto positionnée en haut à gauche apparaissent*/

Tant que quitter=0

Quitter=ecouteutilisateur(affich)

Afficher l'écran /*actualisé par les précédentes manipulations, via la fonction SDL_BlitSurface, voir code*/

Fin tant que

Afficher « Au revoir » sur l'écran SDL /*servira à afficher le résultat du match via la même manipulation SDL*/

NB : Les instructions gérées par des fonctions graphiques dont en rouge, les éléments gérés par des champs de la structure « affichage » sont en bleu.

Fonction ecoute_utilisateur

Entier=ecouteutilisateur(AFFICHAGE affich)

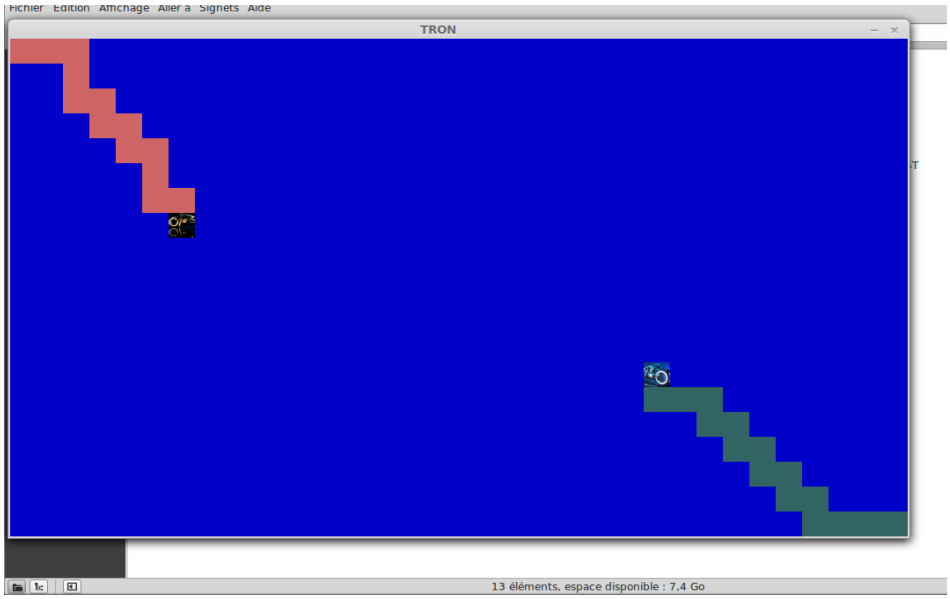
Creationtrainee /*création d'un rectangle de traînée à la position précédente de la moto, qui précède l'évenement clavier*/

Lecture de l'évenement clavier /*Géré par une variable SDL_Event, voir code*/

Actualiser la surface moto par la position du rectangle posMoto en fonction des événements clavier

Actualiser la matrice des positions /*changer les cases à 1 là où la moto est passée, c'est-à-dire à la case « position de la moto x PAS », la position de la moto étant donnée en pixel*/

Retourner 0 /*cette ligne n'est atteinte que lorsqu'on ferme la fenêtre*/



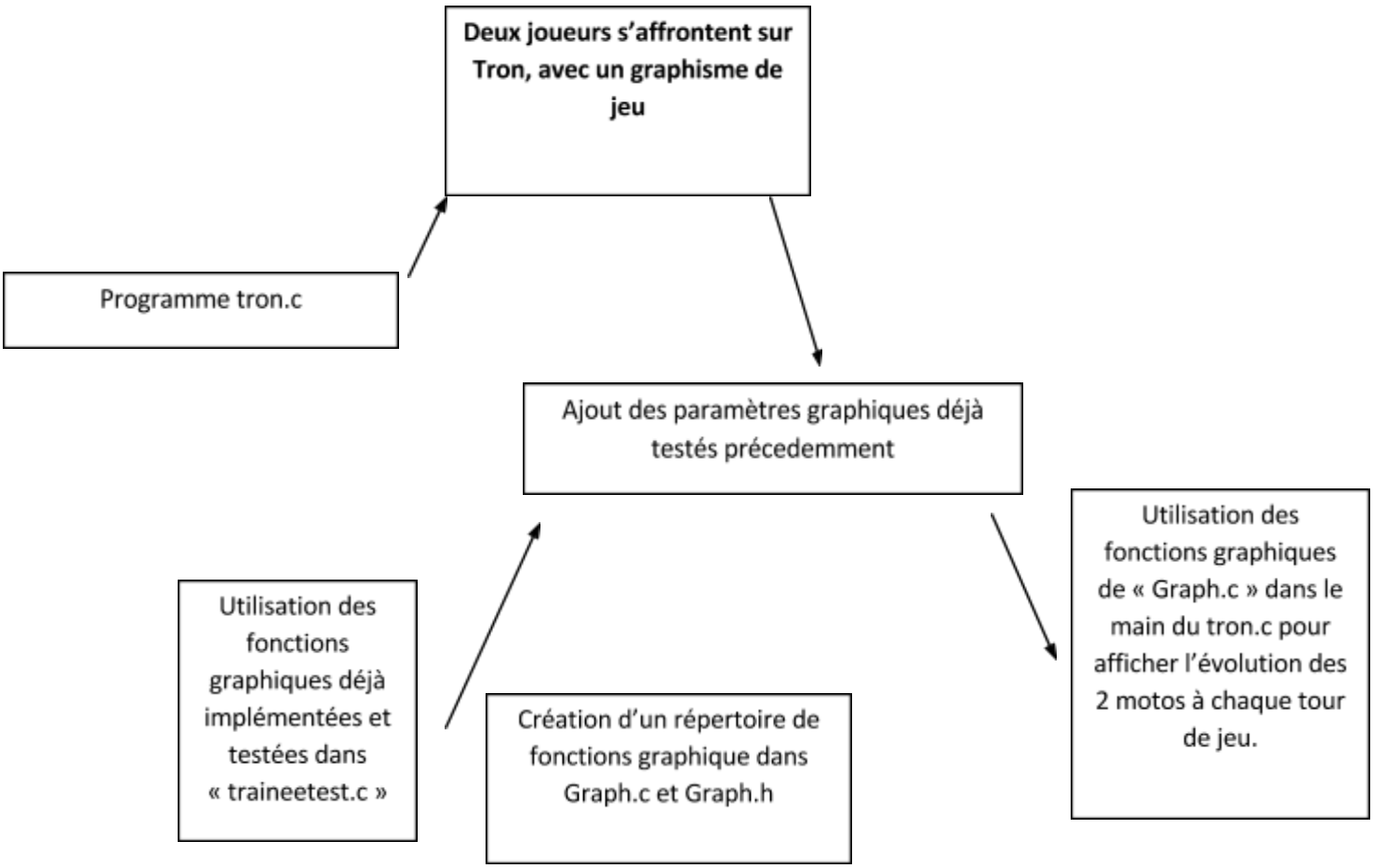
Cependant à ce stade les motos peuvent traverser les trainées, se croiser entre elles et il réside encore quelques problèmes quand les motos arrive sur les bords des carrés.

Ainsi dans la suite de l'étude nous avons tenté de réaliser un mode 2joueurs avec les règles du jeux et sans les problèmes que nous avons cités.

- **Conception du mode 2joueurs**

Par la suite, nous avons réutilisé ce que nous avions fait dans traineetest.c.

Figure : Implémentation des paramètres graphiques sur le Joueur (représenté par le fichier tron.c)



On présente désormais la partie graphique du déroulement d'un tour de jeu dans « tron.c ».

Avant le début des tours :

InitialiserSDL

A chaque tour :

Récupérer les données de position des 2 joueurs /*Voir le code pour plus de détail*/

Position Moto 1=Position 1 donnée par les coordonées x,y x PAS /*Le rectangle SDL posMoto1 est actualisé (voir structure « affichage »)*/

Position Moto 2=Position 2 donnée par le moteur x PAS

Creationtrainee1(affich, Rectangle de position de la moto1) /*crée un rectangle orange à la positon de la moto 1*/

Creationtrainee2(affich, Rectangle de position de la moto2) /*crée un rectangle bleu à la positon de la moto 2*/

Dès que la fin du jeu est détectée :

Afficher message de fin

Utilisation de la matrice représentant la grille de jeu

Il s'agit d'une matrice rectangulaire que l'on choisit de taille 34x20 : les 0 représentent des cases libres d'accès et les 1 représentent les cases à éviter pour gagner.

- On l'initialise à 0 partout sauf sur tous les bords du tableau pour représenter les parois de la fenêtre qu'il ne faut pas heurter.
- Dans le programme tron.c, à chaque tour de jeu, on incrémente à 1 les cases où les motos se sont placées pendant ce tour.

Conception de la fonction de détection de fin de jeu

Celle-ci est essentielle et est appelée à chaque tour de jeu. Elle suppose de connaître tous les événements qui sont susceptibles de donner lieu à la fin du jeu. Plutôt que donner un algorithme, nous allons expliquer dans un tableau le fonctionnement de cette fonction.

Evénement	**Valeur retournée par la fonction de détection de fin de jeu**	**Cause de l'événement**
Victoire du joueur 1	0	La moto2 est positionnée sur une case valant 1 de la matrice de la grille de jeu
Victoire du joueur 2	1	La moto1 est positionnée sur une case valant 1 de la matrice de la grille de jeu
Fin du jeu sur égalité	2	Les motos 1 et 2 font une faute au même tour ou se rentrent dedans
Le jeu continue	3	Rien de ce qui précède

Deux remarques complètent cette description :

- La fonction detection_fin_jeu a donc besoin en argument de la grille de jeu et des 2 structure joueurs (en fait simplement de leur position au tour de test).
- Cette fonction est insérée dans joueur.c dans la boucle « Tant que fini=0 » associée à l'itération des tours de jeu. Si cette fonction a retourné n=3 au tour précédent, on effectue une itération normale, sinon (si n=0,1 ou 2) on incrémente fini à 1 pour sortir de la boucle « tant que ». La fonction affich_victoire affiche selon n le message donnant l'issue du match.

2) La Structure Affichage

La nécessité de la conception de cette structure se dégage par les algorithmes précédents. Pour les détails de son implémentation, se référer au code fourni.

Celle-ci permet comme on l'a vu dans la partie « conception du graphisme du jeu » de regrouper tous les paramètres graphiques dans un seul type de donnée. Il est déclaré comme suit avec les commentaires qui s'y attachent :

```
typedef struct affichage {
```

```
SDL_Surface *ecran, *moto1,*moto2;

/* Les positions des surfaces sont données par des rectangles

dont les attributs .x et .y définissent le coin en haut à gauche

et .h et .w la hauteur et la largeur*/

SDL_Rect posMoto1, posMoto2;

/* variable qui contiendra le son à jouer à chaque coup */

Mix_Chunk *son;

/* variable qui contiendra la police à utiliser pour l'affichage de texte */

TTF_Font *police;

} AFFICHAGE;
```

Les éléments essentiels étant l'écran, qui contient les paramètres de la fenêtre de jeu, et les motos 1 et 2 qui via les rectangles posMoto 1 et 2 qui leur sont respectivement associés contiennent la position graphique (sur l'écran) de ces motos.

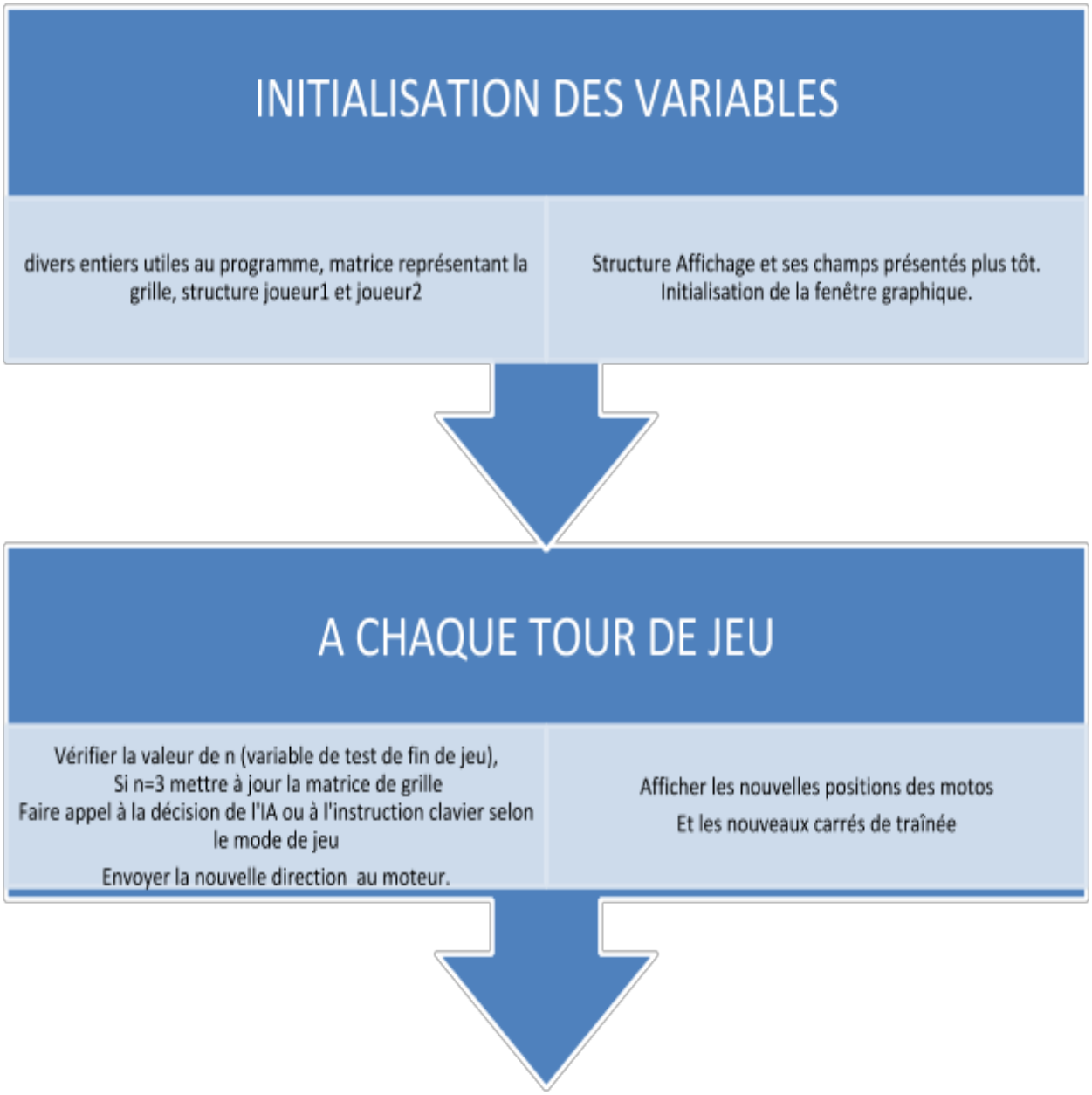
Le type « Affichage » est utilisé tout le long du programme « tron.c » et s'accompagne de différentes fonctions graphiques dont le code se trouve dans Graph.c :

- initSDL
- creationtrainee
- deplacement_moto
- closeSDL
- affich_victoire dans une moindre mesure

On a vu dans les algorithmes et les descriptions antérieures comment on les utilise. Pour plus de détail, se référer au code qui est commenté pour situer ces structures dans l'architecture du programme.

3) Récapitulatif du fonctionnement du tron

A l'aide des outils dont on vient de présenter la conception, voici comment fonctionne la partie main du tron.c de façon globale et synthétique. Sur la colonne de droite figure les opérations relatives au graphisme et sur celle de gauche celles relatives au déroulement intrinsèque du jeu.


INITIALISATION DES VARIABLES
divers entiers utiles au programme, matrice représentant la grille, structure joueur1 et joueur2
Structure Affichage et ses champs présentés plus tôt.
Initialisation de la fenêtre graphique.
A CHAQUE TOUR DE JEU
Vérifier la valeur de n (variable de test de fin de jeu),
Si n=3 mettre à jour la matrice de grille
Faire appel à la décision de l'IA ou à l'instruction clavier selon le mode de jeu
Envoyer la nouvelle direction au moteur.
Afficher les nouvelles positions des motos
Et les nouveaux carrés de traînée

Programmation et Développement

1) *Architecture des fichiers de jeu*

Le graph.c est accompagné de son .h pour la compilation. Cette fonction intervenant dans le tron a été rajouté pour une meilleure vision d'ensemble du programme et surtout une meilleur lisibilité du code du main dans tron.

2) *Stratégie de test*

Celle-ci a permis d'avancer pas à pas dans le projet. L'utilisation de printf pour afficher des valeurs d'éléments dans le terminal à différents stades d'exécution, ou de ddd pour déboguer de manière plus complexe a été récurrente. Nous la résumons dans le schéma suivant :

Test affichage graphique:

-moto

-moto + trainée

-moto + trainée + message de fin

-moto + trainée + message de fin + mémoire position dans une matrice

=> Tests et modifications en fonctions des résultats observés

3) Problèmes rencontrés

- ➢ Du point de vue graphique nous avons rencontré certains problèmes au niveau de la manipulation des bons types d'objets lorsqu'on les passait en argument et nous avons mis du temps à comprendre comment la mémoire était gérée.
- ➢ Du point de vue du jeu nous avons rencontré plusieurs problèmes liés au codage parfait des informations de jeu dans la grille
- ➢ Certains problèmes de mémoire (double free or corruption) ou des « segmentation faults » sont apparus.

Bien sûr de nombreuses erreurs d'inattention sont inévitables et il est important d'avancer étape par étape sans se disperser et avec patience. L'utilité de ddd dans les cas les plus complexes a été démontrée pour le débogage.

4) Exécution

Le tron se compile grâce au procédé makefile, dont le fichier associé est bien sûr joint dans notre dossier joueur.

Il suffit d'exécuter notre Tron. Le Joueur1 utilise les flêches du clavier tandis que le Joueur2 utilise ZQSD.

Conclusion

Ce Projet a été un grand investissement en temps et en énergie. Avant les problèmes techniques qu'il pose de façon manifeste, il a demandé avant tout une organisation assidue des objectifs dans le temps, et ce en binôme.

De plus ce qui a été mis en valeur à nos yeux est le fait qu'un problème colossal se résout en étant décomposé en une multitude structurée de problèmes plus simples. Cette pensée structurée a au fil du Projet permis de repérer de plus en plus facilement nos fautes.

Dans un cadre personnel, nous sommes d'accord pour souligner les apports personnels que la réalisation de ce Projet a entraîné pour nous. Patience, adaptabilité, concentration sur longue durée, communication en binôme, recherche efficace d'informations précises... La diversité des versants du langage C (Graphique, Structure, Architecture des programmes, Raisonnements par adresse...) nous a manifestement offert une formation plus complète et concrète du domaine.

Finalement les efforts investis pour obtenir un programme qui marche et lisible n'ont pas permis le développement d'un programme aussi avancé que nous le pensions.