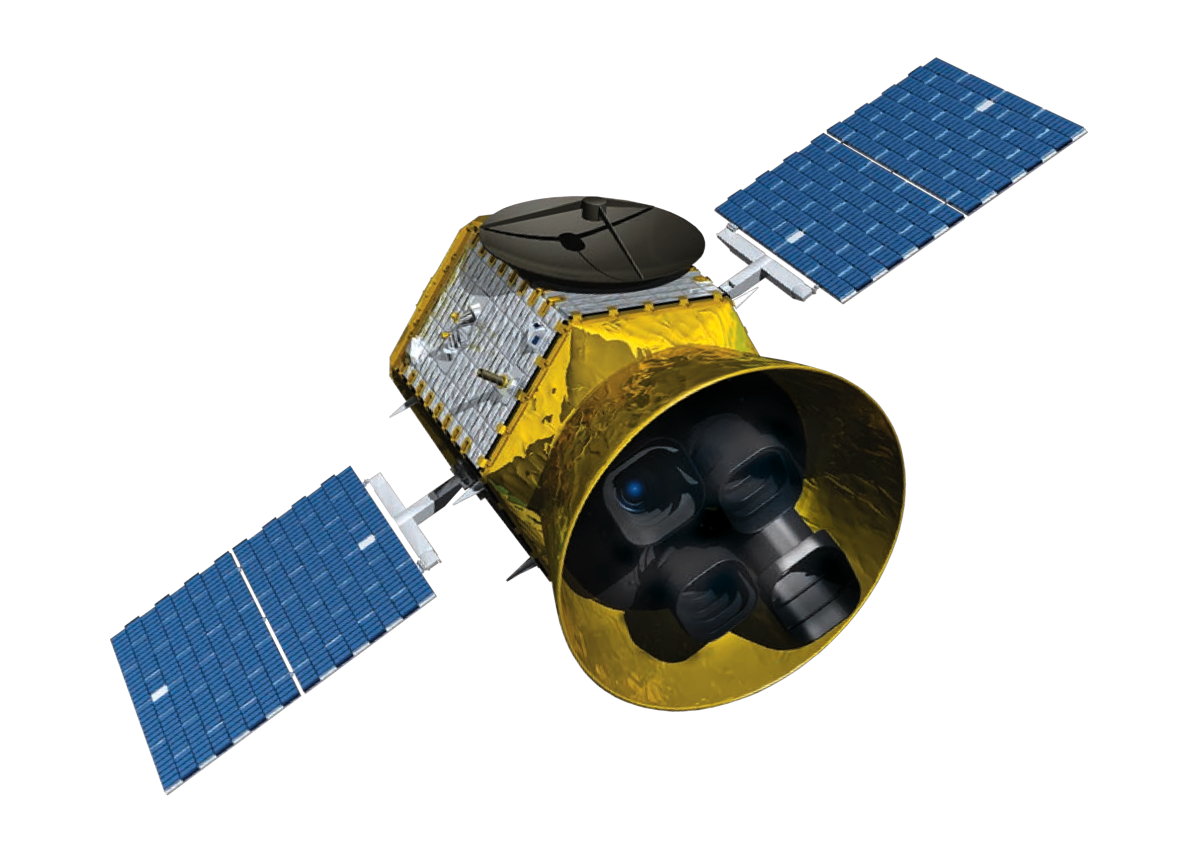
2015/2016

**Projet première année**

**Simulation de déplacement d’un satellite**

**Maxime ARLAUD**

**Vincent CAUDRON**



1. Présentation général

Le projet est de réaliser un système de simulation de déplacement de satellite dans notre système solaire.

Il a pour but de pouvoir déterminer la position exacte du satellite envoyé après un déplacement dans l’espace à un temps T.

Cahier des charges :

1. Créer une matrice pour réaliser notre système solaire. (déterminer la taille de la matrice)
2. Faire une structure planète.
3. Réalisme des informations utilisées (masse, superficie, vitesse).
4. Trouver les équations de graviter applicable à chaque planète.
5. Déterminer les équations de cercle.
6. Faire coller les équations de chaque planète sur celles des cercles
7. Trouver une méthode de redimensionnement de la carte.
8. Mettre en déplacement ce point.
9. Créer un satellite qui sur la matrice
10. Le déplacement doit déprendre de graviter des planètes environnantes et du soleil
11. Trouver une solution de faire des pauses sur le programme pour connaitre la position de l’objet
12. Trouver comment enregistrer un fichier puis le charger.

Liens utilisés pour la recherche :

1. <https://fr.wikipedia.org/wiki/Sph%C3%A8re_de_Hill>
2. <https://fr.wikipedia.org/wiki/Assistance_gravitationnelle>
3. <http://www.phy6.org/stargaze/Ftostars.htm>
4. <https://fr.wikipedia.org/wiki/Cercle>
5. <https://fr.wikipedia.org/wiki/Rotation_plane>
6. <https://openclassrooms.com/>
7. <http://thepoussin.free.fr/TPE2002/Caracteristiques.htm#soleil>
8. Organisation

Pour l’organisation nous avons donc posé des dates butoirs pour certaine phase de notre travail.

* La première dateline a été fixée une semaine après le début de notre projet, elle était sur la faisabilité du système, mais aussi pour que nous puissions nous coordonner sur l’architecture du programme et la façon de la réaliser selon chaque point de vue.
* La seconde était suite à la création des prototypes des fonctions. Nous nous sommes donc repartis les fonctions à écrire.
* La troisième était suite à des problèmes lors de la création de certaine des fonctions, en autre celle avec des lourds calculs mathématique (équation de gravité, équation de rotation).
* La quatrième est pour la phase de mise en conformité du programme et sur les modifications supplémentaires non demandée qui peuvent être apportées.
* La dernière a pour but de faire des tests de non régressions des systèmes suite à l’ajout des nouvelles fonctionnalités.

1. Les prototypes

Pour le projet il y a donc eu 14 prototypes de réalisé. Les fonctions ont été réalisées dans le but simplifier le programme, cela signifie qu’il y a des fonctions appelées à de multiple reprise, mais aussi pour alléger le « main ».

void sleep(int nbr\_miliseconds);

void system\_pause();

void enr\_fichier\_espace(T\_CEL \*T, int t\_cour);

void charger\_fichier\_espace(T\_CEL \*T, int t\_max, int \*t\_cour);

void affiche\_espace(char (\*MAP)[DIMAX],T\_CEL \*T, int dimH, int dimV);

void modif\_espace(T\_CEL \*T);

void modif\_map(char (\*MAP)[DIMAX], T\_CEL \*T, int t\_dim);

void initialise\_espace(T\_CEL \*T, int \*t\_dim);

void zoom\_map(int \*dimH, int \*dimV);

void ajout\_planete(T\_CEL \*T);

void gravitation(T\_CEL \*T);

void color\_espace(T\_CEL \*T, int k);

void rotation(T\_CEL \*T);

void satellite(T\_CEL \*T, T\_CEL \*T1);

1. Réalisation de la matrice

for(i=0;i<DIMAX;i++)

{

for(j=0;j<DIMAX;j++)

{

T[\*t\_dim] = C;

T[\*t\_dim].type = 'o';

T[\*t\_dim].posH = i;

T[\*t\_dim].posV = j;

\*t\_dim = \*t\_dim + 1;

}

}

Pour effectuer une représentation de l’espace nous utilisons un tableau en une dimension, qui nous permet d’avoir une matrice en deux dimensions.

Pour affichage correcte et complet il nous faut donc un affichage carré, de 150 cases par 150.

Comme nous faisons un affichage en un tableau à une dimension nous avons 22500 cases représentées et défini par #define TMAX 22500 .

Mais lors du premier affichage sur le moniteur nous avons fait face à une problématique, l’affichage est trop grand pour la taille de l’écran de l’ordinateur.

Cette problématique implique deux points :

Garder tous les calcule dur une matrice 150 / 150 tout en pouvant visualiser uniquement une matrice 30 / 150.

Cela implique que les paramètres de l’affichage doivent être différents de la partie calcule.

Donc nous avons rentré les paramètres 30/150 dans le « main » ce sont donc les paramètres par défaut.

1. Structure planète

(ou sont les variables qui manque)

typedef struct t\_cel {

char type;

int tps;

int posH;

int posV;

float direction;

double vitesse;

float poids;

float distance;

}T\_CEL;

Pour chaque planète nous avons donc utilisé cette structure. Cette structure nous permettra d’initialiser 3 (pour notre projet), mais laisse une possibilité d’en ajouter d’autre.

Ces 3 planètes sont initialisées dans la fonction void modif\_espace(T\_CEL \*T) alors leur donnée respective.

Mais pour le soleil qui utilise la même structure elle n’est pas initialisée dans la même fonction, mais dans void initialise\_espace(T\_CEL \*T, int \*t\_dim) . La différence entre les initialisations de planète et le soleil est que quoi qu’il arrive le soleil est initialisé à la création du tableau.

Voici un exemple d’initialisation de planète (la Terre) :

memset(&Terre,0,sizeof(T\_CEL)); (a quoi sert le memset)

Terre.type = 'T';

Terre.posH = DIMAX/2;

Terre.vitesse = 365.0;

Terre.poids = 5.972\*pow(10,29);

Terre.distance = 149600000;

Terre.posV = (DIMAX/2) + ceil(Terre.distance/2992000);

1. Fonction de gravité

void gravitation(T\_CEL \*T);

La fonction gravitation a été longue à réaliser. Elle a demandé un énorme travail de recherche et d’adaptation de formule pour pouvoir permettre de la programmer. Nous avons du parcourir de nombreux site pour cela (quelques exemples sont notés dans le document). Mais aussi de nombreux calculs vectoriels faits à la main.

Cette fonction a donc pour particularité de faire du calcule vectoriel en chaque point de la matrice. Ce que nous cherchons grâce à celle-ci est de savoir quelle est la direction et l’intensité de la force en chaque point de l’espace (espace de calcul 150/150), ce besoin est justifié pour l’affichage par couleur void color\_espace(T\_CEL \*T, int k) et pour le déplacement du satellite (exprimer dans les pages suivantes).

Lors de l’écriture de nos formules nous avons fait face à un problème de calcul qui est, que l’ordinateur fait uniquement du calcul en Rad et nous avons besoin de valeur en degré. Car ces formules font des calculs d’angle et nous en avons besoin, en degré. Donc nous avons fait ceci alors en variable de fonction float val = 180.0/PI et avec la valeur de PI en #define PI 3.14159265359.

Suite aux premiers essais un autre problème est survenu, celui-ci était que les calculs étaient conformes avec deux vecteurs par case, mais pas avec 3, donc nous avons dû reprendre tous les calculs, pour faire la résultante avec plusieurs vecteurs en entrées.

1. Redimensionnement de la carte

void zoom\_map(int \*dimH, int \*dimV)

La problématique imposée pour cette fonction était le redimensionnement de la carte.

En vue de notre projet nous nous sommes dit qu’il serait judicieux de faire un recadrage qui resterait toujours centré sur le soleil.

Cette fonction nous permet donc de changer les dimensions du tableau elle demande à l’utilisateur de la taille de l’affichage, pour la hauteur et pour la largueur.

Pour être sûr qu’il n’y ait pas d’erreurs possibles nous avons donc mis une sécurité qui nous empêche de sortir du bornage Max.

do

{

printf("\nSaisir la hauteur de la map (comprise en 0 et %d): ", DIMAX);

scanf(" %d", &i);

}while(i<0 || i> DIMAX);

1. Rotation

void rotation(T\_CEL \*T);

Une fois les planètes créées il se pose donc une nouvelle problématique, qui est de les mettre en rotation.

Une des contraintes est qu’elles doivent tonner autour du soleil, mais aussi elles doivent garder les champs de couleur autour d’elle. Mais la difficulté principale est de trouver les formules pour effectuer cette rotation.

Ces formules de rotation dépende encore des angles il faut donc encore faire des convertirons entre les Rad et les degrés pour rendre les calcule exploitables pour notre programme.

Les calculs se font donc sous cette forme :

T[j].posV = DIMAX/2 + ceil((T[j].distance\*cos(T[j].tps\*vitesse\*Rval))/2992000);

Pour rester centre sur le soleil nous avons donc fait cette opération DIMAX/2 c’est elle qui nous permet la rotation au tour d’un point.

1. Déplacement du satellite

void satellite(T\_CEL \*T, T\_CEL \*T1);

La fonction satellite est celle qui utilise l’automate cellulaire, mais dans notre programme il est utilisé différemment de son utilisation la plus courante. Nous utilisons la propagation pour effectuer un déplacement de point.

Nous allons utiliser ce déplacement pour faire se déplacer ce satellite. Son mode de déplacement va dépendre de la valeur des angles qui vont se trouver dans les cases et comme mentionné auparavant chaque case dépend des champs vectoriels des planètes et du soleil. Et en fonction de ces valeurs il va effectuer un déplacement.

Nous avons donc définie les valeurs pour les 8 cases au tour du satellite.

Les valeurs de ces cases ne sont pas des valeurs fixes, mais des plages de valeur. Nous devons donc repartir 360° sur ses 8 cases, donc une plage de 45° par case.

|  |  |  |
| --- | --- | --- |
| 45° | 45° | 45° |
| 45° | @ | 45° |
| 45° | 45° | 45° |

Pour que la simulation soit correcte et lisible après un déplacement, nous devons effacer l’ancien déplacement sinon il laisserait une trace derrière lui exemple :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 45° | 45° | 45° | 45° | 45° | 45° | 45° | 45° | 45° | 45° |
| 45° | @ | @ | @ | @ | @ | @ | @ | @ | @ |
| 45° | 45° | 45° | 45° | 45° | 45° | 45° | 45° | 45° | 45° |

1. Enregistrement fichier

void enr\_fichier\_espace(T\_CEL \*T, int t\_cour);

void charger\_fichier\_espace(T\_CEL \*T, int t\_max, int \*t\_cour);

Pour l’enregistrement nous avons fait cela en deux fonctions différentes, une pour l’enregistrement déficher et l’autre pour le changement de fichier qui doivent être au préalable enregistré.

Le fichier créé et enregistré est nommé espace.txt in enregistre toutes les informations de la carte que ce soit les positions des planètes, mais aussi celle du satellite, avec les valeurs de chaque point de la matrice.

Une fois que ce fichier est enregistré nous pouvons quitter le programme pour y retourner ultérieurement.

La partie chargement va chercher le fichier du nom mentionné ci-dessus et charge la dernière carte enregistrée (la matrice complète).