CodinGame

QUADRICA

Rapport exercices Codingame

Lead Développeur chez Quadrica : Nicolas Knopf

Candidat apprenti développeur 3D : Jérémie ALCAN

Date de remise : 21/06/2021

Table des matières

1. Températures

Cahier des charges : Le programme doit analyser des relevés de température pour trouver la plus proche de zéro.

Langage sélectionné : C#

Méthode : Utilisation de la valeur absolue pour trouver le nombre le plus proche de 0 comme sur la figure 1 ci-dessous.

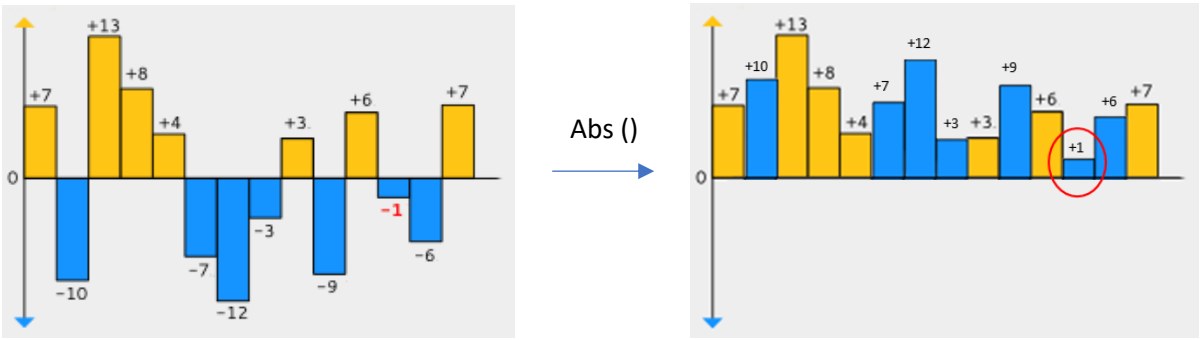


Figure 1 - Utilisation de la valeur absolue pour traiter les entiers positifs et négatifs

Je définis la 1ère valeur comme la valeur la plus proche et dans une boucle je la compare avec la valeur suivante.

Si la valeur suivante est inférieure à la valeur la plus proche, alors cette dernière devient la valeur la plus proche.

Je fais des tests conditionnels pour prendre en compte les cas particuliers :

- pas d'inputs en entrée
- deux nombres sont aussi proches de zéro

Difficulté rencontrée : Pas de difficulté particulière

Autres méthode envisagée : Effectuer un tri dans la liste par ordre croissant et sélectionner la plus petite valeur proche de zéro et positive s'il y a 2 nombres égaux.

Tests Coding Game validés 6/6 → Lien vers l'exercice température

2. ASCII Art

Cahier des charges : Le programme doit afficher une ligne de texte en art ASCII.

Langage sélectionné : Java (1ère utilisation)

Méthode : Utilisation d'un tableau figure 2 ci-contre pour stocker le template des lettres ASCII Art.

J'utilise les index du tableau et le codage en base 10 de la norme ASCII pour retrouver la correspondance entre les lettres du mot donné en entrée et son écriture en ASCII Art.

	0	1	2	3	4	5	6	7
0		#			#	#		
1	#		#		#		#	
2	#	#	#		#	#		
3	#		#		#		#	
4	#		#		#	#		

Figure 2 - Tableau des templates ASCII Art

- Template ASCII Art
- Mot à afficher

Figure 3 - Fonctionnement du programme

Difficultés rencontrées :

- Gérer le remplissage et l'affichage des tableaux → Annoter le code pour expliciter les compteurs de boucles for
- Eviter les index out of bound → Solution dessiner le tableau avec les index
- Lors du debbuging, j'ai été induit en erreur par l'affichage dans l'error stream à cause de la taille de la sortie. Sur la figure 4 ci-dessous, on voit uniquement le début et la fin de la sortie alors que le programme était bon → solution : aller dans le forum de l'exercice pour comprendre le bug.

Figure 4 - Erreur d'affichage à cause de la taille de l'error stream

Autres méthode envisagée : utilisation d'une structure de donnée associative (comme une table de hachage ou un dictionnaire)

Tests Coding Game validés 6/6 → Lien vers l'exercice ASCII Art

3. Mars Lander Episode 1

Cahier des charges : L'objectif est de faire atterrir un vaisseau spatial sur une plateforme en toute sécurité dans un espace 2D (voir figure 5 ci-dessous). Dans l'épisode 1, le contexte est simplifié car la zone d'atterrissage est juste en dessous du vaisseau.

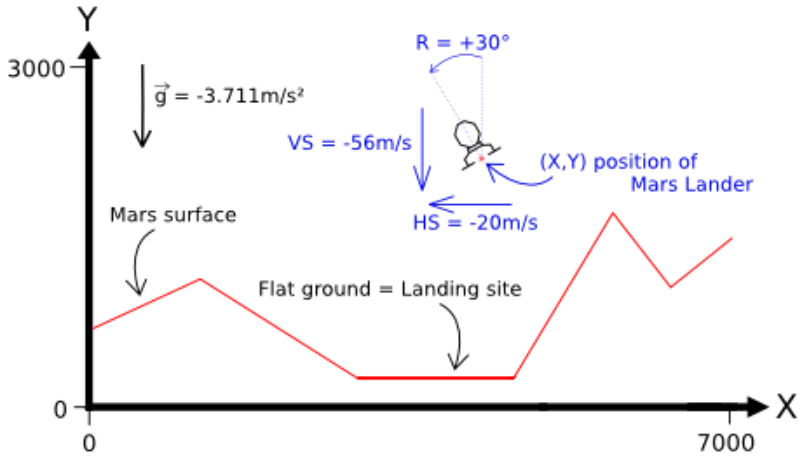


Figure 5 - Espace 2D de la simulation

Langage sélectionné : C++

Méthode : Utilisation de tests conditionnels pour gérer la puissance du moteur en fonction de l'altitude du vaisseau. La séquence de descente est représentée sur la figure 6 ci-dessous.

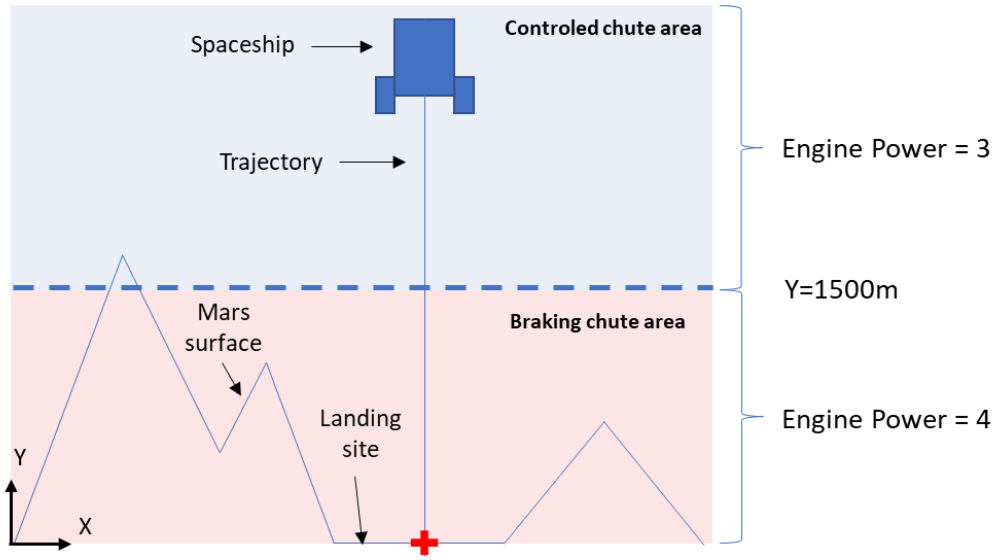


Figure 6 - Séquence de descente

Les valeurs d'altitude et de puissance moteur ont été déterminées expérimentalement sachant qu'une poussée de 4 quasi verticale permet de compenser la gravité de Mars.

Difficultés rencontrées : Pas de difficulté particulière

Test Coding Game validé 1/1 → Lien vers l'exercice Mars Lander Episode 1

4. Mars Lander Episode 2

Cahier des charges : L'objectif est de faire atterrir un vaisseau spatial sur une plateforme en toute sécurité.

Langage sélectionné : C++

Méthode : Après avoir trouvé l'emplacement de la plateforme d'atterrissage, « le séquence manager » détermine le comportement du vaisseau en fonction de sa position et de sa vitesse dans l'espace 2D.

Le vaisseau est piloté grâce à la puissance des moteurs et son orientation dans l'espace.

Les différentes phases de vol sont définies sur la figure 7 ci-contre.

Pour se déplacer dans la phase « Moving » sans perdre d'altitude, le vaisseau prend un angle α calculé pour que l'accélération verticale du vaisseau soit égale à la gravité martienne (voir figure 8 ci-après) :

$$\alpha = \operatorname{acos}\left(\frac{3{,}771}{4}\right) = 22°$$

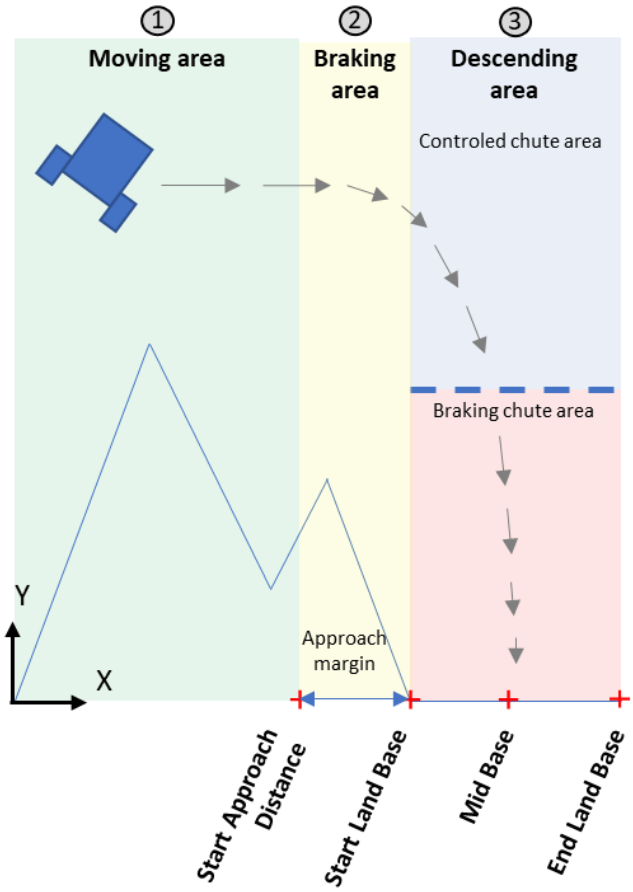


Figure 7 - Découpage des zones de l'espace 2D pour la séquence d'atterrissage

Difficultés rencontrées :

- L'utilisation d'un tableau statique a généré des problèmes de mémoire et un mauvais stockage des points de la surface martienne → solution : utiliser des vectors.
- Au départ difficile de gérer les différentes étapes du vol → solution : créer un « séquence manager »
- S'adapter aux différents tests avec des vitesses initiales positives et des sens différents → solution : annoter le programme pour clarifier les tests conditionnels
- Trouver le bon angle de freinage pour ne pas prendre trop de vitesse verticale lors de la phase « Braking » → solution : tester plusieurs valeurs.

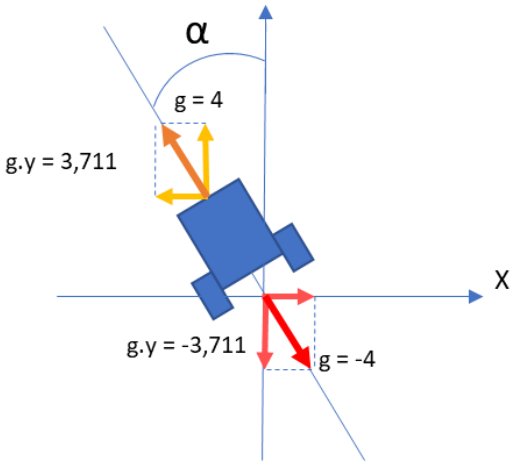


Figure 8 - Détermination de l'angle pour le déplacement horizontal

Autre méthode envisagée : détermination d'une trajectoire courbe optimisée en consommation de carburant avec un intervalle de tolérance à ne pas dépasser lors du déplacement du vaisseau (voir schéma page 8).

Tests Coding Game validés 5/5 → Lien vers l'exercice Mars Lander Episode 2

5. Flood Fill Example

Cahier des charges : A partir d'une grille composée de points visitables et de points non-visitables, et d'autres entités pour les tours de défense. Donnez une map présentant la couverture de chaque tour après que les troupes se soient répandues.

	j-1	j	j+1
i-1		North	
i	West	**Troop**	East
i+1		South	

Figure 9 - Tableau utilisé pour implémenter la fonction de propagation orthogonale

Langage sélectionné : C++

Méthode : Les troupes peuvent se propager de manière orthogonale dans les directions (N,S,W,E) comme on peut le voir sur la figure 9 ci-contre. Le programme suit les étapes suivantes :

1. Collection de la liste des troupes prêtes à se propager
2. Si la troupe n'est pas le long d'une bordure, elle avance d'une case vers les points visitables
3. Pendant un tour, si 2 troupes veulent aller sur la même cellule, il faut noter un + dans la case
4. Après plusieurs itérations affichage de la map s'il n'y a plus de propagation des troupes

Difficultés rencontrées :

- Index out of bound pour les propagations sur les bordures → solution : tester la condition de bordure avant de chercher à faire avancer la troupe
- L'utilisation d'un tableau statique a généré des problèmes de mémoire et un mauvais stockage des troupes sur le tableau de la map → solution : utiliser des vectors.
- Gérer le remplissage et l'affichage des tableaux → solution : Annoter le code pour expliciter les compteurs de boucles for

South

	j-2	j-1	j	j+1	j+2
i-2					
i-1			North		
i		West	**Troop**	East	
i+1		**Potentiel Troop 3**	South	**Potentiel Troop 1**	
i+2			**Potentiel Troop 2**		

Potentiel Troop 1 : {i+1,j+1}
Potentiel Troop 2 : {i+2,j}
Potentiel Troop 3 : {i+1, j-1}

Figure 10 - Tableau utilisé pour implémenter la fonctionnalité des '+'

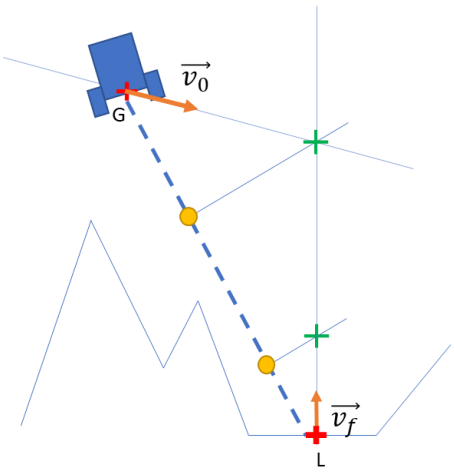
- Je n'ai pas réussi à réaliser la fonctionnalité : Si 2 troupes peuvent atteindre un même point au même moment, mettre un '+' même si les 2 troupes partage le même ID.

  Je me suis aidé du tableau en figure 10 pour essayer de catégoriser toutes les possibilités mais le résultat n'est pas encore celui attendu.

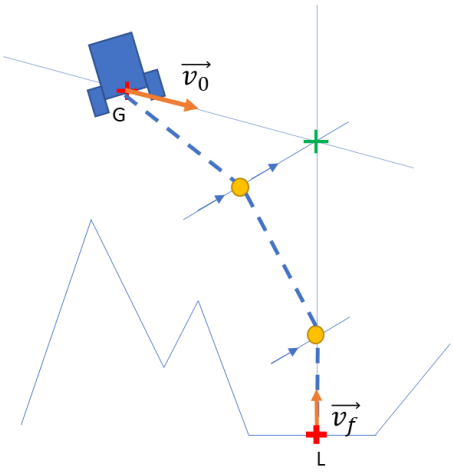
Tests Coding Game non validés 0/5 → Lien vers l'exercice Flood Fill Example

5. Informations complémentaires

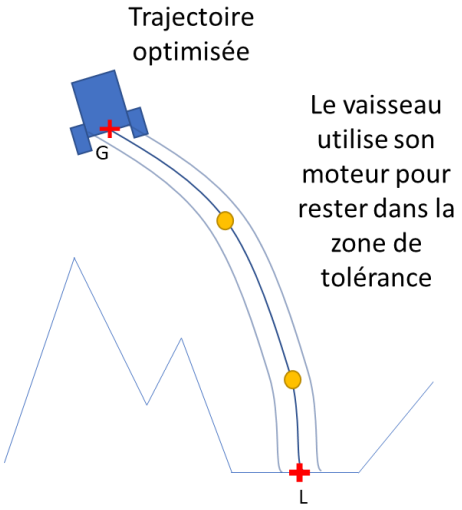
Détermination d'une trajectoire optimisée pour Mars Lander :



Détermination graphique des points de « cassure » de la droite GL



Translation des points jaunes en fonction des vitesses initiales et finales



Approximation par une courbe continue et détermination d'un intervalle de tolérance

J'ai manqué de temps et d'aisance en C++ pour coder cette solution. J'ai préféré me concentrer sur la solution page 6 pour avoir un programme qui valide les tests Codingame.

Néanmoins, avec certaines classes d'objets géométriques et des méthodes associés. Il est envisageable d'obtenir une trajectoire optimisée à condition que le calcul soit rapide.

Idem pour le contrôle du vaisseau, il faudrait développer certaines méthodes qui réagissent au dépassement des intervalles de tolérance sur la trajectoire.