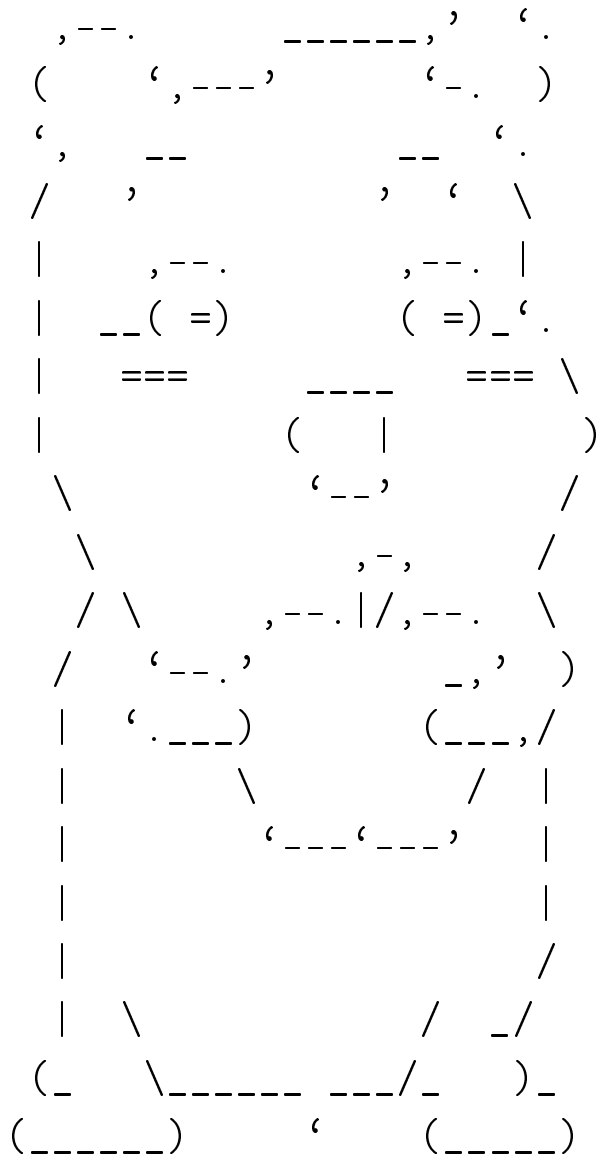
Les Hamsters

Ou la bataille des pommes

Sujet de la finale du Concours National d'Informatique
Vendredi 09 mai 2008 - Dimanche 11 mai 2008

Table des matières



1 Introduction

1.1 Prélude

Combien de dimensions possède l'*Univers* ? Voilà une question à même de déchirer biens des scientifiques ! D'aucuns, naïfs, répondraient trois. En y réfléchissant à deux fois, une quatrième n'est pas de trop. Mais au-delà ? La théorie en suppose de plus en plus, d'abord 11, puis 12, peut-être même 20 ou plus... Mais que contiennent ces dimensions *parallèles*[1] ?

Certains supputent l'existence de *Hammerspace*, des univers divergents peuplés de marteaux qui peuvent apparaître dans vos mains aux moments les plus opportuns[2]. Hugh Everett[3] théorisa l'existence des mondes multiples afin de résoudre le paradoxe[4] du postulat de la réduction du paquet d'ondes de la mécanique quantique.

1.2 Introduction

Les physiciens, aussi brillants soient-ils, ignorent pourtant l'essentiel : non seulement des dimensions divergentes existent, mais en plus elles sont peuplées d'êtres intelligents, voire *supérieurement* intelligents. Heureusement[5], il est encore impossible de faire des petites virées intra-dimensionnelles. Mais dés lors, comment répondre à cette question que tout le monde s'est posé au moins une fois dans sa vie : mais quand est-ce qu'on mange ? Ah, je crois que je me suis trompé de film, je touche l'échec du doigt[6] !

Bref, les dimensions existent, elles sont peuplées, mais vous sont inaccessibles. C'est devant ce terrible constat d'échec que vous avez eu l'*idée* :

Les souris !

[1] Qui ne sont en fait pas parallèles mais divergentes.

[2] La définition de « moment opportun » est laissé à votre discrétion

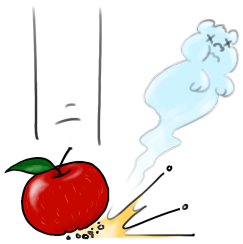
[3] wikipedia ://Hugh_Everett

[4] Le paradoxe étant qu'il semble invraisemblable qu'une fonction d'onde déterministe donne lieu à des observations non-déterministes

[5] Ou pas

[6] Film à voir : La Stratégie de l'Échec

Vous le savez, les souris ne sont pas ce qu'elles semblent être. Bien crédule celui qui imagine que ce ne sont que de petits mammifères friands de fromage, condamnés à vivre au rythme de leur propre respiration stressée dans de lugubres laboratoires pharmaceutiques, ou se contentant de bricoler dans vos combles la nuit. Non, comme vous le savez, les souris ne sont rien de moins qu'une projection dans nos dimensions d'êtres supérieurement intelligents, à l'origine même de la création de la Terre.



Malheureusement en matière de découvertes scientifiques la supériorité intellectuelle ne suffit pas.

L'idée géniale que vous avez eu[7] : parvenir à dresser et innocemment contrôler des souris dans nos dimensions pourrait potentiellement vous donner accès à nombre de privilèges dans cet ordinateur qu'est la Terre[8]...

Seulement voilà, alors que votre *business model* est prêt et que vous vous voyez déjà maître du monde dans l'ombre, un léger soucis vient perturber vos plans : les souris, votre maman n'a jamais voulu en entendre parler. Hors de question d'avoir ces petites bêtes juste bonnes à faire hurler la petite soeur (entre autres) et ronger les pieds de tables. Mais vous n'alliez pas laisser votre plan de conquête multidimensionnelle tomber à l'eau pour si peu ! Après une intense réflexion, vous arrivez à la conclusion qu'un autre animal pourrait fort bien faire l'affaire : une espèce proche, mais plus consensuelle. Fort de votre arbre de l'évolution des espèces, vous envisagez tour à tour le mulot (cela reviendrait au même, jamais vous n'arriverez à faire admettre que ce n'est pas une souris), le rat (accordé, personne ne le confond avec une souris, mais pour autant ce n'est pas vraiment un moindre mal...), le buffle (non là

[7] N'oubliez pas qu'en 2006 nous avions une De Lorean
[8] root@gaia # echo 'Hello World !'

vous avez juste lu la mauvaise branche), l'écureuil (un peu trop éloigné), le furet (trop éloigné également, même si ses capacités extra-sensorielles sont intéressantes[9]), et finalement, le *hamster*. C'est bien un hamster, tout le monde aime les hamsters.

Un problème ne survenant jamais seul, à peine avez-vous le temps de de jeter les bases de votre plan que vous apprenez que vous n'êtes plus le seul *dans la place*. Des dizaines d'autres jeunes talentueux informaticiens[10] seraient déjà en plein perfectionnement de leurs techniques de dressage.

Les souris, quant à elles, bien conscientes que l'anarchie déclenchée revient finalement à faire de l'introspection[11], contre-productive pour la résolution de leur problème, à savoir trouver *la* question, ont donc décidé d'organiser un grand tournoi de dressage. Celui-ci réglera une bonne fois pour toutes la question de savoir quel humain aura l'extrême *privilège* d'être invité à leur table[12]…

[9] « Cyrano, va embêter maman ! »

[10] À croire que seuls les informaticiens lisent les bons guides touristiques

[11] Rappelons que s'ils sont pratiques, les langages introspectifs sont souvent moins performants en terme de rapidité.

[12] Ça ne vous dérange pas de perdre votre cerveau, n'est-ce pas ?

1.3 Épilogue

Arthur :
The, um, talking... talking mice. Cool.
Mice :
Sit, Earthman.
Arthur :
Oh, thank you. Thank you for that.
Mice :
Please, drink.
Arthur :
Um, uh, excuse me.
Mice :
Now, to business.
We've spent a lot of time on your planet looking for this ultimate question.
Only to have it blow up in our faces. Literally.
Which is why you're here.
We've been offered a lucrative contract to do several 5-D TV chat shows.
But here's the point. We must have product.
We need the ultimate question, or one that sounds ultimate.
Arthur :
Of course.
Mice :
We've rebuilt the planet.
Now all we need is the missing piece of the puzzle.
Which happens to be your brain.
More tea ?
Arthur :
Sorry, did you just say you need my brain ?
Mice :
Yes, to complete the program.
Arthur :
You can't have my brain. I'm using it.
Mice :
Hardly.
Arthur :
Hardly ? Cheeky... Cheeky mouse.
What was in that food ? What was in my tea ?
Mice :
Don't worry. You won't feel a thing.
Just wait a sodding minute.
Arthur :
You want a question that goes with the answer 42 ?
What about, what's 6x7 ?
Or, um, uh, how many Vogons does it take to change a light bulb ?
Here's one : How many roads must a man walk down ?
Mice :
Hey, that's not bad.

2 Le match

2.1 Votre objectif

Voilà donc tout plein de prétendants à la domination du monde fin prêts à déballer leur mètre-ruban pour mesurer le diamètre de leurs biceps, ou plus pragmatiquement à comparer le résultat de leur dressage. Malgré les divergences qui opposent tout ce beau monde, il faut bien se mettre d'accord sur au moins une chose : un opérateur de comparaison permettant d'établir un classement, autrement dit, des règles du jeu.

C'est une fois encore les souris qui les imposent[13] : il va s'agir de faire de la recherche de pommes, le hamster ayant une capacité de transport utilitaire proprement impressionnante, grâce à deux poches communément appelées « joues », mais qui dans son cas mériterait plutôt l'appellation de « stock », « grenier », « charge utile », ou encore « supermarché ».

Votre objectif en tant que dresseur de hamster est de leur donner les meilleurs ordres possibles afin qu'ils aient récupérés un maximum de pommes au terme de la partie. Vous serez toujours exactement deux joueurs ayant chacun trois hamsters.

Les points sont comptabilisés de manière très simple : si au terme de la partie vous avez plus de pommes que votre adversaire vous gagnez trois points, si vous êtes à égalité vous gagnez chacun un point et si vous perdez vous ne gagnez pas de points. Lors du tournoi finale les souris organiseront un grand tournoi de plusieurs dizaines de milliers de matchs qui détermineront donc un classement par points.

2.2 Le plateau de jeu

Le terrain sur lequel vos hamsters évoluent est un grand plateau recouvert de copeaux de bois, idéal pour eux. Il se découpe en trois zones : deux zones de départ et une zone neutre. Il y aura également des murs infranchissables et des tranchées. En règle générale sa taille variera entre 10x10 et 20x20 cases, sachant que rien n'exclu des plateaux plus grands ou plus petits.

2.2.1 Les zones de départ

Les zones de départ sont les deux rangées en haut et en bas du plateau, chaque joueur ayant évidemment la sienne. C'est de là que partiront vos hamsters et c'est là qu'ils devront

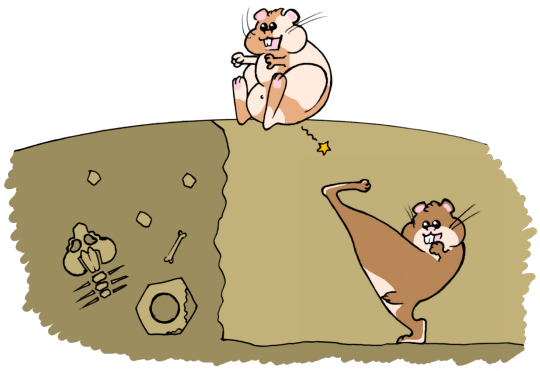
[13] Vous n'êtes toujours pas convaincu de leur supériorité ?

déposer les pommes pour qu'elles soient comptabilisées. Évidemment rien ne vous empêche d'aller vous fournir en pommes dans la zone adverse, et réciproquement...

Les points seront comptabilisés à la fin de la partie de manière très simple : toute pomme déposée dans votre zone vous rapporte un point.

2.2.2 Les tranchées

Certaines cases du plateau peuvent être des trous. Si plusieurs cases adjacentes sont des trous alors elles forment une tranchée dans laquelle il est possible de se déplacer. Les trous sont suffisamment profond pour que vos hamsters ne puissent en sortir seuls. Pour ce faire, il leur faudra soit se faire pousser par un autre hamster, soit se faire tirer à l'aide du grappin d'un autre hamster resté en surface. Référez-vous à la section 2.3 page 8 sur les hamsters pour plus de détails.



2.3 Les hamsters

Même si les hamsters sont la projection dans nos dimensions d'être supérieurement intelligents, leurs limites physico-biologico-morphologico-corporelles les empêchent d'être en pleine possession de leurs moyens. Il va donc vous falloir les guider tout au long de l'épreuve.

2.3.1 La communication

Vous êtes libres d'utiliser le moyen qui vous semble le meilleur pour communiquer avec vos hamsters : signaux de fumée, cris, gestes, wifi... Cependant, les règles stipulent clairement qu'il vous est interdit de toucher ou d'aider vos hamsters physiquement.

Quel que soit votre moyen de communication, vous allez devoir gérer la lenteur de vos petits protégés[14]. En effet, à chaque fois que vous voulez communiquer il leur faut s'arrêter, se retourner vers vous et essayer de décrypter ce que vous essayez de leur faire passer.

Pour éviter de perdre l'essentiel de votre temps à essayer de dire à vos hamsters quoi faire, vous avez la possibilité de leur donner jusqu'à trois ordres. Ceci vous permet d'effectuer plus d'actions mais vous oblige à extrapoler le jeu sur trois tours et donc à deviner les intentions de votre adversaire.

Pour éviter d'avantager l'un ou l'autre des participants, les souris ont pris soin d'établir un ordre de jeu équitable. Le joueur qui commence est choisi au hasard au premier tour. Au second tour, c'est au tour de l'autre joueur de commencer, et ainsi de suite en alternant. Ensuite chaque hamster, choisi alternativement dans chaque équipe, effectuera une action. L'ordre dans lequel ils sont choisi dans chaque équipe est défini par l'ordre chronologique dans lequel le joueur leur a donné leur premier ordre.

2.3.2 Les déplacements

Vos hamsters n'ont aucune idée globale du plateau dans lequel ils évoluent. Ils n'ont donc pas de notion de case, malgré le fait que le plateau de jeu soit découpé à la manière d'un damier. Ça n'aurait donc pas beaucoup de sens de leur demander d'aller sur la case (42, 51)[15]. Vos ordres de direction seront plutôt d'avancer dans une direction (voir les constantes de direction 2.4.3 page 13).

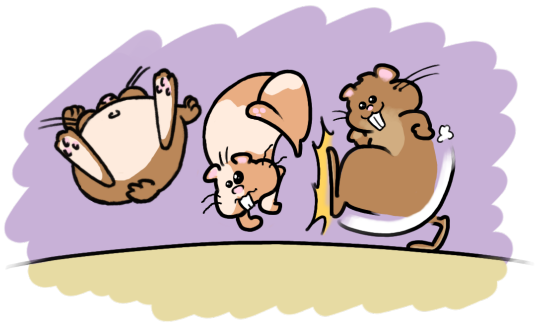
Le hasard et le découpage du plateau de jeu faisant bien les choses, lorsque vous donnez un ordre de déplacement dans une direction à l'un de vos hamsters il se trouve qu'il avance d'exactement une case.

Vous pouvez également demander à votre hamster de prendre son élan. Si vous donnez l'ordre n-fois à votre hamster de prendre de l'élan il avancera de n+1 cases lorsque vous lui demanderez d'avancer. Malheureusement, votre hamster perd cet élan lorsqu'il se retourne vers vous pour demander ce qu'il doit faire. En pratique donc, n sera soit 1, soit 2.

Si lors de son déplacement votre hamster tente de se déplacer sur une case où un autre hamster est déjà présent alors votre hamster va pousser ce dernier pour prendre sa place. Si ce n'est pas possible car l'autre hamster est contre un mur, rien ne se passe. Votre hamster n'aura aucun problème à pousser plusieurs autres hamsters lors de son déplacement.

[14] D'une manière générale, dés qu'un réseau informatique contient des éléments animaliers tout devient très lent, comme en témoigne ce test de TCP/IP-over-pigeon-voyageur : http ://www.blug.linux.no/rfc1149/

[15] Nombres évidemment choisis totalement aléatoirement, ou plutôt, chaotiquement.



2.3.3 Le grappin

Pendant leur entraînement, vos hamsters ont appris à utiliser deux techniques fort intéressantes, la première étant le lancer de grappin.

Pour le grappin, il s'agit d'un petit crochet attaché à une petite ficelle qui leur permet d'attraper et de tirer un autre hamster. En pratique la ficelle fait exactement la longueur d'une case.

La technique consiste donc à demander à l'un de vos hamsters de lancer son grappin dans une direction donnée et ensuite de se déplacer dans une autre direction afin de tirer le hamster « grappiné ». Ce dernier se trouve alors tiré sur la case où était votre hamster.

Comme pour l'élan, votre hamster n'est pas capable de garder son grappin accroché pendant qu'il se retourne vers vous, vous ne pourrez donc le garder accroché plus d'un tour. Libre à vous de le relancer ensuite.

Le grappin n'étant pas d'une accroche très solide, il est très facile pour un hamster de s'en défaire, il lui suffit simplement de se déplacer. Notez également que pour tirer un hamster le déplacement doit être volontaire. C'est à dire que si vous avez « grappiné » une innocente victime et que vous vous faites pousser, elle ne se fera pas tirer et vous perdrez votre accroche. Par contre si cette innocente victime « grappinée » vous pousse, le grappin restera accroché.

2.3.4 Les trognons de pommes

De même que l'on exacerbe le talent d'un cochon truffier en lui faisant goûter de temps à autres une truffe, des pommes ont été généreusement distribuées à votre petite boule de poils super-dimensionnelle. La consommations de celles-ci ayant abouti au stockage d'autant

de trognons dans sa réserve[16], cela fait autant de projectiles qu'il sera par la suite possible d'éjecter grâce à une astucieuse détente adiabatique [17].

Vous vous imaginez bien que vos petits hamsters ne peuvent cependant transporter qu'un nombre fini[18] de trognons. En fait, ils ne pourront en emporter que deux, un par joue[19]

Cependant, avec toutes ces pommes à disposition, il paraît évident que vos hamsters pourront récupérer autant de trognons qu'ils veulent. Seulement, c'est oublier votre but premier : la collecte des dites pommes. Il est donc hors de question de les sacrifier ! De toute façon, la complexité de l'algorithme d'ingestion d'une pomme par un hamster est pire qu'exponentielle et il est probable que l'Univers termine avant qu'il n'ait récupéré le trognon. Il sera donc plus prudent de considérer que vous avez deux trognons par hamster et par partie.

Vos hamsters pourront propulser un trognon à une vitesse vertigineuse dans la direction que vous lui indiquerez. Une fois lancé, il traverse quasi instantanément le plateau en ligne droite. Si vous êtes mauvais, le précieux projectile ira lamentablement s'écraser contre un mur. Dans le cas contraire, s'il touche un hamster, il le poussera d'une case dans la direction du lancé et lui fera perdre la pomme qu'il portait, si toutefois il en avait une. Cette dernière se retrouve alors sur la case où était le hamster avant d'être poussé. S'il se trouve que le hamster était contre un mur il ne sera évidemment pas poussé mais perdra quand même sa pomme qui tombera sur la case où il se trouve.

2.3.5 Les pommes

Les pommes sont l'enjeu critique de la partie. C'est leur nombre dans votre zone qui décideront si vous gagnez ou perdez la partie. En avoir plus que l'adversaire est donc vital[20] pour vous.

Pour prendre une pomme vous devez vous déplacer jusque sur la même case qu'elle et ordonner à votre hamster de la prendre. Une fois prise, il vous faut la déposer dans votre zone. Au vu de la taille de ces dernières, il ne pourra y avoir qu'une pomme par case.

[16] Votre hamster saura retrouver dans quelle étagère de quelle rangées de ses joues exactement, aucune inquiétude à avoir

[17] C'est tout de même plus classe que de dire « en crachant », non ? Pour ceux qui n'ont pas suivi, une recherche de «Sadi Carnot« sur Wikipédia sera un bonheur.

[18] Les sac-à-dos infinis n'existant que pour vous poser d'épineux sujets de demi-finales.

[19] Pour rappel, le terme « hamster » vient de l'allemand « hamstern » qui signifie « faire des réserves ».

[20] « Mangez des pommes ! » ©

2.3.6 Vision

Étant donné que vous surplombez le terrain de jeu, vous pouvez tout voir, quelque soit la position de vos hamsters.

2.4 Constantes

2.4.1 Les codes d'erreur

Constante : BAD_ARGUMENT
Valeur : -1
Description : Un des arguments passés à la fonction est incorrect.

Constante : TOO_MANY_ORDERS
Valeur : -2
Description : Vous avez envoyé trop d'ordres à votre hamster.

Constante : SUCCESS
Valeur : 0
Description : La commande s'est correctement effectuée.

Constante : INFINI
Valeur : 10000
Description : La distance est infinie

2.4.2 Les constantes de terrain

Constante : NORMAL
Valeur : 0
Description : Case de terrain qui n'est ni une tranchée, ni un obstacle

Constante : TRANCHEE
Valeur : 1
Description : Tranchée du terrain

Constante : OBSTACLE
Valeur : 2
Description : Obstacle du terrain

2.4.3 Les constantes de directions

Constante : HAUT
Valeur : 0
Description : Vers le haut

Constante : GAUCHE
Valeur : 1
Description : Vers la gauche

Constante : BAS
Valeur : 2
Description : Vers le bas

Constante : DROITE
Valeur : 3
Description : Vers la droite

Constante : ICI
Valeur : 4
Description : Cette constante n'est utilisée que pour la fonction lacher_pomme

2.5 Fonctions d'information

Toutes les fonctions peuvent renvoyer les constantes **BAD_ARGUMENT** quand au moins un des arguments est incorrect (même si cela n'est pas précisé pour chaque fonction). Par exemple, cela se produit si vous appelez la fonction `type_case` avec `x=13` et `y=5142`.

- **taille_carte_x**

```
int taille_carte_x()
```

Description : Renvoie la largeur de la carte

- **taille_carte_y**

```
int taille_carte_y()
```

Description : Renvoie la hauteur de la carte

- **numero_tour**

```
int numero_tour()
```

Description : Renvoie le numero du tour

- **nombre_tours**

```
int nombre_tours()
```

Description : Renvoie le nombre de tours d'une partie

• **commence**

```
int commence()
```

Description : Fonction pour savoir si on joue en premier ce tour-ci

• **pos_x**

```
int pos_x(int id)
```

Description : Renvoie la position x du hamster id
Parametres : *id* : id du hamster

• **pos_y**

```
int pos_y(int id)
```

Description : Renvoie la position y du hamster id
Parametres : *id* : id du hamster

• **porte_pomme**

```
int porte_pomme(int id)
```

Description : Indique si le hamster id porte une pomme
Parametres : *id* : id du pomme

• **distance**

```
int distance(int x1, int y1, int x2, int y2, int sort_tranchee)
```

Description : Renvoie la distance minimale pour aller de (x1,y1) à (x2,y2). ce est calculée en tenant compte des déplacements possibles d'un seul hamster. Cette fonction a un temps de calcul nul. Lorsque sort_tranchee vaut false, la distance est calculée en tenant compte des déplacements possibles d'un seul hamster. Lorsque sort_tranchee vaut true, la distance est calculée comme s'il n'y avait pas de tranchees.

Parametres :
x1 : la colonne du premier point
y1 : la ligne du premier point
x2 : la colonne du second point
y2 : la ligne du second point
sort_tranchee : indique le mode de calcul de la distance

• **type_case**

```
int type_case(int x, int y)
```

Description : Renvoie le type de la case (x,y) (NORMAL, TRANCHEE ou OBSTACLE)
Parametres : *x* : la colonne
y : la ligne

• **pomme**

```
int pomme(int x, int y)
```

Description : Indique s'il y a une pomme non portée sur la case (x,y)
Parametres : *x* : la colonne
y : la ligne

• **trognons_restants**

```
int trognons_restants(int id)
```

Description : Renvoie le nombre de trognons de pomme restants du hamster id
Parametres : *id* : identifiant du hamster

2.6 Fonctions d'action

Toutes les fonctions peuvent renvoyer les constantes **BAD_ARGUMENT** quand au moins un des arguments est incorrect (même si cela n'est pas précisé pour chaque fonction), ou **TOO_MUCH_ORDERS** quand vous avez envoyé trop d'ordres à l'un de vos hamsters.

• **deplacer**

```
int deplacer(int id, int direction)
```

Description : Demande un déplacement du hamster
Parametres : *id* : l'identifiant de votre hamster
direction : l'une des quatre directions

• **turbo**

```
int turbo(int id)
```

Description : Incrémente le turbo d'un hamster
Parametres : *id* : l'identifiant de votre hamster

• **lacher_pomme**

```
int lacher_pomme(int id, int direction)
```

Description : Fait lacher la pomme à votre hamster, dans l'une des cinq directions
Parametres : *id* : l'identifiant de votre hamster
direction : l'une des cinq directions vers laquelle vous voulez lacher la pomme

• **ramasser_pomme**

```
int ramasser_pomme(int id)
```

Description : Demande à votre hamster de ramasser la pomme sur sa case courante
Parametres : *id* : l'identifiant de votre hamster

• **grappin**

```
int grappin(int id, int direction)
```

Description : Demande à votre hamster de lancer le grappin
Parametres : *id* : l'identifiant de votre hamster
direction : l'une des quatre directions vers laquelle vous voulez lancer le grappin

• **trognon**

```
int trognon(int id, int direction)
```

Description : Demande à votre hamster id de lancer un trognon dans une direction
Parametres : *id* : l'identifiant de votre hamster
direction : l'une des quatre direction vers laquelle vous lancez le trognon

• **attendre**

```
int attendre(int id)
```

Description : Ordonne à votre hamster de ne rien faire
Parametres : *id* : l'identifiant de votre hamster