Examen Réparti 1 - CPS 2016 - `Lemmings`

Durée: Deux heures.
Documents: Documents personnels sur papier autorisés.

Le but de l'examen réparti est de spécifier une version du jeu de réflexion *Lemmings*, développé par *DMA Design* et édité par *Psygnosis* en 1991.

Consignes

L'objectif de l'examen réparti est d'écrire une spécification cohérente, non-redondante, et complète. Ainsi:

- Une bonne spécification couvrant une petite partie du cahier des charges sera plus valorisée qu'une spécification médiocre en couvrant une grande partie. Il n'est absolument pas nécessaire de traiter toutes les questions pour obtenir la note maximale.
- La description donnée dans l'énoncé est ambigüe, plusieurs spécifications correctes sont possibles, il faut faire des choix (et expliciter ces choix au maximum dans la copie).
- La spécification doit être semi-formelle. Sa syntaxe peut s'écarter de celle vue en cours et TD si elle est suffisamment claire (quantificateurs $\forall\exists$, ensemble d'éléments, appel d'opérateurs, ...)
- Si les questions sont relativement indépendantes, il est conseillé de lire l'intégralité du sujet avant de commencer à rédiger une réponse et de traiter d'abord les questions 1, 2 et 3, avant de traiter (au choix) au moins une des questions suivantes.

Sujet: Description informelle

Lemmings est un jeu de réflexion en temps réel se déroulant dans des niveaux en deux dimensions vus de côté. Ces niveaux sont composés de cases pleines (terre ou métal) ou vides.

A intervalle régulier, des personnages (appelés **lemmings**) apparaissent d'un point du niveau (l'entrée). Le joueur n'a pas de contrôle direct sur les lemmings, qui marchent dans la même direction jusqu'à rencontrer un obstacle (ils se mettent alors à marcher dans l'autre sens) ou une falaise/un trou (ils tombent et, si la chute n'est pas mortelle, continuent à marcher dans la même direction).

Les niveaux contiennent tous une case de sortie. Si un lemming marche sur la sortie, il disparait (et est considéré comme "sauvé"). Le but du jeu est de sauver le plus de lemmings possible.

Les actions du joueurs prennent une unique forme: le joueur choisit (en temps réel) un lemming et lui assigne une **classe**, qui modifie son comportement: par exemple en le faisant creuser à travers des obstacles ou en lui faisant construire un pont au dessus d'un trou. Le jeu se termine quand un nombre fixé à l'avance de lemmings a été créé et quand il n'y a plus de lemmings dans le niveau, le score du joueur correspond au pourcentage de lemmings sauvés par rapport aux lemmings apparus.

Service: Terrain de jeu

Le terrain de jeu `Level` est une **grille** `height` $\times$ `width`. Les cases de la grille sont d'une des trois natures suivantes: `EMPTY`, `DIRT`, `METAL`. Le terrain peut être en mode édition (`editing`) ou jeu (`play`). On peut accéder à la nature d'une case à tout moment.
En mode `editing`, on peut changer la nature d'une case sans restriction.
Un opérateur permet de passer du mode `editing` au mode `play`, elle nécessite que les cases sur le bord de la grille (première et dernière ligne, première et dernière colonne) sont toutes des cases `METAL`.
En mode `play`, la nature d'une case peut changer de deux manières: on peut détruire (`remove`) une case `DIRT` et la transformer en `EMPTY`, et on peut construire (`build`) une case `EMPTY` et la transformer en `DIRT`.

Question 1

Ecrire la spécification d'un service `Level` pour le terrain de jeu.
Il est possible de s'inspirer du squelette donné en Annexe 1; cela n'est toutefois pas obligatoire (**Rappel:** bien lire le sujet en entier avant de commencer à rédiger). Dans ce cas, il faut compléter le squelette avec les préconditions et les observations nécessaires.

Service: Moteur de jeu

Le moteur de jeu `GameEng` est un service qui s'occupe de l'évolution du jeu et des conditions de victoire/défaite (et de score). Cette évolution se fait par *tours*. Le moteur de jeu se lie à un terrain de jeu et maintient un ensemble de lemmings actifs. A chaque tour de jeu, si la partie n'est pas finie, il appelle l'opérateur `step` de chaque lemming actif.
Le moteur de jeu contient un observateur `obstacle` qui prend en entrée les coordonnées d'une case et décide si la case est un obstacle. Initialement, une case est un obstacle si sa nature est `DIRT` ou `METAL`.
Le moteur de jeu est initialisé avec une constante strictement positive `sizeColony` qui indique le nombre total de lemmings qui vont être créés dans cette partie, et un entier strictement positif `spawnSpeed` qui indique la vitesse de création des lemmings. La taille de la colonie est le nombre maximum de lemmings créés, un lemmings créé est actif jusqu'à ce qu'il meurt ou qu'il soit sauvé. Des observateurs et opérateurs doivent permettent de gérer l'ensemble (numéroté) des lemmings actifs.
Le moteur de jeu contient un observateur `gameOver` qui décide si la partie est terminée. La partie se termine quand tous les lemmings ont été créés, et qu'il n'y a plus aucun lemming actifs en jeu. Le score du joueur, accessible, quand le jeu est terminé, avec une observateur `score`, est composé du pourcentage de lemmings sauvés (cf. plus bas) et du nombre de tours de jeu.

Question 2

Ecrire la spécification d'un service `GameEng` pour le moteur de jeu.

Service: Lemming - Marcheur et Tombeur

Un lemming a une **direction**: soit il est droitier (il se déplace vers la droite quand il marche), soit il est gaucher (il se déplace vers la gauche quand il marche). Il est toujours créé droitier. En l'absence de mention explicite dans les explications, le comportement d'un lemmings ne modifie pas sa direction. Même s'il ne marche pas (par exemple quand il tombe), un lemming se souvient de sa direction.

La case au dessus de celle ou se trouve un lemmings doit toujours être vide (graphiquement, c'est là que se trouve sa tête). Il peut y avoir plusieurs lemmings sur la même case, leur comportement lors d'un tour de jeu est indépendant.
A la base, un lemming est un **marcheur** (`walker`) ou un **tombeur** (`faller`).
Le comportement du marcheur à chaque tour de jeu est le suivant (une seule de ces actions, dans cet ordre de priorité - l'Annexe 2 présente des schémas décrivant ce comportement):

1. Si la case juste en dessous du marcheur est vide, le marcheur devient un tombeur.

2. Sinon, si le marcheur est droitier (resp. gaucher) si la case à droite (resp. gauche) et en haut de sa case est un obstacle **ou** si la case à droite (resp. gauche) de sa case et celle à droite (resp. gauche) et deux cases en haut sont des obstacles, le marcheur devient gaucher (resp. droitier). Il reste un marcheur.

3. Sinon, si le marcheur est droitier (resp. gaucher), et que la juste case à sa droite (resp. gauche) est un obstacle, le marcheur se déplace vers la droite et vers le haut (en diagonale) et reste un marcheur.

4. Sinon, si le marcheur est droitier (resp. gaucher), il se déplace vers la case juste à droite (resp. gauche) et reste un marcheur.

Le comportement du tombeur à chaque tour de jeu est le suivant (l'Annexe 3 présente des schémas décrivant ce comportement):

1. Si la case juste en dessous du tombeur est un obstacle et que le tombeur tombe depuis moins de 8 cases, le tombeur devient un marcheur.

2. Si la case juste en dessous du tombeur est un obstacle et que le tombeur tombe depuis 8 cases ou plus, le lemming meurt (il est retiré des lemmings actifs).

3. Sinon (c'est à dire si la case en dessous du tombeur est vide) le tombeur se déplace d'une case vers le bas et reste un tombeur.

Question 3
Ecrire la spécification d'un service `Lemming` (qui peut être marcheur ou tombeur, droitier ou gaucher). Le service devra se lier à un moteur de jeu et contenir (entre autres) un opérateur `step`, qui calcul le comportement du lemming pour le tour de jeu courant.

Modification: Entrée et Sortie

Au moment du passage de terrain en mode `play`, deux cases de nature `EMPTY` sont choisies pour être l'entrée et la sortie du terrain de jeu.

- `entrance`: L'entrée est telle que la case en dessous d'elle et la case au dessus d'elle sont `EMPTY`. A partir du début du jeu, tous les `spawnSpeed` tours, un tombeur vers la droite est créé sur la case d'entrée tant que `sizeColony` lemmings n'ont pas été créé.

- `exit`: La sortie est telle que la case en dessous d'elle est `METAL` et la case au dessu d'elle est `EMPTY`. A chaque tour, si un lemming est présent sur la case de sortie, il disparait du jeu et le compteur des lemmings sauvés est incrémenté.

Question 4
Expliciter les modifications à apporter au terrain du jeu au moteur de jeu pour prendre en compte le fonctionnement de l'entrée et la sortie.

Modification: Lemming - Creuseur

Un lemming peut être désigné par le joueur pour être un **creuseur** (`digger`), et creuser un puit vers le bas.
A chaque tour, le creuseur se comporte ainsi (l'Annexe 4 présente des schémas décrivant ce comportement):

1. Si la case en dessous de lui est `EMPTY`, il devient un tombeur (et n'est plus creuseur).
2. Si la case en dessous de lui est `METAL`, il devient un marcheur (et n'est plus creuseur).
3. Si la case en dessous de lui est `DIRT`, il détruit (`remove`) la case en dessous de lui, ainsi que la case en dessous de lui à gauche si elle est `DIRT` (sinon il ne la modifie pas) et la case en dessous de lui à droite si elle est `DIRT` (sinon il ne la modifie pas), puis il se déplace vers le bas.

Le comportement du creuseur ne modifie pas sa direction.

Question 5
Proposer une modification de la spécification qui prend en compte les creuseurs.

Modification: Lemming - Grimpeur

Un lemming peut être désigné pour être un **grimpeur** (`climber`). Un grimpeur reste grimpeur jusqu'à sa mort ou son sauvetage (il peut aussi être marcheur, tombeur ou creuseur). Si un grimpeur droitier (resp. gaucher) est un marcheur, et si les deux cases à sa droite (resp. gauche) (celle juste à sa droite (resp. gauche) et celle à sa droite (resp. gauche) et une case vers le haut) sont des obstacles et que la case située deux cases au dessus de lui est vide, alors il peut se déplacer d'une case vers le haut. Il effectue ce déplacement même si la case sous ses pieds est vide (et ne devient donc pas un tombeur). (l'Annexe 5 présente un schéma décrivant ce comportement.)

Question 6
Proposer une modification de la spécification qui prend en compte les grimpeurs.

Modification: Lemmings spéciaux

Cette section (à traiter dans le projet) décrit d'autre classes de lemmings et peut être traitée à l'examen (en plus, ou à la place des trois questions précédentes). Les descriptions sont moins formelles (et donc laissent plus de place à l'interprétation) que les précédentes.

- **constructeur** (`builder`): le constructeur construit un escalier en posant une dalle tous les trois tours (qui permet aux autres lemmings de traverser des trous ou de monter sur des falaises). Si le constructeur est droitier, et si les trois cases à la droite sont libres, alors il attend trois tours, remplace ces trois cases par de la `DIRT` et se déplace d'une case vers le haut et de deux cases vers la droite. Il s'arrête si les trois cases à sa droite ne sont pas toutes libres, ou si il a posé 12 dalles.
- **flotteur** (`floater`): le flotteur est cumulable avec les autres statuts. Si un flotteur devient tombeur, il tombe deux fois moins vite qu'un tombeur et ne meurt pas en touchant le sol, peu importe la hauteur de chute.
- **exploseur** (`bomber`): l'exploseur est cumulable avec les autres statuts. Cinq tours apres avoir été designé exploseur, le lemming explose (il meurt) et détruit les 14 cases les plus proches de lui si elles sont `DIRT`.
- **stoppeur** (`stopper`): le stoppeur devient immobile et sa case et la case en haut sont considérées comme des obstacles pour les autres lemmings. Il permet, par exemple, d'arrêter les lemmings se dirigeant vers une falaise.
- **pelleteur** (`basher`): le pelleteur creuse un tunnel de hauteur 3 dans sa direction. S'il est droitier, si la case en dessous de lui est un obstacle si aucune des trois cases à droite n'est `METAL`, il enleve ces trois cases et se déplace vers la droite. Il s'arrete lorsqu'il a creusé 20 fois, s'il rencontre une case de `METAL` ou si la case en dessous de lui est `EMPTY`.

- **mineur** (`miner`): le mineur creuse un tunnel en diagonale (vers le haut ou vers le bas).

Question 7
Proposer une modification de la spécification qui prend en compte un ou plusieurs de ces lemmings.

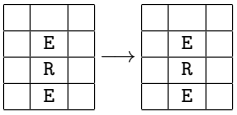
Annexe 1: Squelette du service `Level`

Service:	Level
Types:	bool, int, `enum Nature{EMPTY, DIRT, METAL}`
Observators:	`height`: [Level] → int
	`width`: [Level] → int
	`editing`: [Level] → bool
	`nature`: [Level] × int × int → `Nature`
Constructors:	`init`: int × int → [Level]
Operators:	`setNature`: [Level] × int × int × `Nature` → [Level]
	`goPlay`: [Level] → [Level]
	`remove`: [Level] × int × int → [Level]
	`build`: [Level] × int × int → [Level]

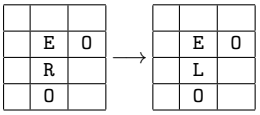
Annexe 2: Comportement du marcheur

Les schémas suivants décrivent le comportement du marcheur droitier (le comportement pour le gaucher est symétrique). Un `R` (resp. un `L`) indique une case vide contenant le lemming considéré, qui est droitier (resp. gaucher), un `E` indique que la case est vide (sa nature est `EMPTY`, elle peut par contre contenir d'autre lemmings), un `O` indique que la case est un obstacle, une case blanche ne donne pas d'information (la case peut être de n'importe quelle nature)

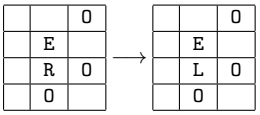
Cas 1 - La case en bas est vide, le lemmings devient un tombeur (il commencera donc sa chute au prochain tour).



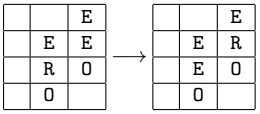
Cas 2.a - La case en haut à droite est un obstacle, le lemmings rebrousse chemin.



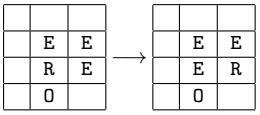
Cas 2.b - Les cases à droite et la case à droite, deux cases en haut sont des obstacles, le lemmings rebrousse chemin.



Cas 3 - La case à droite est un obstacle, mais les deux cases au dessus sont libres, le lemmings monte.



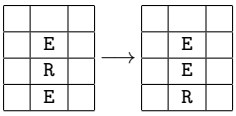
Cas 4 - Les deux cases à droite sont libres, le lemmings avance.



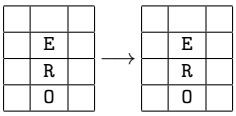
Annexe 3: Comportement du tombeur

Les schémas suivants décrivent le comportement du tombeur droitier (le comportement pour le gaucher est symétrique).

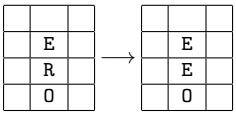
Cas 1 - La case en bas est vide, le lemmings continue à tomber.



Cas 2 - La case en bas est un obstacle, le lemmings tombe depuis moins de 8 cases, il devient un marcheur.



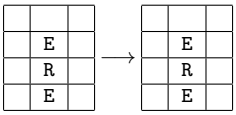
Cas 3 - La case en bas est un obstacle, le lemmings tombe depuis plus de 8 cases, il meurt.



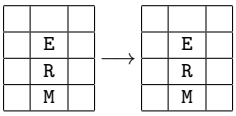
Annexe 4: Comportement du creuseur

Les schémas suivants décrivent le comportement du creuseur droitier (le comportement pour le gaucher est symétrique). Toutes les possibilités du cas 3 ne sont pas données. (`M`: case metal, `D`: case terre)

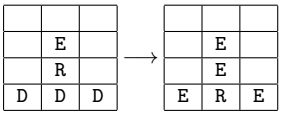
Cas 1 - La case en bas est vide, le lemmings s'arrête de creuser et devient un tombeur (il commencera donc sa chute au prochain tour).



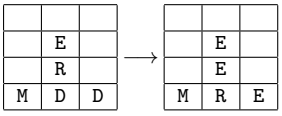
Cas 2 - La case en bas est métal, le lemmings s'arrête de creuser et devient un marcheur.



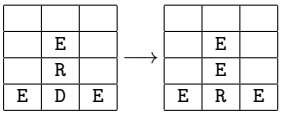
Cas 3.a - La case en bas est terre, le lemmings creuse (ici les deux autres cases en bas sont terre aussi) et se déplace.



Cas 3.b - La case en bas est terre, le lemmings creuse (ici une des deux autres cases en bas est métal) et se déplace.



Cas 3.c - La case en bas est terre, le lemmings creuse (ici les deux autres cases en bas sont vides) et se déplace.



Annexe 5 - Comportement du Grimpeur

Les schémas suivants décrivent le comportement d'un marcheur et grimpeur droitier (le comportement pour le gaucher est symétrique).

Cas 1 - Les deux cases à droite sont des obstacles et la case située deux cases en haut est vide, le lemming grimpe.