université PARIS DIDEROT PARIS 7

Interfaces Graphiques 2022-2023

# Projet de programmation

Le but de ce projet est d'écrire en `java`, à l'aide de la librairie `swing`, une application permettant de jouer au jeu vidéo "Energy"[1].

## 1 Le jeu Energy

Le jeu Energy se joue sur un damier de taille quelconque, dont les cases peuvent être soit carrées, soit hexagonales. Sur ces cases sont posées des tuiles représentant les éléments d'un circuit électrique.

- À chaque tuile est associé un sous-ensemble quelconque de l'ensemble de ses bords, appelé l'ensemble des bords *connectés* de cette tuile.
- Chaque tuile peut également contenir un unique *composant*, parmi trois sortes de composants possibles : *source* d'énergie, *lampe*, ou *borne* wifi.

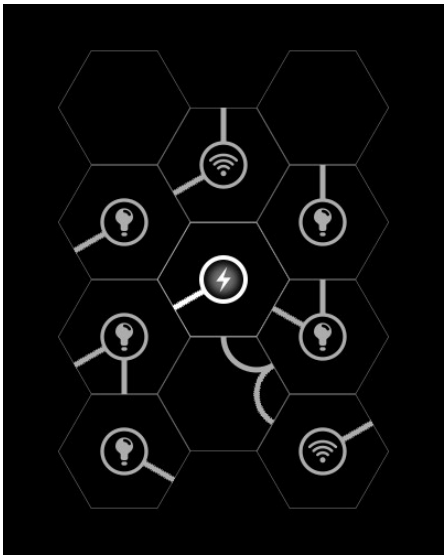
Au début du jeu, le plateau est présenté au joueur avec toutes ses tuiles, chaque tuile ayant subi une rotation aléatoire. L'objectif du joueur est de tourner les tuiles de manière à alimenter en énergie chaque lampe du circuit. Les règles de diffusion de l'énergie sont les suivantes :

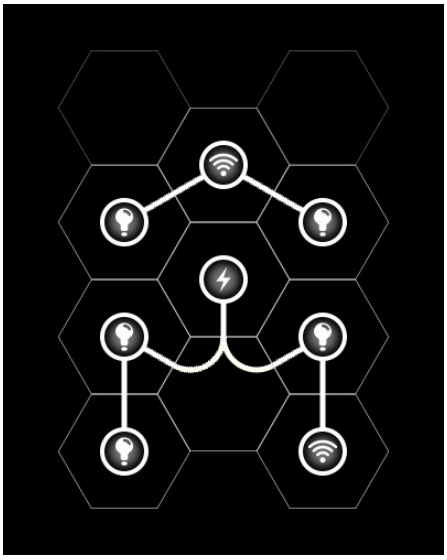
- *Partage.* Si l'un des bords connecté d'une tuile reçoit de l'énergie, tous les autres bords connectés de cette tuile en reçoivent aussi.
- *Propagation.* Si un bord connecté d'une tuile est face à un bord connecté d'une autre tuile et si le premier bord reçoit de l'énergie, le second en reçoit aussi.
- *Émission.* Une source émet de l'énergie vers chacun des bords connectés de sa tuile.
- *Réception.* Une lampe reçoit de l'énergie si les bords connectés de sa tuile reçoivent de l'énergie.
- *Transmission* Une borne reçoit de l'énergie si l'un des bords connectés de sa tuile reçoit de l'énergie, ou si une autre borne quelconque du plateau reçoit de l'énergie[2].

[1] Voir par exemple https://poki.com/en/g/energy pour une des formes possibles de ce jeu, similaire à celle présentée ici.

[2] Autrement dit, si une borne quelconque reçoit de l'énergie, *toutes* les bornes reçoivent de l'énergie.

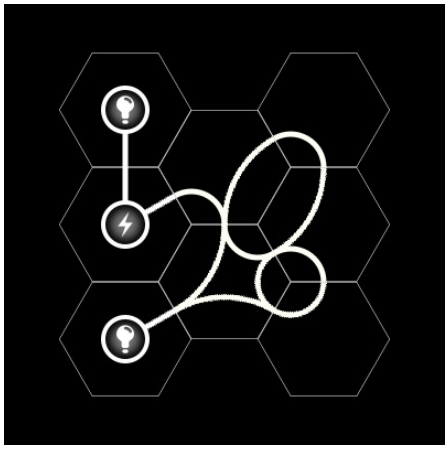
Les deux images suivantes[3] montrent deux états du jeu, en début et en fin de partie :





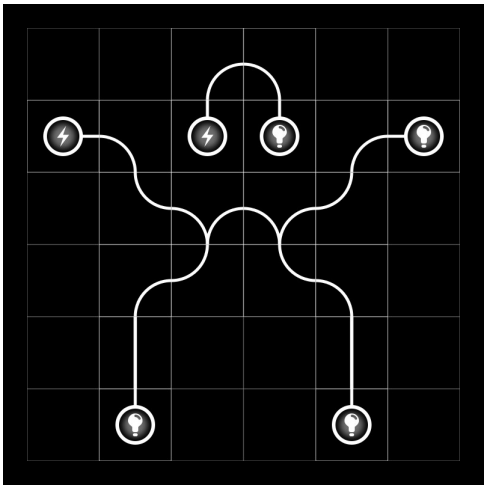
La tuile centrale est ici l'unique tuile munie d'une source. En dessous se trouve une tuile sans composant, à trois bords connectés. Les autres tuiles contiennent une lampe ou une borne, avec un ou deux bords connectés. On notera qu'en fin de jeu, la borne en bas à droite est alimentée par la source centrale, ce qui alimente par transmission la borne de la partie supérieure du plateau.

Comme indiqué ci-dessus, l'ensemble des bords connectés d'une tuile peut être quelconque. Voici un autre exemple de configuration finale, avec des câblages plus exotiques :



[3] Toutes les images du sujet sont des captures d'écran de la maquette du projet.

Voici enfin un autre exemple d'état final du jeu, avec cette fois des tuiles carrées :



# 2 Implémentation du jeu Energy en Java

Votre projet devra permettre de jouer au jeu Energy sur un ensemble de configurations initiales – des *niveaux* encodés par des fichiers textes. Les images de cet énoncé ne font qu'illustrer le principe du jeu et de son interface et ne sont données qu'à titre indicatif : il ne vous est pas demandé de les reproduire de manière exacte dans votre implémentation, mais seulement de respecter la spécification détaillée dans cette section.

Le design de l'interface, les graphismes, le placement des différents éléments, etc., sont libres, mais les contraintes de comportement ci-dessous devront toutes être respectées, à commencer par la première d'entre elles : le jeu doit être intégralement jouable à la souris, sans jamais avoir besoin de recourir au clavier – ce qui ne vous empêche en rien de définir des raccourcis clavier pour certaines actions.

## 2.1 L'écran de jeu

L'écran d'accueil du programme devra permettre de sélectionner une banque de niveaux parmi deux banques possibles puis, pour la banque choisie, de sélectionner un niveau parmi un ensemble de niveaux numérotés par de simples entiers. L'écran sera alors remplacé par un écran de jeu, affichant le plateau après avoir fait subir une rotation aléatoire à chacune de ses tuiles.

**Mise à l'échelle de l'affichage.** Les dimensions de l'écran de jeu ne doivent pas excéder des valeurs maximales choisies de manière à ce qu'il ne recouvre pas une trop grande portion de l'écran de la machine. L'échelle d'affichage sera adaptée de manière à respecter cette contrainte.

**Interfaçage.** L'écran de jeu doit également contenir un bouton permettant de revenir à l'ecran d'accueil, ainsi que tout autre élément que vous jugerez utile (aide, solution, score, etc).

**Visuel.** Les aspects visuels suivants, présents dans la version commerciale du jeu, pourront être repris à l'identique :

- Affichage de liens courbes entre les différents bords connectés d'une tuile si elle ne contient aucun composant. Afin de ne pas surcharger l'affichage, on peut convenir du fait qu'un lien ne sera affiché entre un bord connecté et un autre que si le second est le bord connecté le plus proche.
- Lorsqu'une tuile contient un composant, affichage de liens radiaux, partant du centre d'un composant et partiellement masqués par celui-ci, allant vers chaque bord connecté de la tuile.
- Luminosité accrue pour tous les éléments alimentés, mise à jour de cette luminosité après chaque rotation de tuile.

Après chaque nouvelle rotation le programme devra bien sûr pouvoir détecter si la configuration courante est gagnante, c'est-à-dire si toutes les lampes du plateau sont alimentées – il peux y avoir éventuellement plus d'une configuration gagnante. Le programme devra dans ce cas en informer visuellement le joueur, passer s'il existe au niveau suivant, ou revenir à l'écran d'accueil si le niveau joué est le dernier.

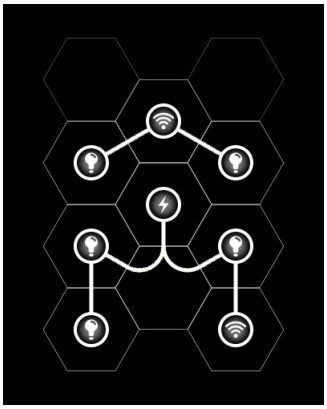
## 2.2 Fichiers de niveaux

Un *fichier de niveau* est un fichier texte dont le nom est de la forme `level`*num*`.nrg` où *num* est le numéro du niveau. Un tel fichier doit décrire une configuration gagnante du jeu. Il est structuré de la manière suivante :

- Le fichier commence par deux entiers $h$ $w$, suivi de la chaîne `"S"` (square) si les cases du niveau sont carrées, ou de `"H"` (hexagon) si elles sont hexagonales. Dans le premier cas, le plateau contiendra exactement $h \times w$ cases. Dans le second, le plateau contiendra $w$ colonnes, les colonnes de rang pair (en commençant par 0) contiendront $h$ lignes, et celles de rang impair $h-1$ lignes, leurs cases étant décalées vers le bas.
- Le fichier contient ensuite une suite d'entrées décrivant, dans la configuratioin finale du jeu, le contenu des tuiles posées sur chacune des cases. Une entrée commence par l'un des chaînes suivantes :
  - `"."` si la tuile ne contient aucun composant,
  - `"S"` si elle contient une source,
  - `"L"` si elle contient une lampe,
  - `"W"` si elle contient une borne.

Cette chaîne est ensuite suivie d'une suite strictement croissante de numéros de bords (*e.g.* une suite vide, ou encore `"3 4"`) indiquant les bords connectés de la tuile. Les bords seront numérotés à partir de 0 en partant du bord nord, et en tournant dans le sens indirect (le sens des aiguilles d'une montre).

Voici par exemple le contenu du fichier encodant le niveau présenté plus haut :



```
4 3 H
. W 2 4 .
L 1 S 3 L 5
L 2 3 . 0 1 5 L 3 4
L 0 . W 0
```

Le choix d'un niveau dans l'écran d'accueil sera suivi de la lecture du fichier de niveau associé, afin de reconstruire en mémoire une configuration finale du niveau, avant de la mélanger par rotation puis de la soumettre au joueur.

## 2.3 Éditeur de niveau

Les deux banques proposées par l'écran d'accueil n'ont pas le même statut. Si l'utilisateur sélectionne la seconde banque, l'écran d'accueil lui donnera la possibilité de jouer à l'un des niveaux de manière ordinaire, mais aussi de passer en mode édition. Les choix d'interfaçage pour effectuer les actions décrites ci-dessous sont libres, mais toutes devront pouvoir être effectuées à la souris de façon fluide. Toute action irréversible – modification effective d'un fichier de niveau, abandon de modifications – devra être confirmée par l'utilisateur.

En mode édition, l'utilisateur doit pouvoir depuis l'écran d'accueil vider un niveau de tous ses composants et bords connectés sans modifier sa géométrie, altérer la géométrie d'un niveau en le vidant implicitement de tous ses éléments – les deux actions mettrons bien sûr à jour le fichier du niveau – ou éditer un des niveaux de la banque sans encore modifier son fichier.

Une demande d'édition de niveau entraîne le passage à un écran de jeu affichant la solution du niveau. L'utilisateur doit pouvoir : ajouter ou supprimer un bord connecté à une tuile donnée, ajouter, retirer ou modifier le type d'un composant d'une tuile. Lors d'une demande de retour à l'écran d'accueil, si le plateau est dans un état gagnant, le fichier du niveau doit être mis à jour. Sinon, les modifications faites sur le niveau seront abandonnées.

# 3 Critères d'évaluation

## 3.1 Qualité de la conception objet et du code

Ce cours est aussi un cours de programmation, plus précisément un cours de programmation objet. Servez-vous de l'héritage, des interfaces de la liaison dynamique et des énumérations partout où ces éléments améliorent la factorisation et la clarté du code. Prenez le temps de réfléchir à la hiérarchie nécessaire, et à la répartition des données et des traitements, et ne programmez pas (jamais) par copier-coller.

Noter qu'une des difficultés de ce projet est la coexistence de deux géométries distinctes pour les cases du plateau (carré, hexagone). En faisant les bonnes abstractions, il est toutefois possible d'éviter de définir deux classes de plateau distinctes, une pour chaque géométrie - c'est un des problèmes centraux de ce projet, qui nécessite une bonne réflexion préliminaire sur la papier.

## 3.2 MVC

Je parlerai en séance de l'architecture MVC, ce type de projet se prêtant idéalement à son usage : votre code sera à la fois mieux organisé, plus facile à développer, plus facile à débugger, et valorisé par le choix de cette architecture si elle est réalisée dans sa forme la plus académique.

## 3.3 Ergonomie de l'interface

Comme son intitulé l'indique, ce cours est un cours d'interfaces graphiques. Votre interface doit être souple, intuitive, naturellement utilisable par un utilisateur même s'il la découvre pour la première fois.

Une image `png`, sa version `svg` et quelques fichiers de niveaux vous sont fournis. L'image contient tous les éléments nécessaires pour afficher les éléments de l'écran de jeu sans perdre de temps à les dessiner[4]. Ces éléments, destinés à être affichés sur un fond sombre, sont tracés en blanc sur un fond transparent - un fichier `pdf` fourni vous indique leur placement et leur dimensions dans l'image. Je vous invite à ajouter à votre rendu des niveaux que vous aurez vous-mêmes créés – ce qui devrait être facile une fois l'éditeur complété.

# 4 Modalités

## 4.1 Groupes

Le projet doit être impérativement réalisé en binôme. En cas d'impossiblité majeure de satisfaire cette contrainte (groupe en nombre impair, conditions d'études particulières, etc.), je vous invite à en discuter avec moi le plus tôt possible - aucun travail non réalisé en binôme ne sera accepté sans cet accord

[4] L'esthétique des graphismes est sans importance pour ce projet (elle est toujours améliorable) et ne sera pas pris en compte dans l'évaluation.

explicite. La répartition des tâches au sein d'un groupe doit être raisonnablement équilibrée. En cas de déséquilibre avéré, les notes finales pourront être individualisées.

## 4.2 Individualité de chaque projet

De manière évidente, votre code doit être strictement personnel. Nous pouvons tolérer l'adaptation d'exemples repris sur le Swing Tutorial d'Oracle ou dans les introductions des API des classes, mais pas la reprise de code trouvé sur le web ou venant d'une quelconque "aide" trouvée sur un forum[5], encore moins le partage de code entres groupes[6]. Il relève de votre responsabilité de faire en sorte que votre code reste inaccessible aux autres groupes : par exemple, si vous vous servez d'un dépôt `git`, ce dépôt doit être privé.

## 4.3 Forme du rendu

Les modalités et dates de rendu seront précisées ultérieurement. Votre rendu consistera en:

- un code-source écrit en `java`, compilable et utilisable tel quel sous Linux et/ou sur les ordinateurs de l'UFR,
- un fichier texte nommé README contenant vos noms,
- un rapport de quelques pages au format PDF décrivant votre projet, et expliquant et justifiant les choix de conception ou d'implémentation,
- tout autre fichier nécessaire à la compilation et à l'exécution.

Tous ces éléments seront placés dans une unique archive compressée en `.tar.gz`.

L'archive devra s'appeler *nom1*`-`*nom2*`.tar.gz`, et s'extraire dans un répertoire *nom1*`-`*nom2*`/`, où *nom1* et *nom2* sont les noms des deux personnes constituant le groupe. Par exemple, si vous vous appelez Denis Diderot et René Descartes, votre archive devra s'appeler `diderot-descartes.tar.gz` et s'extraire dans un répertoire `diderot-descartes/`.

[5] Vous savez vous servir d'un moteur de recherche - vos enseignants aussi.

[6] La soutenance de projet est un examen comme un autre. Le plagiat en projet constitue une fraude aux examens, passible au pire de lourdes sanctions discplinaires – ce cas s'est produit plus d'une fois dans cette UFR.