# Projet de POOIG

Le jeu de l'oie et la famille des jeux qu'on peut lui associer est la source d'inspiration de ce projet.

- Il est particulièrement recommandé que vous le fassiez en **binômes**. Si vous souhaitez le faire seul, informez-en votre chargé de TD/TP, vous pouvez également chercher un partenaire sur le forum dédié de moodle.
- Les soutenances sont prévues en janvier. Les notations pourront être individualisées, il faut donc que chacun maîtrise l'ensemble du travail présenté.

**Quelques conseils :**

- Ne soyez surtout pas tentés de faire une modélisation "à plat" où il n'y aurait qu'une seule classe n'utilisant aucune des spécificités du langage objet. Au contraire nous vous invitons, dès la lecture, à bien analyser les objets qui interviennent, à compartimenter votre code, et à profiter le plus possible de la réutilisabilité que permet java pour pouvoir écrire les jeux demandés et leur variantes.
- Sauvegardez régulièrement votre travail, à chaque modification importante conservez la version antérieure.

Dans la suite de ce document vous trouverez :

- la présentation du jeu de l'oie et de quelques-unes de ses variantes, puis un second jeu, *numeri*, qui est bien moins connu que le premier, mais qui a suffisamment de points communs pour pouvoir faire un travail de modélisation intéressant. Relevez bien les points communs et les différences de ces deux jeux.
- Concernant les affichages, vous devrez d'abord le faire en mode texte, puis plus tard vous pourrez l'adapter au fur et à mesure de l'avancement du cours sur les interfaces graphiques. Il est important de se concentrer tout d'abord sur la modélisation du jeu.
- Enfin nous reprendrons plus précisément les détails du travail attendu.

## Le jeu de l'oie

On dit qu'il y en a 10000 versions différentes, vous pouvez prendre pour thème de base celle que vous préférez, dès lors qu'elle comporte les éléments suivants :

- le plateau est constitué d'un seul tracé linéaire ;
- les joueurs en nombre illimité lancent chacun à leur tour deux dés ;
- les effets des cases sont équivalents à ceux décrits dans wikipédia (saut, passer son tour, attendre une libération, rejouer ...).

Partant de ce premier travail, il vous faudra écrire des variantes. Idéalement en réécrivant une méthode dans une sous-classe, mais cela dépend de votre modélisation. (Evitez de copier-coller du code !) Voici les variantes demandées :

1. le jeu termine quand un pion dépasse la ligne d'arrivée.
2. le jeu termine quand un pion se pose exactement sur la case d'arrivée. Dans le cas où les dés donnent un nombre trop grand alors le pion reculera.
3. deux pions ne peuvent cohabiter dans une case, si le cas se présente, le dernier arrivé ira sur la case précédente (si elle est libre...)
4. lorsque vous arrivez sur une case, vous devez répondre à une question. Les joueurs accumulent alors un score en fonction du nombre de bonnes réponses. Inspirations possibles : jeu de l'oie de l'Unicef [1], jeu de l'oie d'apprentissage du français [2] ou plus simplement vous pouvez poser une question de calcul mental.

1. https ://www.unicef.fr/sites/default/files/userfiles/Jeudeloie_unicef.html
2. https ://fr.islcollective.com/resources/printables/worksheets_doc_docx/jeu_de_loie_verbes_en_er/actions/87206

## Le jeu *Numeri*

La figure suivante illustre le plateau du jeu du commerce. Vous pourrez vous rendre compte des similarités avec le jeu de l'oie. En voici les règles.

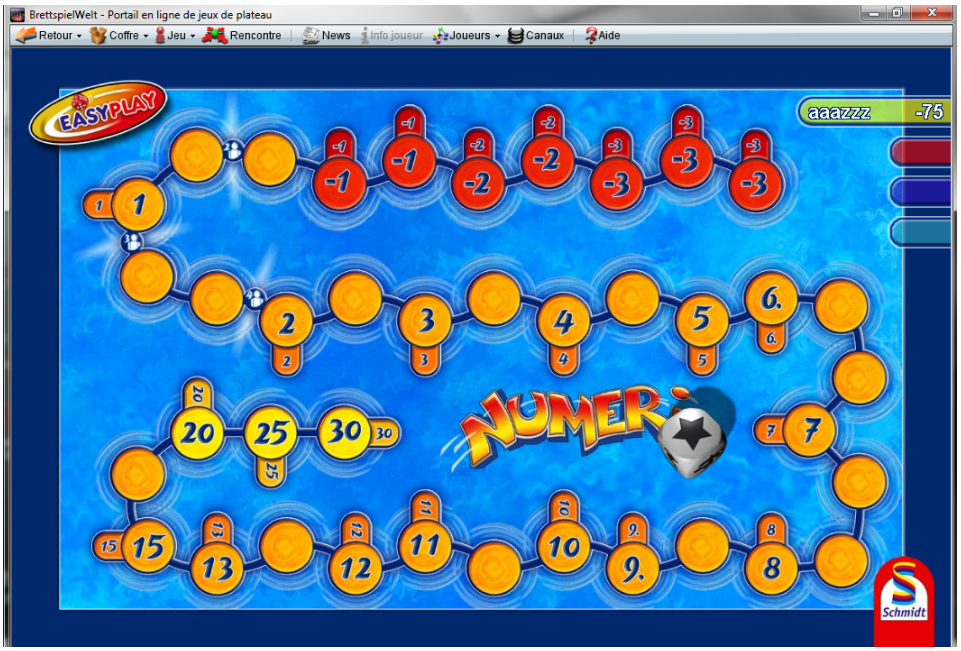
Chaque joueur dispose de 6 pions, affectés d'un coefficient de 1 à 6. Au départ ils sont placés juste avant la première case du plateau, celle qui porte le numéro $-3$ rouge, la plus à droite.

Lorsque c'est son tour le joueur lance un dé et obtient une valeur $x$. Il sélectionne un certain nombre de ses pions dont la somme des coefficients est égale à $x$. Ainsi par exemple s'il lance son dé et obtient 4, il a la possibilité de sélectionner le pion 4 seul, ou alors les pions 1 et 3 ensembles, mais il ne peut rien faire avec les pions 2, 5 ou 6. Il avance ensuite son/ses pions, en les prenant dans l'ordre qui lui convient, jusqu'à la prochaine case libre. Il profite ainsi de la configuration pour *sauter* par dessus d'autres pions, et avancer plus ou moins rapidement ; d'une case dans le pire des cas, de plusieurs s'il peut sauter par dessus d'autres pions.

Et ainsi de suite, chaque joueur à son tour. Le jeu s'arrête lorsque les 3 dernières cases, marquées ici 20, 25 et 30 sont occupées.

Pour déterminer quel joueur a gagné, on évalue la position qu'occupent les pions sur le plateau au moment où le jeu s'arrête. Chaque case, ou presque, porte déjà une valeur, les autres sont considérées comme étant nulles.

Le score d'un joueur est obtenu en multipliant la valeur des cases par le coefficient du pion qui s'y trouve. Si le joueur a plusieurs pions en jeu il cumule ces scores. Le total permet de déterminer le vainqueur.




Vous devrez également imaginer des variantes, par exemple :

— si à la fin de son tour un joueur crée un alignement de trois de ses pions alors il pourra rejouer.

— si le dé comporte une face supplémentaire avec un zéro, alors pour ce tirage le joueur fera reculer un pion d'un adversaire avant de rejouer.

## Séparer la vue du modèle

Le développement des règles du jeu est largement indépendant des aspects graphiques. Si on souhaite faire évoluer notre programme pour l'adapter avec une autre présentation, par exemple sur un téléphone ou une console, l'essentiel du jeu sera tout de même préservé, les changements

à faire relèvent tous de ce qu'on appelle *la vue*. Que le programmeur puisse localiser l'ensemble du code concerné simplifiera donc grandement le développement ultérieur. C'est pourquoi on choisit, très classiquement, de séparer dès la conception les aspects relatifs à la *vue*, des aspects propres au modèle de données.

Dans notre cas, vous allez dans un premier temps tout faire en mode texte. Puis, lorsque vous serez plus avancé dans votre réflexion, que vous aurez quelques connaissances d'interface graphique, et que vous aurez mieux cerné ce que vous souhaitez apporter au projet selon votre sensibilité, vous serez naturellement libre de la modifier, de redéfinir les vues, via une interface, ou une classe plus abstraite. L'essentiel est que vous sépariez modèles en général, et vues en général.

L'association entre les modèles et les vues se fera en introduisant un champs `VueGenerale` dans les modèles de votre jeu, et réciproquement si besoin. Et pour rendre compte d'une modification, graphique ou textuelle, le jeu s'adressera à sa vue.

## Travail d'interfaces graphique

Nous vous demandons d'implémenter graphiquement quelques vues simples, ayant une régularité géométrique. Vous pourrez si vous le désirez placer vos cases dans une grille, utiliser `GridLayout` et considérer les cases comme des boutons. Il serait très souhaitable que la grille puisse se redimensionner lorsqu'on agrandit la fenêtre.

— une vue en colonne :

```
1 - Départ
2 - Case Prison
3 - Case Oie
...
```

— une vue en rectangle :

```
1 - Départ       | 2 - Case Oie  | 3 - Case Pris. |
4 - Case Norm. | ...
```

— une vue en zigzag :

```
1 - Départ      | 2 - Case Oie  | 3 - Case  Pris. |
                                ...   |4 - Case Norm.|
```

— une vue en spirale :

```
Case 1  | Case 2  | Case 3  | Case 4 |
Case  12| Case 13| Case 14| Case 5 |
Case 11 | Case 16| Case 15| Case 6 |
Case 10 | Case 9  | Case 8  | Case 7 |
```

— En bonus, vous pourrez essayer de proposer une solution basée sur une image classique du jeu.

## Rapport et soutenance

Les travaux seront à rendre sur `Moodle` sous la forme d'une archive nommée *nom1-nom2* selon la constitution de votre binôme, et qui s'extraira dans un répertoire *nom1-nom2*. Elle devra contenir :

- les sources et ressources (images ...) de votre programme. Attention à bien utiliser des noms relatifs pour que nous puissions tester votre code sans avoir à modifier quoi que ce soit.
- un fichier nommé `README` qui indique comment on se sert de votre programme, et ce qu'il nous faut regarder en premier.
- un rapport au format *PDF* de quatre ou cinq pages expliquant les parties traitées, les problèmes connus, et les pistes d'extensions que vous n'auriez pas encore implémentées. Il devra contenir impérativement une représentation graphique du modèle. (Le plus simple est qu'elle soit manuscrite, puis scannée)
- pensez à préparer toutes choses utiles pour rendre la soutenance fluide.
- la soutenance devra pouvoir se faire sur une machine du script à partir des sources que vous avez déposées. Elle se déroulera devant un ou deux enseignants dans un mélange de questions et de tests. Il pourra vous être demandé de modifier votre code pour répondre à une question spécifique.
- Enfin, organisez-vous pour avancer petit à petit, sans par exemple vous perdre complètement sur les aspects graphiques.

Source des images :
http ://www.brettspielwelt.de/Hilfe/Anleitungen/Numeri/ ?nation=fr