DIAMANTS

Diamants est un projet d'implémentation d'un jeu de société

OBJECTIF GÉNÉRAL

L'objectif de ce projet est d'implémenter le jeu de société DIAMANTS, aussi connu sous le nom de INCAN GOLD (l'or des incas). Il s'agit d'un jeu de Bruno Faidutti et Alan R. Moon, édité en français chez IELLO.

Le principe du jeu est simple, se jouant essentiellement avec des cartes et quelques jetons. Le mécanisme du jeu est dit de « stop ou encore » : à chaque étape, les joueurs doivent décider s'ils sécurisent leur gain, ou s'ils prennent des risques pour gagner davantage, au risque de tout perdre !

RESSOURCES POUR COMPRENDRE LE JEU

Avant tout, vous pouvez suivre un tutoriel du jeu sur **BoardGame Arena (BGA)**. Il faudra probablement vous créer un compte (gratuit).

* **tutoriel** : https://boardgamearena.com/tutorial?game=incangold&tutorial=575

Le jeu est jouable en ligne sur BGA (entre amis ou contre des joueurs du monde entier), néanmoins il est nécessaire qu'un des joueurs dispose d'un compte payant (2€ pour un mois pour un des joueurs mais ne dépensez pas juste pour cette SAE). Ceci dit, on vous donne plein de ressources ci-dessous pour bien comprendre le jeu.

* **Règles du jeu** : https://iello.fr/wp-content/uploads/2022/07/DIAMANT_regles.pdf

Ressources vidéos qui présentent et expliquent les règles :

* Présentation rapide par LudoChrono : https://youtu.be/AZ_gcm2UYP8

* Détail des règles par Vidéorègles.net : https://youtu.be/_CttteyEScY

* Détail des règles par Professor Board Game : https://youtu.be/1wEKFj_Qdzg

. Vous pouvez également voir des **parties jouées** sur ces vidéos de Domingo Replay :

* https://youtu.be/0gYRlDf3Pns

* https://youtu.be/c07WxhoWnck

(vous pouvez trouver d'autres parties et présentations du jeu sur youtube)

Le matériel de jeu nécessaire

Seul le paquet de cartes est réellement nécessaire pour jouer. Il contient :

* 15 cartes trésor de valeurs 1, 2, 3, 4, 5, 5, 7, 7, 9, 11, 11, 13, 14, 15, 17

* 15 cartes danger : 3 araignées, 3 serpents, 3 laves, 3 boulets, 3 béliers

* 5 cartes Relique

Tout le reste du matériel est **inutile pour une version digitale**. Par exemple :

- les jetons (rubis, diamants) peuvent être représentés par des compteurs, variables, etc. On peut également avoir un seul type de point (pas besoin de jetons qui valent 5), et on peut afficher un nombre.

- les coffres permettant de cacher les jetons sont inutiles. De plus, un joueur attentif peut très bien mémoriser le montant acquis par les autres joueurs donc il n'est pas obligatoire de cacher cette information.

- les cartes décisions, tuiles barricades (le « plateau » pour compter les manches) n'ont pas non plus lieu d'être dans une version digitale.

Ce qui peut ou doit être fait : objectifs d'implémentation

L'objectif final « idéal » de ce projet est de réaliser une version du jeu semblable dans l'esprit à ce que l'on trouve sur Board Game Arena, avec une interface graphique permettant à plusieurs joueurs de jouer sur une même machine (en « hot seat »), ou bien à un seul joueur de jouer contre des intelligences artificielles rudimentaires.

Cependant, l'objectif décrit ci-dessus est certainement très complexe et vous n'irez peut-être pas jusqu'au bout. Aussi nous détaillerons les niveaux de réalisation suivants :

Niveau 0 : Version jouable en ligne de commande, où les joueurs entrent leurs actions dans le terminal (on peut imaginer qu'un arbitre entre les décisions des joueurs qui les révèlent simultanément), avec une seule manche, qui ne prend pas en compte les reliques.

Niveau 1 : Comme la version 0, mais le jeu se joue en 5 manches, on prend en compte les reliques, et on procède à la suppression d 'une carte danger si un piège est déclenché (bref toutes les règles du jeu sont simulées exactement).

Niveau 2 : Comme le niveau 1, mais le jeu dispose maintenant d'une interface graphique permettant de faire les choix à la souris et il affiche les résultats dans l'interface également (le terminal n'est plus utilisé).

Niveau 3 : comme le niveau 2, mais l'interface graphique est travaillée pour afficher les cartes comme sur BGA (prenez des images et formes simples, le graphisme n'a pas besoin d'être élaboré) et l'ensemble est plus esthétique et ergonomique.

Bonus (pouvant intervenir à chaque étape, et très pratique pour tester) : il est possible de remplacer certains des joueurs par des IA élémentaires (par exemple, qui ont une certaine probabilité de CONTINUER ou de SORTIR).

ÉTAPES À SUIVRE POUR LE DÉVELOPPEMENT

ÉTAPE 1 : COMPRENDRE LE JEU

Pour cette première étape vous devez comprendre en détail les règles du jeu en lisant de près le livret de règles, en prenant des notes, et en regardant les vidéos fournies (pas obligatoirement tout, mais essayez de regarder un peu des règles et une partie).

Nous insistons sur les points suivants de règles suivants :

- le jeu se joue de 3 à 8, en 5 manches

- les rubis qui ne sont pas distribués entre les joueurs restent sur les cartes trésor déjà dévoilées. Il est possible de regrouper tous ces rubis en un seul tas (afficher le total). Lors de l'étape de révélation des décisions, les joueurs qui décident de SORTIR se partagent ces rubis ; il peut à nouveau y avoir un reste, et alors il est posé sur n'importe quelle carte.

- pour obtenir une carte relique il faut SORTIR tout seul ; si deux joueurs ou plus sortent en même temps la carte relique reste en place. NB : dans certaines versions du jeu, les cartes reliques ne valent pas 5,5,5,10,10 mais ont des valeurs écrites ; cela ne change pas grand-chose au fonctionnement du jeu.

- les rubis accumulés en cours de manche sont sécurisés seulement si le joueur fait l'action SORTIR avant de tomber dans un piège (deux cartes danger identiques révélées), sinon tout ce qui a été accumulé pendant la manche est perdu.

- Si deux cartes danger identiques sont révélées, on retire du jeu l'une de ces deux cartes pour les manches suivantes. Cela influence donc les prises de risque pour les manches suivantes.

Étape 2 : faire une version minimale, en ligne de commande

Cette étape est indispensable, ne commencez pas directement une version graphique. Il est en effet important que toute la mécanique du jeu soit gérée par une collection de fonctions qui sont indépendantes et bien séparées de l'aspect graphique. Ce qui va différer entre cette version et la version graphique, ce sont les entrées/sorties, c'est-à-dire la manière dont les informations sont affichées (terminal / interface graphique) et aussi récupérées ; toute la mécanique du jeu est la même et est assurée par les mêmes fonctions.

Pour cette étape, il est important de penser à la structure générale de votre programme et de le découper en fonctions les plus simples possibles. Concevez également les structures de données qui vont garder en mémoire l'état actuel du jeu et qui vont être passées en paramètres aux différentes fonctions : une fonction pour gérer le début, une pour la boucle principale, une qui gère les manches, une fonction qui fait l'affichage, une pour gérer le deck… vous êtes libres mais gardez à l'esprit que les fonctions doivent être courtes (on dit en général maximum 15 à 20 lignes de code, à titre indicatif) et remplir un objectif précis.

Cette étape consistant à penser la structure du code et le découper en fonction doit être effectuée **avant** d'écrire le contenu des fonctions proprement dit. Idéalement, on écrit tous les prototypes et *docstrings* des fonctions et on fait un schéma décrivant comment les fonctions s'appellent entre elles ; une fois que ceci est effectué il reste à écrire les codes des fonctions, et à les tester individuellement (test unitaires).

A la fin de cette étape, débuggez au maximum votre code jusqu'à obtenir un programme simple mais stable. Quand cette étape est terminée, gardez une copie du code, par sécurité. Gardez également les fonctions de test (regroupées dans un fichier séparé).

Étape 3 : Améliorer votre programme et ajoutez les graphiques

C'est le moment d'ajouter du chrome. Ajoutez les points de règles que vous avez négligés, et remplacez toute l'interaction avec le terminal par une interaction avec une interface graphique. C'est aussi à cette étape que vous pouvez commencer à ajouter des IA simples

permettant de tester facilement votre code. Dans la mesure du possible ne modifiez pas la structure de votre code de l'étape précédente, changez juste les fonctions d'affichage dans le terminal par des fonctions d'affichage dans l'interface, et idem pour la récupération des décisions de l'utilisateur.

Votre interface graphique peut être très simple (affichage de texte, quelques boutons pour cliquer) ou plus complexe et élégante. Vous avez intérêt à la penser et préparer des croquis avant de vous lancer. Il va aussi falloir qu'elle s'adapte au nombre de joueurs ! Dans un premier temps vous pouvez fixer le nombre de joueurs à 4, par exemple.

VOTRE TRAVAIL

PENDANT LA SAÉ

- Travail en **binôme obligatoire**. Exceptionnellement par 3, si pas d'autre possibilité et uniquement sur autorisation de vos enseignant.e.s. Il est important de trouver un binôme qui soit à peu près du même niveau que vous, avec qui vous pouvez travailler à deux et vous répartir le travail (vous devez cependant maîtriser l'ensemble du code à la fin). Le travail en projet est une occasion de beaucoup progresser, aussi ne laissez pas l'autre tout faire ! Vous seriez également pénalisés lors de l'évaluation.
- Vous suivrez la progression par étapes proposée dans ce sujet afin que votre programme soir bien structuré. La note maximale sera donnée aux projets qui atteignent le dernier niveau et qui sont clairs, bien écrits, stables ; mais nous préférons un programme bien écrit, stable, et qui va moins loin dans les niveaux, plutôt qu'un programme très sophistiqué mais mal conçu, pas clair et et plein de bugs .
- Toutes les fonctions doivent être documentées avec des spécifications (docstring), et raisonnablement commentées. Les noms des fonctions et des variables doivent être judicieusement choisis. Comme précisé plus haut, les spécifications des fonctions et docstrings servent à développer intelligemment votre programme, et il est dommage voire inutile de les faire à la fin, on doit s'appuyer dessus pour bâtir le programme !
- S'aider d'un groupe à l'autre est possible, recopier un code tel quel est interdit. Nous avons des moyens informatiques de comparaison des codes (JPlag, entre autres). Le code de votre binôme doit être le vôtre, et chacun.e des membres du binôme doit être capable de l'expliquer intégralement, que ce soit « votre partie » ou non.

UNE FOIS LA SAÉ TERMINÉE

- Date de rendu des projets: **SAMEDI 17/12 à 23h59. SOUTENANCES LA SEMAINE DU 3/01**.
- Vous déposerez votre projet dans la **rubrique Travaux de l'espace Initiation au Développement sur eCampus**, sous la forme d'une **archive** au format

`NOM1_NOM2.tar.gz` où NOM1 et NOM2 sont les noms des deux membres du binôme.
ATTENTION: le non respect précis de cette consigne (mauvais nommage, mauvaise compression, dépôt en retard) entraîne une diminution de la note finale.

- Le projet doit être fonctionnel **SUR LES MACHINES DE L'IUT**. À vous de tester votre projet sur ces machines avant de le rendre.
- Dans votre archive, vous joindrez un document **README** indiquant la procédure pour lancer votre jeu et y jouer, le travail accompli, les problèmes rencontrés, les bugs connus, les idées originales…etc Une partie du README peut-être demandée en anglais (voir avec l'enseignante).

ÉVALUATION

Des soutenances en binôme auront lieu début janvier.

Les **compétences** qui seront évaluées sur ce projet seront à priori :

- *Respect du cahier des charges "dépôt" (deadline, README, archive nettoyée et au bon format, programme fonctionne directement sans erreurs et sans rien installer de nouveau sur les machines de l'IUT)*
- *Respect du cahier des charges minimal du programme : vous êtes allés au bout des niveaux niveaux 0 et 1, la mécanique du jeu est fonctionnelle.*
- *Spécifications et tests: chaque fonction doit documenter sa **spécification**, et les fonctions critiques doivent être accompagnées de leur **fonction de test** (test unitaire).*
- *La qualité du code : commentaires, découpage en fonctions, constantes, nommage correct…*
- *Considérer tous les cas particuliers d'un programme (ex: tel paramètre fait-il planter le programme?)*
- *Vous êtes allés plus loin en fournissant une interface graphique basique.*
- *Votre interface graphique est avancée, pratique, s'adapte en fonction du nombre de joueur.*
- *Incorporer des solutions techniques innovantes : soyez imaginatifs! Ici on note ce que vous apportez en plus sur le programme par rapport à ce qui est demandé. Attention : vous ne devez pas changer les règles du jeu.*
- *Vous avez implémenté des IA élémentaires pour pouvoir jouer sans adversaire.*

ET MAINTENANT?

Go ! N'hésitez surtout pas à questionner vos enseignant.e.s si vous avez des questions. De plus, un salon dédié à la SAé permettra de dialoguer sur Discord. Bon travail! Amusez-vous bien…