Rapport de l'unité Projet math/info

Thoma Lessage
Pascal Padilla
Raphaël Sellam

14 mars 2020

Table des matières

Appel à projet

Proposition de projet dans le cadre du module **Projet Mathématiques et Informatique** du **DU CCIE**.

Description du projet

Ce projet sera l'occasion de **développer une application**.

Idée de départ (étude de cas)

L'idée de cette application est partie du problème algorithmique suivant :

Je suis un prof de maths qui motive ses 30 élèves à pratiquer le calcul mental en organisant un **tournoi** au sein d'une classe. Mais je me fixe comme contrainte que **chaque participant affronte tous les autres**.

Pour ma classe de 30 élèves, il faut donc que chaque élève participe à 29 matchs, ce qui fait que je dois gérer **435 matchs** !

Il y a là un problème algorithmique d'**optimisation** mais je suis confiant : puisque je suis prof de math, j'espère bien organiser ces 435 matchs en seulement **29 tours** (je pense pas que l'on puisse faire moins...).

Objectifs du projet

Objectif principal

1. Créer l'algorithme `générer_tournoi`

```
Algorithme : générer_tournoi
Entrée : liste des participants
Sortie : ensemble de tous les matchs d'un tournoi,
            regroupement des maths par tours,
```

2. Optimiser l'algorithme pour assurer que le nombre de tours est **minimal**

3. Développer une application **multi-plateforme**

Objectif secondaire

— Enrichir l'application avec des fonctionnalités (cf. Annexe Carnet de fonctionnalités)

Organisation du projet

Langage de développement

Pour développer cette application, l'équipe de développement utilisera le moteur de jeu **Godot** (site officiel [1] et page wikipédia [2]).

Ce moteur de jeu possède de nombreux avantages :

— création d'applications **multi-plateformes** (plateformes mobiles (iOS, Android), ordinateur (Windows, macOS, Linux) et web (HTML5 et WebAssembly)
— logiciel libre et gratuit

1. <https://godotengine.org/>
2. <https://fr.wikipedia.org/wiki/Godot_\char‘%{}28moteur_de_jeu\char‘%{}29>

— plateforme de développement : Linux, Windows, Mac
— programmation de scripts en (au choix) : Godot Script (équivalent Python), Python, C++, C#, etc.

Utiliser Godot apportera à l'**équipe de développement** les compétences suivantes :

— découverte pour l'équipe d'un moteur de jeu (par exemple les liens entre interface et scripts)
— outil professionnel utilisé dans l'industrie dujeu (grand studio et **indépendant**)
— très **attractif** pour les élèves

Planification et phases de développement

— **Phase 1** (6 semaines)
  — projet : appropriation du problème
  — algo : premiers algorithmes (langage de programmation libre)
  — Godot : installation et découverte Godot : Hello World ! + export multi-plateforme
  — interface : premiers schémas/dessins d'interfaces
— **Phase 2** (6 semaines)
  — algo : vérification / optimisation algorithme
  — Godot : transposer l'algorithme sur Godot
  — Godot : création d'un interface
— **Phase 3** (6 semaines)
  — Godot : évolution/correction application (fonctionnalités)
  — Godot : évolution/correction interface

ANNEXE

Carnet de fonctionnalités

Pour clarifier mes attentes vis à vis de l'application, je propose le carnet de fonctionnalités suivant. Pour établir cette liste de besoins, je me suis mis à la place de l'**utilisateur** de l'application, du **participant** au tournoi et aussi du **prof** qui utilise cette application dans sa classe.

Cette liste est un idéal **ordonné** mais évidemment non définitif. Des fonctionnalités nouvelles peuvent être **ajoutées**, des fonctionnalités inutiles pourront être **supprimées**.

1. En tant qu'utilisateur, je veux créer des tournois facilement (où tous les participants s'affrontent entre eux), afin de connaître pour chaque tour du tournoi la répartition des participants, match par match
2. En tant qu'utilisateur, je veux utiliser l'application indifféremment sur un ordinateur/tablette/smartphone, afin de pouvoir utiliser l'application en toute circonstances
3. En tant qu'utilisateur, je veux afficher à l'écran le tableau matchs/scores, afin de projeter le tableau et faire suivre les résultats en direct
4. En tant qu'utilisateur, je veux que le tournoi puisse aboutir à un classement entre participants, afin d'atteindre le but du tournoi qui est de définir un ou plusieurs vainqueurs
5. En tant qu'utilisateur, je veux que le score d'un participant soit calculer de façon équitable, afin de ne pas démotiver les plus faibles
6. En tant que participant, je veux connaître à l'issue du tournoi mon score final, afin de m'autoévaluer
7. En tant que participant, je veux comprendre le mode de calcul du score, afin de (1) vérifier le calcul et (2) comprendre comment fonctionne l'application et (3) m'améliorer
8. En tant qu'utilisateur, je veux accéder à l'historique de tous les résultats, afin d'établir un bilan général sur l'application et les tournois
9. En tant que participant, je veux que l'application fonctionne hors connexion (sans wifi par exemple), afin de mettre en place un tournoi sans contrainte de connexion
10. En tant qu'utilisateur, je veux importer facilement la liste de participants, afin de démarrer le plus rapidement possible le premier tour du tournoi
11. En tant qu'utilisateur, je veux importer les identifiants des participants à partir de fichiers aux formats variés (par ex. XLSX, CSV, ODS, XLS, etc.), afin de ne pas avoir à saisir à la main la liste des participants

12. En tant qu'utilisateur, je veux créer un nouveau tournoi à partir d'un tournoi précédents, afin de (1) créer rapidement un tournoi, (2) ne pas avoir à saisir manuellement la liste des participants
13. En tant qu'utilisateur, je veux regrouper en différents groupes les participants à des tournois précédents, afin de (1) saisir facilement les groupes, (2) proposer aux profs de gérer des classes, (3) créer des groupes de niveaux
14. En tant qu'utilisateur, je veux modifier la liste des participants d'un tournoi en direct, afin de (1) mettre à jour une liste importée, (2) ajouter des participants absents (3) supprimer (temporairement) un participant absent
15. En tant qu'utilisateur, je veux importer les identifiants des participants par copier/coller, afin de personnaliser le tableau d'affichage maths/scores
16. En tant qu'utilisateur, je veux récupérer/exporter les scores du tournoi, afin de pouvoir exploiter de façon personnalisée les résultats
17. En tant qu'utilisateur, je veux créer un tournoi facilement (regrouper les participants par poules pour qualifier les finalistes), afin de (1) gérer un trop grand nombre de participant (trop de matchs), (2) gérer des groupes de niveaux hétérogènes
18. En tant que participant, je veux générer des diplômes du tournoi pour les participants, afin de féliciter les participants : (1) bien classés, (2) motivés mais pas forcément avec un bon score, (3) qui ne lâches rien mais par forcément avec un bon score
19. En tant que participant, je veux obtenir mon diplôme sur le tournoi auquel j'ai participé, afin de (1) me motiver, (2) me féliciter du travail accompli
20. En tant qu'utilisateur, je veux exporter les diplômes des participants au format PDF, afin de (1) pouvoir imprimer les diplômes (2) féliciter les participants
21. En tant qu'utilisateur, je veux pouvoir personnaliser les diplômes, afin de (1) m'adapter au tournoi et aux regroupements de participants et (2) motiver les participants
22. En tant qu'utilisateur, je veux gérer l'historique des tournois, afin de (1) créer à posteriori des regroupements de tournois et (2) obtenir un classement à partir de différents tournois
23. En tant qu'utilisateur, je veux que le score obtenu puisse dépendre de l'historique des tournois précédents, afin de proposer des tournois motivants avec de l'enjeu
24. En tant qu'utilisateur, je veux avoir un bilan par participant et par groupe sur l'évolution des scores entre les tournois, afin d'établir un bilan sur le groupe
25. En tant qu'utilisateur, je veux pouvoir obtenir un score (classement ?) par participant sur l'ensemble des tournois auquels ils ont participé, afin de motiver les participants sur la durée
26. En tant que prof, je veux pouvoir évaluer les participants par compétences, afin d'intégrer la pratique du tournoi avec les recommandations et presrciptions du BO
27. En tant qu'utilisateur, je veux partager facilement les scores d'un tournoi avec les participants, afin de donner accès à l'historique du tournoi aux participants (sans oublier d'anonymer les identifiants RGPD)
28. En tant qu'utilisateur, je veux partager facilement les scores de tous les tournois avec les participants, afin de (1) donner accès à l'historique des tournois aux participants (sans oublier d'anonymer les identifiants RGPD) (2) établir un classement international entre les pays du monde !
29. En tant qu'utilisateur chinois, je souhaite utiliser l'application afin de créer des tournois de millions de personnes.

Générateur automatique de doc

La documentation est actuellement disponible sur le github : https://github.com/Pskalou/tournoi/tree/master/source/doc

Principe

La documentation est générée à partir de commentaires ajouté avant la déclaration des classes, méthodes et attributs.

Par exemple, voici le début de la classe `Score_manager.gd`

```
# Cette classe gère l'état du tournoi actuel.
# Dans un second temps, l'état est converti en score.
class_name Score_manager


# Barême pour une victoire.
# Par défaut égal à 10
#
# TODO : privatiser la propriété (set/get)
var win_points  :int= 10
```

Qui permet de générer le fichier https://github.com/Pskalou/tournoi/blob/master/source/doc/Score_manager.md [3]

How To

Nous avons installé le plugin gdscript-docs-maker

— https://github.com/GDQuest/gdscript-docs-maker

Sous `bash`, il faut se rendre dans le dossier `gdscript-docs-maker`. Puis lancer la commande `./generate_reference ./../source/ ./../source/doc/`. Les fichiers générés seront déplacés dans le dossier `source/doc`.

3. <https://github.com/Pskalou/tournoi/blob/master/source/doc/Score_manager.md>

Road map

Le développement du projet est découpé en 4 phases temporelles :

Phase 1 (5 semaines)

Les objectifs pour l'équipe sont les suivants :

— échéance : 22/01/2020
— livrer la version **alpha**

Gestion de projet

— **Déployer les outils**. Mise en place d'un serveur et installation d'un Redmine. Créations de comptes Github et Github classroom.
— **S'approprier les outils 2**. Prendre en main les différents outils proposés (Redmine, Github et visioconférence sur https://meet.jit.si/).
— **Méthode Scrum**. Comprendre le fonctionnement général de la méthode agile Scrum. S'approprier les notions de cycles, de backlogs et de revues. Mettre en place les sprints.

Problématique du projet

— **S'approprier la problématique**. Définir un vocabulaire commun et comprendre les objectifs de l'application. Trouver des modélisations possibles.
— **Premiers développements**. Se limiter à des cas simples. Choix libre dans le langage de programmation. Partager le code.

Application

— **Découverte de Godot**. Installation et découverte Godot. Premiers programmes et tutoriels divers. Exports multi-plateforme.
— **Interface de l'application**. Premiers schémas/dessins d'une interface possible.

Phase 2 (7 semaines)

Les objectifs pour l'équipe sont les suivants :

— livrer la version beta
— échéance : 15/03/2020

Problématique du projet

— **Modéliser la problématique**. Choisir un modèle pertinent généralisant la problématique.
— **Généralisation des algorithmes**. Choix d'un langage de développement. Programmer les fonctionnalités essentielles de l'application.

Application

— **Godot**. Transposer les fonctionnalités essentielles en Godot (GDscript et/ou Python et/ou C#)
— **Interface de l'application**. Développer l'interface de l'application.

Phase 3 (6 semaines)

Les objectifs pour l'équipe sont les suivants : * livrer la version Release Candidate * échéance : 30/04/2020

Application

— **Godot**. Transposer les fonctionnalités supplémentaires en Godot (GDscript et/ou Python et/ou C#)
— **Interface de l'application**. Développer l'interface de l'application et créer les écrans supplémentaires
— **Application**. Déployer l'application, tester et debugger

Phase 4 (4 semaines)

Les objectifs pour l'équipe sont les suivants :

— livrer la version Stable
— échéance : 31/05/2020

Application

— **Application**. Correction et évolution.

Version alpha

Nous sommes heureux d'annoncer la mise en ligne de la version alpha de notre application

Vous pourrez trouver sur notre dépot : * fichiers sources : source:source * exports en différents formats : source:source/pascal/exports@7ba7f97a

et tester la version en ligne * **version en ligne**[4]

À PROPOS DE LA VERSION ALPHA

Deux versions de l'application ont vue le jour :

— version **Python** partagée au format Jupyter notebook
— version **Godot Engine**

Ci-dessous, diagramme montrant le nombre de demandes en fonction du temps. On avance, on avance...

PYTHON

Développement en 2 temps :

— recherche et preuve algorithmique (plutôt vérification sur les 500 premiers cas...)
— mise en forme

01premier**algorithme_python.ipynb**

— Recherche d'un algorithme générant un tournoi
— Vérification algorithmique que les tournois générés répondent à la problématique

Fichier à télécharger (notre dépot) : source:notebook/01premieralgorithme_python.ipynb@7ba7f97a Fichier à consulter en ligne (visionneuse Github) : visualisation en ligne[5]

02algo**plus_lisible.ipynb**

— Mise en forme du code
— Clarification des différentes parties

Fichier à télécharger (notre dépot) : source:notebook/02algoplus_lisible.ipynb@7ba7f97a Fichier à consulter en ligne (visionneuse Github) : à visualiser en ligne[6]

GODOT ENGINE

Nous avons tenté une version Godot. Dans un premier temps nous avons cherché à mettre en forme l'interface puis, dans un soucis d'efficacité nous avons seulement tenté d'implémenter l'algorithme trouvé en Python dans un projet Godot.

Ensuite nous avons exporté notre projet sous différentes formats (source:source/pascal/exports@7ba7f97a)

4. <https://pskalou.github.io/tournoi/source/pascal/exports/html/Tournoi_html5.html>
5. <https://github.com/Pskalou/tournoi/blob/master/notebook/01_premier_algorithme_python.ipynb>
6. <https://github.com/Pskalou/tournoi/blob/master/notebook/01_premier_algorithme_python.ipynb>

— html (version en ligne[7] sur GitHub)
— linux
— mac_osx
— windows
— windows_universal

Source à télécharger (notre dépot) : source:source

/!\ Attention , comme le développement se fait de façon continue, les captures d'écran et les versions exportées peuvent différées suite aux multiples évolutions.

7. <https://pskalou.github.io/tournoi/source/pascal/exports/html/Tournoi_html5.html>

Scrum

L'organisation du projet est inspirée de la méthode Scrum.

Avant le projet, la liste des fonctionnalités : le Backlog du produit

Avant le moindre développement, il y a une personne (un client, le commanditaire) qui imagine un produit : c'est le **propriétaire du produit**. Afin de l'aider à expliciter ses besoins, le propriétaire du produit est assisté par le **maître de mêlée**. Le maître de mêlée ne fait pas parti de l'équipe de développement, il n'est pas un chef de projet ni un supérieur. Il est là pour faire en sorte que la méthodologie Scrum soit appliquée. Il met en place les rituels ou aide à la communication entre propriétaire du produit et équipe de développement. À eux deux, propriétaire de produit et maître de mêlée vont écrire le **backlog du produit**. C'est la liste ordonnée des fonctionnalités du produit. Dans la méthodologie Scrum, cette liste **n'est pas figée**. Le propriétaire de produit peut changer à discrétion l'ordre des éléments, en ajouter, modifier le découpage en éléments, modifier leur description, ou supprimer des éléments qui n'ont pas encore été réalisés.

Pendant le projet, des cycles itératifs : les Sprints

Une fois que le projet est lancé, la méthode Scrum consiste en une **itération d'événements** simples de durées courtes. Toute la durée de développement du projet est segmentée en une répétition de cycle (appelés **sprint**), chacun d'une durée de 1 à 4 semaine (**à définir**).

1. **Avant** de débuter un sprint, l'équipe de développement défini le **Backlog du sprint** c'est à dire les fonctionnalités qui vont être développées durant le sprint. Ces fonctionnalités sont issues du Backlog du produit.

— **Pendant** le sprint, l'équipe se réuni quotidiennement : c'est la **mêlée quotidienne**. Cette discussion de 15min maximum est l'occasion pour chaque membre de l'équipe de présenter ce qu'il a réalisé la veille, ce qu'il compte réaliser le jour même pour atteindre l'objet du sprint et les obstacles qui empêchent l'équipe d'atteindre le but du sprint.
— **Après** un sprint,
  — l'équipe présente propriétaire du produit les fonctionnalités entièrement achevées : c'est donc une version améliorée de l'application
  — l'équipe se réunit et débriefe pour améliorer le fonctionnement du sprint suivant

La littérature concernant cette méthode est très dense et variée.

— https://fr.wikipedia.org/wiki/Scrum_(d%C3%A9veloppement)
— https://www.bettercalldave.io/methodologie/definition-scrum/
— lister les sites qui nous ont aidés...
— ...

Projet Tournoi

En ce qui concerne le projet tournoi, les rôles de propriétaire de produit et de maître de mêlée ont été joués par une seule personne : **le chef de projet**. En effet, c'est au chef de projet qu'incombe la responsabilité du produit, la définition de ses fonctionnalités ainsi que ses limites. Par ailleurs, en imposant la méthode de travail Scrum, le chef de projet doit expliquer et accompagner l'équipe dans cette méthodologie. Il est donc aussi maître de mêlée.

Il y a plusieurs difficultés pour appliqué la méthode Scrum dans le cadre d'un projet de DU en ligne. D'abord les personnes ne travaillent pas dans un même lieu géographique, mais à **distance**. Ensuite, la quantité de travail que chaque personne peut fournir est très limitée puisque nous ne sommes pas une véritable équipe de travail qui accorde au moins 5h par jour au projet, et ce 5 jours sur 7 ! Cette méthode est organisée autours d'évènements comme la mêlée quotidienne qu'il n'est alors pas possible de réaliser.

Pour tenter d'appliquer au mieux la méthode Scrum, nous faisons donc en sorte de planifier les évènements avec précision.

Nous travaillons donc de la façon suivante :

- Chaque sprint a eu une durée de **2 semaines**.
- 1 visioconférence de 1h a été réalisée au début de chaque sprint pour
  - clôturer le sprint précédent : **revue de sprint**
  - établir la **planification du sprint** suivant
- Concernant la **mêlée quotidienne**, . . . j'ai pas d'idée. . .

Le Backlog du produit est rédigé sur la vue Agile du projet (https://pp.irem.univ-mrs.fr/projects/application-tournoi/agile/board). On y voit une succession de post-it. Chaque bloc est une User Story : fiches détaillant une fonctionnalité attendue. Elles sont rédigées sous la forme "en tant que". . . "je veux que". . . "afin de". . .

C'est a partir de ce Backlog du produit que nous établissons de Backlog du sprint.

Sprint et visioconf

Dates et lieux des différents sprints

1. ~~((sprint du 29 novembre))~~, 7h15 (https://meet.jit.si/sprintduprojet)

— ~~((sprint du 16 décembre))~~, 14h30 (https://meet.jit.si/sprintduprojet)
— ~~((sprint du 31 décembre))~~, 08h30 (https://meet.jit.si/sprintduprojet)
— ~~((sprint du 20 janvier))~~, 19h00 (https://meet.jit.si/sprintduprojet)
— ~~((sprint du 3 février))~~, 17h00 (https://meet.jit.si/sprintduprojet)
— ~~((sprint du 2 mars))~~, 17h00 (https://meet.jit.si/sprintduprojet)
(sprint du 16 mars), 17h00 (https://meet.jit.si/sprintduprojet)

Godot

Prise en main Ci-joint le lien vers le tutoriel pour la prise en main de Godot : https://docs.godotengine.org/fr/latest/gettingstarted/stepbystep/yourfirst_game.html * + : Très bien décrit,très détaillé * - : Un peu éloigné du type de jeu qu'on veut mettre en place & sans base de programmation le tutoriel peut vite devenir compliqué (je ne suis pas arrivé au bout)

Exportation Pour l'exportation web/smartphone etc... https://docs.godotengine.org/fr/latest/gettingstarted/stepby_step/exporting.html

Pas mal de manip â faire a priori mais tout est géré par Godot.

Dépôt git

— Pour **consulter** le dépôt de fichier : c'est ici [8] sur le Redmine (cf onglet **Dépôt**).
— Pour **modifier** les fichiers du dépôt : c'est sur Github [9] (en effet, pour des questions de sécurité, l'hébergement du Redmine ne permet pas l'écriture sur le serveur très simplement).

Github

Sources d'information :

— Tutoriel (anglais) - comment collaborer avec github [10]
— Page Wikipedia - Git [11]
— Page Wikipedia - GitHub [12]

Pour travailler à plusieurs sur les mêmes fichiers, il semble indispensable d'utiliser un système de dépôt de fichiers. Cette technologie permet d'avoir l'historique de modification de tous les fichiers et de synchroniser les fichiers entre tous les utilisateurs.

Pour synchroniser (pull)

1. mettre les fichiers sur le serveur collectif **github** : https://github.com/Pskalou/tournoi
2. mettre en place la synchronisation d'un dossier de son ordinateur avec le dépôt : `git clone https://github.com/Pskalou/tournoi`
3. vérifier les mises à jour : `git fetch`
4. récupérer les mises à jour : `git pull`

Pour publier (push)

1. posséder un compte github
2. ajouter le compte github au projet
3. vérifier les mises à jours `git fetch` et les récupérer `git pull`
4. envoyer ses propres mises à jour : `git push`

8. <https://pp.irem.univ-mrs.fr/projects/application-tournoi/repository>
9. <https://github.com/Pskalou/tournoi>
10. <https://code.tutsplus.com/tutorials/how-to-collaborate-on-github--net-34267>
11. <https://fr.wikipedia.org/wiki/Git>
12. <https://fr.wikipedia.org/wiki/GitHub>

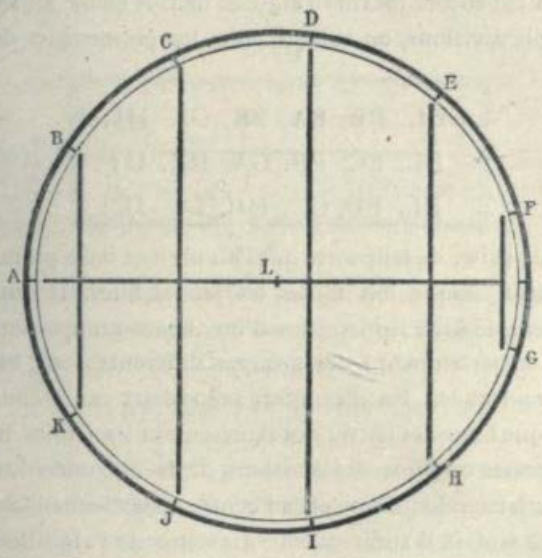
Une approche mathématique

LES PROMENADES DU PENSIONNAT.

Un pensionnat renferme un nombre pair de jeunes filles qui se promènent tous les jours deux par deux ; on demande comment il faut disposer les promenades de telle sorte qu'une jeune fille se trouve successivement en compagnie de toutes les autres, mais ne puisse s'y trouver plus d'une fois.

Supposons, pour fixer les idées, que le pensionnat renferme douze demoiselles que nous désignerons par les douze premières lettres de l'alphabet. Divisons une circonférence en onze parties égales; plaçons au centre l'une des lettres, L, et les autres aux points de division de la circonférence, dans un ordre quelconque ; puis, traçons les lignes droites de la *fig.* 89;

Fig. 89.



nous grouperons les jeunes filles deux par deux, pour la première promenade, en plaçant A avec L, et les autres suivant les lignes parallèles, de telle sorte que la première promenade se compose de six groupes :

I. AL, BK, CJ, DI, EH, FG.

Pour obtenir les groupes de la seconde promenade, nous con-

sidérerons l'ensemble des lignes droites de la figure comme une aiguille mobile que nous ferons tourner d'une division dans le sens des aiguilles d'une montre, pendant que les lettres resteront immobiles sur le contour; nous aurons ainsi pour la seconde promenade les groupes

II. BL, CA, DK, EJ, FI, GH.

Si l'on fait encore tourner l'aiguille dans le même sens de une, deux, trois divisions, on obtient pour les promenades des jours suivants:

III. CL, DB, EA, FK, GJ, HI;

IV. DL, EC, FB, GA, HK, IJ;

V. EL, FD, GC, HB, IA, JK;

et ainsi de suite, de telle sorte que l'on obtient onze promenades, comprenant chaque fois toutes les jeunes filles. Il nous reste à montrer que deux jeunes filles d'un même groupe d'une promenade appartiennent à des groupes différents pour toutes les autres promenades. En effet, nous avons deux cas à considérer, suivant que l'une des lettres qui représentent les jeunes filles occupe le centre ou l'une des divisions de la circonférence. Pour que deux lettres dont l'une est au centre appartiennent au même groupe, il faut et il suffit que le diamètre de l'aiguille mobile passe par l'autre, ce qui arrive une fois et une seule. Pour que deux lettres du pourtour soient dans le même groupe, il faut et il suffit que la droite qui les joint soit perpendiculaire à l'axe de l'aiguille; or les directions de l'axe ou celles des perpendiculaires sont différentes dans les onze positions de l'aiguille.

On peut encore résoudre le problème des promenades du pensionnat par l'emploi du Taquin. Soient AB... KL les cubes du Taquin élémentaire (*fig.* 90) servant à distinguer les jeunes

Fig. 90.

personnes. Convenons de lire chaque promenade en groupant les cubes deux à deux par colonnes verticales, c'est-à-dire pour la première promenade suivant l'ordre

AL, BK, CJ, DI, EH, FG.

Pour obtenir la seconde promenade, il suffit d'enlever le cube L, d'abaisser le cube A, d'avancer d'un rang vers la gauche les cubes de la ligne supérieure, de monter le cube G, puis de déplacer d'un rang vers la droite tous les cubes de la ligne inférieure; enfin de replacer le cube L sur la case vide, de telle sorte que L est revenu au même endroit.

On obtient les mêmes promenades que par l'emploi de l'aiguille; cela revient, en effet, à supposer immobile l'aiguille de la *fig.* 89, et à déplacer en même temps toutes les lettres de la circonférence de une, deux, trois, etc., divisions dans le sens opposé à celui de l'aiguille. Si l'on déforme le polygone de l'aiguille, en supposant égales toutes les cordes, ainsi que le périmètre de la circonférence, de telle sorte que celle-ci se trouve remplacée par le périmètre d'un rectangle, on retrouve identiquement le procédé du Taquin élémentaire.

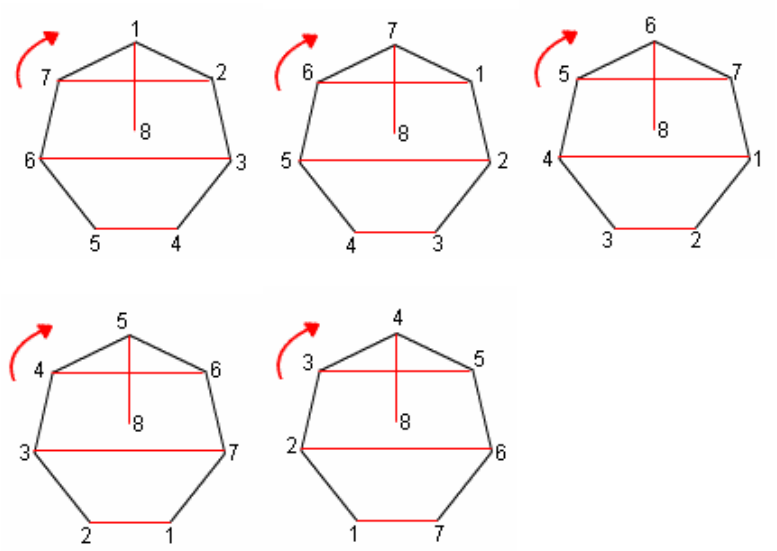
Nous avons supposé, dans les deux paragraphes précédents, que les jeunes pensionnaires se trouvent en nombre pair, le problème étant impossible pour un nombre impair. Cependant, dans ce dernier cas, on peut modifier l'énoncé de la manière suivante :

Un pensionnat renferme un nombre impair de jeunes filles qui se promènent tous les jours deux par deux, à l'exception d'une seule qui joue le rôle de monitrice ; on demande comment il faut disposer les promenades de telle sorte que chaque jeune fille se trouve une seule fois en compagnie de toutes les autres et une seule fois monitrice.

En admettant qu'il y ait onze jeunes filles, on divise la circonférence en onze parties égales, et l'on construit la *fig.* 89 en ne tenant pas compte de la lettre L placée au centre. On fait tourner l'aiguille de une, deux, trois, etc., divisions ; la monitrice est indiquée par la lettre placée à l'extrémité du diamètre de l'aiguille, tandis que les écolières se trouvent groupées deux par deux, aux extrémités des cordes perpendiculaires.

Le problème se déduit encore du problème des douze pensionnaires, en supprimant dans les promenades la lettre L. On peut d'ailleurs se servir encore du procédé du Taquin.

Remarque. — Si l'on écrit toutes les promenades des élèves, en les supposant soit en nombre pair soit en nombre impair, on forme évidemment le tableau complet de toutes les combinaisons des élèves deux à deux, ainsi qu'il est facile de le vérifier par un calcul direct.



One more rotation will bring the polygon back to its original position.

If A, B, C and D are the fields / pitches / courts, the schedule could look like this:

Round	A	B	C	D
I	7, 6	1, 5	2, 4	3, 8
II	6, 5	7, 4	1, 3	2, 8
III	5, 4	6, 3	7, 2	1, 8
IV	4, 3	5, 2	6, 1	7, 8
V	3, 2	4, 1	5, 7	6, 8
VI	2, 1	3, 7	4, 6	5, 8
VII	1, 7	2, 6	3, 5	4, 8

We can also rotate the teams around so that each team plays in every field / pitch / court at least once (at present team 8 always plays in D).

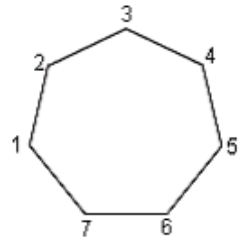
Round	A	B	C	D
I	7, 6	1, 5	2, 4	3, 8
II	6, 5	7, 4	1, 3	2, 8
III	**1, 8**	6, 3	7, 2	**5, 4**
IV	4, 3	5, 2	**7, 8**	**6, 1**
V	3, 2	4, 1	5, 7	6, 8

VI	2, 1	**5, 8**	4, 6	**3, 7**
VII	1, 7	2, 6	3, 5	4, 8

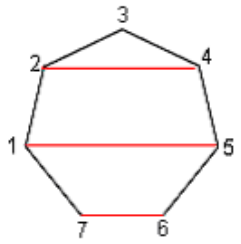
Round Robin scheduling: Odd number of teams.

Let N = number of teams in the tournament. There will be N rounds (since each team will play every other team once, and **will be idle for exactly one round**).

Let us work out the schedule for 7 teams, numbering the teams from 1 to 7. Draw a regular N-gon (heptagon for 7 teams). Each vertex represents one team.

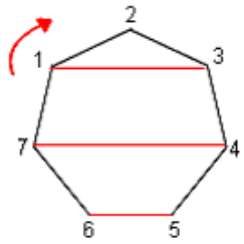


Draw horizontal stripes as shown below. The vertex that has been left out gives the idle team. Each segment represents teams playing each other in the first round.

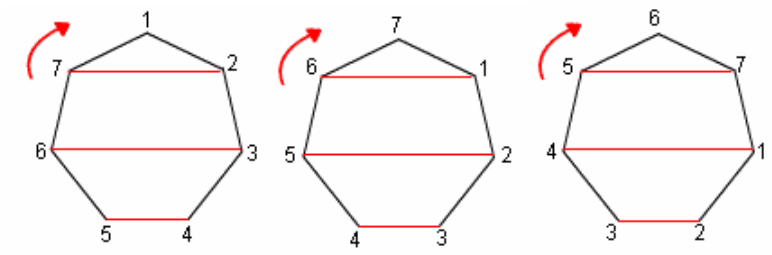


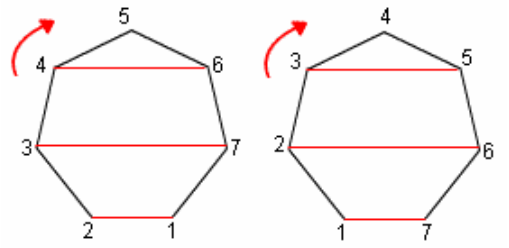
So (7, 6), (1, 5) and (2, 4) play in the first round.

Rotate the polygon 1/Nth of a circle (i.e. one vertex point.) The new segments represent the pairings for round two.



Continue rotating the polygon until it returns to its original position.





One more rotation will bring the polygon back to its original position. Therefore, the schedule could look like this:

Round	A	B	C
I	7, 6	1, 5	2, 4
II	6, 5	7, 4	1, 3
III	5, 4	6, 3	7, 2
IV	4, 3	5, 2	6, 1
V	3, 2	4, 1	5, 7
VI	2, 1	3, 7	4, 6
VII	1, 7	2, 6	3, 5

Why does this work?

The restriction that no vertex has more than one segment drawn to/from it ensures that no team is scheduled for more than one game in each round.

Restricting ourselves to horizontal stripes ensures that no segment is a rotation or reflection of another segment. This means that no pairing will be repeated in a future round.

Notice that in the case where N (no. of teams) was odd, by having only one idle team in each round, the tournament can be completed in the minimum number of rounds.

Arunachalam Y. is a member of the HeyMath! team

Générateur automatique de doc

La documentation est actuellement disponible sur le github : https://github.com/Pskalou/tournoi/tree/master/source/doc

Principe

La documentation est générée à partir de commentaires ajouté avant la déclaration des classes, méthodes et attributs.

Par exemple, voici le début de la classe `Score_manager.gd`

```
# Cette classe gère l'état du tournoi actuel.
# Dans un second temps, l'état est converti en score.
class_name Score_manager


# Barême pour une victoire.
# Par défaut égal à 10
#
# TODO : privatiser la propriété (set/get)
var win_points  :int= 10
```

Qui permet de générer le fichier https://github.com/Pskalou/tournoi/blob/master/source/doc/Score_manager.md [13]

How To

Nous avons installé le plugin gdscript-docs-maker

— https://github.com/GDQuest/gdscript-docs-maker

Sous `bash`, il faut se rendre dans le dossier `gdscript-docs-maker`. Puis lancer la commande `./generate_reference ./../source/ ./../source/doc/`. Les fichiers générés seront déplacés dans le dossier `source/doc`.

13. <https://github.com/Pskalou/tournoi/blob/master/source/doc/Score_manager.md>

<!– Auto-generated from JSON by GDScript docs maker. Do not edit this document directly. –>

Duel_button

Extends : TextureButton [14]

Description

Classe associée à la scène `duel_button.tscn`.

Affiche 2 boutons associés à 2 adversaires : le playerid et son opponentid. Un clic sur les boutons modifie l'état du match (`_state`) en fonction de la victoire, défaite ou non joué.

L'information se transmet via les signaux.

Functions

Type | Name — | — void | func setstate(newstate : int) -> void void | func initialisation(playerid : int, opponentid : int) -> void

Signals

— signal is_pressed() :

Method Descriptions

set_state

```
gdscript func set_state(new_state: int) -> void
```

Défini le nouvel état dans le match opposant playerid et opponentid.

— 0 : non joué
— 1 : player_id gagne
— 2 : player_id perd

initialisation

```
gdscript func initialisation(player_id: int, opponent_id: int) -> void
```

Initialise l'instance de duel_button avec les identifiants :

— player_id : joueur de référence
— opponent_id : adversaire

14. <../TextureButton>

<!– Auto-generated from JSON by GDScript docs maker. Do not edit this document directly. –>

Game_generator

Description

Cette classe à en charge la gestion de la répartition des matchs.

Pour chaque tour, l'algorithme génère la liste des matchs.

En notant n le nombre de participant, le modèle de donné utilisé est une matrice carrée M de dimension n. Pour deux participants d'identifiants les nombres entiers i et , M(i,j) est égal au tour durant lequel les participants i et j se rencontrent.

Functions

Type | Name — | — int | func opponent(playerid : int, roundindex : int) -> int

Method Descriptions

opponent

```
gdscript func opponent(player_id: int, round_index: int) -> int
```

Retourne l'identifiant de l'adversaire de 'playerid' lors du tour 'roundindex'. S'il n'y a pas d'adversaire (tour impair par exemple), la méthode retourne -1.

<!– Auto-generated from JSON by GDScript docs maker. Do not edit this document directly. –>

Round_buttons

Description

Cette classe a en charge la création et la gestion les boutons de duels. Elle fait le lien entre les clics et le score.

Functions

Type | Name — | — var | func kill() Array | func build(roundindex : int, currentnode : Node) -> Array

Method Descriptions

kill

```
gdscript func kill()
```

Fonction utilisée pour vider le tableau d'objets. Supprime tous les anciens boutons.

build

```
gdscript func build(round_index: int, current_node: Node) -> Array
```

Crée l'ensemble des boutons pour un round donné

entrée :

— round_index : le round concerné
— current_node : le node dans lequel on va instancier les boutons

sortie :

* tableau de tous les boutons

<!– Auto-generated from JSON by GDScript docs maker. Do not edit this document directly. –>

Score_manager

Description

Cette classe gère l'état du tournoi actuel. Dans un second temps, l'état est converti en score.

Properties

Type | Name — | — int | winpoints int | losepoints int | no_point

Functions

Type | Name — | — bool | func existresult(id1 : int, id2 : int) -> bool void | func setresult(playerid : int, opponentid : int, state : int) -> void int | func getresult(playerid : int, opponentid : int) -> int String | func scoreen_texte() -> String

Property Descriptions

win_points

```
gdscript var win_points: int = 10
```

Barême pour une victoire. Par défaut égal à 10. (Les points pour une victoire sont paramètrables dans le menu "options")

TODO : privatiser la propriété (set/get)

lose_points

```
gdscript var lose_points: int = 5
```

Barême pour une défaite. (Les points pour une victoire sont paramètrables dans le menu "options") Par défaut égal à 5 car nous souhaitons valoriser la participation à un match.

TODO : privatiser la propriété (set/get)

no_point

```
gdscript var no_point: int = 0
```

Barême pour une absence de match. Par défaut égal à 0.

TODO : privatiser la propriété (set/get)

Method Descriptions

exist_result

```
gdscript func exist_result(id1: int, id2: int) -> bool
```

Cette méthode permet de savoir si un match a eu lieu. Retourne vrai si les joueurs id1 et id2 se sont affrontés.

set_result

```
gdscript func set_result(player_id: int, opponent_id: int, state: int) -> void
```

Défini l'état d'un match entre playerid et son adversaire opponentid.

- state : 0 => non joué
- state : 1 => player_id gagne
- state : 2 => player_id perd

TODO : les états 0/1/2 doivent être internes à la classe. Les autres classes n'ont pas à connaître ce fonctionnement.

get_result

```
gdscript func get_result(player_id: int, opponent_id: int) -> int
```

Retourne l'état d'un match entre playerid et son adversaire opponentid.

- retourne : 0 => non joué
- retourne : 1 => player gagne
- retourne : 2 => player perd

TODO : les états 0/1/2 doivent être internes à la classe. TODO : utiliser des prédicats ? retourner uniquement les scores ?

score_en_texte

```
gdscript func score_en_texte() -> String
```

Transforme le tableau des états en score en utilisant le barême.

TODO : version CLI à passer en GUI