Sujet 6:

Rendu d'un logiciel livrable minimaliste.

(durée max: 4h)

➔ **Finalisez votre projet.**

Note:

- *Avez-vous suivit les règles de bonne programmation?*
- *Assurez vous que l'ensemble de vos fonctions sont correctement commentées.*
- *Assurez vous de respecter une programmation par contrats. Les fichiers .h sont ils commentés, les corps des fonctions respectent-ils les contrats?*
- *Gérez-vous correctement le mot clé const?*
- *L'ensemble des tests-ont ils été réalisés ? Tous les cas d'erreurs sont-ils testés?*
- *Votre* `Makefile` *est-il à jour, est-il lisible? Peut-on factoriser des arguments?*

➔ Synthétisez les tests effectués et expliquez votre démarche ainsi que vos choix dans un **compte-rendu** de 5 à 10 pages.

Enoncé, sujet 6

Travail supplémentaire.

Si vos déplacements fonctionnent et sont testés suffisamment.
Mettez en place la gestion de la fin de partie. Vous placerez pour cela, dans un fichier nommé `victoire_siam`, la gestion du joueur gagnant.
Vous créerez pour cela une fonction prenant en entrée au minimum un état du plateau après qu'une poussée ait eut lieu et la position d'une pièce à l'origine du déplacement. Cette fonction supposera qu'une poussée vient d'avoir lieu et que celle-ci a aboutit à la sortie d'un rocher. La fonction doit renvoyer une structure de type `condition_victoire` telle que le champ victoire soit mis à 1, et le joueur corresponde au joueur gagnant (relisez bien les règles pour connaître le joueur gagnant, ce n'est pas forcément la pièce à l'origine de la poussée). Vous ferez appel à cette fonction dans la gestion de l'API.

Si le jeu est entièrement jouable et que la condition de fin de partie est traitée, vous pouvez, au choix, réaliser différentes extensions parmi lesquels on pourra noter:

- Jeu paramétrable .
  - Modification des règles (telles qu'indiquée par les règles du jeu).
  - Taille du plateau différente
  - Un troisième (ou nième) joueur

  On pourra supposer que les paramètres sont soit écrit en dur dans le programme, soit paramétrable à partir d'un fichier texte ou xml (niveau plus avancé). (Pour éviter d'interférer avec le code existant passant les tests, créez un autre projet dans un répertoire spécifique).

- Mise en place d'une intelligence artificielle.
  Votre implémentation d'intelligence artificielle devra se lancer à l'aide d'un appel unique:
  `void ai_joue_tour(jeu_siam* jeu_courant);`
  qui jouera un tour du joueur courant de manière automatique.
  Si plusieurs intelligences artificielle sont proposées par différents binôme, un tournoi pourra être mis en place.

- Amélioration de l'interface graphique.
  - Amélioration du code de l'interface
  - Thème différent de jeu
  - Autre visualiseur (ex. visualiseur 2D avec gestion de la souris en SDL, Qt, etc).

- Création d'une librairie dynamique pour la gestion du jeu.
  - Réalisez les tutoriaux sur la mise en place d'une librairie dynamique.
  - Mettez la en place pour votre projet en rassemblant la gestion du jeu dans une librairie séparée de l'interface.