Modalités de rendu du projet et évaluations

Modalités du projet

Le projet est à traiter en groupes de 2 personnes au plus (les projets soumis par des groupes de 3 personnes ou plus ne seront pas acceptés).

Le travail doit être réalisé via un dépôt git pour chaque groupe. L'usage de ce gestionnaire de versions (VCS) permet d'éviter toute perte de donnée, et d'accéder si besoin à vos anciennes versions. Ceci nous permettra aussi de garder la trace et les dates de vos contributions, de vérifier la régularité de votre travail et son équilibrage.

Votre dépôt git doit être hébergé par le serveur GitLab de l'UFR d'informatique : https://gaufre.informatique.univ-paris-diderot.fr Voir le fichier GIT.md pour plus de détails sur son usage. Aucun autre code ne sera pris en compte, même hébergé par un autre serveur. Votre dépôt doit être rendu **privé** dès sa création, avec accès uniquement aux membres du binôme et aux enseignants de ce cours (au minimum l'utilisateur `letouzey` comme Maintaineur). Tout code laissé en accès libre sur Gaufre ou ailleurs sera considéré comme une incitation à la fraude, et sanctionné.

Il va de soi que votre travail doit être strictement personnel : aucune communication de code ou d'idées entre les groupes, aucune "aide" externe ou entre groupes. Nous vous rappelons que la fraude à un projet est aussi une fraude à un examen, passible de sanctions disciplinaires pouvant aller jusqu'a l'exclusion définitive de toute université.

Modalités de rendu

Ce projet est organisé en deux rendus, le premier le 15/12/2022 et le second le 10/01/2023. Nous prendrons en compte l'état de vos dépôts git à minuit à ces dates. Attention, nous considérerons uniquement la branche "master" de vos dépôts, pensez à fusionner vos éventuelles branches de développement dans "master". Attention à ce que vos dépôts soient bien lisibles par l'équipe enseignante (ajouter au moins l'utilisateur `letouzey` comme Maintainer) mais *pas* par tous (visibilité privée).

Rapport.txt

Un fichier texte Rapport.txt (ou Rapport.md) à la racine de votre dépôt devra nous fournir toutes les informations spécifiées ci-dessous, clairement séparées en sections distinctes. Ce rapport n'est pas une simple formalité : il sera pris en compte dans l'évaluation.

1. (Identifiants) Commencez par mentionner, pour chacun des membres du groupe : nom, prénom, identifiant sur GitLab, numéro d'étudiant.

2. (Fonctionnalités) Donnez une description précise des fonctionnalités implémentées par votre rendu - sujet minimal, extensions éventuelles, éventuellement parties non réalisées ou non encore fonctionnelles.

3. (Compilation et exécution) Documentez ensuite de façon précise la manière dont votre projet doit être compilé (normalement via dune) et exécuté (en donnant les options acceptées par votre

programme). Pour ce projet, aucune bibliothèques externes n'est autorisé a priori. Nous contacter si cela vous semble problématique.

4. (Découpage modulaire) Donnez une description des traitements pris en charge par chaque module (.ml) de votre projet. Précisez le rôle et la nécessité de chaque module ajouté au dépôt initial.

5. (Organisation du travail) Cette partie est plus libre dans sa forme. Indiquez la manière dont les tâches ont été réparties entre les membres du groupe au cours du temps. Donnez une brève chronologie de votre travail sur ce projet au cours de ce semestre, avant et après le confinement.

6. (Misc) Cette partie est entièrement libre : remarques, suggestions, questions...

Critères d'évaluation

Cette grille n'est donnée qu'a titre indicatif. Elle récapitule tout ce que nous attendrions d'un projet "parfait" - faites simplement au mieux pour vous rapprocher de ce projet idéal.

1. Rapport.txt (ou Rapport.md) complet et bien rédigé.

2. Usage approprié et régulier de GitLab. Commits (ou branches et merge-requests) correspondant à des tâches précises et effectués par la personne ayant effectué la tâche, équilibrage de l'usage entre les membres du groupe.

3. Découpage du code en modules consistant. Un rôle précis attribué à chaque module, masquage approprié des éléments utilitaires dans les .mli.

4. Projet minimal pleinement et correctement réalisé, et de manière flexible.

5. Choix de nouveaux types ou de nouveaux modules justifiés et appropriés.

6. Écriture fonctionnelle pure privilégiée. Éviter autant que possible les références, tableaux, boucles et enregistrements à champs mutables. Justification précise de chaque usage restant de ces éléments impératifs, via un commentaire détaillant pourquoi une solution fonctionnelle pure était impossible ou trop lourde à cet endroit.

7. Qualité de l'écriture fonctionnelle. Usage approprié de fonctions d'ordre supérieur (les "fonctionnelles"), de la récurrence terminale, des fonctions de la bibliothèque standard.

8. Qualité d'écriture du code. Pas de redondances, aucun copier-coller, code bien factorisé, choix de noms de fonctions parlants.

9. Code pleinement nettoyé, commenté et mis en page lors du rendu final. Pas de code résiduel laissé en commentaire. Des lignes limitées à 80 caractères. Des commentaires concis mais suffisants, souvent mais pas trop.

10. Eventuellement, des extensions pertinentes et intéressantes - ce point est moins important que les précédents.

Sur les points (8) et (9), rappelons-le : nous relirons (très attentivement) votre code. Facilitez au mieux cette lecture.

Cordialement, L'équipe pédagogique de pf5.