IUT, Université Jean Monnet, Saint-Étienne

SAé 2 Info2 : Projet d'Algorithmique et Programmation informatique

DOCUMENT DE SUIVI ET D'ÉVALUATION

Année 2021-2022

Fabien Momey, Bruno Bernard, Paul Grandamme, Franck Gérossier, Vincent Grosso, Cyril Mauclair, Thierry Lagniet

OBJET DU PRÉSENT DOCUMENT :

Ce document sert de référence pour le suivi et l'évaluation des groupes projet par leurs enseignants-évaluateurs respectifs.
Il contient notamment, pour chaque Jalon-Livrable, les critères d'évaluation et barèmes associés, afin d'être homogènes dans les points à vérifier et à évaluer (mis en regard des compétences).

RESSOURCES À DISPOSITION :

Les ressources dédiées aux enseignants-évaluateurs pour assurer le suivi se trouvent sur **IUTSE-GEII_S1_M1103_INFO1**. Un onglet **SAé2 Info2 - Ressources Enseignants-Évaluateurs** a été créé dans lequel vous retrouverez le **présent document** ainsi que des **ressources dédiées** *(à consulter régulièrement)*.

Résumé des notions à maîtriser par les étudiants :

- **Connaissances techniques :**
  - **Bases d'algorithmique et programmation informatique.**
  - Variables, types, structures de contrôle (conditions, boucles).
  - Tableaux, **structures**, **pointeurs**.
  - **Fonctions** : appel de fonctions, définition de fonctions, notion de **passage par valeur (entrées en « lecture seule » E)** et **passage par adresse (entrées/sorties en « lecture/écriture » E/S)**.
  - ***Allocation dynamique de mémoire.***
- **Mise en œuvre d'une solution algorithmique complète exploitant ces connaissances.**
- **Implémentation d'une solution algorithmique complète dans un langage de programmation (C++).**
- Structure d'un programme informatique :
  - fichier source principal "`main.cpp`".
  - fichiers d'en-tête "`.h`" : prototypes des fonctions et déclaration des structures.
  - fichiers sources "`.cpp`" : définition des fonctions.
- Écriture du code et compilation dans un environnement de développement intégré (IDE) :
  - Code::Blocks.
- **Vérification du code** à l'aide de **procédures de test** et **débogage**.

COMPÉTENCES & APPRENTISSAGES CRITIQUES MIS EN JEU :

C1-N1-CONCEVOIR

C1-N1-AC1 - Produire une analyse fonctionnelle d'un système simple

- Pour toutes les étapes de ce projet, les étudiants doivent proposer une solution algorithmique complète et détaillée (découpage fonctionnel, algorithme principal, procédures de test), indépendamment du langage de programmation choisi, qui respecte les contraintes imposées dans le cahier des charges.

C1-N1-AC2 - Réaliser un prototype pour des solutions techniques matériel ou logiciel

- Les étudiants doivent ensuite implémenter dans une application console C++ chaque étape du projet, en traduisant fidèlement la solution algorithmique élaborée.

C1-N1-AC3 - Rédiger un dossier de fabrication à partir d'un dossier de conception

- Les étudiants doivent rédiger une **documentation complète** de leur programme, sous forme algorithmique et illustrée par des exemples issues de leur prototypage en C++. **Ce rapport doit pouvoir servir de référence à un programmeur tierce qui souhaiterait implémenter leur solution dans un autre langage de programmation**.
- Concernant leur programme C++, ce dernier devra aussi être **rigoureusement commenté**, et notamment toutes les fonctions avec leurs **{ Rôle, Entrées, Entrées/Sorties, Sortie }** clairement définis.

C2-N1(N3)-VERIFIER

C2-N1-AC1 - Appliquer une procédure d'essai

- Pour chaque étape du projet, lors de la phase de conception algorithmique et d'implémentation, les étudiants doivent réfléchir à une batterie de tests unitaires à mettre en place - **sous forme de procédures à implémenter ou à réaliser avec l'outil débogueur par exemple** - pour valider chaque élément de leur solution (programme principal, fonctions, structures, etc.). **Toutes les procédures de test devront être rigoureusement décrites et documentées dans leur rapport. Si ce sont des procédures de test implémentées, elles devront être intégrées au code (dans une librairie dédiée par exemple) et rigoureusement commentées**.

C2-N1-AC2 - Identifier un dysfonctionnement

- L'application des procédures d'essai que les étudiants auront définies doivent leur permettre de vérifier le bon fonctionnement de leur programme et d'identifier des dysfonctionnements éventuels.

C2-N1-AC3 - Décrire les effets d'un dysfonctionnement

- En cas d'identification d'un dysfonctionnement, les étudiants doivent garder trace de leurs tests et analyses en rédigeant une petite description dans leur journal de bord.

C2-N3-AC2 - Proposer une solution corrective à un dysfonctionnement

- Bien entendu, tout dysfonctionnement de leur programme devra être corrigé et l'application de leur procédure d'essai devra permettre de valider la modification. Cette dernière sera également notifiée dans leur journal de bord.

JALONS & LIVRABLES :

Tout au long de l'élaboration des phases du projet, un certain nombre de jalons seront à respecter, avec des dates fixées et des livrables à fournir (codes, journaux de bord, état de la documentation). **Ces livrables seront à déposer sur IUTSEGEII_S1_M1103_INFO1 dans l'onglet dédié SAé2 Info2 - Livrables pour le projet d'informatique**.

Les jalons sont indiqués par le cartouche suivant à la fin des phases concernées :

Jalon & Livrable n°i

Chacun de ces jalons sera évalué par les enseignants-évaluateurs, et des entrevues pourront être prévues. **Voici les dates butoires à respecter (les espace de dépôt sur IUTSEGEII_S1_M1103_INFO1 seront ouverts ces jours) :**

Jalon & Livrable n°1 ⇒ 01/04/2022

~~**Jalon & Livrable n°2 ⇒ 15/04/2022**~~

Jalon & Livrable n°2 ⇒ 06/05/2022

~~**Jalon & Livrable n°3 ⇒ 06/05/2022**~~

Jalon & Livrable n°3 ⇒ 24/05/2022

~~**Jalon & Livrable n°4 ⇒ 24/05/2022**~~

Jalon & Livrable n°4 OU PLUS ⇒ 10/06/2022

Jalon & Livrable n°5 à atteindre si possible au 10/06/2022

CONVENTIONS DE CODAGE À RESPECTER :

- Uniquement des caractères alphanumériques pour le nommage des éléments de code (variables, fonctions, etc.). Pas de caractère accentué, pas d'espace, pas de point, seul le caractère « underscore (tiret du 8) » est toléré.
- les **variables** doivent être nommées en minuscule.
  Exemple :
  ```
  int mavariable ;
  ```
- les **constantes** doivent être nommées en majuscule.
  Exemple :
  ```
  const int MACONSTANTE ;
  ```
- les **structures** doivent être nommées par un mot commençant par une majuscule.
  Exemple :
  ```
  struct Mastruct { } ;
  ```
- les **champs** de structures doivent être nommées en minuscules et commencer par le préfixe `m_`.
  Exemple :
  ```
  int m_champ1 ;
  ```
- les **fonctions** doivent nommées par un ou plusieurs mots attachés chacun commençant par une majuscule.
  Exemple :
  ```
  void MaFonctionSuperGeniale() ;
  ```
- les **entrées** et **entrées/sorties** des fonctions doivent être nommées en minuscule et commencer par le préfix `input_` pour les entrées et `output_` pour les entrées/sorties.
  Exemple :
  ```
  void MaFonctionSuperGeniale(const int* input_var1, float* output_var2) ;
  ```
  Note : les entrées (lecture seule) doivent être précédées du mot-clé `const` *pour bien les identifier comme étant en lecture seule ...*

DANS LA SUITE DE CE DOCUMENT, POUR CHAQUE PHASE, ON DONNE LES POINTS À ÉVALUER ET LES CRITÈRES ASSOCIÉS LORS DE L'ÉVALUATION DU JALON CONCERNÉ.
CES CRITÈRES SONT RETRANSCRIS DANS LE Tableau de suivi et d'évaluation. CHAQUE CRITÈRE CORRESPOND À UNE COLONNE À NOTER AVEC UN POURCENTAGE TRADUISANT LE "NIVEAU DE RÉUSSITE" DE L'OBJECTIF VISÉ.

Phase 1 : création du conteneur `Vaisseau` générique

↬ Points à vérifier pour respecter le **cahier des charges** :

- Il faut créer une **structure** `Vaisseau` avec 4 champs : 2 chaînes de caractères et 2 entiers. Toute autre solution peut être acceptable en partie si elle est correctement justifiée et documentée mais à terme, ne pas utiliser une structure risque de rendre le code compliqué à gérer.
- La librairie de fonctions doit permettre de manipuler facilement des **variables de type** `Vaisseau`. Le vaisseau "appelant" doit être passé en premier paramètre de la fonction en tant que pointeur. *Le grosse difficulté va être la gestion des pointeurs sur structure dans les fonctions ⇒ déréférencement puis accès au champ de la structure :* `output\_vaisseau->m_nom` ou `(*output\_vaisseau).m_nom`
- En tout il devrait y avoir 1 fonction **Afficher**, **AffecterCarac** (affectation de tous champs dans la même fonction), 4 fonctions **Affecter*Champ*** (1 par champ), 4 fonctions **Renvoyer*Champ*** (1 par champ). *Attention au respect de la "généricité" des fonctions par les étudiants qui ont tendance à "rajouter des fonctionnalités" dans ces fonctions qui doivent avoir un rôle unique. Ces fonctions doivent constituer les outils de base pour créer ensuite des fonctions plus complexes utilisant ces dernières.*
- L'affectation des chaînes de caractères doit se faire préférentiellement à l'aide de la librairie `string.h` (fonction **strcpy**). Attention également aux fonctions renvoyant des chaînes de caractères.
- Les fonctions doivent nommées suivant une "nomenclature" commune et claire. L'idéal est de créer une librairie dédiée (`.h` et `.cpp`) avec un nommage clair : par exemple `lib_vaisseau`.
- *Il faut bien comprendre que dans cette phase, on implémente surtout l'architecture du jeu et non le jeu lui-même, c'est-à-dire toutes les fonctionnalités qui gérer des **"objets" `Vaisseau`**, à la manière du **paradigme orienté objet**.*
- **Interdiction d'utiliser des classes ! ! ! Les étudiants ne connaissent pas ce paradigme et l'aborderont en deuxième année.**

Phase 2 : « duel de vaisseaux »

↬ Points à vérifier pour respecter le **cahier des charges** :

- Il y a une fonction à rajouter à la librairie `lib_vaisseau` : **SubirDegatsCoque** qui permet d'ôter un nombre entier de points de vie au vaisseau "appelant". *Il faut penser à gérer si la résistance de coque tombe en-dessous de 0.*
- Le programme de duel entre 2 vaisseaux doit utiliser la librairie `moteur_de_jeu`.
- *Attention dans cette phase à ce que les étudiants respectent bien la simplicité de l'algorithme demandé et "ne partent pas dans tous les sens".*

Jalon & Livrable n°1 ⇒ 01/04/2022

Pour ce jalon, Les étudiants doivent fournir dans une archive `ZIP` intitulée `Jalon_1_n°binome.zip` :

↬ leur code fonctionnel et correctement documenté.

↬ leurs journaux de bord, personnels et de groupe, au format numérique PDF.

↬ **On ne regarde pas encore la partie "algorithmes" qui sera évaluée à partir du Livrable n°2.**

Si ils ne sont pas parvenus jusqu'à ce point, ils doivent rendre **leur projet en l'état** à la date prévue.

CRITÈRES D'ÉVALUATION :

Ces critères sont retranscris dans le **Tableau de suivi et d'évaluation**. Chaque critère correspondant à une colonne dans lequel

1. C1-N1 (**éval. de groupe**) :
   - ➟ respect du cahier des charges.
   - ➟ qualité d'implémentation de la structure + librairie `lib_vaisseau` (phase 1) : code clair, commenté.
   - ➟ qualité d'implémentation du programme de duel de vaisseaux (phase 2) : code clair, commenté.
2. C2-N1 (**éval. de groupe**) :
   - ➟ implémentation de procédures ou programmes de tests unitaires clairs et commentés.
3. Implication de travail du groupe C1-N1 / C2-N1 (**éval. de groupe toutes compétences**) :
   - ➟ tenue du journal de bord **de groupe**.
   - ➟ implication du groupe dans le projet : questions aux tuteurs, demandes de rendez-vous.
   - ➟ C1-N1 ⇒ les choix faits, la répartition des tâches, les tâches réalisées, les notes techniques.
   - ➟ C2-N1 ⇒ la description des tests réalisés, l'identification de dysfonctionnements et solutions correctives.
4. Implication de travail personnel C1-N1 / C2-N1 (**éval. individuelle toutes compétences**) :
   - ➟ tenue du journal de bord **personnel**.
   - ➟ implication personnelle de l'étudiant dans le projet : travail, participation dans les rendez-vous tuteurs.
   - ➟ C1-N1 ⇒ les tâches réalisées, les notes techniques.
   - ➟ C2-N1 ⇒ la description des tests réalisés, l'identification de dysfonctionnements et solutions correctives.