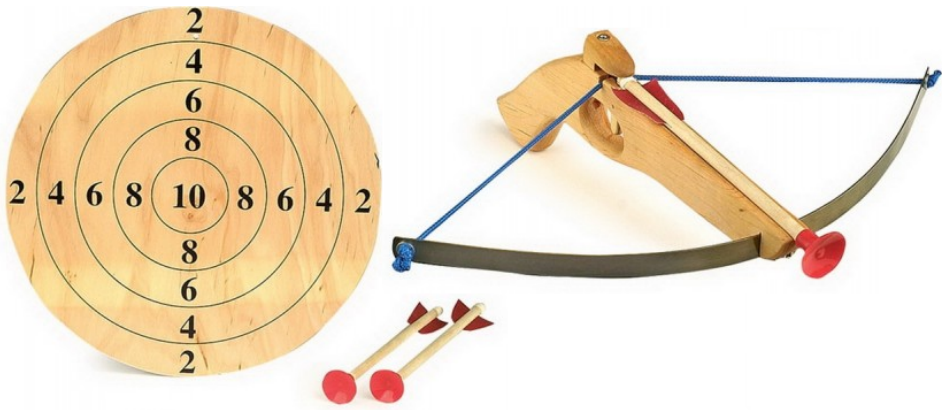
ENSICAEN
ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN
& CENTRE DE RECHERCHE

Génie Logiciel et Patron de Conception

Regis.Clouard@ensicaen.fr
Sylvain.Vernois@ensicaen.fr
Julien.Mahier@ensicaen.fr

Jeu en réseau de tir à l'arbalète



Ce projet vise la mise en œuvre des techniques de génie logiciel pour la réalisation d'un projet concret. Une attention particulière sera donc accordée à la qualité de la conception de votre logiciel. Il est difficile dans le cadre de séances de travaux pratiques de proposer un projet d'une taille suffisante pour se rendre compte qu'une conception mal structurée conduit inévitablement à un logiciel rigide, fragile et immobile qui vire à l'usine à gaz. Compte-tenu des contraintes de temps, nous ne pouvons que proposer un projet modeste et qui pourrait être conçu sans compétences particulières en génie logiciel. Néanmoins, dans un but pédagogique, il vous est demandé de bien vouloir jouer le jeu de la gestion de projet et de livrer un logiciel démontrant des qualités de robustesse, extensibilité et réutilisabilité, accompagné de ses tests unitaires. Un projet ne présentant pas toutes les fonctionnalités demandées mais conçu selon les règles de l'art sera beaucoup mieux évalué qu'un projet complètement fonctionnel mais bricolé.

1 Le sujet

Vous venez de créer une entreprise dans le domaine du jeu vidéo avec une offre spécialisée dans les jeux en réseau. Un client (rôle joué par votre encadrant de TP) qui représente une célèbre entreprise de fabrication d'arbalète de compétition vous demande de concevoir pour leur activité de promotion un jeu vidéo de tir à l'arbalète en réseau.

1.1 Description du jeu de tir à l'arbalète

Une compétition de tir à l'arbalète fait s'affronter deux archers en deux fois 10 volées de 3 flèches soit 60 flèches. La cible est découpée en 10 cercles concentriques. La valeur 1 est attribuée au premier cercle et la valeur 10 au centre. Toute flèche en dehors de la cible ne sera pas comptabilisée. Le comptage des points est la somme des valeurs des cercles atteints par chaque flèche. Plusieurs distances et tailles de cible existent, par exemple 90 m sur une cible de 122 cm de diamètre pour le tir en extérieur dit FITA.

1.2 Exigences du client pour le jeu

Les exigences du client par rapport au logiciel de jeu de tir à l'arbalète en réseau sont les suivantes :

1. Chaque joueur accède au jeu via un ordinateur en réseau. Un des deux joueurs prend le rôle du serveur où vient se connecter l'autre participant en tant que client.
2. Une partie consiste en 1 volée de 10 flèches. Pour tirer sa flèche, le joueur doit choisir le vecteur vitesse de la flèche.
3. Chaque joueur doit avoir un affichage de sa propre zone de tir et l'affichage rapprochée de la cible de son adversaire avec les impacts de flèches.

4. L'affichage de la zone de tir de chaque joueur doit montrer une vue depuis l'arbalète, une vue de côté et une vue rapprochée de la cible. Il doit être possible à tout moment de passer d'une vue à une autre mais une seule est affichée à chaque instant.
5. L'application est responsable des lancers, de l'affichage du jeu, de l'affichage du score et de l'application du règlement.

2 Organisation du projet

La réalisation et l'organisation du travail sont soumises à des contraintes que chaque groupe devra respecter.

2.1 Contraintes sur la réalisation du projet

- ☑ **UML :** Les modélisations seront réalisées avec le langage UML. La rédaction des diagrammes UML devra se faire avec un atelier de génie logiciel tel que ArgoUML (installé sur les machines) ou un autre atelier de votre choix.
- ☑ **JAVA** : Le langage d'implémentation sera le Java.
- ☑ **JAVAFX** : L'interface graphique se basera intégralement sur la bibliothèque JavaFX.
- ☑ **GIT** : La gestion des versions sera réalisée avec Git. Le dépôt central sera localisé sur gitlab.ecole.ensicaen.fr et le développement quotidien des développeurs sera géré par Git sous Netbeans. La récupération des livrables finaux sera faite par votre encadrant directement à partir du dépôt Gitlab dont vous lui communiquerez l'adresse.
- ☑ **TDD** : La conception du logiciel doit être faite avec un processus de développement dirigé par les tests (TDD). Pour cela, vous devez utiliser JUnit.

- ☑ **Patrons de conception** : L'architecture devra bien entendu faire appel aux patrons de conception, dont la pertinence devra être justifiée dans le rapport et la soutenance.

2.2 Méthode de gestion de projet

Vous devrez former des équipes de 8 personnes. Le projet se déroule sur 8 séances de 2 heures.

Il vous est demandé d'utiliser une méthode de gestion de projet à base de **cycles de développement itératifs**. Dans notre cadre, une itération correspond à deux séances. Une itération commence par la définition et la sélection d'un sous-ensemble de tâches à réaliser (de granularité assez fine) qui correspondent à des fonctionnalités du futur logiciel. Ce choix doit être fait en relation avec votre client.

Au cours de l'itération, les tâches sont réalisées par les développeurs de l'équipe.

Chaque fin d'itération correspond à la **démonstration** d'un prototype opérationnel montrant à votre client les fonctionnalités développées jusque-là. Cette démonstration est un moment d'évaluation de votre capacité à concevoir le logiciel demandé. Elle doit donc être préparée et réalisée de manière formelle. Le trop fameux « effet démo » est ici une erreur qui sera sanctionnée.

2.3 Rôles et fonctions dans chaque équipe

Chaque équipe doit répartir les responsabilités en nommant des personnes aux postes suivants :

- ☑ **Chef de projet** : son rôle est de coordonner les activités de l'équipe, de superviser les travaux et d'interagir avec le client ;
- ☑ **Architecte** : son rôle est de concevoir l'architecture du

logiciel, d'identifier les points fonctionnels et les principales abstractions ;

- ☑ **Ingénieur de conception** : son rôle est de concevoir l'architecture des classes et de l'organisation de la conception en paquets et de prendre les décisions tactiques relatives à cette réalisation ;
- ☑ **Développeur** : son rôle est de concevoir les classes et leur organisation pour réaliser les fonctionnalités qui lui sont attribuées. Avant de développer une classe, le développeur définit les tests qui lui semblent le mieux évaluer l'effectivité de la classe. Seules les classes réellement graphiques ne sont pas accompagnées de tests unitaires.
- ☑ **Responsable de version** : cet ingénieur est le responsable du dépôt sur Gitlab. C'est lui qui fait l'intégration continue du travail des développeurs et réalise le prototype de démonstration qui sera présenté à l'issue de chaque itération. Il est aussi le garant de la propreté du code et de la présence des tests dans les contributions des développeurs.

Il est à noter qu'une même personne peut endosser plusieurs rôles et qu'un rôle peut être assumé par plusieurs personnes.

2.4 Livrables

Différents documents, regroupés dans un rapport, devront être produits. Le rapport devra au moins contenir les documents suivants :

- ☑ **Analyse des risques** : la liste des risques majeurs de ne pas parvenir à développer le logiciel avec les moyens d'y remédier.
- ☑ **Analyse des besoins** : les cas d'utilisation du logiciel,

certains détaillés si nécessaire.

- ☑ **Conception UML** : La modélisation du logiciel en UML où les patrons de conception seront mis en exergue.
- ☑ **Diagramme de paquet** : Il décrit vos choix d'organisation de la conception en paquets ;
- ☑ Code source : Le projet Netbeans avec le code Java du logiciel avec le code des tests unitaires.

Cette liste n'est en aucun cas exhaustive et pourra être complétée avec d'autres documents qui vous paraîtront pertinents dans la limite du raisonnable (au sens Agile du terme).

2.5 Soutenance

À l'issue des huit séances de travail, une soutenance est organisée pour la validation finale du logiciel. La soutenance devra au moins faire apparaître clairement :

- ☑ L'architecture de votre système.
- ☑ La réponse de votre conception à la robustesse, la réutilisabilité et la maintenance.
- ☑ L'organisation de l'équipe et la gestion de projet conduite.

La soutenance sera conclue par une démonstration finale.

Si la planification de votre projet s'est avérée irréaliste et que l'avancement réel en est très éloigné, il vous est demandé de prendre du recul et d'analyser les causes du dysfonctionnement. Le travail d'une équipe qui saurait analyser des dysfonctionnements et proposer des solutions sera bien mieux apprécié que celui d'une équipe qui chercherait à dissimuler d'éventuelles déconvenues derrière un produit fini, même spectaculaire.

Lors de la présentation orale, tous les membres de l'équipe

doivent présenter l'aspect du projet dont ils sont responsables. Il ne sera pas accepté qu'un seule personne présente l'ensemble du projet de l'entreprise.

2.6 Évaluation

La note de TP de chaque membre d'une équipe sera la moyenne des notes obtenues pour chacune des quatre évaluations :

- ☑ la pertinence de la conception ;
- ☑ la « propreté » du code et des tests ;
- ☑ la qualité de la soutenance ;
- ☑ la perception du client sur votre capacité à produire le logiciel demandé.

3 Documentation fournie

Sur la plateforme pédagogique vous trouverez les documentations suivantes :

- ☑ Le travail de développement de projet avec Git (Git Workflow).
- ☑ Un exemple introductif de code JavaFX.
- ☑ Les tests unitaires avec JUnit.
- ☑ Un exemple de compte-rendu de projet.