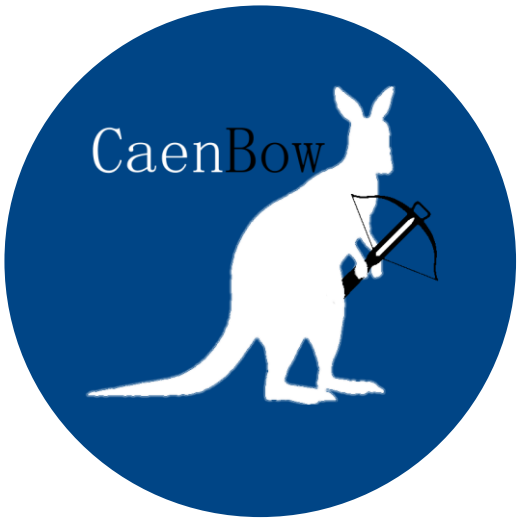


ENSICAEN
École Nationale Supérieure d'Ingénieurs de Caen

CaenBow, jeu de tir à l'arbalète en réseau

Rapport du projet Génie Logiciel

Guaquière Thibaut
Jorand Jean Baptiste
Durand Valentin
Moraine Guillaume
Ngalula Jean-Paul
Elyadari Sami
Djiomo Loïk

Début du projet : 30 novembre 2016
Fin du projet : 11 janvier 2017

Encadré par : Mahier Julien

ENSICAEN
6, boulevard Maréchal Juin – CS 45 053 – F- 14050 Caen Cedex 4
Tél. +33 (0)2 31 45 27 50
Fax +33 (0)2 31 45 27 60

Table des matières

Description du projet

Ce projet vise à mettre en œuvre des techniques de génie logiciel, tout en gérant un projet de grande envergure. Pour ce faire, il nous à été demandé d'implémenter un jeu d'arbalète en réseau en suivant les exigences de notre client.

Ce projet s'est conduit du 30 novembre 2016 au 11 janvier 2017 avec huit séances de deux heures en présence de notre client - professeur encadrant Mahier Julien.

Les exigences de base du projet consistaient à créer une compétition de tir à l'arbalète en réseau, où pouvaient s'affronter deux joueurs en deux fois deux volées de dix flèches. La cible devait être divisée en dix zones afin de pouvoir compter un score de 0 à dix lors du tir, zéro étant le cas ou le flèche n'atteint pas celle-ci. Chaque joueur doit pouvoir choisir le vecteur vitesse de son tir.
L'un des joueurs doit prendre le rôle du serveur et l'autre doit venir s'y connecter.

Mais en plus de devoir répondre à ces exigences, nous avons dû nous soumettre à celles de notre client.

Exigences client

Afin de répondre aux exigences de notre client, nous avons du implémenter une version solo de notre jeu d'arbalète afin que le joueur puisse s'entraîner si il le souhaite.

Notre version multi-joueurs devait permettre le jeu en ligne jusqu'à quatre joueurs par partie.

Toutes nos vues devaient pouvoir s'adapter à la taille de l'écran de l'ordinateur sur lequel était lancé le jeu.

Nous devions prendre en compte le vent dans nos calculs de trajectoire mais aussi permettre un jeu sans vent.

Chaque joueur devait avoir le choix entre trois arbalètes de puissance et de précision différentes.

Les arbalètes devaient posséder différents cran de puissance afin de pouvoir augmenter ou diminuer la puissance de celles ci.

Le choix du vecteur vitesse lors du tir devait se faire à la souris, le joueur devait pouvoir choisir ses angles de tirs verticaux et horizontaux par ce biais.
Le nombre de volées devait pouvoir être modifié en début de partie par exemple une partie pourrait se dérouler en quatre volées de dix flèches.

Lors de la partie le joueur devait être capable de regarder non seulement la cible de ses adversaires mais aussi la sienne dans un espace dédié.

Tâches réalisées

Au vue de toutes ces exigences nous avons décidé de scinder le projet en cinq grandes parties : la conception de l'architecture de notre logiciel, la création des différentes vues, l'implémentation du modèle et du contrôleur de la partie, la vérification de celui par test unitaire et enfin l'implémentation du réseau qui permettait de jouer en multi-joueurs.

1. Conception architecture

La conception d'architecture du système a été cruciale, puisque que toute l'implémentation du logiciel de jeu se basait dessus. Un premier modèle a donc été établi, celui ci se mettant par la suite à jour au vue des nouvelles solutions que nous pouvions apporter lors de la conception du logiciel.

L'architecture finale de notre logiciel sera développée dans la suite de ce rapport.

2. Conception des vues

La conception des vues à elle même été subdivisée en trois sous-modules : les vues d'initialisation de la partie, la vue principale, et enfin la vue finale.
Les différentes vues d'initialisation permettaient au joueur de choisir entre partie solo et partie multi-joueurs.

➔ Vues d'initialisation de la partie

Une première vue permet au joueur de choisir entre une partie solo ou une partie multi-joueurs (figure 1).

Figure 1 : Écran d'accueil

Dans le cas d'une partie solo, une nouvelle vue à été créée afin que le joueur puisse choisir les paramètres de sa partie (figure 2). Le choix de la force du vent et de la distance à la cible sont gérés par paliers permettant de déterminer aléatoirement les coordonnées du vecteur vent (x,y) et la distance d.

Pour le vent nous avons décidé d'utiliser les paliers :

- nul avec (x, y) = (0,0)
- faible avec x et y compris entre 1 et 15 $m.s^{-1}$
- moyen avec x et y compris entre 16 et 45 $m.s^{-1}$
- fort avec x et y compris entre 46 et 100 $m.s^{-1}$

Et pour la distance nous avons utilisé :

- proche avec d compris entre 10 et 50 m
- moyen avec d compris entre 51 et 125 m
- éloigné avec d compris entre 126 et 200 m

Figure 2 : Écran de paramétrage de partie solo

Si le joueur choisi de joueur une partie multi-joueurs, un nouvel écran lui demande alors si il choisit d'être le client ou l'hôte de cette partie.

Dans le cas où il choisi d'être le client, il accède à une nouvelle vue lui permettant de choisir le serveur de jeu sur lequel il souhaite se connecter (figure 4). Enfin une dernière vue lui permet d'indiquer son pseudo.

Figure 4 : Écran client

Dans l'autre cas le joueur hôte accède à un écran lui permettant de paramétrer la partie multi-joueurs (figure 5).

Figure 5 : Écran hôte, paramétrage de la partie multi-joueurs.

➔ Vue principale de la partie

La vue principale de la partie peut se découper en deux blocs : une zone dédiée à la visée et au jeu, et une autre permettant d'afficher les cibles des différents joueurs et de changer d'arbalète (figure 6).

Dans le bloc réservé au jeu, un cadre permet au joueur de connaître toutes les informations nécessaires sur le tir en cours.

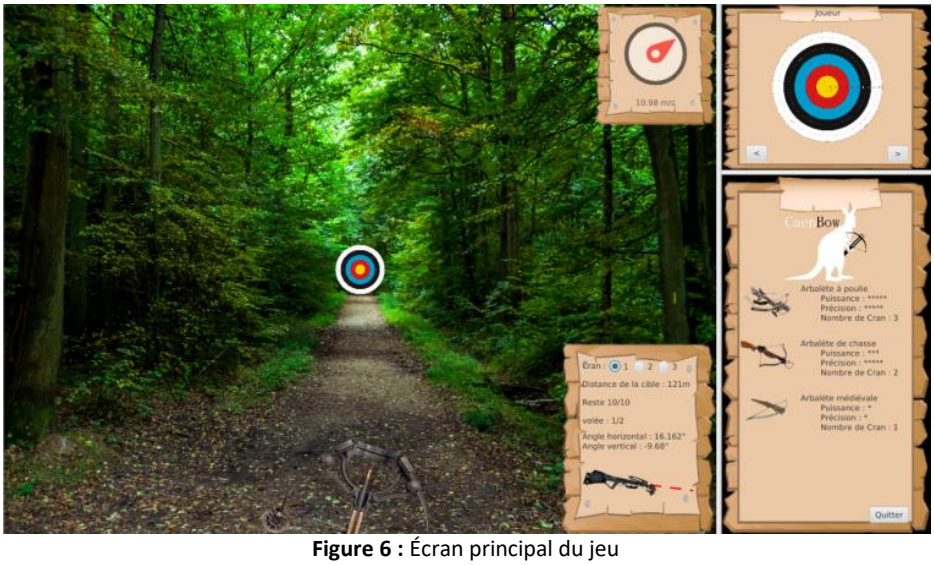


Figure 6 : Écran principal du jeu

➔ Vue finale de la partie

La vue finale permet au joueur de connaître le nom du vainqueur de la partie et affiche également flèche par flèche le résultat de chaque joueur (figure 7).

Figure 7 : Écran de fin de partie

3. Implémentation du modèle et du contrôleur de la partie

Cette implémentation consiste à veiller au bon déroulement des différentes actions des vues, tout en respectant l'architecture définie.
Elle contient toutes les données correspondantes aux calculs de trajectoire, aux interactions entre les différentes vues.

Nous avons décidé de calculer la trajectoire suivant la deuxième loi de newton :

$$\sum_i \vec{F}_i = m\vec{a}$$

Avec m la masse de la flèche, a son accélération et F_i les forces en présence, ici seuls le poids et la force du vent exercée sur la flèche sont pris en compte.

- **Vérifications par test unitaires**

Afin de vérifier la validité du modèle et du contrôleur de la partie, nous avons implémenté des tests unitaires validant ou non les méthodes que nous avions codées précédemment.

- **Implémentation du réseau**

Afin d'implémenter le mode multi-joueurs de notre jeu, nous avons décidé d'utiliser un serveur externe au logiciel : caenbow.youspin.it.

Tous les messages clients-serveurs sont faits avec le protocole TCP et sont formatés en Json. Les échanges entre le logiciel et le serveur sont documentés en annexe 1.

Lors de la création d'une partie, les différents joueurs envoient tous une requête sur le port 2014 du serveur distant permettant de vérifier que celui-ci est bien en ligne. Le serveur renvoie alors un port disponible au joueur hôte, qui créé alors la partie en entrant tout les paramètres de jeu. Les joueurs clients reçoivent de leur coté la liste des parties disponibles ainsi que leur port, et, choisissent alors la partie sur laquelle ils souhaitent se connecter (figure 8).

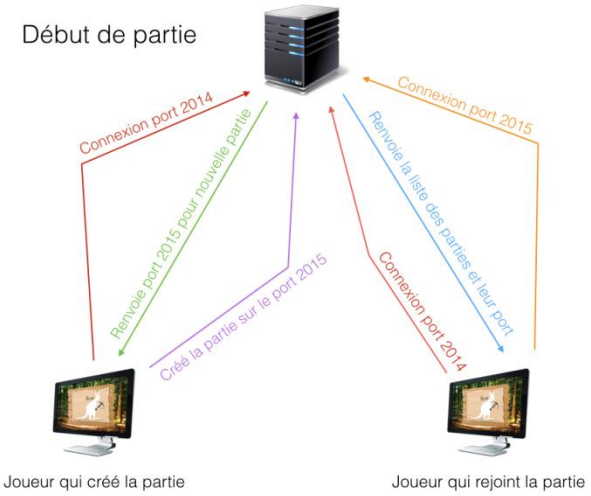


Figure 8 : Echanges partie/serveur lors de la création d'une partie multi-joueurs

Lors de la partie, à chaque tir les joueurs envoient leur résultat au serveur, qui les renvoie à leur tour à tous les autres joueurs (figure 9).

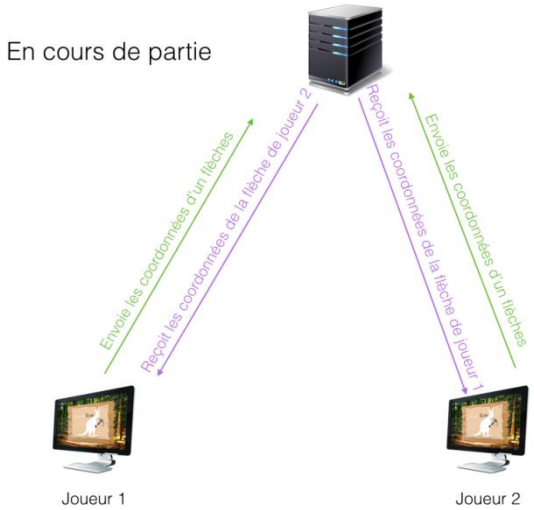


Figure 9 : Echanges partie/serveur lors d'une partie multi-joueurs

A la fin de la partie, le serveur envoie à tout les joueurs la liste compte des joueurs classé selon leur points avec le résultat de chaque flèche lancée par ceux-ci (figure 10).

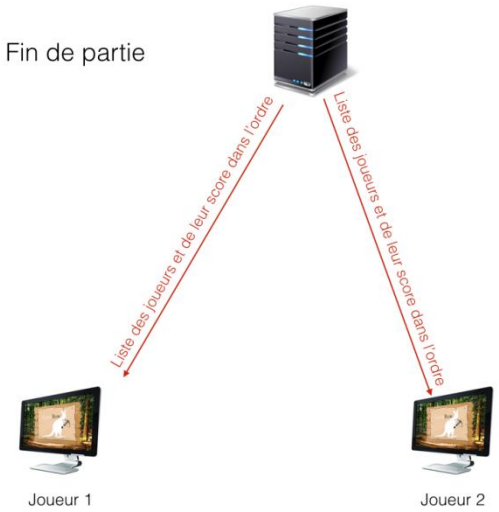


Figure 10 : Echanges partie/serveur lors de la fin d'une partie multi-joueurs

Architecture

Notre modélisation du jeu est représentée sous forme d'un diagramme de classes en annexe 2.

1. Les patrons de conception

Le diagramme des classes de ce projet a été divisé en cinq packages. Deux patrons de conception ont été utilisés :

- un MVC (Modèle-Vue-Contrôleur) ;
- un Client-serveur.

Le MVC est le patron d'architecture permettant la gestion d'un jeu multi-joueurs. En effet, grâce à celui-ci, plusieurs vues peuvent être créé à partir du même modèle. Chaque joueur aura ainsi sa propre vue, indépendante de toutes les autres. Un contrôleur sera créé par vue pour interagir avec le modèle, partagé par tous les joueurs.

L'établissement d'un jeu multi-joueurs en ligne a nécessité un patron Client-serveur. De celui-ci, nous avons pu créer un client pour chaque joueur, se connectant sur le serveur du jeu. Ce dernier contient le modèle du jeu, i.e. toutes les règles du jeu partagées par les joueurs. Ainsi donc, à chaque début de partie, le réseau initialisera certaines données comme la distance du joueur à la cible et la force du vent dans le modèle.

2. Les différents packages

- **modèle :**

La package modèle contient toutes les différentes parties possibles. La partie est un concept abstrait. Deux classes dérivent de celui-ci, les parties multi-joueurs et les parties solos. La classe Partie va donc initialiser toutes les données nécessaires au jeu et les renvoyer à chaque joueur.
La création de la Partie sera faite sur le serveur pour une partie multi-joueurs ou elle sera créée localement dans le cas d'une partie solo.

Par la suite, la classe Partie va créer des instances de Joueur au fur et à mesure que des joueurs rejoignent le jeu.

- **player :**

Le package player représente la classe créée pour un joueur lorsqu'il rejoint la partie. Celle-ci ne se définit que par l'identité du joueur et son score. Un tableau est aussi enregistré pour un Joueur, sauvegardant chaque flèche placée par ce même joueur.

- **vue :**

  La vue est composée de trois sous modules :
  - l'initialisation de la partie
  - l'interface du jeu
  - la fin de partie

Le premier permet soit l'établissement d'une PartieMulti, soit celui d'une PartieSolo. De celle-ci, nous pouvons initialiser les paramètres du jeu selon quelques niveaux de difficultés.

Le deuxième est la vue principale du jeu. Cette interface graphique représente l'arbalète du joueur et sa cible, ainsi que toutes les données dont le joueur à accès, à savoir les différentes arbalètes possibles, les cibles et les impacts des flèches de chaque joueur adversaire, la force courante du vent, et l'inclinaison courante de l'arbalète du joueur.

Le troisième symbolise les résultats de la fin de la partie. Une fois que chaque joueur a fini de jouer, cet écran de fin s'ouvre, affichant les différents résultats, ainsi que les scores de chacune des flèches tirées.

- **outil :**

Ce package contient tous les éléments nécessaires à la partie, comme l'arbalète, la cible, les flèches et le calcul de la trajectoire.
Ainsi, chaque élément pourra être développé individuellement en fonction des paramètres de la partie. La taille de la cible pourra être modifiée à partir de la Partie, ainsi que sa distance au joueur.

- **serveur :**

Le package serveur est le code nécessaire pour l'ouverture du serveur et la création des clients, en suivant le modèle de l'annexe 1.

Conception du projet :

- **Observateur**

Dans la vue principale de tous les joueurs, il est possible de voir toutes les cibles adverses ainsi que les impacts des flèches de chacun des adversaires. Un observateur s'avère utile ici afin que dès qu'un joueur tire, ses adversaires en sont informés et la cible concernant ce joueur soit mise à jour chez tous les autres joueurs.

Néanmoins, par manque de temps, nous n'avons pas implémenté cette classe Observateur nous permettant de mettre à jour toutes les cibles adverses.

- **Etat**

Afin d'optimiser la création des arbalètes et le choix entre les différentes arbalètes possibles, un patron de conception Etat aurait permis de mettre à jour l'arbalète que le joueur sélectionne.

A chaque fois que le joueur clique sur l'image d'une arbalète sur l'interface du jeu, l'arbalète se serait mise à jour et adapté au choix du joueur.

Gestion d'équipe

Pour mener à bien notre projet, il a été primordiale de se répartir convenablement les tâches entre chacun de nous afin d'être efficace, ce qui a été réalisé selon le tableau de la figure 11.
On avait au préalable défini les rôles de chacun comme il suit :

- **Chef de projet** : T. Guaquière ;
- **Architecte** : G. Moraine ;
- **Ingénieur de conception** : J-B. Jorand ;
- **Responsable de version** : V. Durand ;
- **Développeur graphique** :
  → vues de début de partie : J-P. Ngalula et V. Durand,
  → vue principale : J-B. Jorand,
  → vue de fin de partie : T. Guaquière ;
- **Développeur java** : G. Moraine, L. Djiomo et S. Elyadari ;
- **Développeur réseau** : V. Durand.

	Équipe							Client
Tâches	T. Guaquière	J-B. Jorand	V. Durand	G. Moraine	S. Elyadari	L. Djiomo	J-P. Ngalula	J. Mahier
Conception architecture	A	R	I	R	I	I	I	C
Implémentation contrôleur / modèle	A	I		R	R			C
Interface Graphique	A, R	R	R				R	C
Test Unitaire	A			I	I	R		C
Réseau	A		R				I	C

Figure 11 : Matrice des responsabilités

A : assume
R : réalise
I : informé
C : consulté

Risques

La matrice des risques est représentée figure 11.

Nature du risque	Description	Action préventive	Action corrective	Responsable
Humain	Manque de compétence en réseau et/ou en graphisme	Se former en la matière à l'aide de documentation sur le sujet	Consulter de nouveau la documentation	V. Durand T. Guaquière J-B. Jorand
Humain	Mauvaise ambiance au sein de l'équipe	Définir clairement les rôles de chacun	Établir des réunions pour parler des différents litiges	T. Guaquière
Humain	Le chef de projet est absent	Prévoir un remplaçant	Le remplaçant devient chef de projet temporairement	J-B. Jorand
Humain	Les résultats des tests ne sont pas ceux escomptés	Prévoir dans le planning du temps pour corriger d'éventuel bogue	Faire des tests régulièrement et en informer du résultat le développeur concerné	L. Djiomo
Technique	Le réseau de l'ENSICAEN tombe en panne lors d'une séance de travail	Prendre sa machine personnelle avec soi	Utiliser sa machine personnelle	Tout le monde
Technique	Les logiciels nécessaires ne sont pas présents sur les ordinateurs fournis par l'école	Prendre sa machine personnelle avec soi	Utiliser sa machine personnelle	Tout le monde

Figure 11 : Matrice des risques

Dysfonctionnements

Finalement, les deux modes de jeu, solo ou multi-joueurs sont fonctionnels. Même s'il persiste toujours quelques problèmes tel que l'impossibilité d'affichage de la vue de la cible adverse, un problème de précision lors de l'affichage des impacts des flèches ou encore le fait que lors d'une partie multi-joueurs, les joueurs n'attendent pas la fin de partie des autres, ce qui entraine un problème d'affichage du score final des joueurs finissant en premier.

La partie multi-joueurs fonctionne correctement, malheureusement, le proxy de l'ENSICAEN nous empêche d'accéder au server distant. Nous avons pris connaissance de ce problème trop tard mais avons trouvé une solution, bien qu'inexploitable, qui implique de recompiler le code. Il suffit en effet de remplacer la ligne 59 de la classe Client du package par ip = adresseIp; où adresseIp est l'adresse IP de l'ordinateur interne à l'ENSICAEN ayant lancé le .jar du serveur.

Ce projet dysfonctionna principalement sur le plan humain, avec les retards, erreurs d'implémentations de chacun, nous perdîmes beaucoup de temps, ce qui s'en ressentit lors de la finalisation du projet.

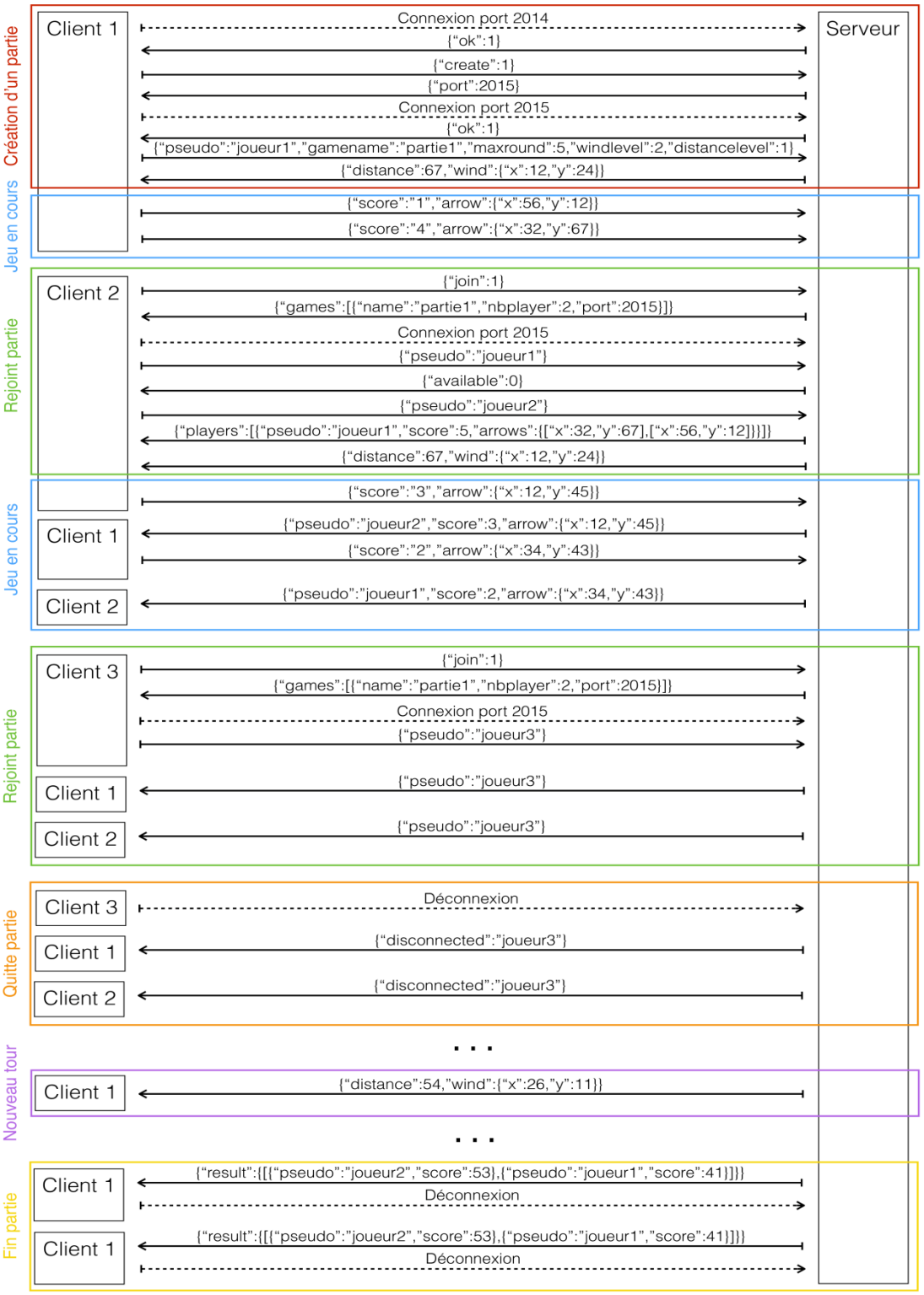
Bilan

Ce projet à été mené à son terme le 11 janvier dernier, pour autant bien que les délais furent difficiles à respecter, le logiciel final est fonctionnel.

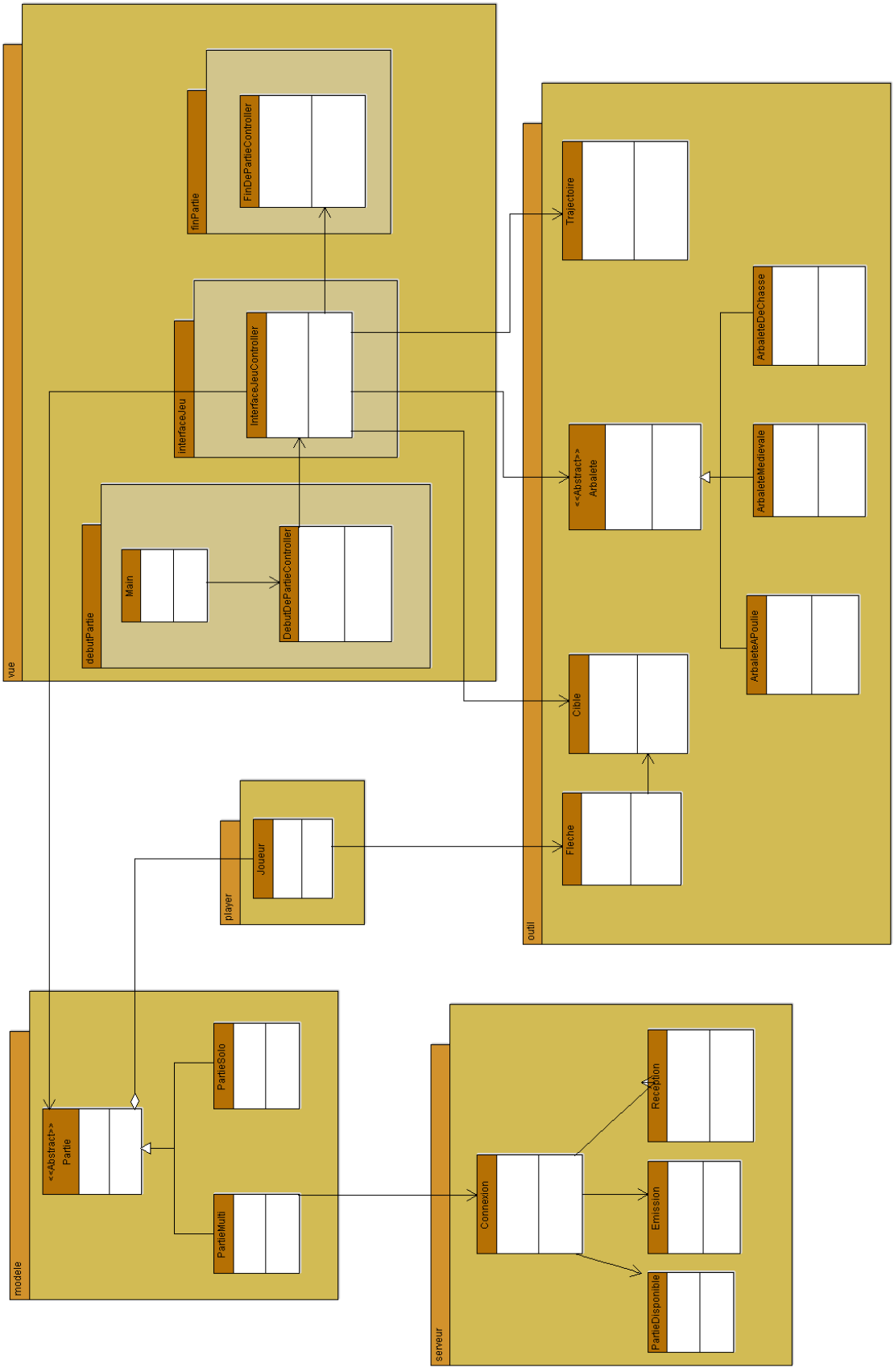
Il est possible de lancer une partie solo facilement et sans problème. De plus, malgré les légers problèmes évoqués précédemment pour une partie multi-joueurs, le jeu peut quand même être lancé en ligne avec plusieurs joueurs.

Par ailleurs, toutes les fonctionnalités exigées par le client sont implémentées et ont été respectées.

Annexe



Annexe 1 : Diagramme des dialogues client-serveur



Annexe 2 : Digramme des classes