# Protocole d’échange entre le client et le serveur

Vous trouverez en annexe le protocole JSON d’échange entre le client et le serveur des principaux messages de communications.

A chaque message sera associé un scénario d’utilisation correspondant selon le cahier des charges définis dans les premières étapes du projet. Nous avons utilisé ce protocole pour nous permettre de nous répartir les tâches entre l’implémentation du client et du serveur. En effet chacun avait ce document comme référence de la syntaxe et de la sémantique de chaque message. Il s’agit donc du lien entre les clients considérés comme des vues et le serveur considéré comme le modèle et le contrôleur.

# Suivi du projet : itérations par itérations

Tout au long des itérations que nous avons planifiés, nous nous sommes confrontés à divers soucis, problèmes ou voir même bonnes surprises qui ont conduit à diverses replanifications. Nous avons également tiré un bilan de nos erreurs et des choses que nous avons bien faites.

## Itération 1 – Serveur d’enregistrement + Interface graphique

Dans cette itération, le but était de mettre en place le serveur d’enregistrement en utilisant les protocoles et outils réseaux développés, ainsi que d’intégrer dans le programme la partie qui utilises ce serveur, permettant de voir les parties enregistrées.

### Problèmes rencontrés

Nous avons rencontré des soucis lors de la mise en place des tests unitaires JUNIT sur cette partie, du fait des méthodes bloquantes de la classe TCPCanal développée pour faire l’interaction TCP entre le client et le serveur.

Nous nous sommes également confrontés à des problèmes conceptuels concernant l’adresse IP à transmettre au serveur d’enregistrement. En effet, nous avions le choix entre l’adresse locale (celle du réseau local visiblement uniquement par les utilisateurs sur le même broadcast) et l’adresse Internet (l’adresse du routeur, vue par un éventuel client ou serveur externe). Nous avons du mettre en place un protocole de contournement, ainsi qu’une fonctionnalité permettant de se connecter à un serveur directement depuis une adresse IP, sans passer par le serveur d’enregistrement.

### Bilan

Nous tirons de cette itération un bilan très positif : en effet nous avons à notre disposition un serveur d’enregistrement tel que décris dans le cahier des charges fonctionnel. Cependant, les soucis d’adressages rencontrés nous ont bien rendus attentif à ce genre de détails pour la suite du projet.

### Replanifications

Malgré la résolution des soucis inattendus relatifs aux adresses IP, nous avons pu rester dans les délais et ne pas avoir eu à replanifier les itérations suivantes.

### Remarques

Lors de la livraison de l’itération, le client nous a semblé content du résultat. Les retours que nous avons eu nous on permit de rester motiver pour la suite du projet.

## Itération 2 – Serveur de Jeu + Architecture

Dans cette itération, le but était de mettre en place le serveur de jeu ainsi que l’architecture associée, tant coté client que serveur, incluant les protocoles de communications entre le serveur et le client.

### Problèmes rencontrés

Malheureusement le protocole tel que présenté dans le rapport préliminaire et le cahier des charges n’était pas totalement adapté à nos besoins. Nous avons du donc nous replonger dedans pour corriger quelques incohérences et ajouter quelques fonctionnalités nécessaires. Ce n’était pas à proprement parlé un problème, mais une rapide remise en question et adaptation du travail fourni précédemment.

Nous avons également rencontré au niveau protocole des soucis concernant non pas le protocole d’échange, mais le protocole de connexion (échange des versions, des ID des clients, etc…). En effet, ce protocole était défini de manière empirique et suivant les règles de bons sens. Nous avons donc du nous mettre d’accord sur un bref ensemble de messages à changer à la connexion.

Cependant, nous avons expérimenté un léger stress avant la présentation de l’itération du à ces refontes. Cependant nous avons su en n’ajoutant pas d’heures supplémentaires mais en augmentant l’efficacité de notre travail durant les heures planifiées arriver à bout des soucis que nous avons rencontré.

### Bilan

Encore une fois cette itération s’est bien passée. Nous avons également effectué des tests de non régressions sur la version solo pour vérifier que l’adaptation de la structure ne supprime pas de fonctionnalités déjà vérifiées dans la version précédente.

L’implémentation du protocole tant coté client que serveur s’est également bien passée, nous avons très rapidement pu arriver à une version fonctionnelle du dialogue entre le serveur et le client. Par exemple le client demande l’ajout d’une tour, le serveur reçoit correctement la requête et donne en retour une valeur témoin.

### Replanifications

Malgré les changements souvent esthétiques du protocole, nous n’avons pas subis de retard entrainement une replanifications futures.

### Remarques

Encore une fois la présentation du résultat au client s’est correctement déroulée. Nous avons mis en place une petite application de test en interface par lignes de commandes pour l’échange de messages texte à la manière d’un t’chat IRC pour illustrer le fonctionnement correcte de notre protocole.

## Itération 3 – Intégration du serveur de jeu + Interface du Jeu en réseau

Dans cette itération, notre but a été de mettre en place le lien entre les modules clients/serveur et la structure de base du jeu (modèles, vues, contrôleurs) déjà mise en place dans les itérations précédentes.

### Problèmes rencontrés

Nous nous sommes confrontés à des soucis de compréhension des bases déjà mises en place au sein de l’équipe de développement. En effet, les responsables de chaque parties ont du comprendre et par la lecture de la documentation mise en place et par le dialogue direct la manière dont chaque partie fonctionne.

Cette mise à niveau de chaque membre sur les sections des autres s’est révélée être complexe du fait du nombre de concepts mis en place par chacun. Cependant, nous avons pu grâce à une réunion de crise éclaircir pour chacun les points problématiques ainsi que de pouvoir recentrer nos efforts.

### Bilan

Dernière étape avant la terminaison du projet, cette itération s’est révélée délicate. Malgré quelques soucis de communication au sein du groupe, nous avons pu mettre en place des moyens de résolution rapides des problèmes survenus, et ceux à moindre coût humain et en temps.

Nous avons donc pu présenter dans les temps l’itération selon le cahier des charges.

### Replanifications

Ici encore une fois pas de replanification nécessaire, mais une heure supplémentaire de réunion elle a été obligatoire pour se mettre d’accord sur les façons de lier le serveur/client aux parties applicatives.

### Remarques

La présentation au client de l’itération s’est bien passée. Nous avons pu faire une démonstration d’une pose de tour par le client ainsi que l’affichage de l’objet chez chaque client connecté.

## Itération 4 - Lifting de la GUI + Game Design + Amélioration Mode Solo

Dans cette itération nous avions prévu de mettre en places des refontes cosmétiques de notre application, tant au niveau de l’interface qu’au niveau de la jouabilité.

### Problèmes rencontrés

Pas de problèmes majeurs rencontrés, à part l’hétérogénéité des bonnes idées d’améliorations (changement du prix des créatures, de la manière dont nous faisions gagner de l’argent aux joueurs, etc…) entre les membres du groupe. En effet, M.Da Campo en tant que chef de projet ainsi que M.Farjallah en tant que testeur ont énormément apporté de très bonnes idées permettant d’améliorer grandement le coté ludique du jeu. Dû au coté prenant et motivant de cette itération, quelques heures supplémentaires on été faites.

### Bilan

Après cette itération nous avons un jeu pleinement jouable malgré notre faible expérience dans le développement de jeux vidéo. La difficulté des différents modes est contrôlée, la jouabilité est également au rendez-vous. Nous ressortons enthousiastes de cette itération et très fière de présenter notre travail au client.

### Replanifications

Ici également pas de replanifications nécessaire du à l’investissement d’heures supplémentaires pour peaufiner les détails ludiques du jeu.

Cependant des membres du projet par forcement concerné au premier degré par l’équilibrage du jeu sont venus en renfort des membres responsables de cette partie.

*(ici peut être donner un nombre d’heure, j’ai un peu de mal à estimer le volume sans doute énorme !)*

### Remarques

Rien de plus à ajouter.

*(Si tu veux ajouter quelque chose Aurélien ici, libre à toi !)*

## Itération 5 – Serveur Web (facultatif)

Dans cette itération nous avions prévu de mettre en ligne l’affichage des scores avec une gestion client/administrateur tel que suggéré dans la structure du projet.

### Problèmes rencontrés

Le principal problème que nous avons rencontré est le délai de livraison du projet ne nous laissant pas le loisir dans le laps de temps que le client nous a accordé de mettre en place cette itération définie également avec le client comme facultative.

### Bilan

Cette itération ne sera donc pas livrée à la date butoir fixée avec le client. Si nous avions eu de l’avance dans les précédentes itérations, nous aurions pu avoir le temps de mettre en place l’itération cinq. Cela n’a malheureusement pas été le cas.

Cette itération reste donc non livrée, en conformité avec le cahier des charges.

### Replanifications

Cette itération va être replanifiée dans une amélioration future du projet hors délai accordé par le client pour le projet.

### Remarques

Rien d’autre à ajouter, sinon insister sur le fait que la non livraison de cette itération était un cas prévu dans le cahier des charges avec le client. Nous avons donc en tous points respectés le contrat précédemment admis par les deux partis.

# Stratégie d’intégration du code de chaque participant

Repartant d’un projet déjà existant, nous avions une structure de code déjà mise en place sur notre SVN (un serveur public fourni par Google : <http://code.google.com/p/asd-tower-defense/> )

La stratégie a été la suivante : nous avons fait un clone du code existant de la version 1.0 pour conserver, par soucis historique, une base tel qu’elle a été présentée dans le cours d’ASD2.

Puis nous avons restructuré le code pour permettre la mise en place des paquets destinés au réseau (créations des outils par M.Farjallah, écriture des parties clients et serveur par M.Poulain et M.Putallaz, intégration de la gestion du réseau dans le modèles par M.Da Campo).

A la suite de la mise en place des bases de travaille, chacun a modifié les parties dont il était responsable au fil des itérations en concurrence avec toujours SVN. Cette méthode nous permettait d’avoir un suivi très régulier des modifications par les notes que nous avons intégrés à chaque *commit*, de pouvoir revenir en arrière en cas d’erreur grâce à la fonction *revert* ainsi que de gérer les éditions concurrente du même fichier ou du même pacquage.

*(je ne sais pas quoi d’autre ajouter, peut être une capture d’écran d’une page de commit du SVN…)*

# Ce qu’il resterait à développer

(ici je ne donne juste qu’une idée pour le maillage, en complément à ce qu’Aurélien pourrait dire)

## Maillage

Hérité du projet d’ASD2, le maillage parcouru par un algorithme de recherche de chemin le plus court d’un point à un autre par Dijkstra pourrait être grandement amélioré.

En effet, la version actuelle souffre d’un manque de performance flagrant pour un grand nombre de créatures, du fait de la redondance de certains calcules. L’algorithme de Dijkstra est très performant pour trouver un chemin d’un point A à un point B dynamiques pour chaque créatures à chaque position sur le maillage. Cependant, dans notre problème, nous n’avons d’un seul point d’arrivée et d’un seul point de départ pour toutes les créatures

Nous aurions donc la possibilité de restreindre les contraintes du problème et de trouver une solution plus performante pour la recherche de chemin à travers le graphe.

L’idée serait de considérer l’entier du maillage (avec les arcs dynamiques) comme un système de flot. Le flot partirait du point de départ des créatures et se terminerait dans leur point d’arrivée. Chaque créature suivrait donc ce flot déjà calculé (et recalculé de manière différentielle à chaque modification du maillage –pose de tour, suppression d’une tour, etc… -) au lieu de recalculer à chaque modification son propre chemin de son point courant au point invariant d’arrivée.

Un gros travail algorithmique avancé serait donc nécessaire pour optimiser les performances de la recherche de chemin. Cependant le travail se limiterait uniquement à la classe *Maillage*, les méthodes d’accès à cette classe ainsi que la manière de l’utiliser ne seraient pas modifiés par cette refonte du fait de l’architecture de notre projet.