Diceforge201920

Rendu intermédiaire

Groupe : TP2 DFA

Équipe

GILLES Marco

LEFEUVRE Maxence

MERCURI Sabrina

CAMORANI Alexis

DESIRE Stéphane

Professeur référent

FERRY Nicolas

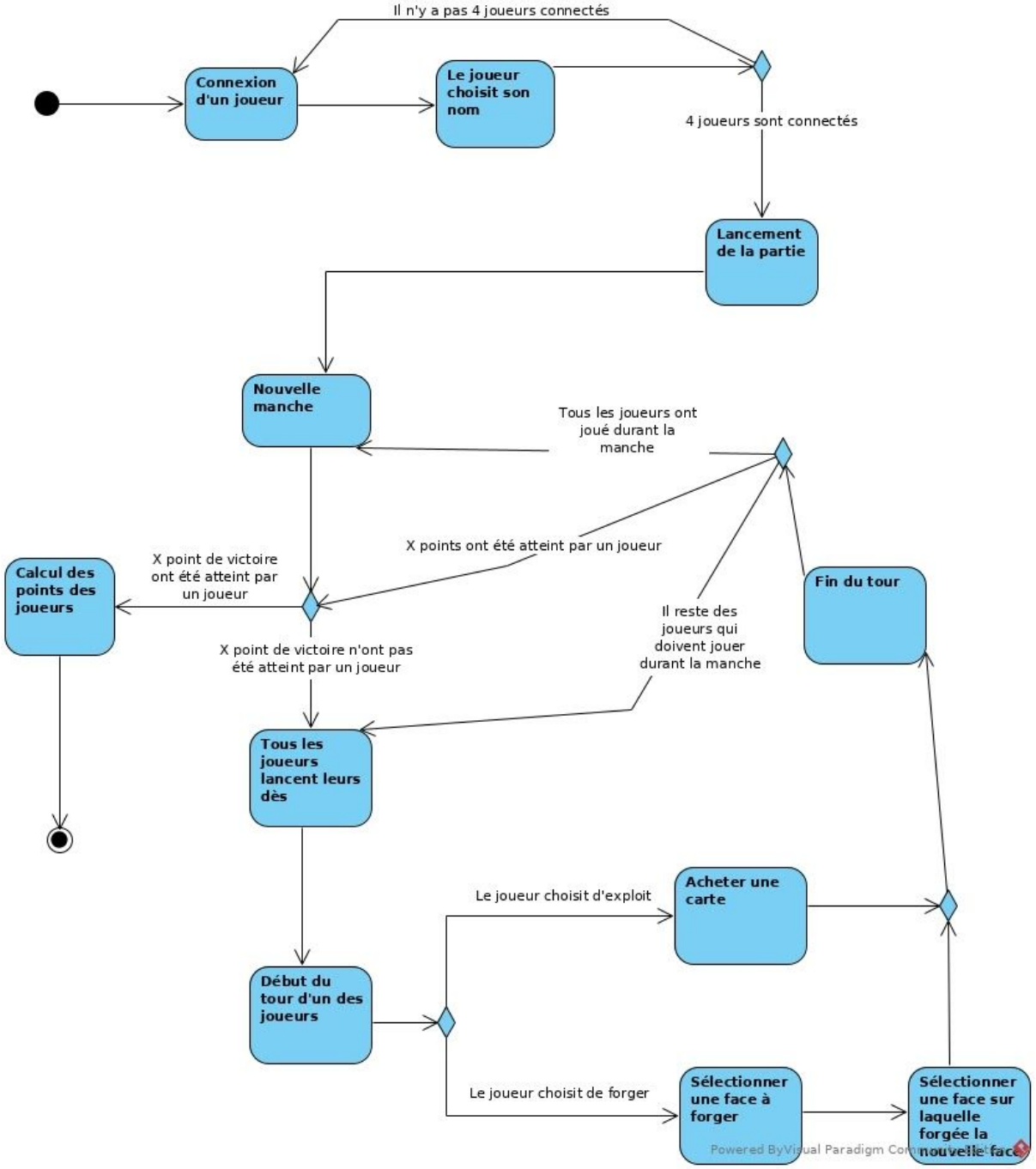
04/03/2020

Point de vue général

Glossaire

Joueur : Quand on parlera de Joueur on parlera de la classe java Joueur pas de l'utilisateur
Utilisateur : L'utilisateur est la personne/l'entité qui jouera avec le Client
Client : Le Client désigne le programme qui sert à l'utilisateur pour pouvoir communiquer avec le serveur.
Forge : La forge ou aussi le sanctuaire.
Sanctuaire : L'endroit où le joueur peut acheter des faces de dé
Catégorie de la forge : Les différents pallier de coût pour acheter une face

Représentation générale



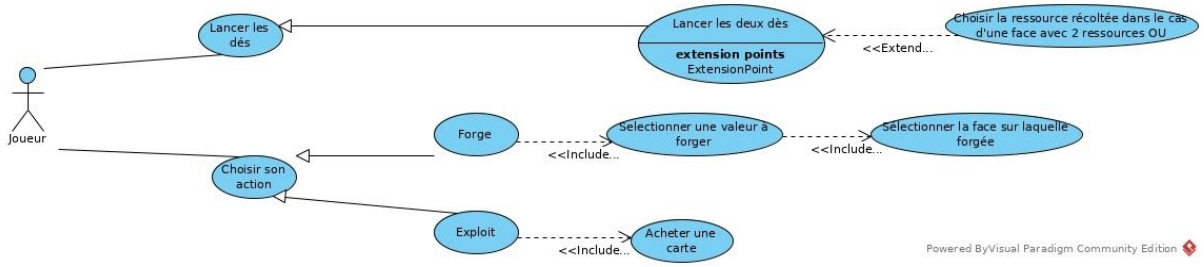
Fonctionnalités traitées

Durant les itérations 1, 2, 3 nous avons traité les fonctionnalités suivantes :

- Connexion du client au serveur
- Le serveur peut accepter ou refuser les connexions des clients
- Attribution du pseudo de leur choix aux utilisateurs
- Démarrage de la partie
- Modélisation de la forge
- Modélisation des différentes catégories de la forge
- Modélisation des îles
- Création des cartes (sans les effets pour le moment)
- Création de l'inventaire
- Création des dés et des faces
- Lancer des dé et transmettre les résultats des lancers
- Acheter des faces de dé et transmettre l'achat de la face
- Placer les faces de dé et transmettre le placement de la face
- Acheter des cartes et transmettre l'achat de la carte
- Condition de victoire (pour le moment sur un nombre X de points de victoire)

Client/Joueur

Analyse des besoins



Un joueur peut effectuer les actions suivantes.

- Lancer les deux dés.
- Le joueur peut choisir une action qu'il effectue durant son tour, soit il forge et achète des faces de dé, soit il exploit et achète une carte.

User story

Titre : Lancer un dés

Description :

En tant que joueur,

je souhaite pouvoir lancer les deux dés de mon inventaire

afin de pouvoir obtenir des ressources.

Tests d'acceptances

Scénario :

Etant donné que j'ai lancé les dés

Et que j'ai obtenu une ressource sur chaque dé

Quand je veux récupérer les ressources

Alors je les récupère

Scénario :

Etant donné que j'ai lancé les dés

Et que j'ai obtenu une ressource sur un dé et un multiplicateur sur l'autre

Quand je veux récupérer la ressource

Alors je la récupère autant de fois que le multiplicateur

Scénario :

Etant donné que j'ai lancé les dés

Et que j'ai obtenu un multiplicateur sur les 2 dés

Alors rien ne passe

Scénario :

Etant donné que j'ai lancé les dés

Et que j'ai obtenu une ressource ou un multiplicateur sur un dé et que j'ai obtenu un portail sur l'autre

Quand je veux récupérer les ressources

Alors je sélectionne une ressource chez un autre joueurs et je la récupère en appliquant le multiplicateur si j'en ai obtenu un, sinon je la récupère simplement.

Titre : Acheter des faces de dés

Description :

En tant que joueur je peux changer la face d'un de mes dés afin de l'améliorer.

Test d'acceptances :

Scénario :

Etant donné que je veuille forger un dé, si j'ai assez de ressources pour

avoir l'amélioration choisie

Alors je sélectionne la face de mon dé que je veux changer, puis la nouvelle face de mon dé.

La nouvelle face prend la place de l'ancienne.

Titre : Acheter des cartes

Description :

En tant que joueur,

je peux acheter des cartes afin de pouvoir

améliorer mon équipement ou gagner des points de victoire.

Test d'acceptances

Scénario :

Etant donné que j'ai réuni assez de ressources

je peux acheter une carte.

Je l'obtiens.

Conception logicielle

Pour ce diagramme de classe, nous avons fait des choix au niveau des relations :

- La classe Plateau contient 4 Joueurs, des îles qui contiennent des Cartes qui héritent de BuyableItem et une Forge qui contient des Faces qui héritent aussi de BuyableItem.
- Un Joueur a un Inventaire qui comprend une liste de dés, une liste de cartes, une liste de faces.

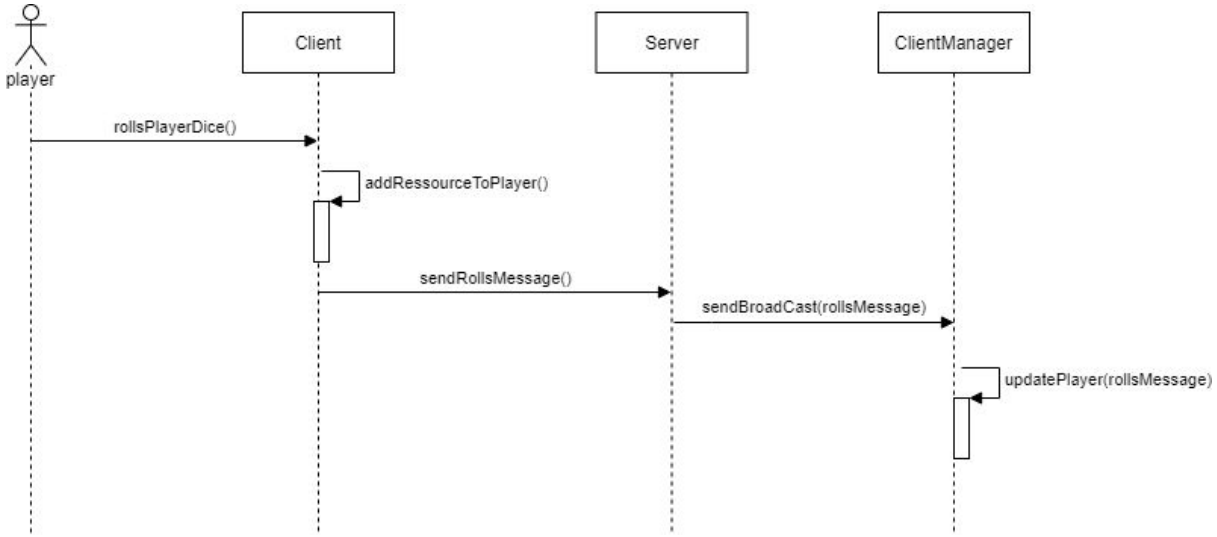


Diagramme de séquence : Lancer les dés

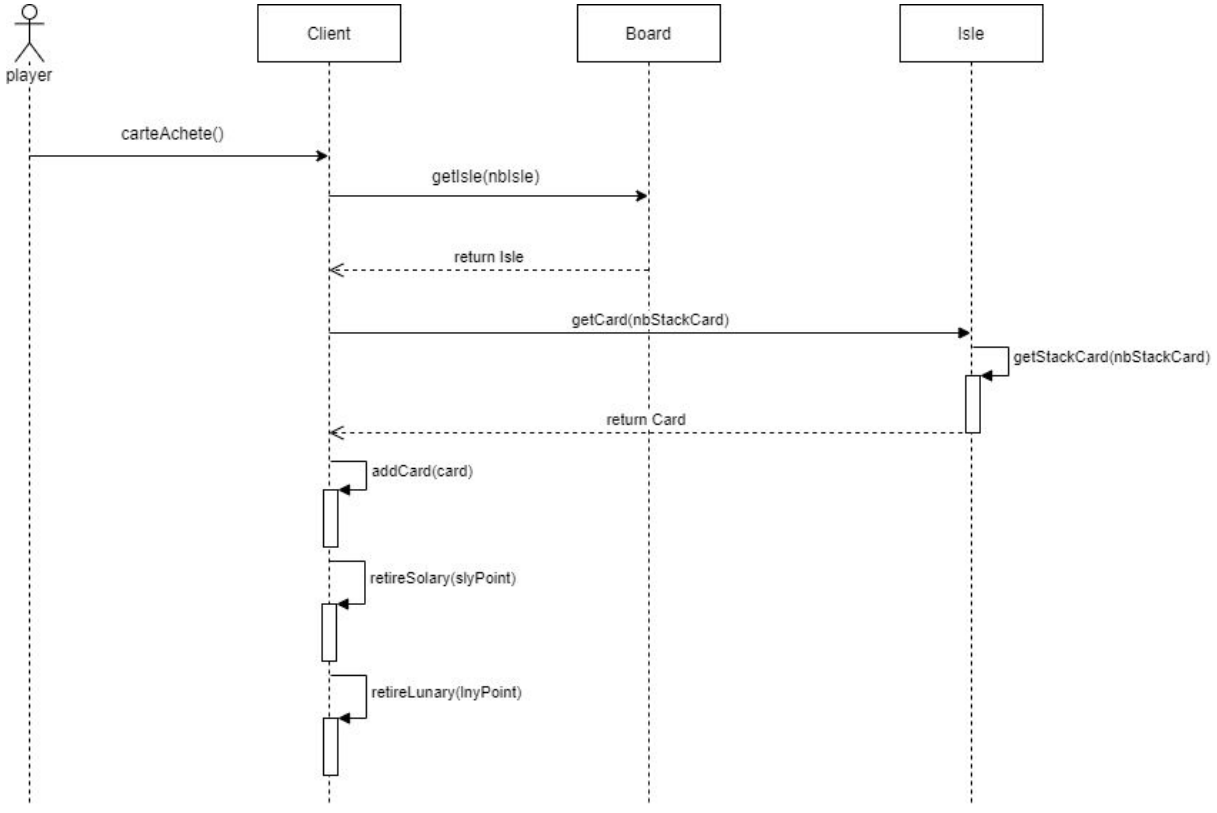


Diagramme de séquence : Exploit

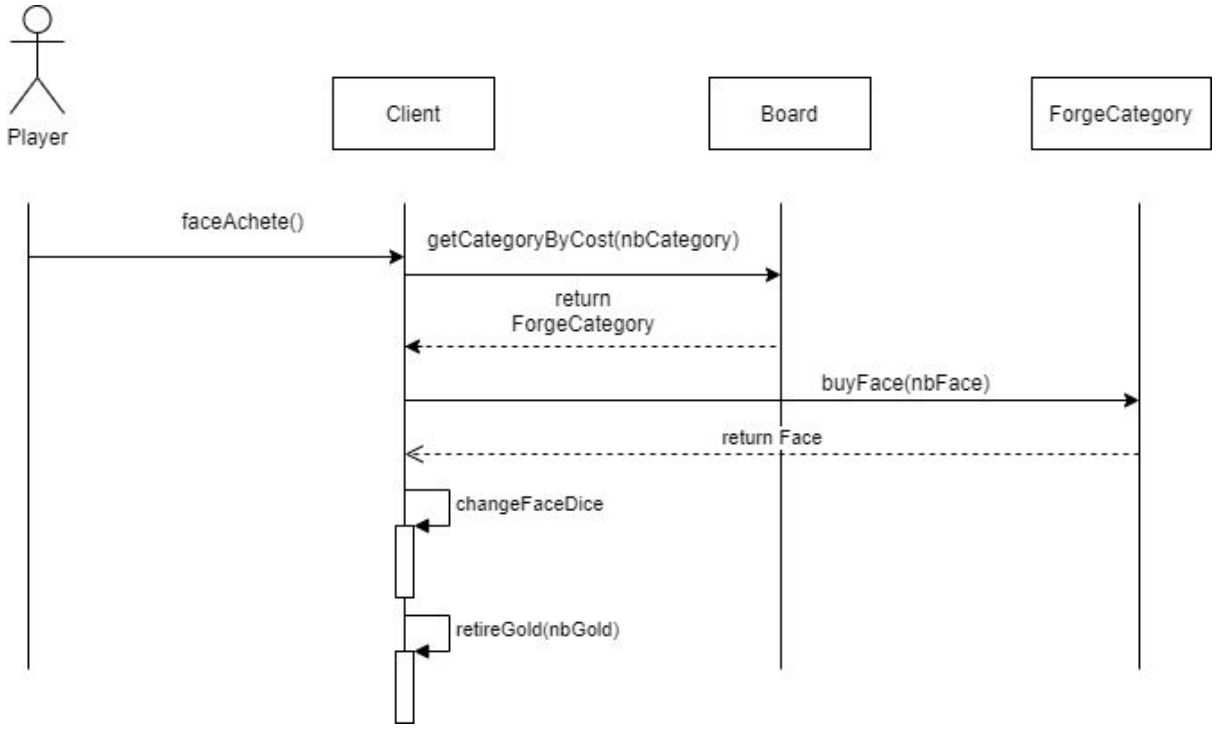
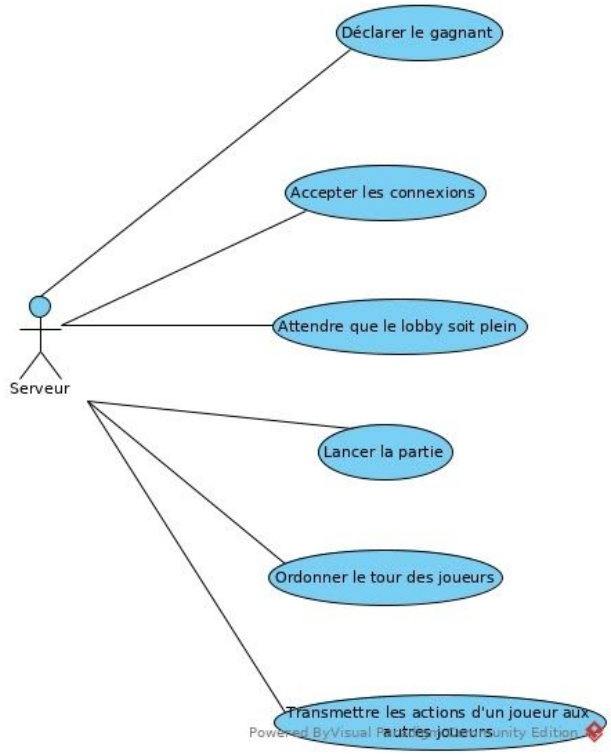


Diagramme de séquence : Forge

Serveur/Moteur de jeu

Analyse des besoins



Le serveur peut effectuer les actions suivantes :

- Déclarer un gagnant
- Accepter les connexions des clients ou les refuser dans le cas ou la partie est pleine
- Attendre que le lobby soit plein pour lancer la partie
- Lancer la partie une fois que le lobby est plein
- Choisir l'ordre de jeu des joueurs.
- Transmettre les actions d'un joueur aux autres joueurs

User Story

Titre : Accepter les connexions de plusieurs clients

Description

En tant que serveur,

Je souhaite accepter les connexions de plusieurs clients

Afin de pouvoir lancer une partie

Tests d'acceptances :

Scénario :

Etant donné que j'ai pour objectif de lancer une partie

Et que pour lancer une partie il me faut entre 2 à 4 joueurs

Alors je vais accepter les différentes connexions de joueurs

Scénario :

Etant donné que ma partie est déjà composé de 4 joueurs

Et que pour lancer une partie il me faut entre 2 à 4 joueurs

Alors je refuse la connexion du joueur

Scénario :

Etant donné que ma partie est déjà en cours

Et qu'un joueur essaye de se connecter

Alors je lui refuse l'accès

Titre : Attendre la connexion des autres joueurs pour que la partie soit pleine

Description

En tant que serveur, je souhaite attendre la connexion de tous les joueurs avant de lancer la partie afin que la partie se joue avec 4 joueurs au maximum et 2 au minimum.

Note : Le jeu se joue entre 2 et 4 joueurs (3 inclus).

Critère d'acceptance :

Lancer la partie quand le nombre de 4 joueurs est atteint et notifier les joueurs du début de la partie.

Lorsqu'il y a moins de 2 joueurs, il ne faut pas lancer la partie

Lorsqu'il y a entre 2 et 3 joueurs on peut lancer la partie à partir d'un certain temps d'attente.

Titre : Lancer la partie

Description

En tant que serveur,

Je souhaite pouvoir lancer la partie immédiatement lorsque 4 joueurs sont connectés et la lancer après un délai lorsque 2 ou 3 joueurs sont connectés afin de pouvoir commencer à jouer.

Critère d'acceptance :

Scénario :

Etant donné que 4 joueurs se sont connectés

Et qu'ils sont prêts

Quand je veux lancer la partie

Alors la partie se lance et une notification est envoyée aux joueurs pour les prévenir du début de partie.

Scénario :

Etant donné que 2 ou 3 joueurs se sont connectés

Et qu'ils sont prêts

Quand je veux lancer la partie

Alors la partie se lance après un délai et une notification est envoyée aux joueurs pour les prévenir du début de partie.

Titre : Choisir un ordre de jeu des joueurs

Description

En tant que serveur, je souhaite donner un ordre de tour aux joueurs, afin que que ceux-ci puissent jouer chacun à leurs tours.

Critère d'acceptance :

Sur chaque manche faire jouer les 4 joueurs, puis à la prochaine manche refaire jouer les joueurs dans le même ordre

Titre : Retransmettre les actions des autres joueurs

Description

En tant que serveur,

Je souhaite pouvoir prévenir les autres joueurs des actions de chacun en tant réel afin qu'ils soient au courant du déroulement de la partie.

Critère d'acceptance :

Scénario :

Etant donné qu'un joueur a effectué une action

Et que l'action est terminée

Quand je veux prévenir les autres joueurs

Alors j'envoi un message aux joueurs en détaillant l'action qui vient d'être effectué et qui l'a effectué.

Scénario :

Etant donné qu'un joueur n'a pas effectué d'action

Et que son tour est fini

Quand je veux prévenir les autres joueurs

Alors je ne fais rien et je passe au tour du joueur suivant.

Titre : Fin du jeu

Description

En tant que serveur,

Je souhaite mettre un terme à la partie

Afin de pouvoir définir un gagnant

Tests d'acceptances :

Scénario :

Etant donné que 9 tours sont passés

Et qu'ils sont terminés

Quand je veux terminer la partie

Alors je calcule les points et je défini un gagnant

Titre : Transmettre les actions d’un joueur aux autres joueurs

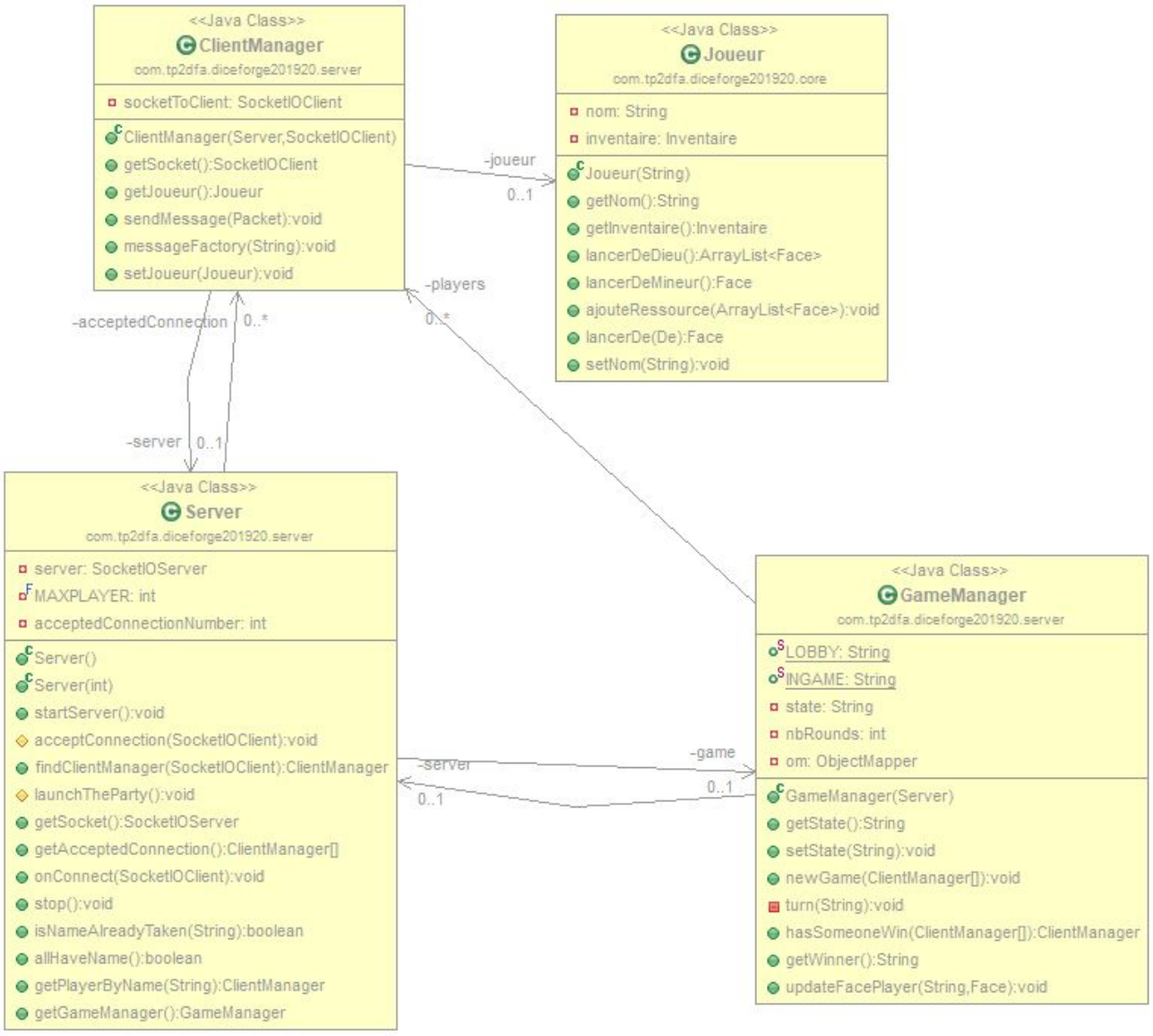
Description

En tant que serveur, je souhaite transmettre les actions d’un joueur aux autres joueurs afin qu’ils puissent mettre à jour leurs versions du plateau

Test d’acceptance

Quand un joueur m’envoie un message de ce qu’il vient d’effectuer, si ce message a une influence sur le plateau alors je le retransmet aux autres joueurs.

Conception logicielle



Pour ce diagramme de classe, nous pouvons voir que la classe ClientManager est la référence du joueur côté serveur. Il comprend aussi une référence au serveur.

Le serveur a une liaison avec le GameManager et le ClientManager.

Le GameManager a une liaison avec le ClientManager et le Serveur.

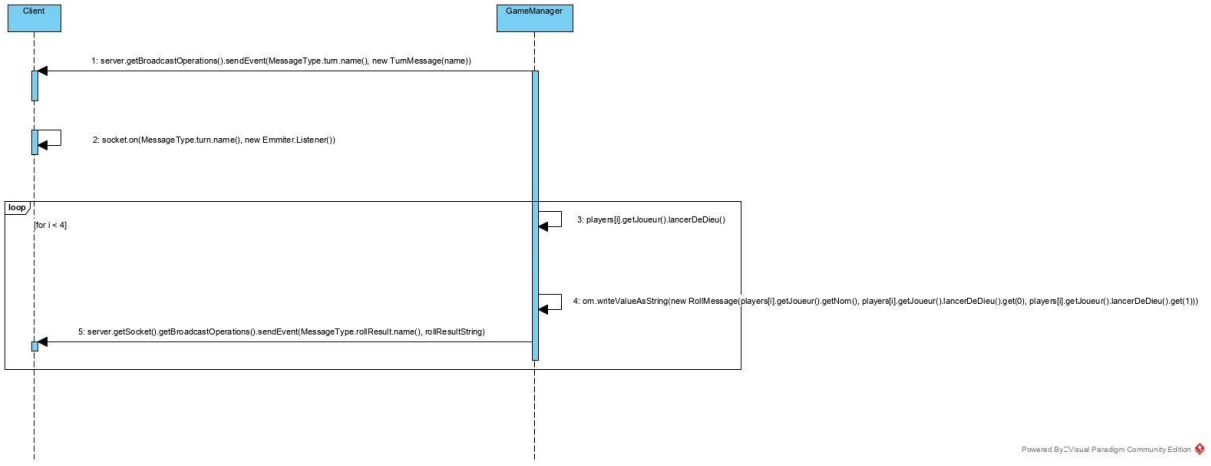


Diagramme de séquence : Ordonner le tour des joueurs

Pour ordonner le tour des joueurs nous envoyons tout d'abord un message aux clients pour leur dire que le tour commence puis le GameManager va faire jouer un par un, grâce à la boucle for, les clients en lançant leurs dés puis il enverra à tous les clients le résultat des lancements de chaque joueur. Dans le cas où le message ne s'envoie pas on lève une exception.

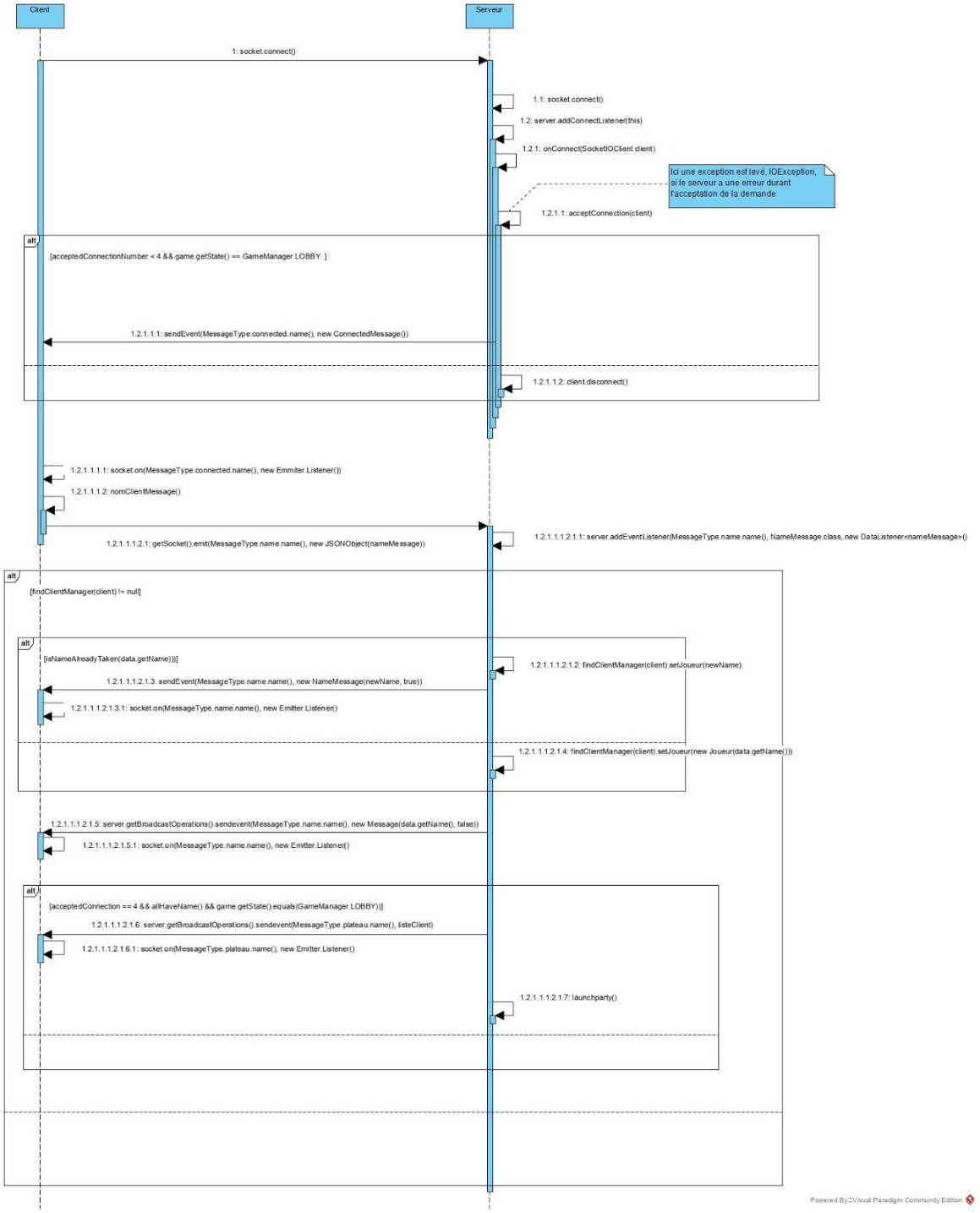


Diagramme de séquence : Accepter connexion

Lorsque le client est créé, il envoie une demande de connexion au serveur qui lui est en attente de demande. Puis il envoie par message au client qu'il est bien connecté. Par la suite le client envoit son nom au serveur pour que celui-ci lui attribue un joueur au client de son côté. Si le nom est déjà pris alors le serveur lui attribue un nouveau nom et envoie ce nouveau nom au client pour qu'il le mette à jour puis ajoute attribue le joueur au client et envoie le nom de Joueur au client. Dès que 4 clients seront connectés alors le serveur envoie au 4 clients un message pour qu'ils mettent à jour leur plateau par rapport aux autres Joueurs. Si une demande de connexion échoue alors on déconnecte le client.

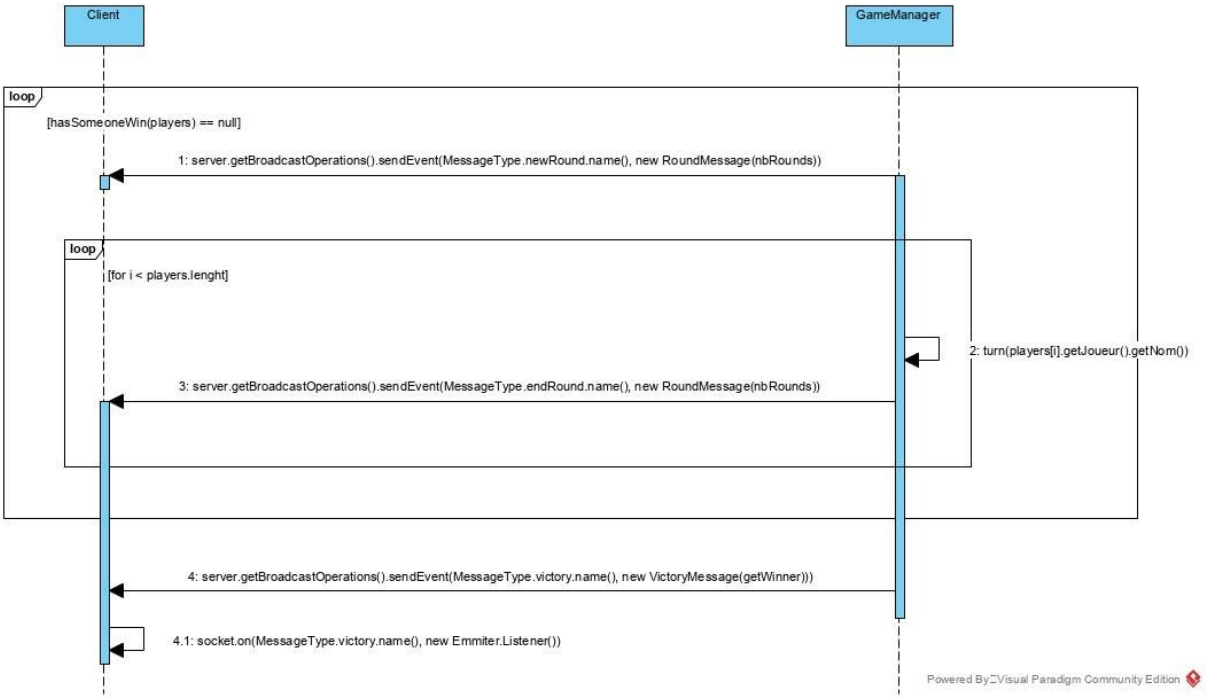
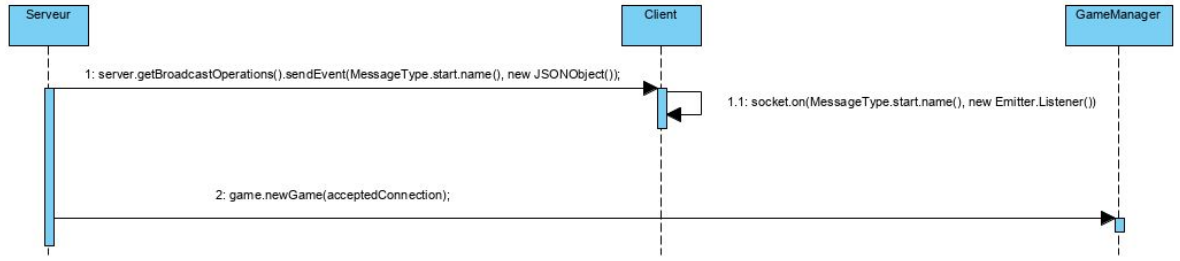


Diagramme de séquence : Déclarer le gagnant

Tant que personne n'a gagné alors on continue de jouer normalement avec les tours. Lors du calcul des points avec la fonction turn() de chaque joueurs et la vérification avec hasSomeoneWin() , dès qu'un des joueurs atteint le nombre nécessaire de points pour gagner alors le serveur envoie un message à tout le monde pour les prévenir que tel joueur à gagner. Dans le cas où le message ne s'envoie pas on lève une exception.



Lancer la partie

Pour le lancement de la partie, le serveur va envoyer un message de début de partie en broadcast à tous les clients et après cet envoi, il va lancer la gestion de la partie dans le GameManager.

Interaction entre le/les joueurs et le moteur

Protocole d'échange

CarteMessage / HuntMessage

CarteMessage qui notifie le serveur puis les autres joueurs que le joueur qui joue à acheter une carte

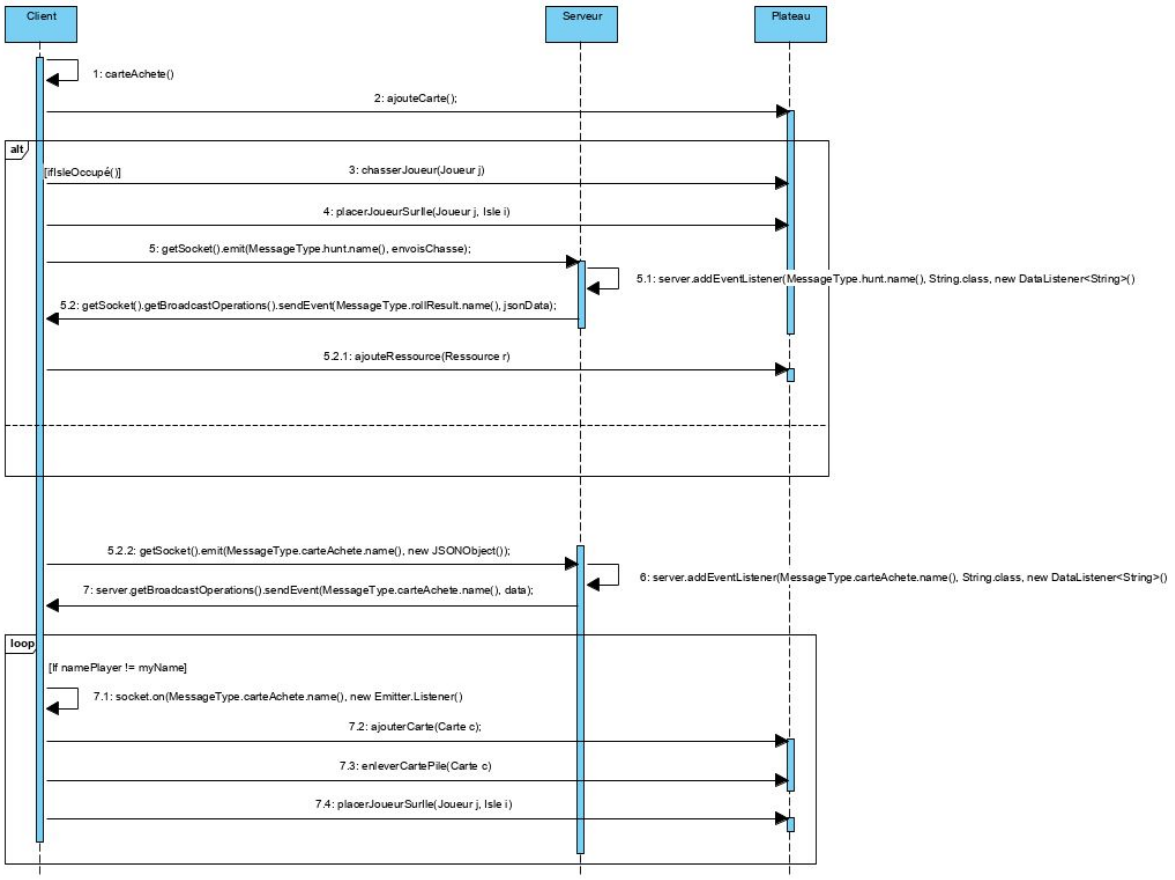
Le CarteMessage contient les informations suivantes

- Le nom du joueur
- La carte achetée
- Le nombre de lunary stone nécessaire à son achat
- Le nombre de solary stone nécessaire à son achat
- Le numéro de l'île sur laquelle elle a été acheté

HuntMessage est envoyé par le client au serveur puis transmis aux autres clients. Le client envoie ce message quand il a du chassé un autre client pour acheter une carte. À la réception de ce message le serveur lance les dés du joueur chassé et transmet le résultat du lancer à tous les joueurs

Le HuntMessage contient les informations suivantes :

- Le nom du joueur chassé



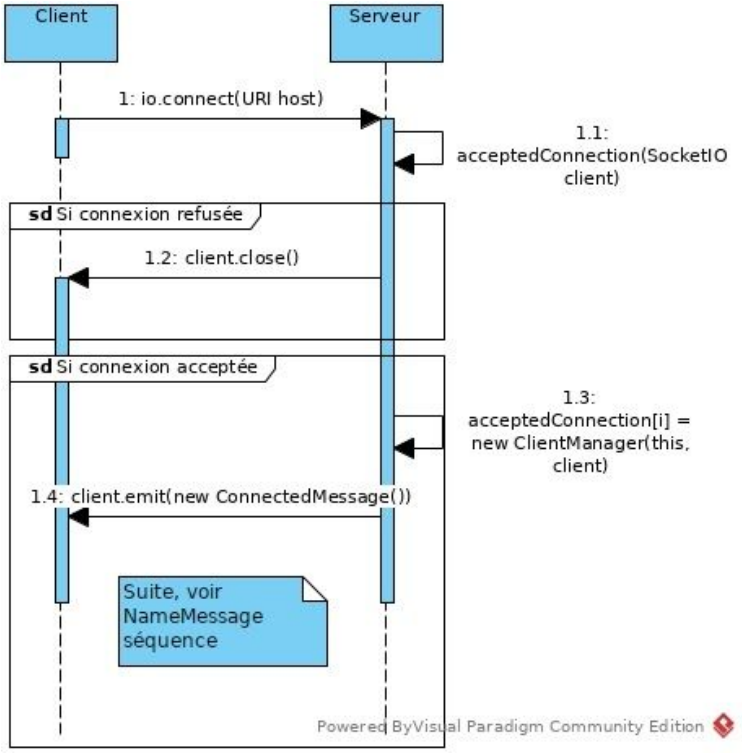
Le client choisit une carte à acheter puis il va la communiquer au plateau pour l'acheter. Si l'île où la carte est acheté est occupé par un joueur, alors le joueur ayant acheté la carte va chasser le joueur positionner sur l'île. L'acheteur est ensuite positionné sur l'île. un message HuntMessage est envoyé vers le serveur qui va roll les dés du joueur chassé. Le résultat du roll est ensuite retourner au joueur chassé qui va ajouter les ressources à son inventaire.

Après l'achat de la carte, le client envoie un CarteMessage vers le serveur qui va ensuite l'envoyer en broadcast aux joueurs. Après la réception, les joueurs qui ne sont pas le joueur qui a acheté la carte, vont ajouter la carte dans l'inventaire du joueur l'ayant acheter, enlever la carte acheté de la pile du plateau et placer le joueur sur l'île.

ConnectedMessage

Message envoyé par le serveur pour notifier à un client que sa connexion a été accepté et qu'il a été ajouté dans la partie

Le message est vide, le serveur envoie juste un événement de type ConnectedMessage



Le client se connecte au serveur, le serveur regarde si les conditions sont remplies pour l'accepter.

Si non il close le socket vers le client.

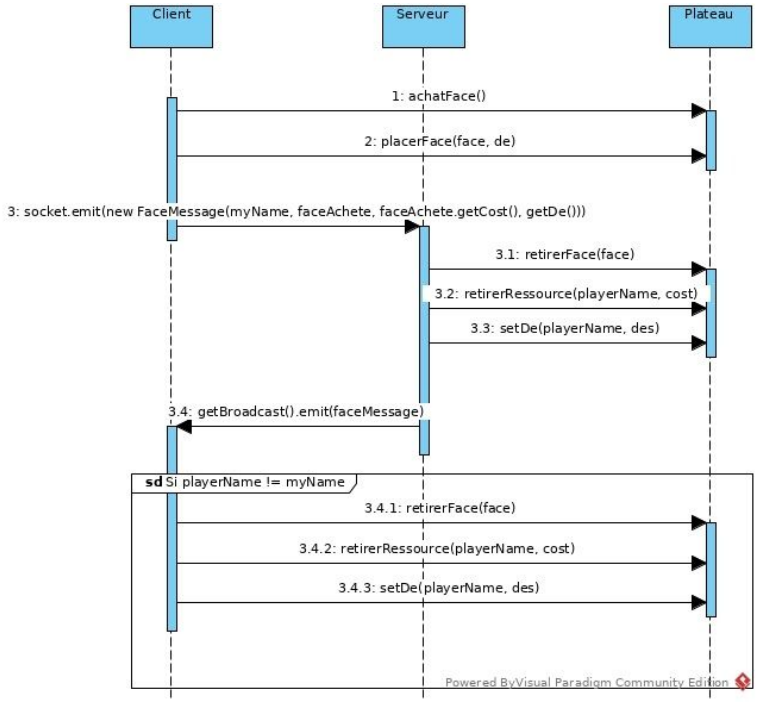
Si oui, il lui alloue un ClientManager et lui envoie un ConnectedMessage pour lui notifier qu'il a été accepté et qu'il attend qu'il lui donne son pseudo.

FaceMessage

Message envoyé par le client pour notifier le serveur puis les autres joueurs qu'il a acheté une Face et qu'il l'a placé sur un de ses dés.

Le message contient les informations suivantes :

- Le nom du joueur
- La face achetée
- Le coût en ressource de la face
- Les dés modifiés du joueur



Le Client achète une face, il place la face sur un de ses dés. Il envoie au serveur un FaceMessage avec son nom, la face, le coût de la face, et ses dés modifiés.

Le serveur retire la face de la forge, il retire les ressources du coût au joueur, puis il actualise les dés avec les dés dans le FaceMessage.

Puis il broadcast le FaceMessage aux clients. Si le client qui reçoit le message a un nom différent que dans le FaceMessage alors il retire la face du message dans la forge, il enlève les ressources du coût au joueur du message et il actualise les dés du joueur.

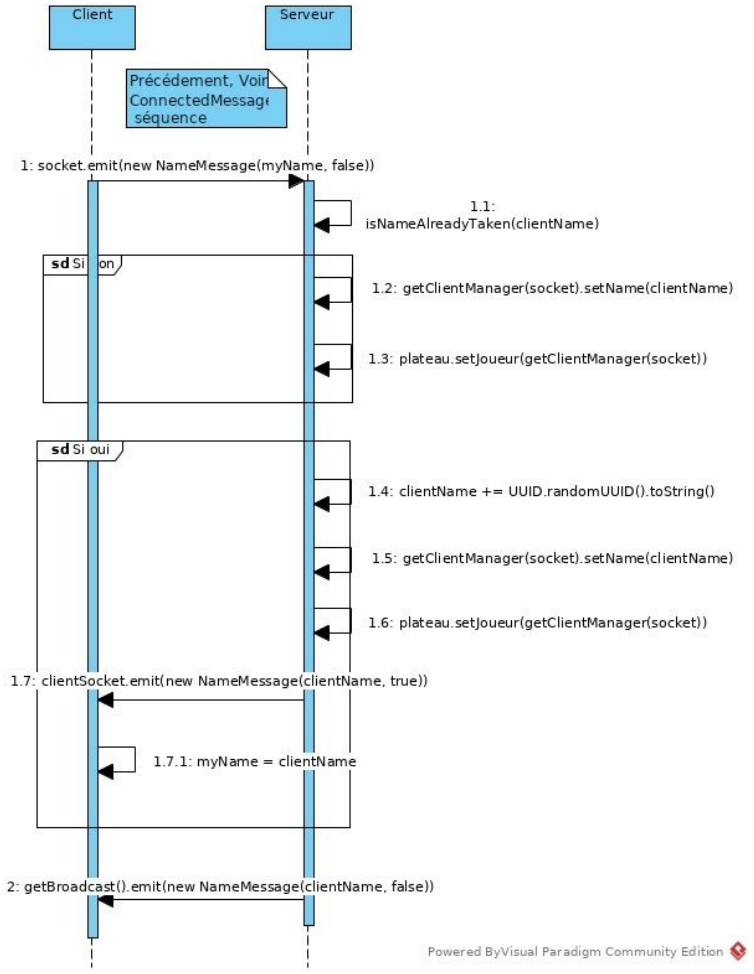
NameMessage

Message envoyé par le Client au Serveur afin que le Client puisse donner son pseudo au Serveur une fois la connexion acceptée.

Si le pseudo donné par le Client est déjà utilisé par un autre Client, le serveur modifie le nom du Client et lui renvoie.

Le message contient les informations suivantes :

- Le nom du joueur
- un boolean qui indique si le serveur a dû modifier le nom du joueur car il était déjà pris.



Le Client envoie son pseudo au serveur, le serveur regarde si le nom n'est pas déjà utilisé par un autre client.

Si non il set le nom du joueur dans son clientManager et il le rajoute dans le plateau.

Si oui il rajoute un UUID derrière le pseudo du joueur, il set le nom et le rajoute dans le plateau. Puis envoie un NameMessage au client avec son pseudo modifié et la valeur du boolean en true, pour indiquer que le pseudo a été modifié.

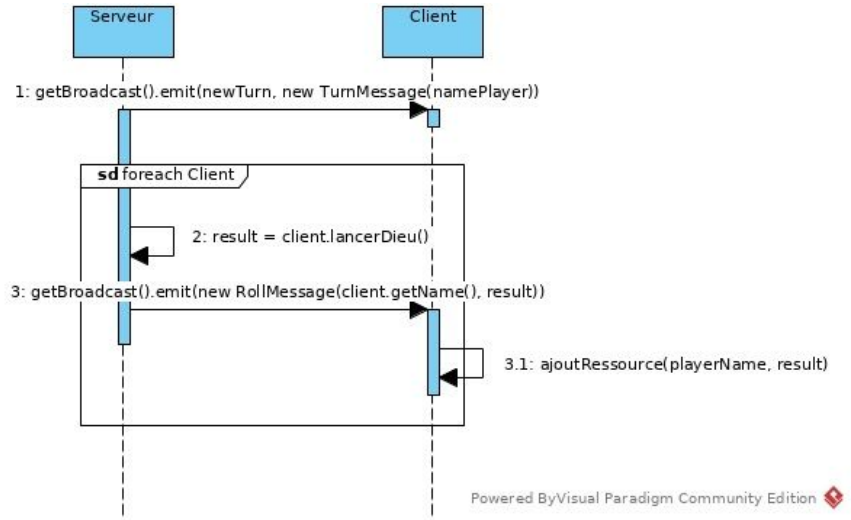
Puis il envoie en broadcast le nom du joueur qui vient de se connecter.

RollMessage

Message envoyé par le Serveur aux Clients afin de leur donner le résultat des lancers de dés.

Le message contient les informations suivantes

- Le nom du joueur à qui est attribué le lancer
- Les faces sur lesquelles il est tombé



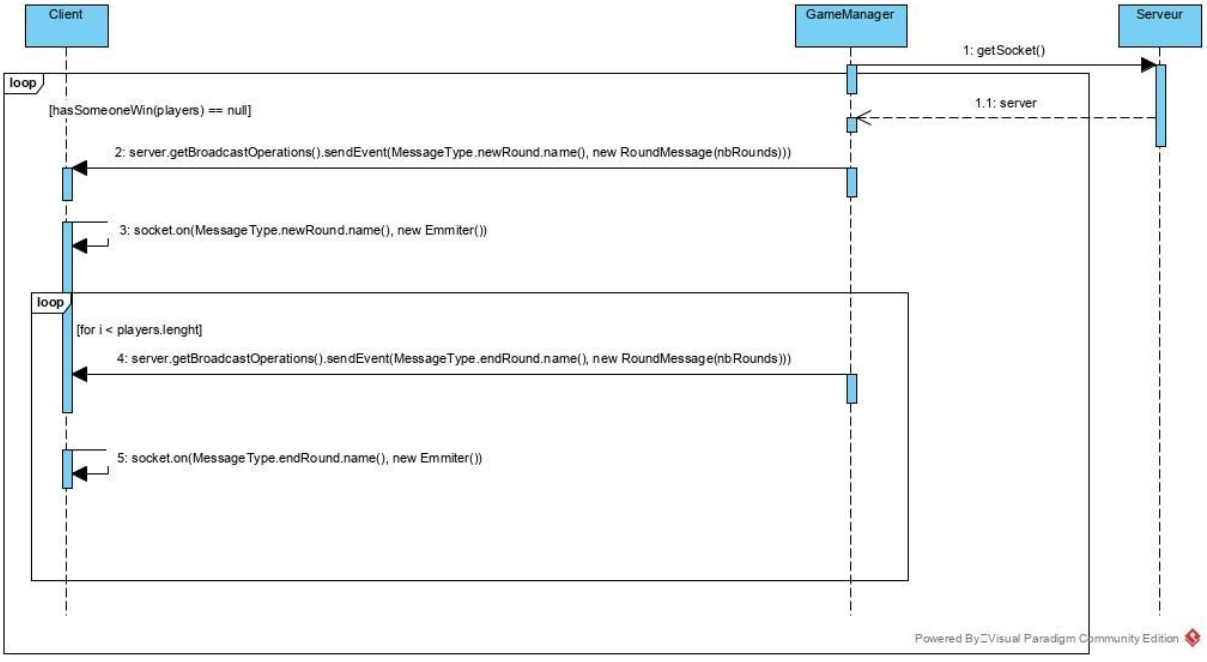
Le serveur broadcast le début d'un nouveau tour en désignant le joueur qui doit jouer. Ensuite pour chaque Client il lance les dés et envoie les résultats en Broadcast avec le nom du joueur et le résultat du lancer. Quand le client reçoit ce message il ajoute les ressources des faces au joueur associé.

RoundMessage

Message envoyé par le serveur aux Clients pour leur indiquer une nouvelle manche

Le message contient les informations suivantes

- Le numéro de manche



Le GameManager broadcast à chaque nouveau tour le numéro de manche tant qu'il n'y a pas de gagnant. Une fois que les joueurs ont joué, le GameManager envoie à chaque joueur un message pour dire que la manche est terminée.

TurnMessage

Message envoyé par le serveur aux Clients pour leur indiquer un nouveau et qui est le joueur qui doit jouer.

Le message contient les informations suivantes

- Le nom du joueur qui doit jouer

Durant la manche, les joueurs reçoivent un message de la part du GameManager qui leurs précise quel joueur est en train de jouer.

VictoryMessage

Message envoyé par le serveur aux Clients pour leur indiquer le gagnant de la partie.

Le message contient les informations suivantes

- Le nom du joueur qui a gagné

Une fois qu'un joueur a atteint le nombre de Victory point nécessaire à la victoire, le GameManager envoie à chaque client quel joueur à gagner et ils l'affichent.

Conclusion

Analyse de votre solution

Point fort

- Comme chaque client a sa version du plateau, quand un client doit effectuer une action il n'a pas besoin de demander toutes les informations du plateau au serveur, on limite donc le nombre de messages envoyés.
- Comme le serveur a aussi sa version du plateau, il peut vérifier si les actions des clients est légal ou non.

Point faible

- Chaque joueurs a sa version du plateau, il ne demande pas en temps réel les informations du plateau au serveur. Cela nécessite de devoir le synchroniser et de voir s'il n'y a pas des bugs de synchronisation/désynchronisation.
- Pour le moment notre solution fonctionne mal avec les interactions serveur/client. Le serveur s'exécute jusqu'à la fin de la partie puis donne les informations aux joueurs. Il faudrait que le serveur attende les interactions des clients quand leurs interactions est nécessaire.

Évolution prévue

Ajouter une classe ServerListener et ClientListener qui reprendront le code qui définit les actions à exécuter en fonction des messages reçus, déjà présent dans les classes Server et Client.

Ajouter un plateau dans le serveur, pour que le serveur ait sa version du plateau et qu'il puisse savoir si un joueur triche ou pas.

Changer la fin de la partie, pour le moment la fin de la partie est sur le nombre de victory point possédé par les joueurs, tant que un nombre n'est pas atteint le serveur créer des nouvelles manches. Il faudrait que l'on change pour que la fin de la partie se déclenche à la fin de la 9ème manche.

Le joueur peut à cet instant acheter des cartes, mais elles ne lui servent à rien car le joueur ne convertit pas les HP des cartes en victory point et les sorts ne fonctionnent pas encore. Il faudrait donc que l'on ajoute qu'à la fin de la partie les HP des cartes sont ajoutés au score du joueur et qu'ils puissent déclencher les effets des cartes.

Pour le moment on ne peut lancer que les 2 dés d'un Joueur. Il faudrait que l'on rajoute le fait que le joueur puisse choisir un dé et le lancer afin de pouvoir réaliser l'effet "*faveur des dieux mineures*".

À la fin de son action, le tour du joueur prend fin. Il faudrait ajouter que le serveur demande au joueur à la fin de sa première action s'il souhaite effectuer une nouvelle action s'il possède 2 Solary Stone. Si la réponse est positive qu'il puisse ré-effectuer une action.