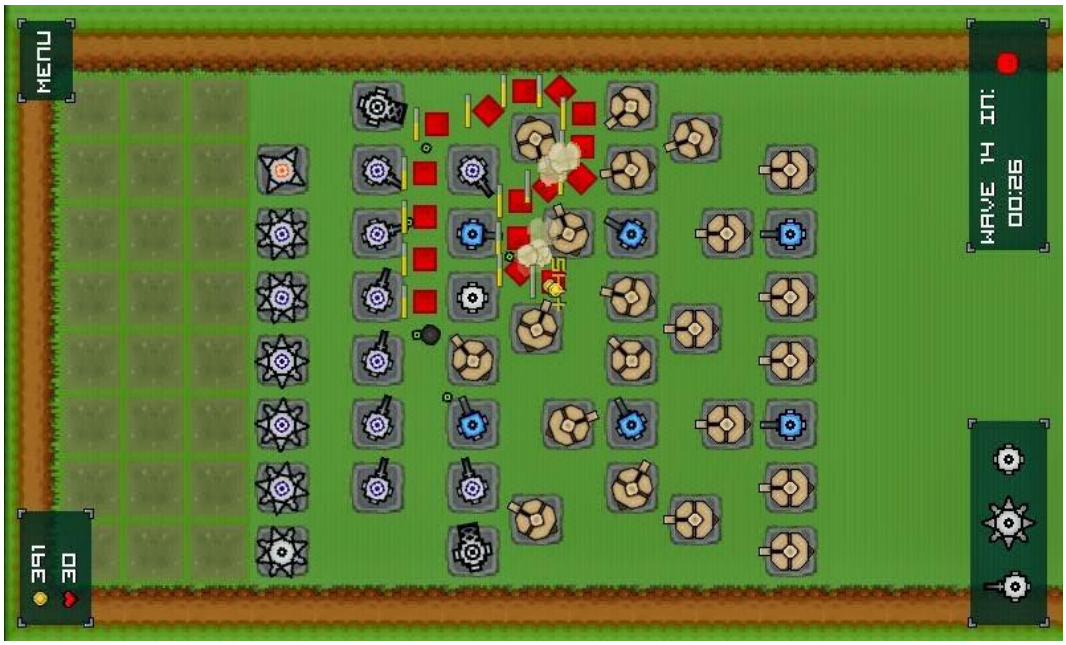
Dossier d'analyse et de conception

TOWER DEFENSE



Lescaret Elisa, Auguste Bastien, Briant Arnaud, Deladerrière Balthazar, Keil Cyril, Mamet Nicolas, Pipino Jérôme

13/11/2014

Table des matières

Introduction

Le projet consiste en la conception et le développement d'un jeu de Tower Defense par différents groupes. Chaque groupe doit créer une application, et toutes les applications développées dans ces groupes doivent être capables de fonctionner entre elles, afin que les différents groupes puissent jouer entre eux, tout en utilisant leur propre application.

Le Tower Defense est un genre de jeu dans lequel deux joueurs s'affrontent en envoyant des monstres dans le camp adverse afin de le détruire. Pour se défendre, chaque joueur a la possibilité de créer des tourelles possédant divers effets. Toute action coûte de l'argent, qui sera gagné à chaque début de tour en fonction des agissements du joueur.

Notre application fonctionnera au tour par tour, découpé en plusieurs phases : une phase de production des monstres, une phase de placement des tourelles, et enfin une phase d'attaque, gérée sans intervention du joueur. Le jeu se déroulera dans un univers steampunk (sous-genre de la science-fiction se déroulant généralement au 19e ou au début du 20e siècle, et devant son nom à l'utilisation massive de technologies en avance sur leur temps, mais à base de machines à vapeur).

Cahier des charges

I. Description du jeu et de ses éléments

A. Le terrain

Le terrain sera découpé dans un quadrillage. Il n'y a qu'un spawn (voir plus bas) de monstres, ainsi qu'une base ennemie.

Il existe différents types de terrains :

- **La plaine** : Terrain de base sur lequel on peut poser toutes les tourelles. Aucun bonus ou malus de terrain pour les monstres.

- **La forêt** : Terrain sur lequel on peut poser toutes les tourelles. Confère un malus sur la vitesse de déplacement du monstre, sauf sur les monstres aériens.

- **La rivière** : Ce terrain ralentit les monstres passant dedans, ou empêchent certains monstres d'y passer. Les monstres aériens ne sont pas affectés par ce terrain. Impossible de placer des tourelles sur ce terrain.

- **Le pont** : Terrain permettant de franchir une rivière. On ne peut pas y poser de tourelle. Aucun bonus ou malus de terrain pour les monstres.

- **La montagne** : Terrain difficile à franchir, réduit la vitesse de monstres sauf des monstres aériens. Les monstres résistants ne peuvent pas franchir la montagne. Seules des tourelles spécifiques à la montagne peuvent y être posées.

- **Le spawn** : Case sur laquelle les monstres apparaissent, et depuis laquelle ils doivent rejoindre la base ennemie. Impossible de placer une tourelle sur cette case. Aucun bonus ou malus de terrain pour les monstres.

- **La base** : Case que les monstres ennemis doivent rejoindre pour l'attaquer. Aucun bonus ou malus de terrain pour cette case. Impossible de placer une tourelle sur cette case.

B. Les monstres

Un monstre fait une case d'épaisseur. Les monstres possèdent tous les mêmes caractéristiques de bases :

- points de vie ;
- vitesse de déplacement ;
- points d'attaque.

Les monstres sont quant à eux classés dans différentes catégories :

- **Monstres de base** : Monstres basiques sans bonus ni malus.
  ⇒ Points de vie : **
  ⇒ Vitesse de déplacement : **
  ⇒ Points d'attaque : **

- **Monstres rapides** : Monstres se déplaçant plus vite que les monstres de base. De par leur légèreté, ils ne peuvent pas passer par la rivière.
  ⇒ Points de vie : *
  ⇒ Vitesse de déplacement : ***
  ⇒ Points d'attaque : *

- **Monstres volants** : Monstres pouvant voler au-dessus des terrains, et ne peuvent être ciblés que par certaines tourelles. Ces monstres peuvent passer au-dessus des rivières et des montagnes sans être ralentis.
  ⇒ Points de vie : *
  ⇒ Vitesse de déplacement : **
  ⇒ Points d'attaque : **

- **Monstres invisibles** : Monstres ne pouvant pas être détectés et donc attaqués que par certaines tourelles.
  ⇒ Points de vie : *
  ⇒ Vitesse de déplacement**
  ⇒ Points d'attaque : **

- **Monstre résistant** : Monstres ne pouvant pas passer les montagnes à cause de leur poids.
  ⇒ Points de vie : ***
  ⇒ Vitesse de déplacement : *
  ⇒ Points d'attaque : ***

Différents types de monstres auront donc parfois un chemin différent pour arriver à la base ennemie. Les monstres ne peuvent pas traverser les tourelles mises en place par l'ennemi. En cas d'impossibilité de trouver un chemin, le monstre deviendra fou, et déclenchera l'arrivée du Goliath pour résoudre le problème.

Le **Goliath** est un monstre différent des autres. Il est invoqué sur le terrain automatiquement, ne coûte rien et est invincible tout en ne faisant aucun dommage à la base ennemie, et permet seulement de nettoyer le chemin vers la base ennemie afin que le jeu puisse continuer en cassant une ou plusieurs tourelles.

C. Les tourelles

- **Zone d'effet** (En croix, en cercle, en ligne droite) ;
- **Portée** (distance d'attaque et de détection des monstres);
- **Vitesse d'attaque** ;
- **Dégâts d'attaque** ;
- **Type** (une tourelle peut avoir plusieurs types) :
    - ralentissement des ennemis touchés ;
    - discernement d'ennemis, pour voir les ennemis invisibles ;
    - anti-aérienne, afin de toucher les ennemis volants ;
    - empoisonnante/brulante : inflige des dégâts pendant un laps de temps à définir au monstre touché par cette tourelle.

Les caractéristiques de base d'une tourelle peuvent être améliorées via un achat en pièces d'or (monnaie du jeu).

D. Autres règles de jeu

L'argent sera distribué au joueur à chaque début de tour. Plus on dispose de tourelles sur le terrain, plus on gagne d'argent au début du tour.

Concernant les phases du tour de jeu, elles sont définies de cette façon :

Phase 1 : Réception de l'argent, choix des unités à produire, puis les deux joueurs valident la fin de la phase 1 afin de passer à la phase 2.

Phase 2 : On reçoit la liste des différents types de monstres qui vont arriver. Après avoir consulté la liste, on achète des tourelles que l'on place sur le terrain. Une fois que l'on a fini de placer ses tourelles, les deux joueurs valident la fin de la phase 2, afin de lancer la phase d'attaque.

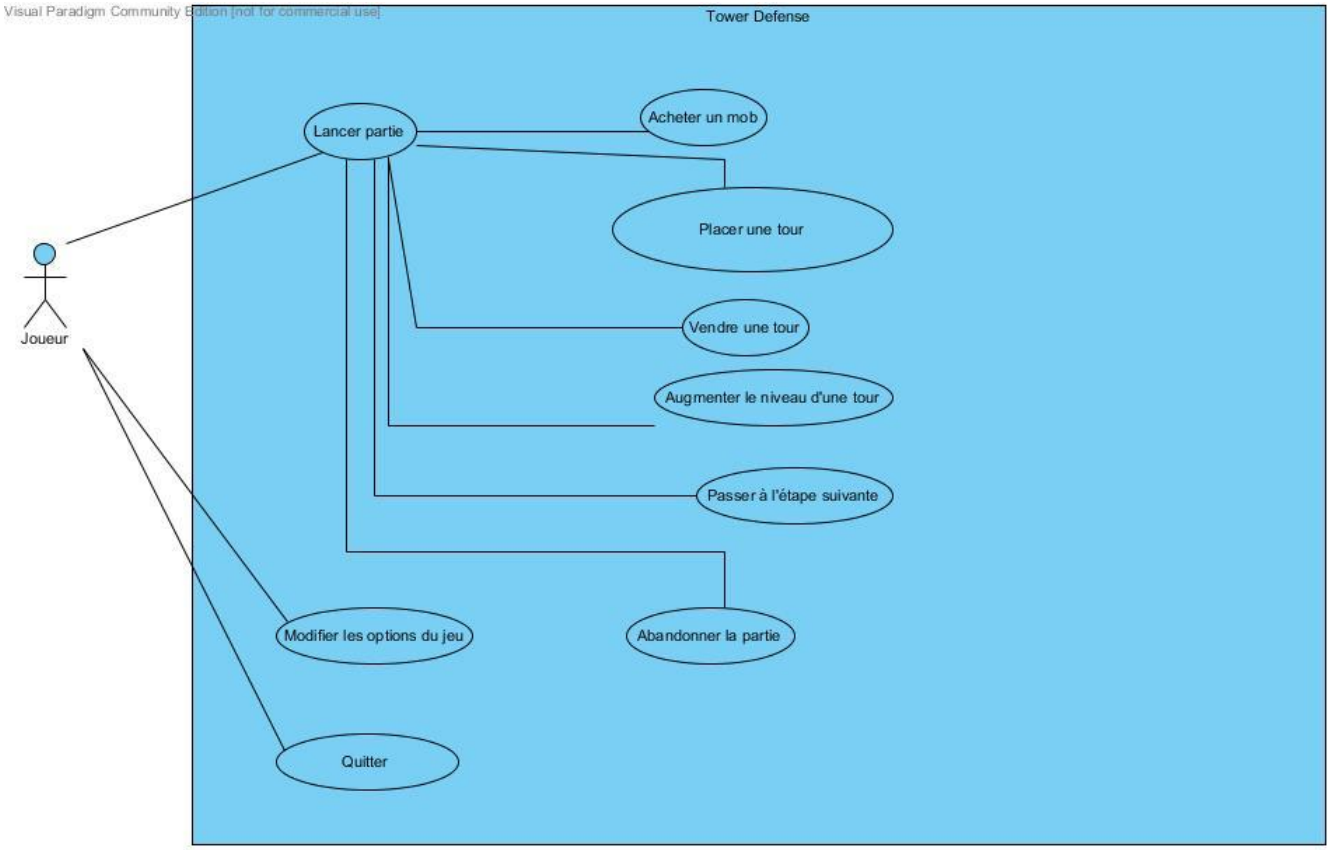
Phase d'attaque : il n'est plus possible d'acheter des monstres, ou de construire des tourelles. (Option envisagée : permettre de dépenser de l'or dans cette phase afin de lancer des attaques sur les ennemis arrivant, ou pour lancer des bonus sur ses monstres).

II. Les acteurs

La responsable de la maitrise d'ouvrage (MOA) est Elisa Lescarret et le chef de projet (MOE) est Nicolas Mamet. L'équipe projet est composée des membres restants, c'est-à-dire : Bastien Auguste, Arnaud Briant, Balthazar Deladerrière, Cyril Keil et Jérôme Pipino.

III. Les cas d'utilisation

A. Diagramme des cas d'utilisation



B. Description textuelle d'un cas d'utilisation

Lancer la Partie : Permet au joueur via un clic sur un bouton dans le menu principal de démarrer une nouvelle partie du jeu de Tower Defense. Cette partie sera débutée au point nul. C'est-à-dire au premier tour, avec aucun achat de réalisé, un terrain vierge, et donc l'argent de base donné par le jeu.

Modifier les Options du Jeu : Permet au joueur via un clic sur un bouton dans le menu principal d'accéder à un menu d'Options dans lequel il pourra modifier certaines caractéristiques du jeu comme le volume général de celui-ci, le volume des SFX, celui de la musique ou encore d'autres options.

Quitter : Permet au joueur via un clic sur un bouton dans le menu principal de quitter l'application.

Acheter un mob : Si le joueur possède suffisamment d'argent, le bouton "acheter" à côté d'un mob dans le menu des mobs sera activé. En cliquant sur celui-ci, le joueur pourra acquérir le mob à côté duquel se trouve le bouton. Celui-ci sera alors ajouté à la vague et sera envoyé à la fin du tour.

Placer une tourelle : Si le joueur possède suffisamment d'argent, le bouton "acheter", à côté d'une tourelle dans le menu des tourelles, sera activé. En cliquant sur celui-ci, le joueur

pourra acquérir la tourelle à côté de laquelle se trouve le bouton. Celle-ci devra ensuite être placée sur le terrain par le joueur. Pour ce faire, celui-ci doit ensuite cliquer sur un emplacement libre du terrain pour positionner la tourelle.

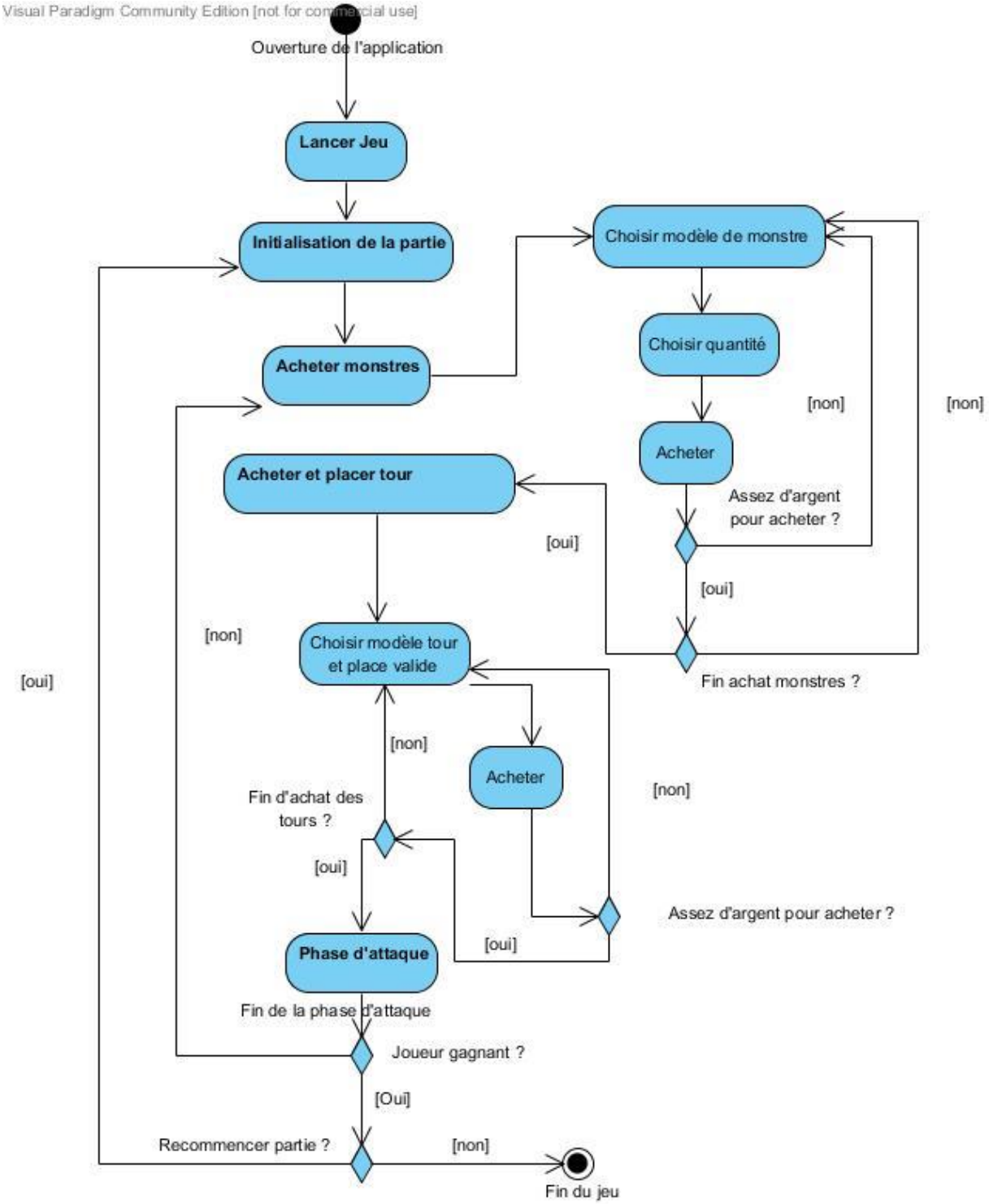
Vendre une tourelle : Lorsque le joueur clique sur une tourelle il peut voir ses caractéristiques. Dans le menu de caractéristiques, un bouton "Vendre" est activé. En cliquant sur celui-ci, le joueur retire la tourelle du terrain et récupère une partie de la somme d'argent qu'il a investi dans celle-ci.

Augmenter le niveau d'une tourelle : Lorsque le joueur clique sur une tourelle il peut voir ses caractéristiques. Dans le menu de caractéristiques, un bouton "Lvl-up" sera activé si le joueur possède suffisamment d'argent. En cliquant sur celui-ci, le joueur va débourser une somme d'argent et augmenter toutes les caractéristiques de la tourelle.

Passer à l'étape suivante : A n'importe quel moment dans une partie. Le joueur peut cliquer sur le bouton "Finir le tour". En faisant ceci il ne peut plus effectuer d'action sur son terrain. Dès que les deux joueurs ont cliqué sur ce bouton, leurs vagues respectives se lancent. Les joueurs reprennent le contrôle dès que la vague est terminée.

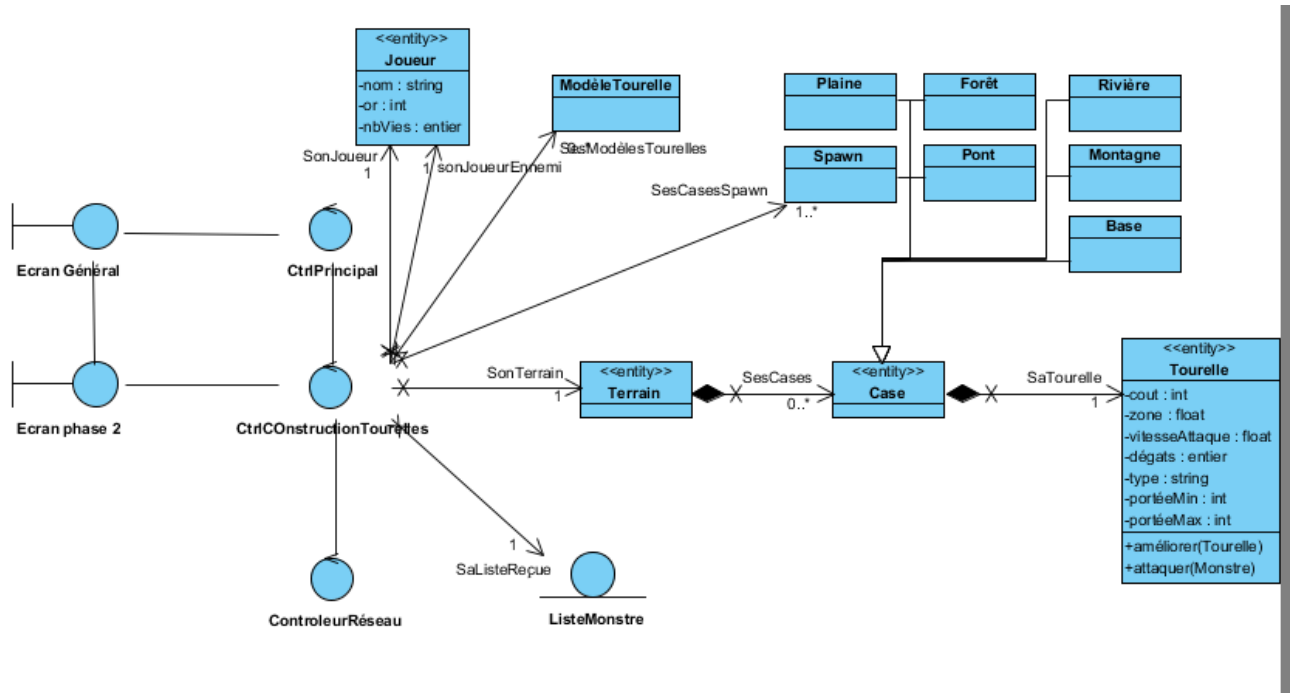
Abandonner la partie : A n'importe quel moment dans une partie. Le joueur peut cliquer sur le bouton "Abandonner". En faisant ceci il abandonne son droit de gagner la partie. Le joueur adverse est déclaré gagnant, et le joueur revient au menu principal.

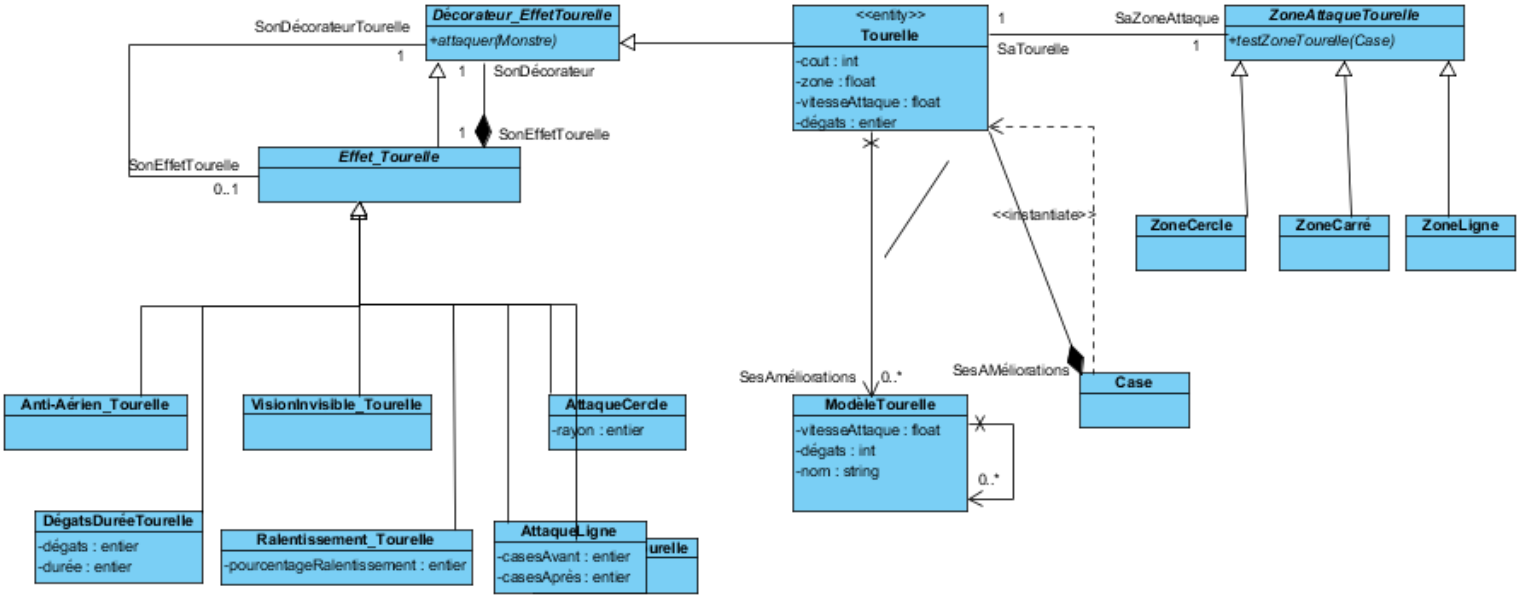
IV. Fonctionnement général de l'application



Analyse et Conception de l'application

I. Les tourelles





La classe tourelle utilise le pattern design décorateur, car on dispose de plusieurs effets sur une tourelle qui peuvent agir en même temps.

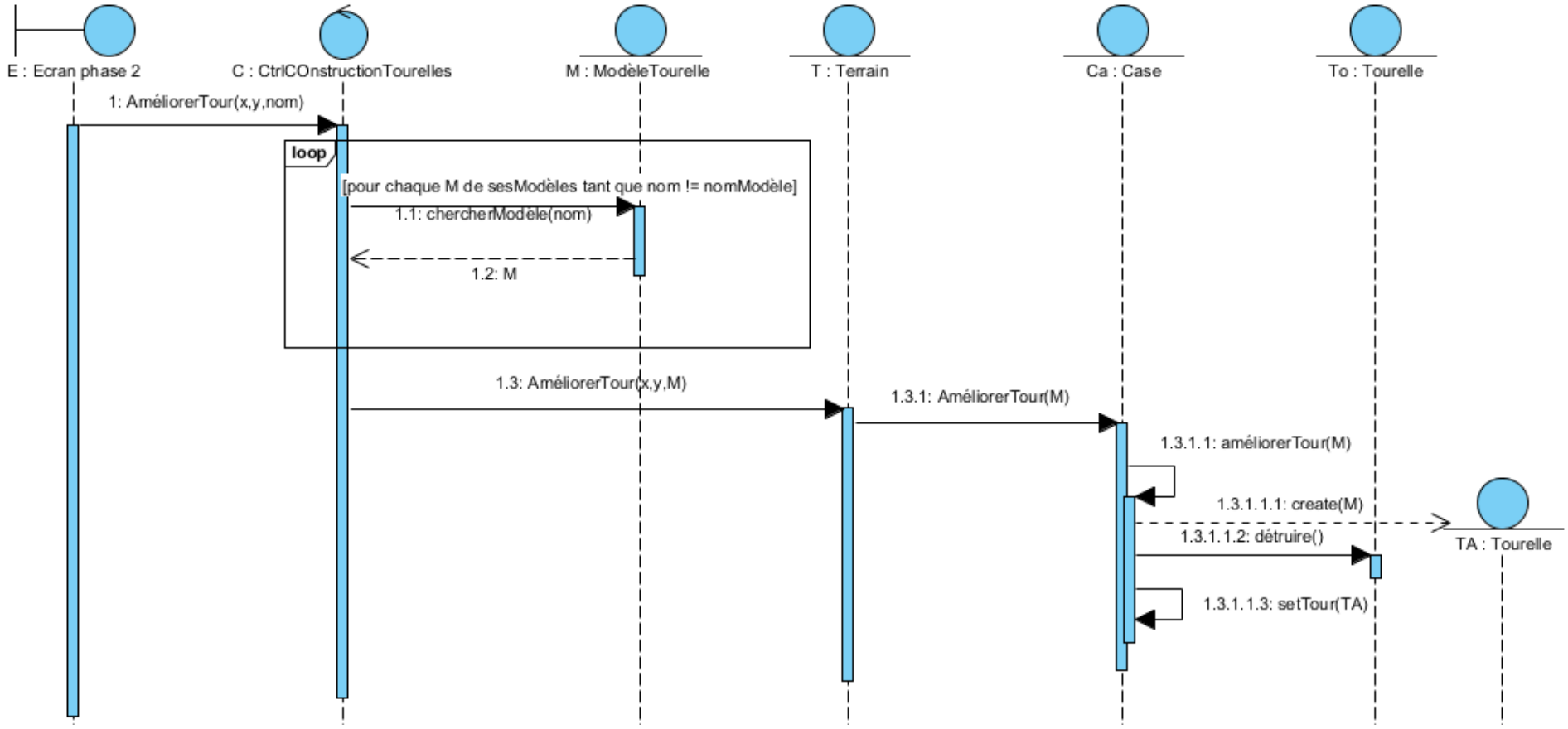
La tourelle dispose d'un effet, qui lui-même dispose d'un autre effet, et ainsi de suite. Les effets sont donc ajoutés à une tourelle de façon récursive, ce qui permet d'avoir un nombre d'effets illimité dans le jeu.
La tourelle peut attaquer les autres monstres. Pour cela, elle doit d'abord tester sur les cases environnantes si elle a un monstre à attaquer. Afin de réaliser ce test, elle doit effectuer des traitements en fonction de sa zone d'attaque. On dispose donc de trois classes ZoneCercle, ZoneCarré et ZoneLigne, qui vont définir la zone d'attaque de la tourelle, dont la classe abstraite ZoneAttaqueTourelle se servira afin de tester si un monstre est bien dans sa zone d'attaque. Les monstres invisibles ne seront pas vus par la tourelle si elle ne dispose pas de l'effet nécessaire.

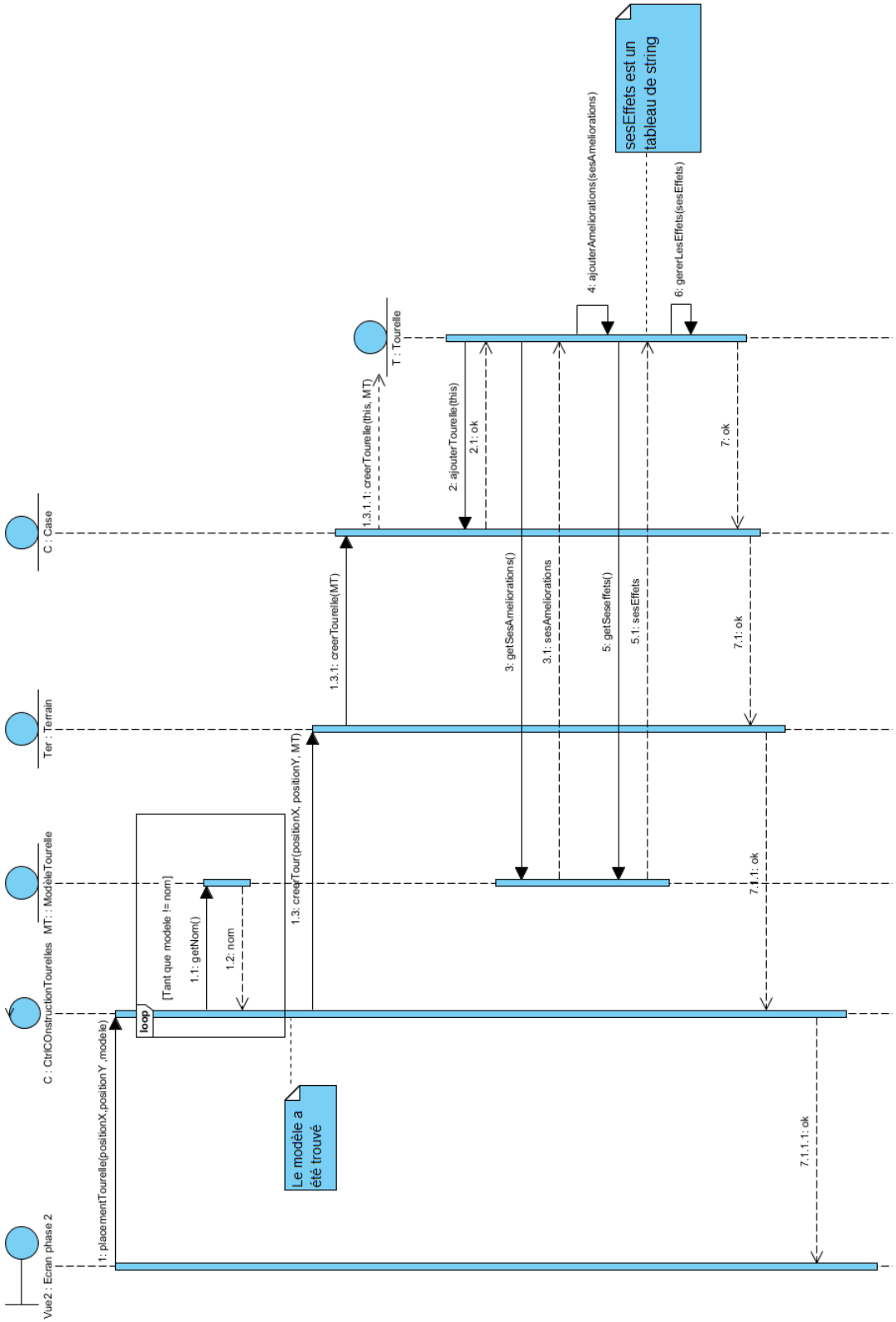
La classe ModèleTourelle contient les caractéristiques d'un type de tourelle. Cette classe permet de créer une tourelle selon le type choisi. Chaque ModèleTourelle contient les références vers les améliorations possibles de tourelles de ce modèle.

La tourelle est placée sur une case. C'est la case qui a la responsabilité de créer une tourelle. La case est contenue dans le terrain.

Une fois que la tourelle a trouvé un monstre, elle doit l'attaquer, grâce à la fonction contenue dans la classe abstraite Décorateur_EffetTourelle contenant les différents effets de la tourelle. Si une tourelle a l'effet « vision invisible », il faut que celui-ci soit enregistré comme l'effet de la tourelle ou comme l'effet du « dernier » effet de la tourelle.
Lorsque la tourelle attaque un monstre, la fonction attaque(Monstre) de la tourelle est lancée. Elle appelle la fonction attaque(Monstre) de son effet. Celle-ci appelle également la fonction attaque(Monstre) de son effet et ainsi de suite.

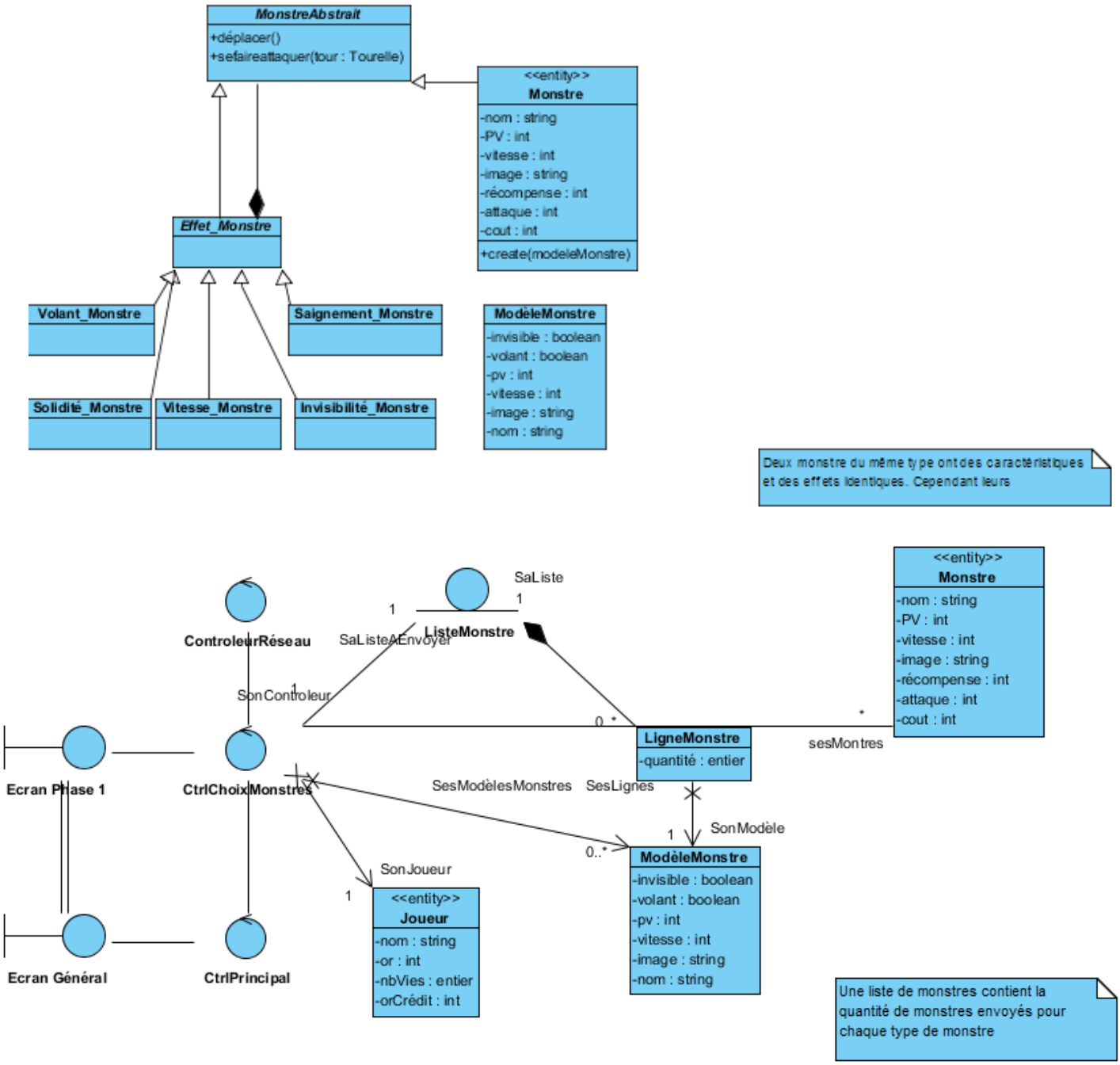
Enfin, concernant l'amélioration de la tourelle, elle est réalisée à travers la classe ModèleTourelle. Les améliorations achetées sur une tourelle ne seront liées qu'à cette tourelle, et elles peuvent affecter différents attributs de la tourelle : La vitesse d'attaque, la zone d'attaque, les dégâts, et également les effets présents sur la tourelle. Les effets présents sur la tourelle ne peuvent être améliorés, seule la tourelle en elle-même peut être améliorée.


E : Ecran phase 2
C : CtrlCOnstructionTourelles
M : ModèleTourelle
T : Terrain
Ca : Case
To : Tourelle
1: AméliorerTour(x,y,nom)
loop
[pour chaque M de sesModèles tant que nom != nomModèle]
1.1: chercherModèle(nom)
1.2: M
1.3: AméliorerTour(x,y,M)
1.3.1: AméliorerTour(M)
1.3.1.1: améliorerTour(M)
1.3.1.1.1: create(M)
TA : Tourelle
1.3.1.1.2: détruire()
1.3.1.1.3: setTour(TA)



Lorsqu'une tour est achetée, il faut la placer sur la carte. Pour cela, on a besoin de ces coordonnées par rapport au terrain puis la case concernée du terrain « construit » la tourelle. Après la création de la tourelle de base, elle ajoute toutes ses améliorations et ses effets à sa panoplie.

II. Les monstres



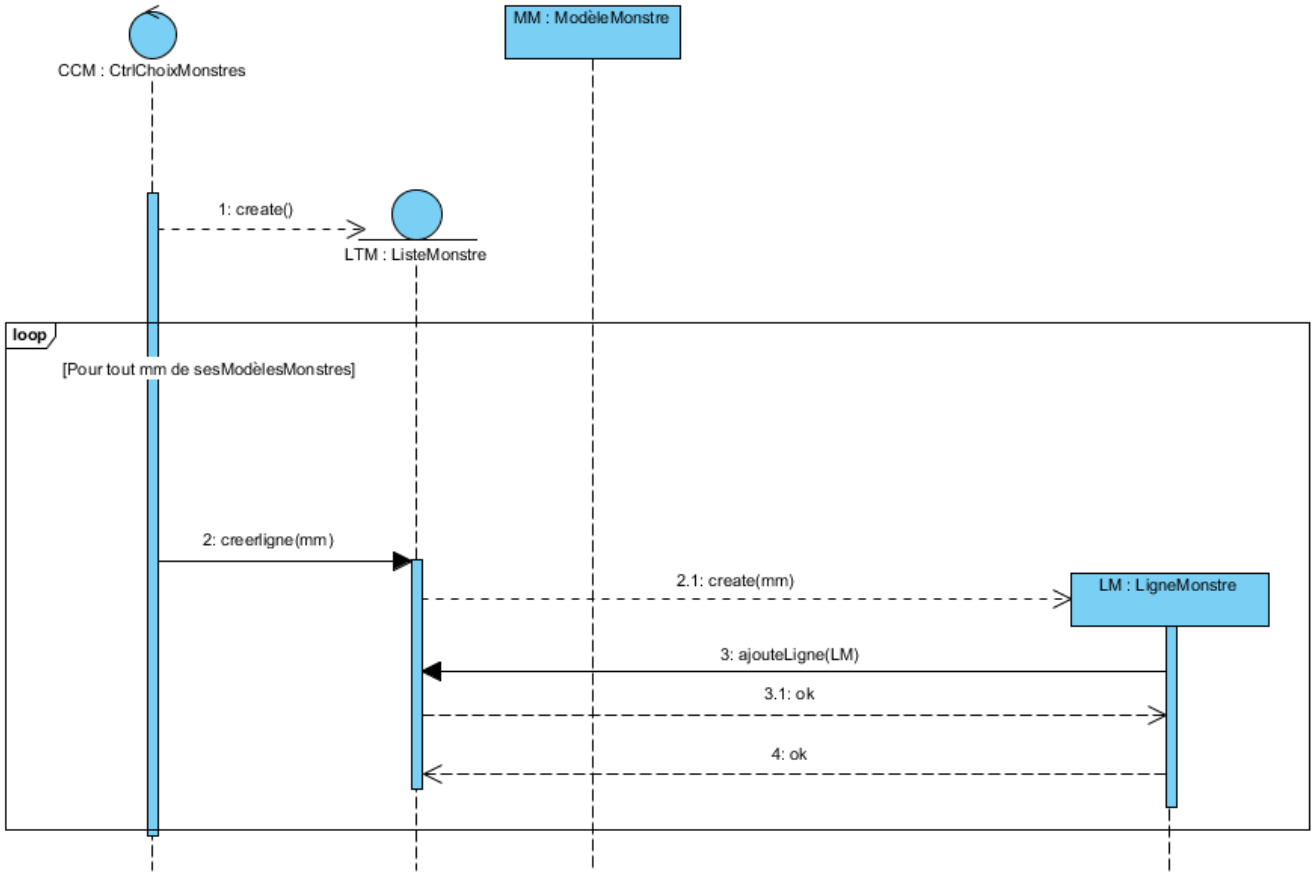
L'entité principale de ce diagramme est la classe Monstre, elle contient les caractéristiques du monstre :
-son nom,
-ses PV (points de vie),
-son coût (en pièces d'or),
-vitesse (en cases par seconde),
-image (élément graphique représentant le monstre),
-récompense (nombre de pièces d'or que gagne le défenseur en tuant ce monstre).

Elle possède le constructeur Monstre(modeleMonstre) qui attend un objet de la classe ModèleMonstre qui contient les paramètres de base de chaque monstre (pv de base, vitesse de base, etc..).

Elle hérite de la classe abstraite MonstreAbstrait qui utilise les opérations « déplacer() », « sefaireattaquer(tour: Tourelle) ».

Effet_Monstre hérite et dépend aussi de MonstreAbstrait, elle possède elle même 5 filles représentant chaque paramètre du monstre, elles implémentent déplacer() et sefaireattaquer().

Déplacer() affecte le déplacement des monstres, c'est-à-dire que selon le monstre qui devra se déplacer sur la carte, celui-ci aura un mouvement différent selon son type, par exemple les volants pourront survoler les tours alors que les autres non. De plus, les monstres ont une certaine vitesse et elle peut être altérée par des ralentissements causés par des tours.
Sefaireattaquer() permet de savoir si un monstre sera cible d'une tour et les effets de cette attaque. C'est-à-dire que si c'est un monstre invisible, il ne se fera pas attaquer par une tour sans « vision invisible ». Les effets de l'attaque peuvent être un ralentissement, ou un empoisonnement.



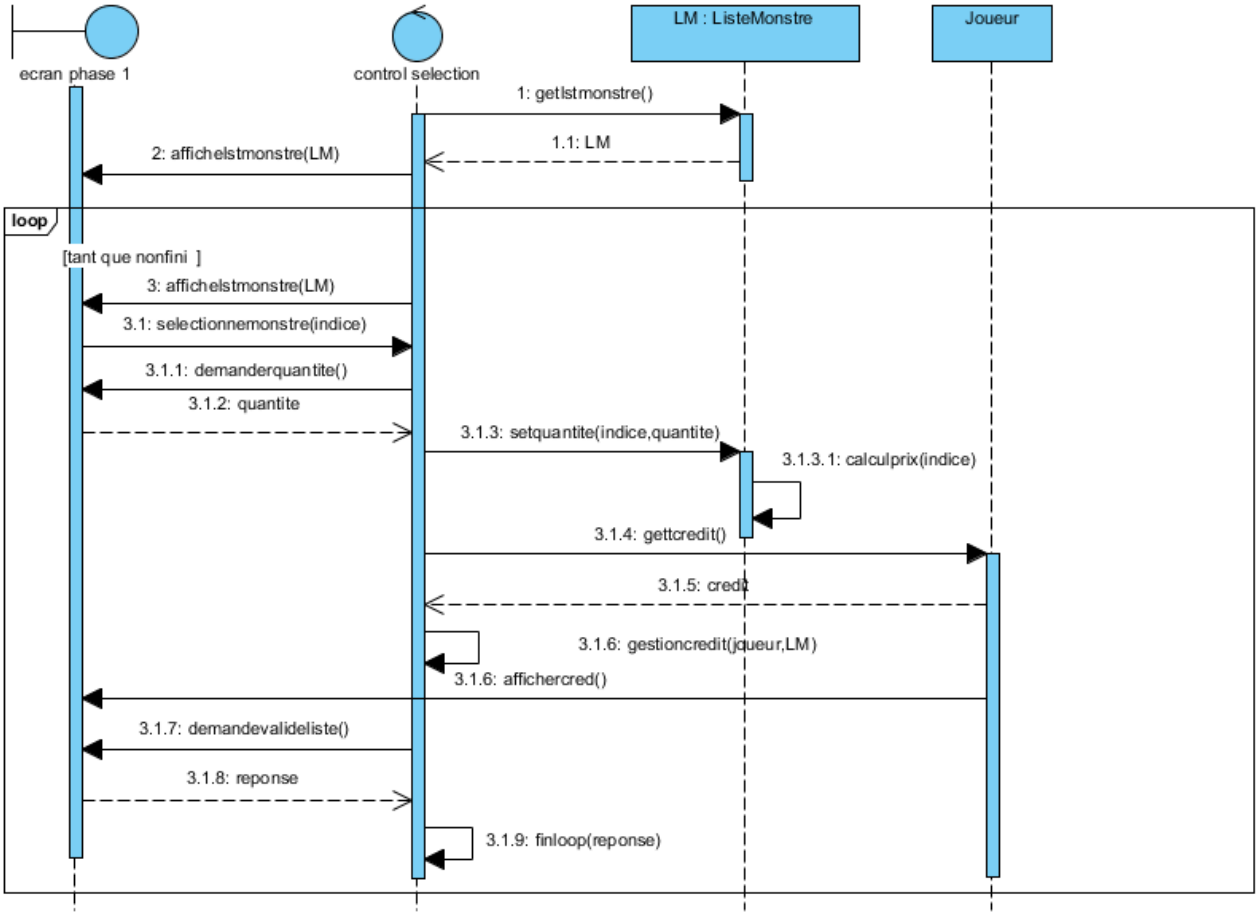
Le diagramme créerListeMonstre modélise la création d'un objet ListeMonstre contenant l'ensemble des différents monstres (modèles) et leur quantité (mis à zéro par défaut).

La création des LigneMonstre est indispensable pour la première phase. En effet lors de cette phase les listes de types de monstres sont créées. Ainsi lorsque le joueur choisi un monstre, il modifie seulement sa quantité.

Methodes utilisées lors de l'appel de creerListeMonstre :

- creerligne(mm) : crée une ligne de ModeleMonstre en appelant sont constructeur avec un Modele Monstre en parametre et met la quantité pour chaque monstre à 0.
- ajouterligne() : permet d'ajouter la ligne créée à ListeMonstre

La boucle permet de répéter l'opération pour tous les ModeleMonstre.



Description générale de sélection monstre:

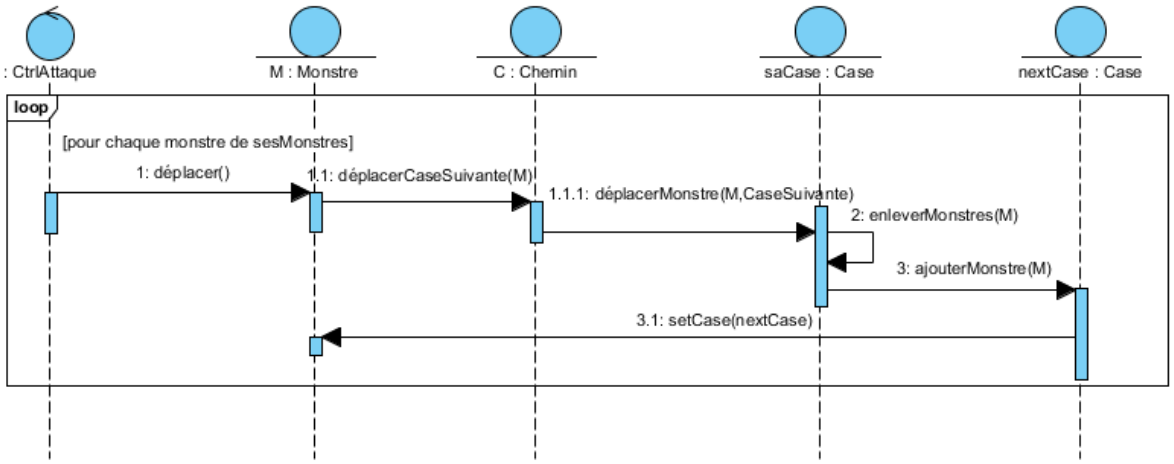
Permet de sélectionner des monstres de ListeMonstre, qui a été initialisée auparavant comme expliqué dans le diagramme créerListe et de leur affecter la quantité voulue par le joueur.

Sélection monstre modélise aussi la gestion des crédits (or) du joueur, les diminier lorsque le joueur achète quelque chose (après avoir controlé qu'il y avait suffisamment d'or pour l'acheter), etc.

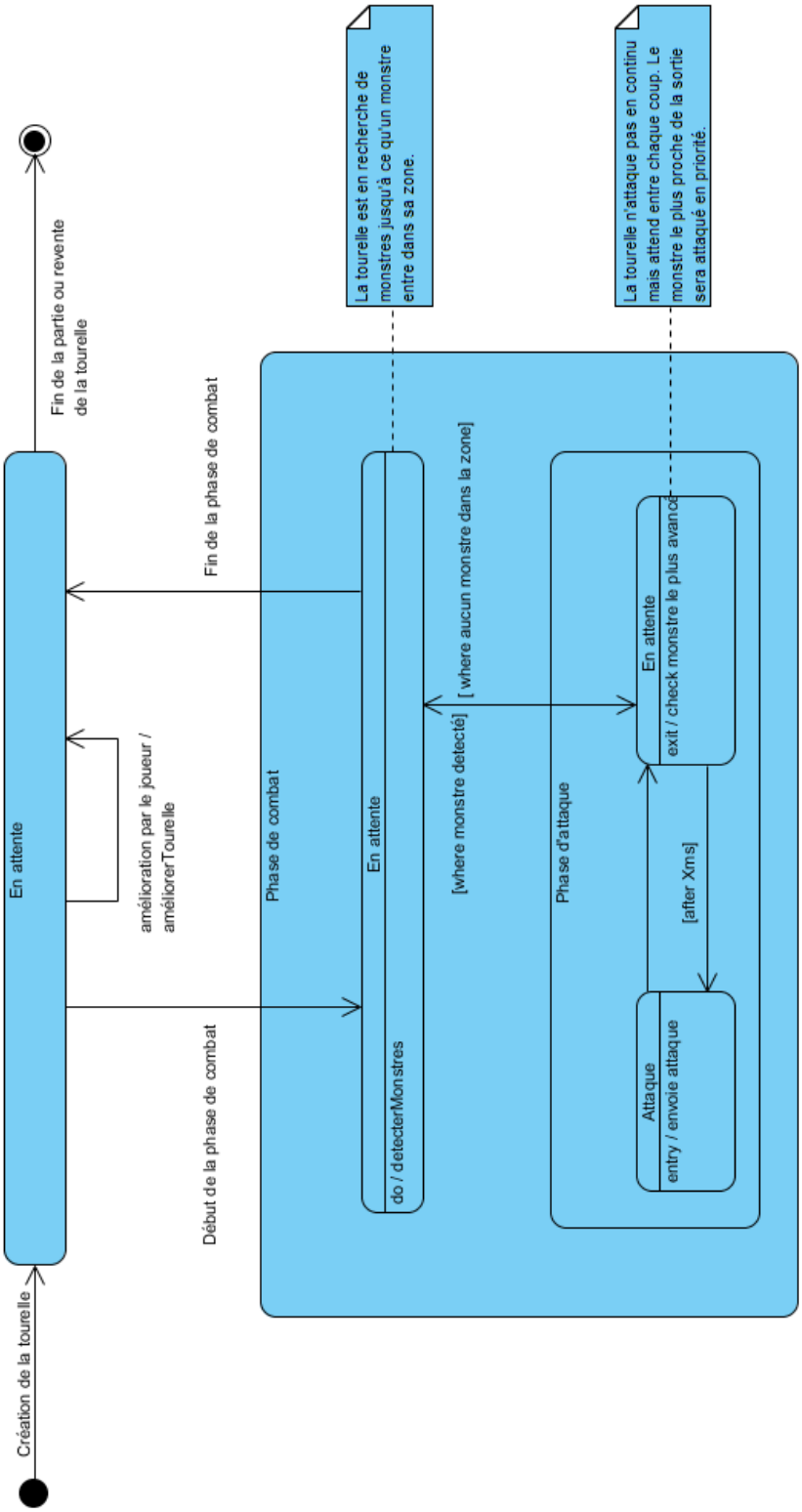
Méthodes:

- getlistmonste() : récupère ListeMonstre
- affichelstmonstre(LM) : affiche ListeMonstre
- selectionnemonstre(indice) : sélectionne un monstre de ListeMonstre
- demanderquantité() : demande au joueur de choisir une quantité pour le monstre sélectionné
- setquantite(indice,quantite) : définit la quantité de monstres pour la ligne de ListeMonstre selectionné.
- gestioncredit(joueur,LM) : gère tout ce qui concerne les crédits (voir diagramme d'activité gestion crédit pour le fonctionnement)

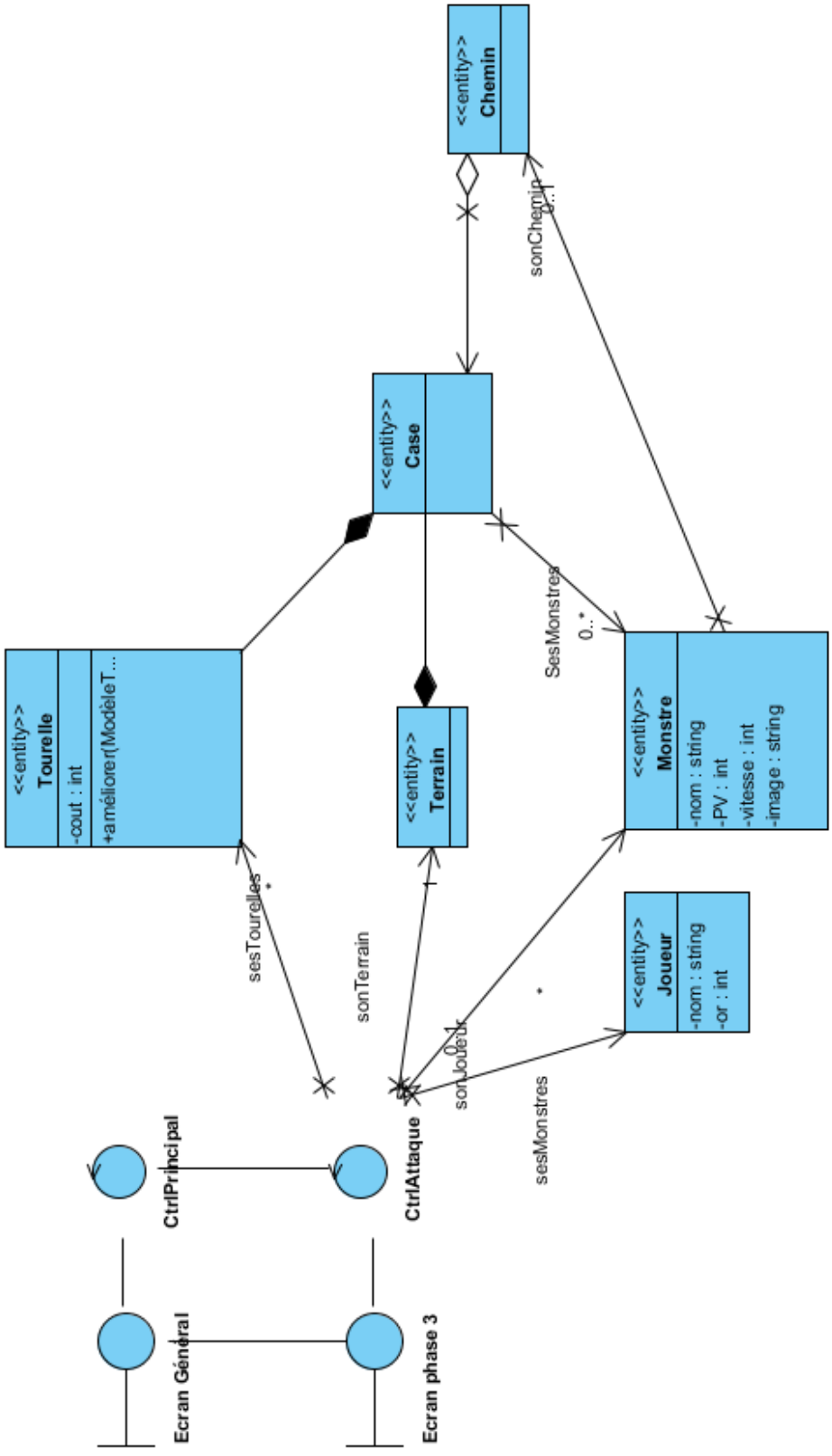
- calculprix(indice) : calcule le prix de la ligne
- affichercred() : affiche le crédit du joueur
- demandervalidéliste() : demande au joueur de valider sa liste
- finloop(reponse) : enregistre la liste, termine la boucle et passe à la phase suivante si la réponse est positive



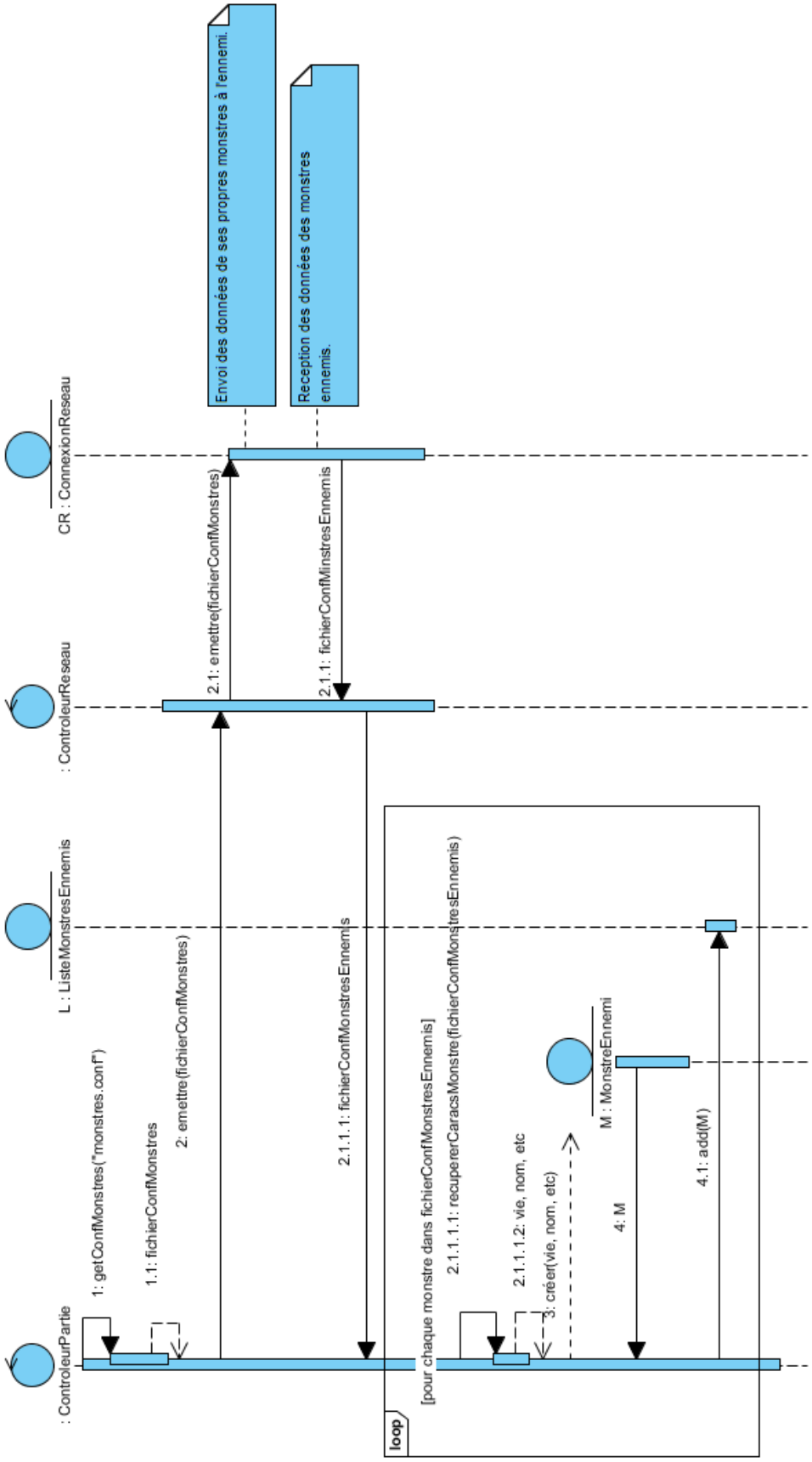
III. Diagramme d'état transition des tourelles

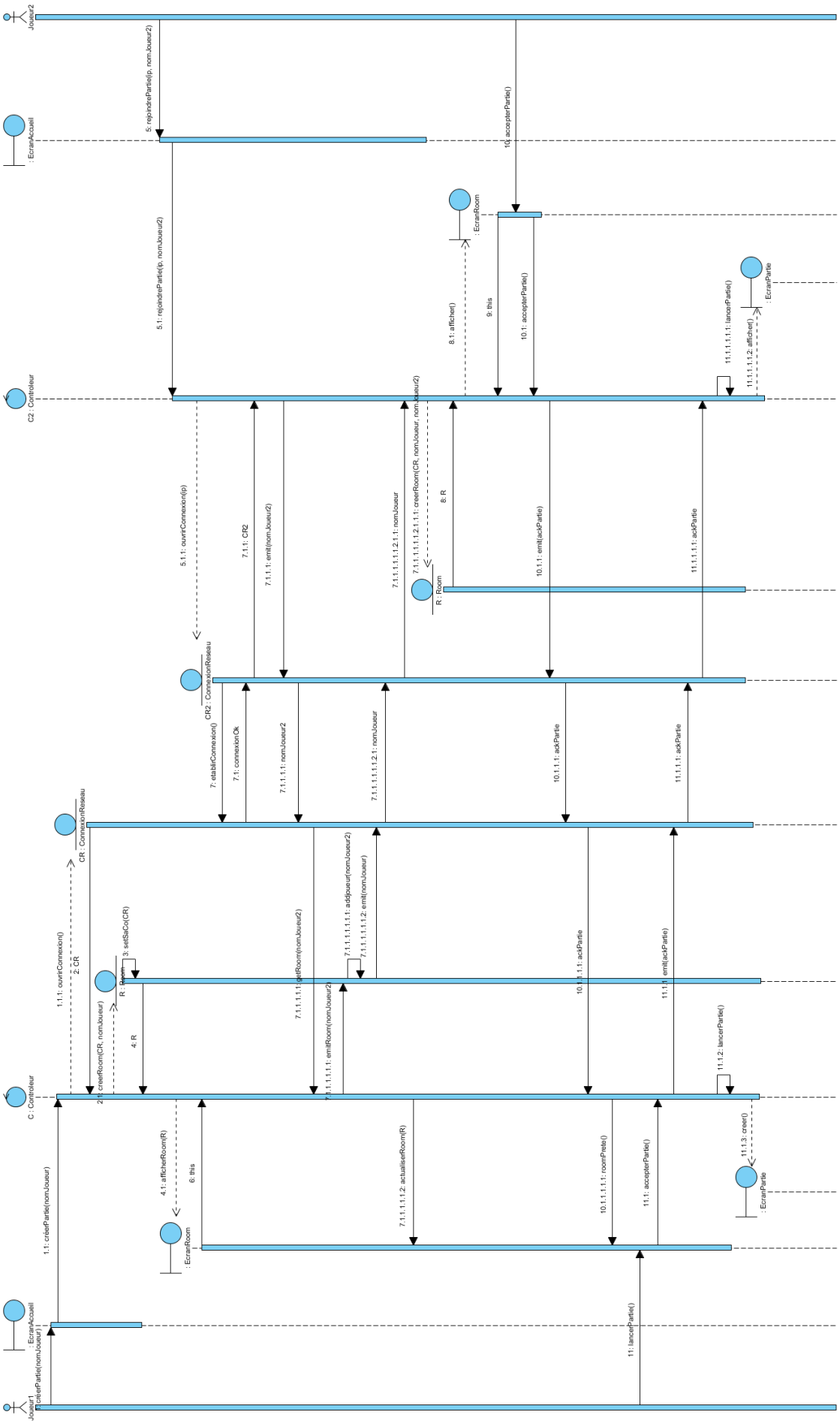


IV. Diagramme phase d'attaque

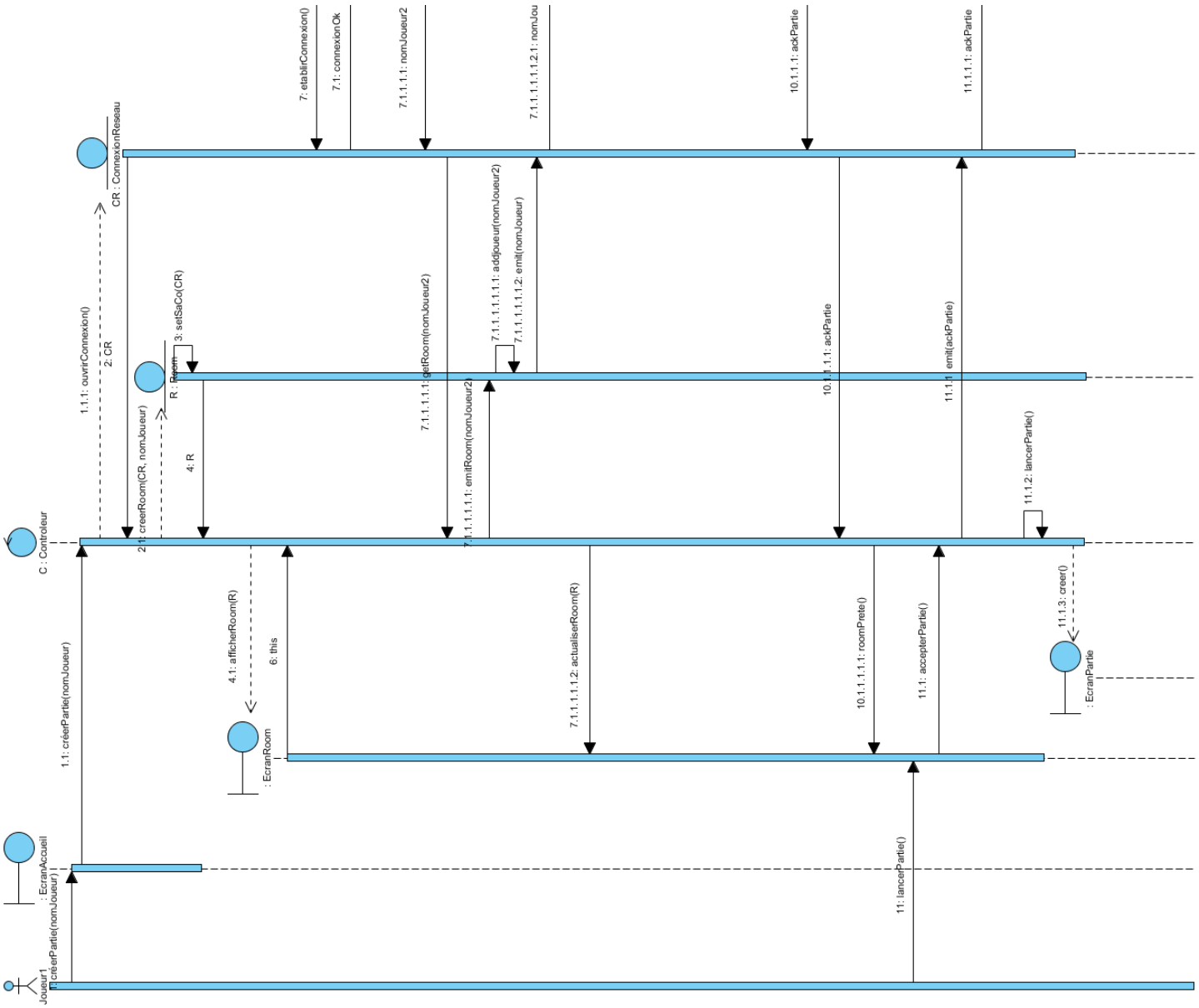


V Diagrammes réseau

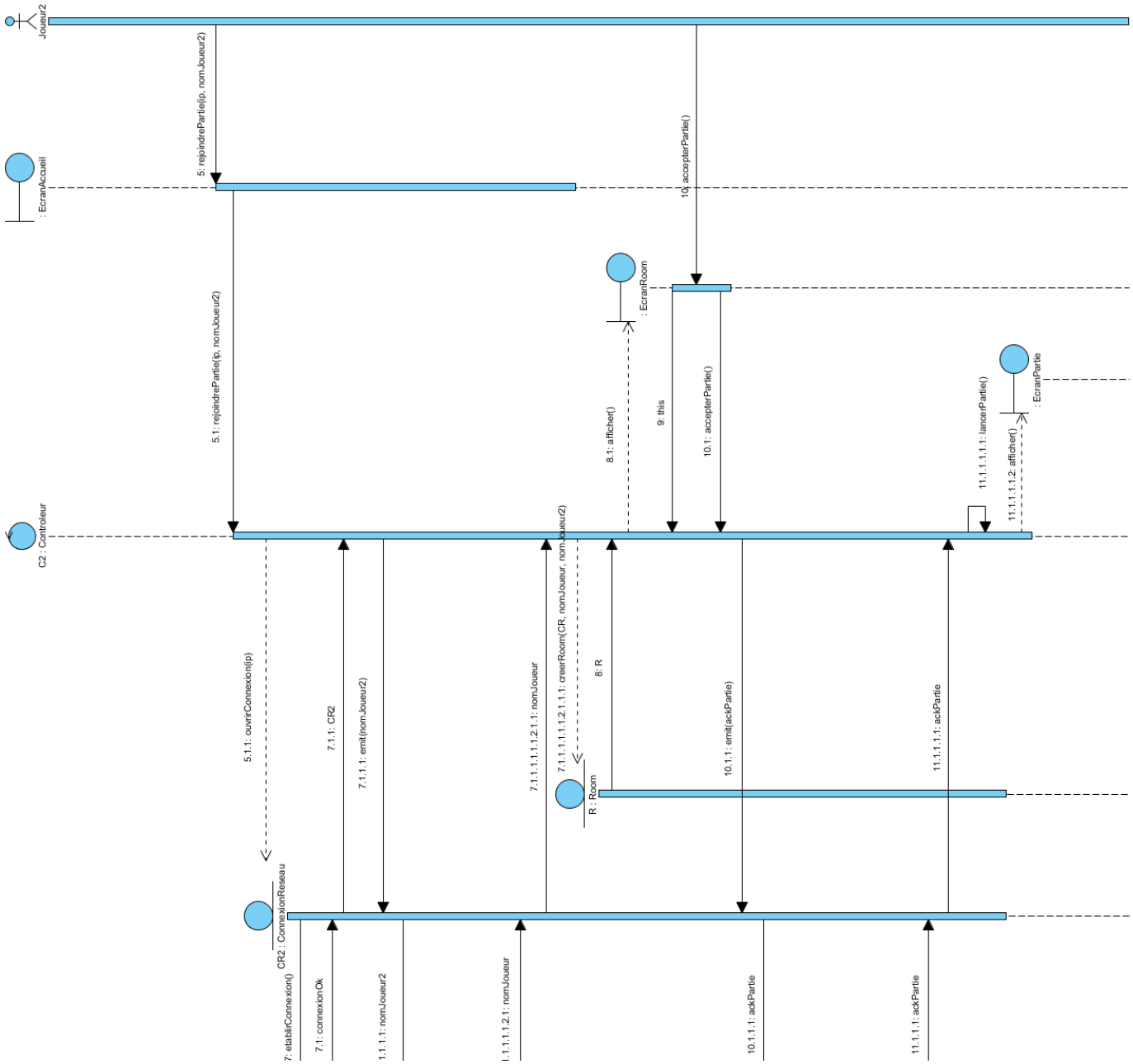



Joueur1
: EcranAccueil
C : Controleur
: EcranRoom
: EcranPartie
1: créerPartie(nomJoueur)
1.1: ouvrirConnexion()
2: CR
2.1: creeRoom(CR, nomJoueur)
3: setSaCo(CR)
4: R
4.1: afficherRoom(R)
5: rejoindrePartie(ip, nomJoueur2)
5.1: rejoindrePartie(ip, nomJoueur2)
5.1.1: ouvrirConnexion(ip)
6: this
7: etablirConnexion()
7.1: connexionOk
7.1.1: CR2
7.1.1.1: nomJoueur2
7.1.1.1.1: emit(nomJoueur2)
7.1.1.1.1.1: getRoom(nomJoueur2)
7.1.1.1.1.1.1: emitRoom(nomJoueur2)
7.1.1.1.1.1.1.1: addJoueur(nomJoueur2)
7.1.1.1.1.1.2: emit(nomJoueur)
7.1.1.1.1.2: nomJoueur
7.1.1.1.1.2.1: nomJoueur
7.1.1.1.1.2.1.1: creerRoom(CR, nomJoueur, nomJoueur2)
7.1.1.1.1.2.1.1.1: creerRoom(CR, nomJoueur, nomJoueur2)
7.1.1.1.2: actualiserRoom(R)
8: R
8.1: afficher()
9: this
10: accepterPartie()
10.1: accepterPartie()
10.1.1: emit(ackPartie)
10.1.1.1: ackPartie
10.1.1.1.1: roomPrete()
11: lancerPartie()
11.1: accepterPartie()
11.1.1: emit(ackPartie)
11.1.1.1: ackPartie
11.1.1.1.1: ackPartie
11.1.1.1.1.1: lancerPartie()
11.1.1.1.1.2: afficher()
11.1.2: lancerPartie()
11.1.3: creer()

Zoom sur le diagramme précédent



Deuxième parti du zoom



Jeux d'essai

Essai : Les unités ne trouvent aucun chemin pour aller à la base adverse.
Evènement attendu : Le Goliath intervient et créer un chemin.

Essai : La base a perdu tous ses points de vie.
Evènement attendu : L'attaquant gagne et le défenseur perd.

Essai : Toutes les vagues ont défilées et le défenseur a encore au moins 1 point de vie.
Evènement attendu : L'attaquant perd et le défenseur gagne.

Essai : Deux unités sont à portées d'une tourelle.
Evènement attendu : La tour attaque une unité selon des critères à définir (la première arrivé, ou celle en pole position par exemple).

Essai : Deux unités sont à portée d'une tourelle en arrivant au même moment (car une est plus rapide que l'autre).
Evènement attendu : La tour attaque l'unité la plus avancée.

Essai : Le joueur a assez d'argent pour acheter une unité/tour.
Evènement attendu : L'unité/tour est achetée par le joueur.

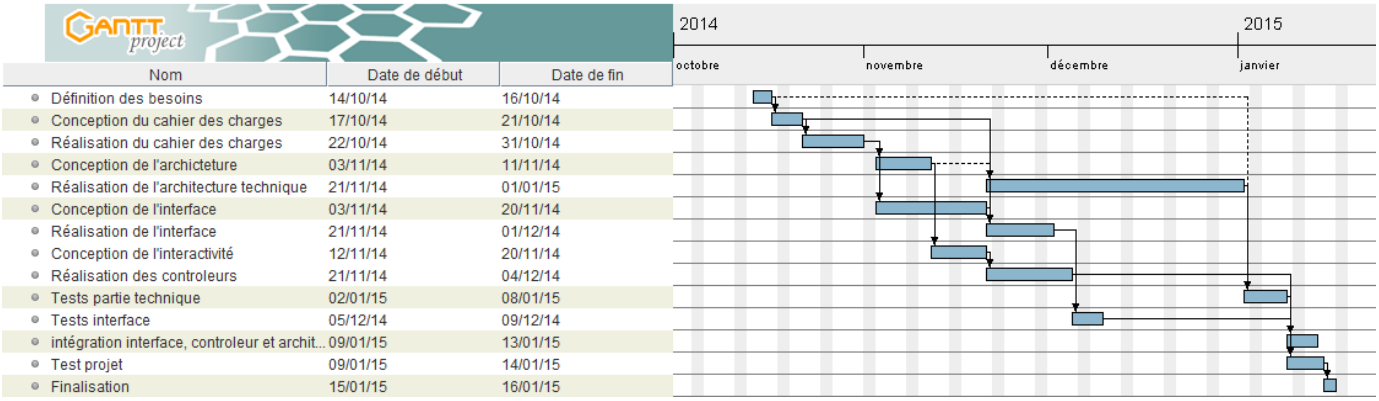
Essai : Le joueur n'a pas assez d'argent pour acheter une unité/tour.
Evènement attendu : Le joueur ne peut pas acheter l'unité/tour.

Essai : Le défenseur améliore sa tourelle en ayant les moyens.
Evènement attendu : La tourelle est améliorée

Essai : Le défenseur **Essai**e de placer une tour achetée sur une autre tour.
Evènement attendu : Le joueur ne peut pas placer sa tour à cet endroit.

Essai : Un joueur abandonne/quitte la partie.
Evènement attendu : L'autre joueur gagne.

Conclusion



Nom	Date de début	Date de fin
Définition des besoins	14/10/14	16/10/14
Conception du cahier des charges	17/10/14	21/10/14
Réalisation du cahier des charges	22/10/14	31/10/14
Conception de l'archicteture	03/11/14	11/11/14
Réalisation de l'architecture technique	21/11/14	01/01/15
Conception de l'interface	03/11/14	20/11/14
Réalisation de l'interface	21/11/14	01/12/14
Conception de l'interactivité	12/11/14	20/11/14
Réalisation des controleurs	21/11/14	04/12/14
Tests partie technique	02/01/15	08/01/15
Tests interface	05/12/14	09/12/14
intégration interface, controleur et archit...	09/01/15	13/01/15
Test projet	09/01/15	14/01/15
Finalisation	15/01/15	16/01/15

Nous avons commencé le projet le 14 octobre en définissant les besoins. Après les avoir définis, nous avons commencé la conception du cahier des charges du 17 Octobre au 21 et l'avons réalisé les 10 jours suivants. Nous comptons démarrer la conception de l'architecture technique et de la conception de l'interface simultanément, juste après avoir rendu ce dossier. Ces tâches seront effectuées simultanément car faites par des groupes prédéfinis.

Après avoir réalisé la conception de l'architecture technique, nous démarrerons la réalisation des contrôleurs et de l'architecture technique. Tandis que la réalisation de l'interface s'effectuera logiquement quand sa conception sera terminée (le 20 Novembre). Ensuite nous testerons la partie technique et l'interface pour effacer des bogues communs. Le 9 Janvier, nous assemblerons l'interface et la partie technique du projet tout en effectuant d'autres tests portants sur cette intégration. Puis nous finaliserons le tout du 15 au 16 Janvier 2015.