VisuaLigue (VL)

Rapport de projet – version 3.0

présenté à

Martin Savoie, Jonathan Gaudreault

par

Équipe 01 — GIFTW

matricule	*nom*	*signature*
111 099 643	Alexandre Gingras-Courchesne	
111 106 456	Jérôme Isabelle	
111 107 781	Maxime Ménard	
111 098 395	Alexandra Mercier	

Université Laval

27 novembre 2016

Historique des versions		
version	*date*	*description*
0.0	18 septembre 2016	Création du document
1.0	2 octobre 2016	Première remise
2.0	6 novembre 2016	Livrable 2
3.0	27 novembre 2016	Livrable 3

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Diagrammes de classe de conception

Ce chapitre présente les différents diagrammes de classe de conception de *VisualLigue*.

1.1 Couche domaine

La figure 1.1 présente le diagramme de classe de la couche applicative (domaine). La classe *Controller* est ce qui permet à la vue d'intéragir avec le domaine. Cette classe gère les différents objets du domaines, l'éditions des stratégies ainsi que le visionnement de celles-ci. Tout objet qui peut être placé sur le terrain est un *GameObject*. Trois classes héritent de *GameObject*, soit *Player*, *Obstacle* et *Projectile*. Les joueurs sont maintenus dans la classe *PlayerPool*, qui peut être sauvegardée. Cela permet à l'entraîneur de réutiliser ses joueurs d'une stratégie à l'autre. Un objet *Sport* regroupe un terrain ainsi qu'une liste de rôles pour les joueurs. Les sports sont maintenus dans la classe *SportPool*, qui peut être sauvegardée, ce qui permet de réutiliser les sports créés pour différentes stratégies. La classe *Frame* représente une image du terrain avec tous les *GameObject* présents à un moment précis. La classe *Strategy* regroupe tous les frames d'une stratégie et les associe à un sport. Elle permet le visionnement ainsi que l'édition d'une stratégie.

Sous la couche domaine se trouve la couche *Utility*, qui contient certaines classes et méthodes utilitaires qui sont utilisées par les classes du domaine. Dans cette couche, on retrouve la classe Vector, qui peut représenter un vecteur 2D ou un point. Cette couche comporte aussi le package serialization, qui contient l'interface permettant la sauvegarde et le chargement des données de l'application. Pour éviter de surcharger le diagramme, les accesseurs et les constructeurs sans paramètre n'ont pas été représentés, bien que ces méthodes soient implémentés.

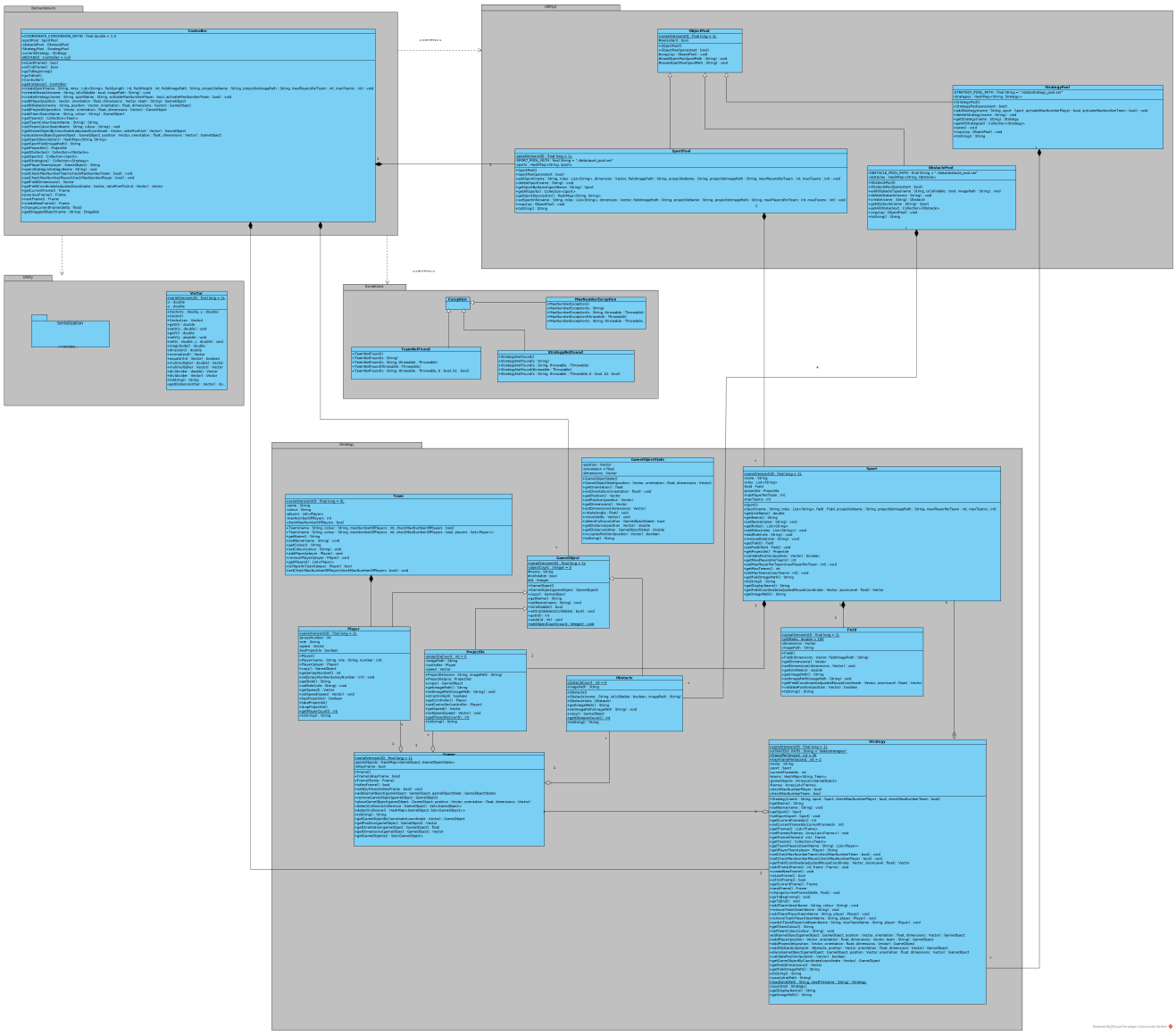

FIGURE 1.1 – Diagramme de classes de conception de la couche domaine

1.2 Couche présentation

La figure 1.2 présente le diagramme de classe de la couche présentation(vue). La technologie JavaFX est utilisée en union avec le constructeur d'interface graphique *Scene Builder*. Il y a donc un fichier FXML associé à chacune de ces classes qui contient le code des éléments graphiques. De prime abord, la classe *RootLayoutController* représente la fenêtre de base de l'application. Celle-ci contient la barre de menu principal, qui permet d'exécuter la majorité des fonctionnalités du logiciel. C'est pour cela que ce contrôleur contient une majorité des méthodes permettant la gestion des événements. Par ailleurs, le patron de conception *Observateur* est utilisé afin de rediriger la gestion de certains événements vers le *RootLayoutController*. Cela permet d'éviter une duplication du code de gestion des événements tout en maintenant le couplage bas. Ainsi, la même fonctionnalité peut être exécutée par la barre de menu ou par un bouton dans un autre contrôleur. Finalement, il est important de comprendre que presque toutes les classes de la vue doivent communiquer avec le contrôleur du domaine. Cela n'est pas représenté dans le diagramme, mais il pourrait y avoir une référence au contrôleur dans presque toutes les classes de la vue.

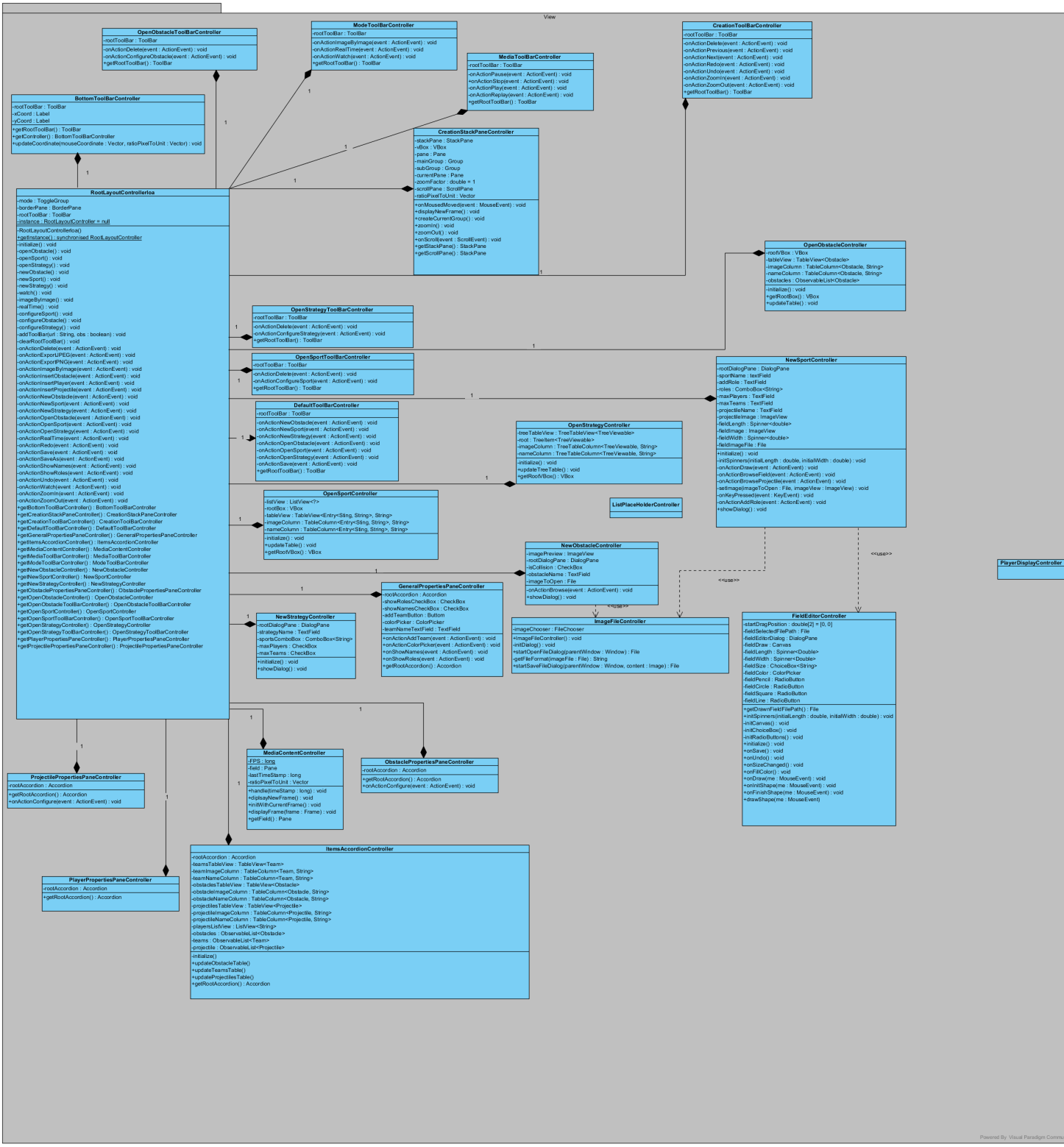

FIGURE 1.2 – Diagramme de classes de conception de la couche présentation

Chapitre 2

Diagramme d'activité

La figure 2.1 représente le diagramme d'activité lors de l'édition de la stratégie en mode image par image. L'initialisation de l'édition d'une stratégie se fait lors de la création ou de l'ouverture de celle-ci. L'édition va toujours commencer sur la première trame. La figure 2.2 représente le diagramme de sous-activité de la création d'une stratégie vide. L'option *Activer le nombre maximum de joueurs par équipe* va, lors de l'édition, empêcher l'utilisateur de placer un nombre de joueurs plus grand que le maximum permi par le sport. L'option *Activer le nombre maximum d'équipes* va, lors de l'édition, empêcher l'utilisateur de placer sur le terrain plus d'équipes que le maximum permis par le sport. La figure 2.3 représente le diagramme de sous-activité de l'édition d'une trame d'une stratégie. Tel que mentionnée dans le diagramme d'état d'un joueur, en 3.1, le nombre de joueur pouvant avoir le projectile est limité au nombre de projectiles.

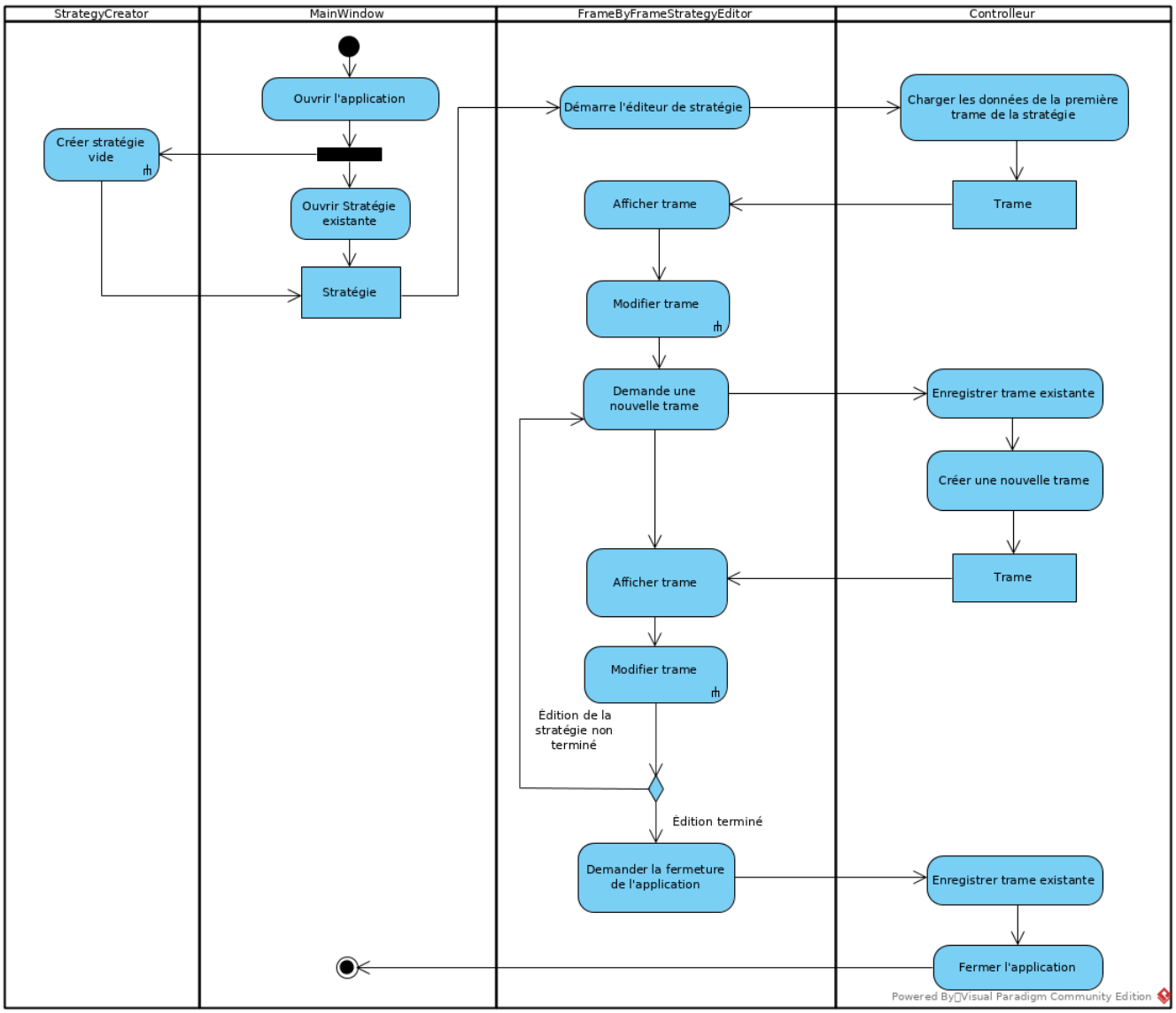

FIGURE 2.1 – Diagramme d'activité de la création d'une stratégie en mode image par image

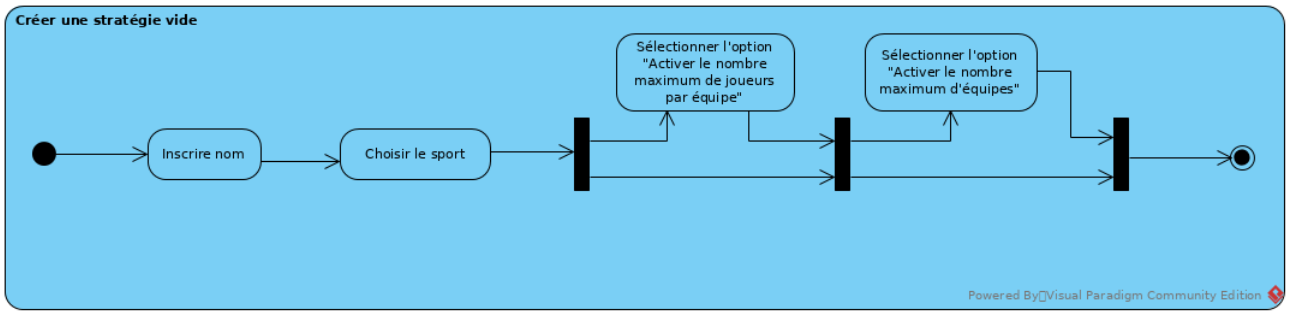

FIGURE 2.2 – Diagramme de sous-activité de la création d'une stratégie vide

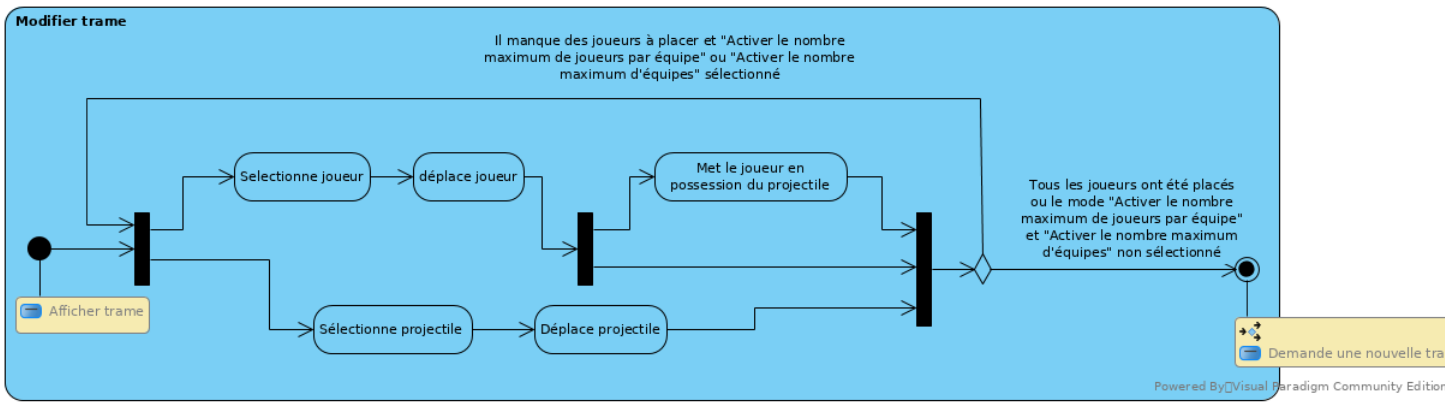

FIGURE 2.3 – Diagramme de sous-activité de l'édition d'une trame d'une stratégie

Chapitre 3

Diagrammes d'état

3.1 Diagramme d'état d'un joueur

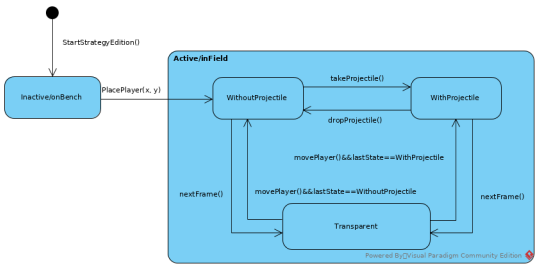

FIGURE 3.1 – Diagramme d'état d'un joueur

La figure 3.1 représente le diagramme d'état d'un joueur lors de l'édition de la stratégie. Au démmarage de la simulation, les joueurs sont inactifs. Cela est équivalent à l'idée que les joueurs sont « sur le banc ». Placer les joueurs sur le terrain les mets en mode actif. Lorsqu'un des joueurs a l'état « hasProjectile », le projectile change d'état, tel que démontré dans la figure 3.2. Aussi, le nombre maximum de joueur pouvant avoir cet état en même temps est limité par le nombre de projectiles sur le terrain. En mode d'édition image par image, créer un nouvelle image mets tous les joueurs sur le terrain en mode transparent. Le joueur revient à son état précédent si l'utilisateur fait une action sur le joueur.

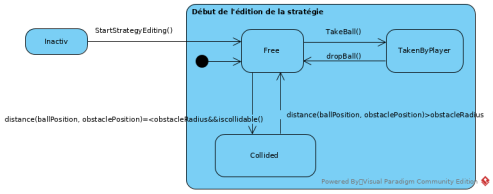

FIGURE 3.2 – Diagramme d'état d'un projectile

3.2 Diagramme d'état d'un projectile

La figure 3.2 représente le diagramme d'état d'un projectile lors de l'édition d'une stratégie. Le projectile sera inactif si l'application n'est pas en mode édition, que ce soit image par image ou temps réel. L'état de la gestion de la collision doit respecter deux conditions. Il faut que le projectile « colle » un obstacle, soit que la distance entre l'obstacle et le projectile est plus petit que le rayon de l'obstacle. De plus, il faut que l'obstacle ait l'option « gère les collisions ». Sinon, le projectile se déplace comme si l'obstacle n'existait pas.

Annexe A

Échéancier

L'image de l'échéancier est disponible sous */rapport/fig/echeancier.png*.

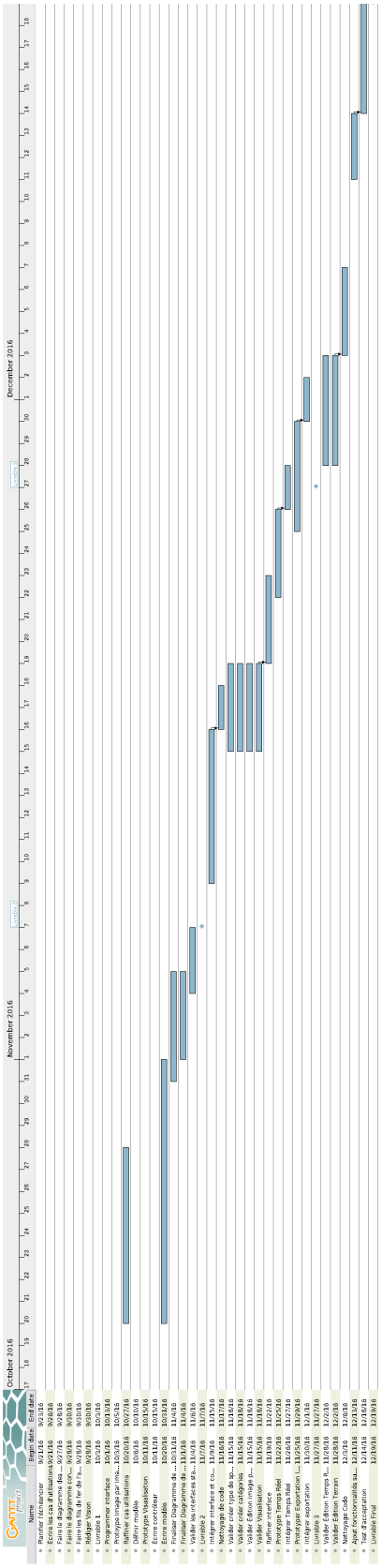

FIGURE A.1 – Gantt pour la planification du projet

Annexe B

Vision

Suite à une rencontre avec le président de l'AEMQ (Association des entraîneurs mineurs du Québec), notre "startup" s'est vu confier le mandat de réaliser l'application VisuaLigue, qui a pour objectif de simplifier l'enseignement des stratégies de jeu pour les entraîneurs. Cette application doit permettre aux entraîneurs de créer facilement des stratégies, puis de les présenter de façon dynamique et en temps réel aux joueurs.

La création des stratégies s'effectue en dessinant les séries d'actions que les joueurs doivent exécuter. Un entraîneur peut créer une nouvelle stratégie ou modifier une stratégie existante. La création d'une stratégie nécessite que chaque joueur ait sa série d'actions. Pour cela, il faut sélectionner un joueur, lui assigner un rôle et dessiner ensuite les actions qu'il doit faire. Quand une stratégie est complétée, il est possible de la sauvegarder et de l'exporter. Un utilisateur, que ce soit un entraîneur ou un joueur, peut visualiser une stratégie précédemment définie. Lors de la visualisation, il est possible d'arrêter, de redémarrer, d'avancer ou de reculer le visionnement. De plus, plusieurs sports peuvent être définis. L'entraîneur peut aussi créer un nouveau sport ou en modifier un existant. Il peut ainsi spécifier un terrain, un projectile et des rôles pour un sport. La création d'un terrain se fait de deux façons : en important une image représentant le terrain ou en dessinant les lignes via l'application. Pour les deux cas, il faut que les dimensions du terrain soient spécifiées.

L'application permet d'obtenir des stratégies que l'utilisateur peut visualiser en temps réel. Puisque la visualisation des stratégies décrites par un entraîneur est le but principal, il est primordial que l'utilisation de l'application soit naturelle et que son interface ne soit pas surchargée.

Annexe C

Maquette des interfaces

Ce chapitre présente les maquettes des principales interfaces de l'application VisuaLigue. Une courte explication des fonctionnalités moins évidentes est également présente à la suite des figures si nécessaire.

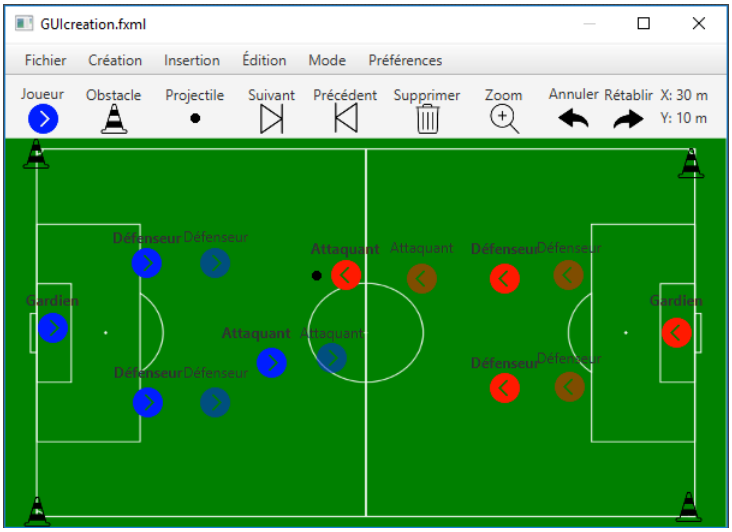

FIGURE C.1 – Interface pour la création des stratégies

L'interface C.1 permet l'édition des stratégies, soit en mode image par image ou en mode temps réel. Les trois premiers icônes en partant de la gauche permettent de glisser des éléments sur le terrain. Les icônes "suivant" et "précédent" permettent respectivement d'avancer ou de reculer d'une image dans le mode image par image. Les étiquettes "x" et "y" à la droite des icônes correspondent aux coordonnées de la souris sur le terrain en unités réelles. La transparence de certains joueurs sur la figure ci-dessus correspond à des éléments qui proviennent d'une image précédente lors de l'édition en mode image par image.

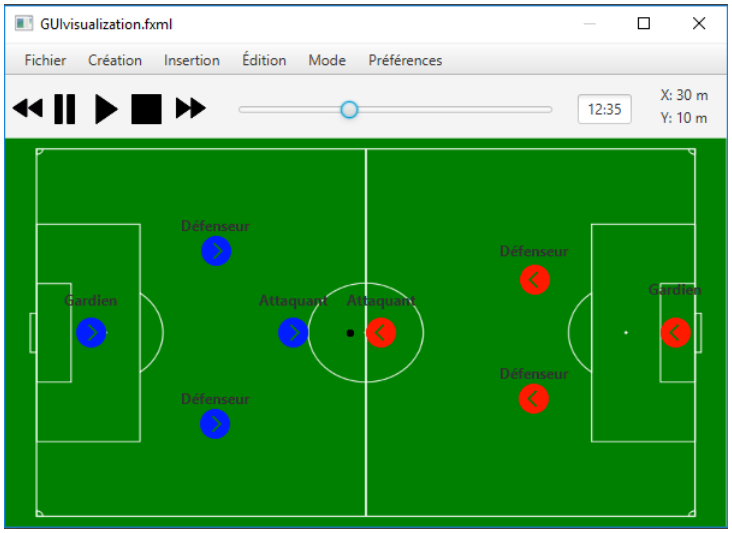

FIGURE C.2 – Interface pour la visualisation des stratégies

L'interface C.2 permet la visualisation des stratégies préalablement créées. En plus des boutons habituels de visionnement de vidéos, on y retrouve un curseur glissant permettant de se déplacer facilement à n'importe quel temps de la stratégie. Ce temps peut aussi être modifié directement dans la boîte de saisie prévue à cet effet à la droite du curseur. Les coordonnées de la souris sont encore affichées.

FIGURE C.3 – Interface de gestion des stratégies

L'interface C.3 permet la gestion des stratégies. Après avoir sélectionné une stratégie, une multitude de choix s'offre à l'entraîneur. Il peut supprimer la stratégie, la visionner, ou bien l'éditer soit en mode temps réel ou en mode image par image. Bien sûr, l'interface permet aussi l'ajout d'une nouvelle stratégie.

FIGURE C.4 – Interface de gestion des sports

L'interface C.4 permet la gestion des sports. Il est possible d'ajouter, de modifier ou de supprimer un sport.

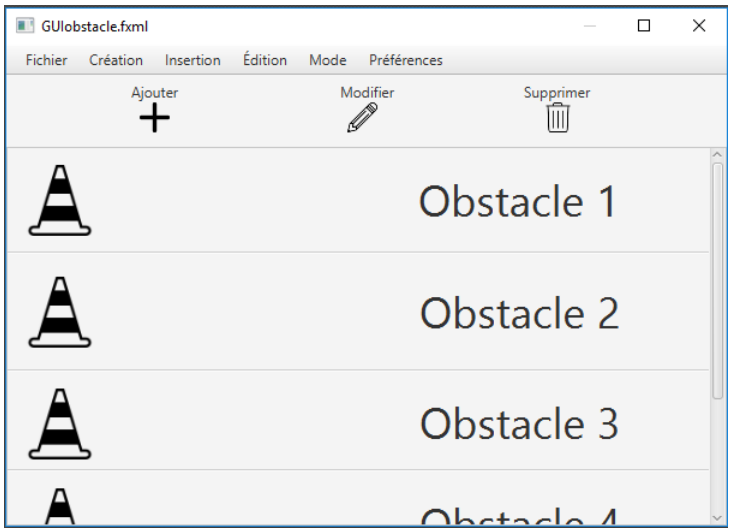

FIGURE C.5 – Interface de gestion des obstacles

L'interface C.5 permet la gestion des obstacles. Les fonctionnalités sont les mêmes que pour la gestion des stratégies.

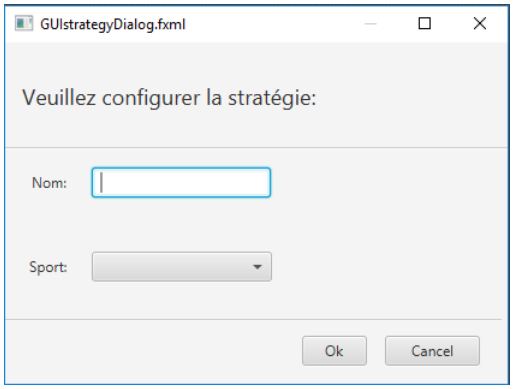

FIGURE C.6 – Interface de configuration des stratégies

La fenêtre de dialogue C.6 permet de configurer une stratégie. Elle apparaît lorsque l'on modifie ou ajoute une stratégie via la fenêtre de gestion des stratégies. On y retrouve une boîte de saisie pour le nom de la stratégie et une boîte de choix pour y associer un sport préalablement créé.

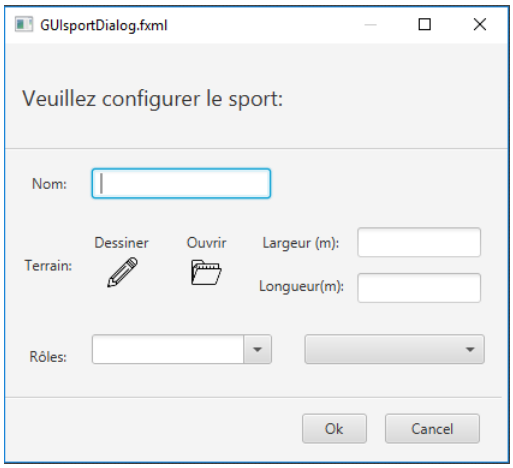

FIGURE C.7 – Interface de configuration des sports

La fenêtre de dialogue C.7 permet de configurer un sport. Elle apparaît lorsque l'on modifie ou ajoute un sport via la fenêtre de gestion des sports. On y retrouve entre autres deux icônes pour dessiner le terrain ou choisir une image déjà existante. Deux autres champs permettent la saisie des dimensions réelles du terrain. Finalement, on peut créer ou choisir un rôle parmi ceux déjà créés afin d'établir la liste des catégories de joueur pour le sport.

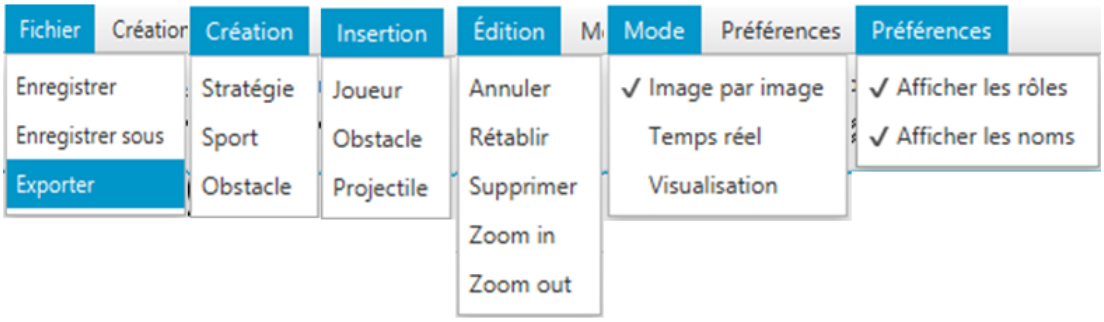

FIGURE C.8 – Interface du menu pour montrer les options

La figure C.8 montre toutes les options du menu présent dans les principales interfaces. La majorité des fonctionnalités du logiciel sont accessibles par ce menu. Il est important de comprendre que le sous-menu "mode" ne permet pas le choix de plus d'une option à la fois, contrairement au sous-menu préférences.

Annexe D

Modèle du domaine

Ce chapitre présente la modélisation du domaine avec un diagramme des classes conceptuelles de l'application. La figure D.1 représente la modélisation du domaine.

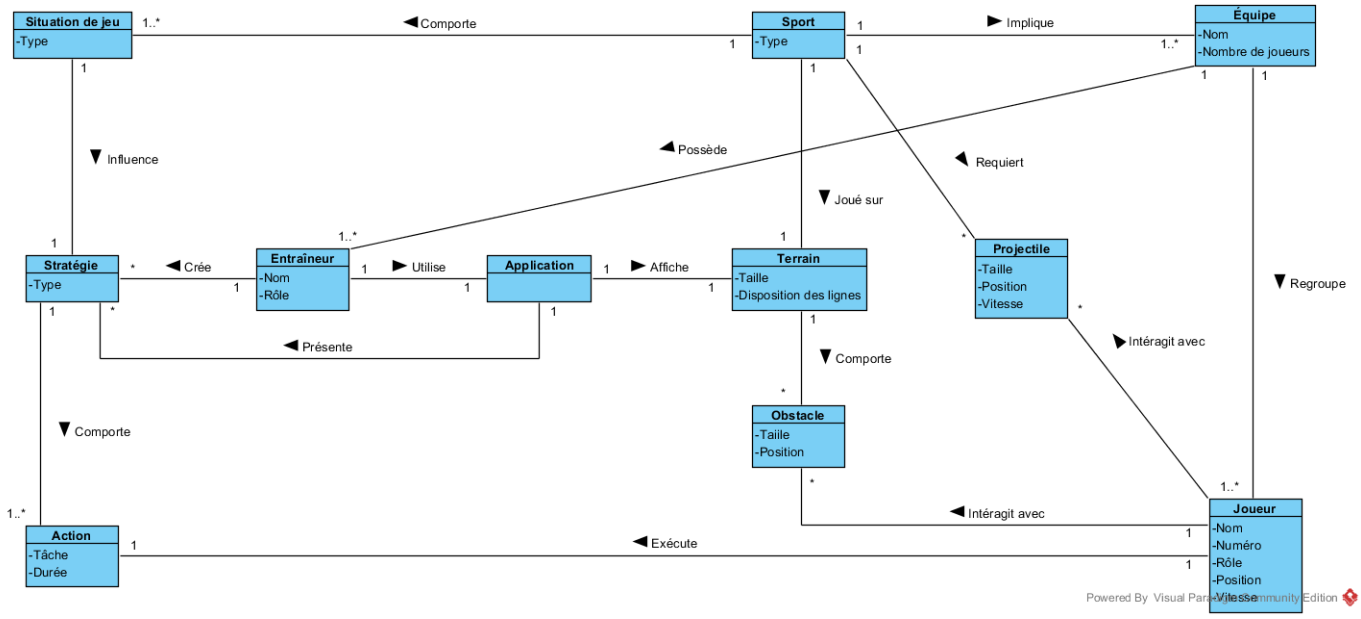

FIGURE D.1 – Modélisation du modèle pour l'application

Annexe E

Cas d'utilisations

Ce chapitre présente les différents cas d'utilisation pour l'application VisuaLigue. La figure E.1 résume les acteurs du système et les cas d'utilisations. La suite du chapitre décrit en détail les cas d'utilisations et s'attarde sur les plus importants.

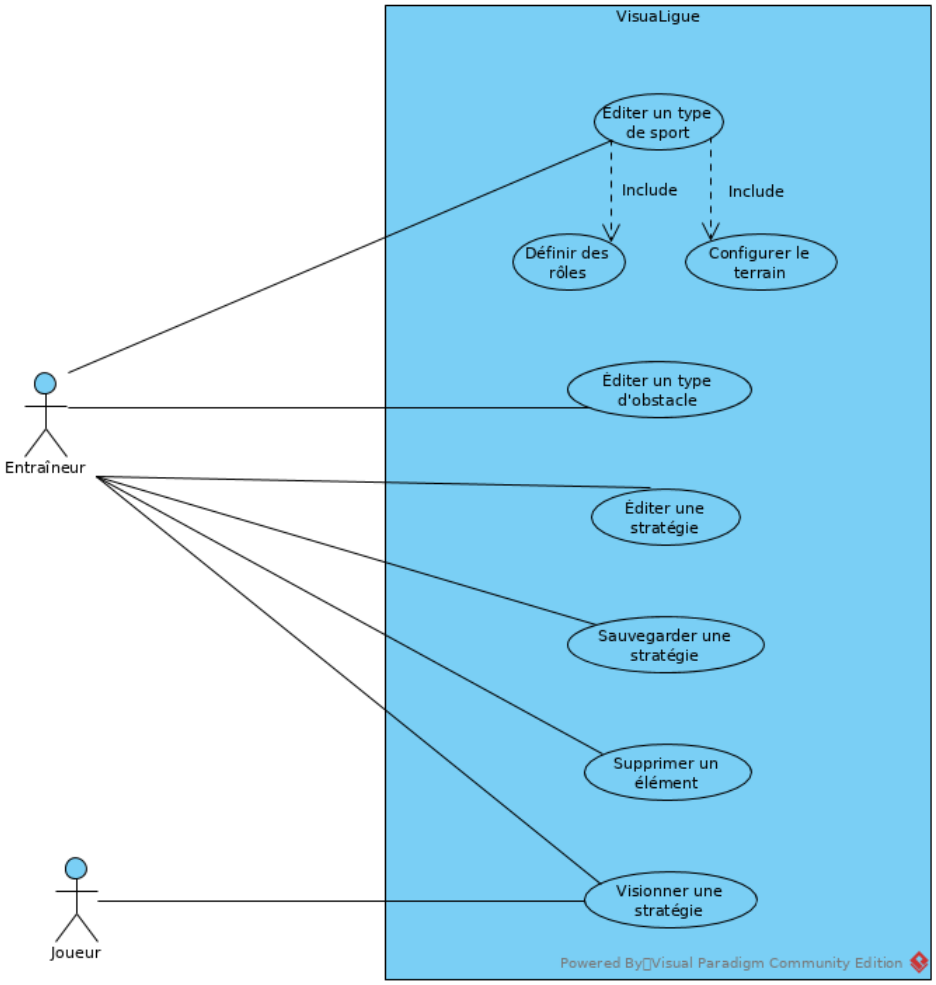

FIGURE E.1 – Diagramme des cas d'utilisations

E.1 Éditer un type de sport

- **Cas d'utilisation :** Éditer un type de sport
- **Système :** VisuaLigue
- **Acteur principal :** Entraîneur
- **Prérequis :** Aucun.
- **Parties prenantes et intérêts :**
  - Entraîneur : Veut pouvoir créer facilement le type de sport que pratique son équipe ainsi que configurer les paramètres de ce sport.
- **Garanties en cas de succès :** Le sport est ajouté à la liste des sports existants et peut être utilisé lors de la création de stratégies.
- **Scénario principal :**
  1. L'entraîneur sélectionne l'outil d'édition de sports.
  2. L'entraîneur crée un sport, le nomme.
  3. Ensuite, il définit les rôles des joueurs du sport. (Cas d'utilisation E.2)
  4. Il ajoute l'image correspondant au projectile.
  5. L'entraîneur configure ensuite les paramètres du terrain. (Cas d'utilisation E.3)
  6. L'entraîneur sauvegarde le sport.
  7. L'application ajoute le sport à la liste des sports créés.
- **Autres situations :**
  - **1a. Sport déjà existant :** Si le sport existe déjà dans l'application, un message d'avertissement apparaît pour signaler que le sport existe déjà. L'entraîneur peut décider d'effacer ce qui était dans le sport existant, d'enregistrer son sport sous un autre nom, ou d'oublier le sport créé.
  - **3a. Ajout d'une image :** L'entraîneur peut ajouter une image plutôt que dessiner les lignes pour représenter le terrain.
- **Fréquence d'utilisation :** Ce cas d'utilisation survient rarement.
- **Diagramme de séquence de système :** Voir E.2.

E.2 Définir des rôles

- **Cas d'utilisation :** Définir des rôles
- **Système :** VisuaLigue
- **Acteur principal :** Entraîneur
- **Parties prenantes et intérêts :**
  - Entraîneur : Veux pouvoir créer une liste de rôles pour un sport rapidement.
- **Prérequis :** Aucun.

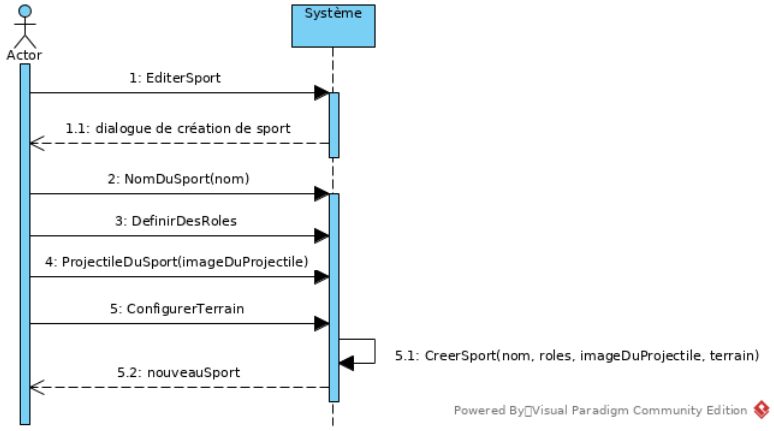

FIGURE E.2 – Diagramme de séquences de système pour Éditer un type de sport

— **Garanties en cas de succès :** La liste de rôles sera accessible lors de la création d'une stratégie du sport associé.
— **Scénario principal :**
  1. L'application affiche les rôles qui avaient déjà été créés pour d'autres sports.
  2. L'entraîneur choisit un rôle à ajouter à la liste de rôles du sport.
  3. L'application ajoute le rôle à la liste de rôles du sport.
  4. L'entraîneur reprend l'étape 2 pour tous les rôles du sport.
  5. L'entraîneur sauvegarde la liste de rôles du sport.
  6. L'application enregistre la liste de rôles du sport.
— **Autres situations :**
  — **2a. Le rôle n'existe pas :** Si le rôle n'existe pas, l'entraîneur peut en ajouter un nouveau.
    1. L'entraîneur choisit de créer un nouveau rôle.
    2. L'entraîneur écrit le nom du rôle.
    3. L'entraîneur sauvegarde le rôle.
    4. Le scénario principal reprend à l'étape 3.
— **Fréquence d'utilisation :** Ce cas d'utilisation survient rarement.
— **Diagramme séquence de système :** Voir E.3

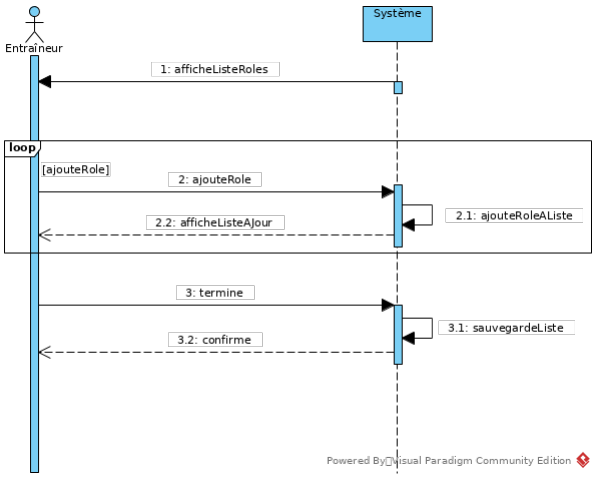

FIGURE E.3 – Diagramme de séquence de système pour Définir des Rôles

E.3 Configurer le terrain

- **Cas d'utilisation :** Configurer le terrain
- **Système :** VisuaLigue
- **Acteur principal :** Entraîneur
- **Prérequis :** Aucun.
- **Parties prenantes et intérêts :**
  - Entraîneur : Veut créer un terrain rapidement et facilement. S'il n'a pas d'image pour le terrain, en tracer un doit être facile.
- **Prérequis :**
- **Garanties en cas de succès :** Le terrain est proportionnel par rapport à ses dimensions réelles.
- **Scénario principal :**
  1. L'entraîneur sélectionne l'option de configuration du terrain.
  2. L'entraîneur importe une image représentant le terrain.
  3. Il spécifie les dimensions du terrain.
  4. L'entraîneur sauvegarde le terrain.
  5. L'application lie le terrain au sport correspondant.
- **Autres situations :**
  - **1.a Dessiner le terrain** L'entraîneur peut choisir de dessiner les lignes du terrain.
    1. L'entraîneur choisit de dessiner le terrain.
    2. Une application de dessin s'ouvre et l'entraîneur trace les lignes du terrain.
    3. L'entraîneur sauvegarde son tracé et continue le scénario principal à partir de l'étape 3.

— **Fréquence d'utilisation :** Ce cas d'utilisation sera utilisé chaque fois que l'on éditera un nouveau terrain.
— **Diagramme de séquences de systèmes :** Voir E.4.

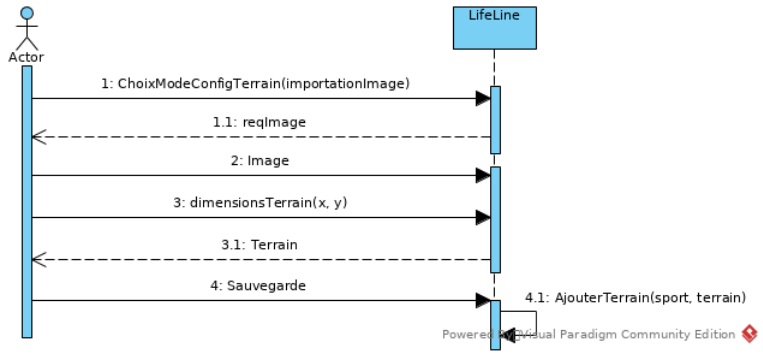

FIGURE E.4 – Diagramme de séquence de système pour le scénario principal de Configurer un terrain.

E.4 Éditer un type d'obstacle

— **Cas d'utilisation :** Éditer un type d'obstacle
— **Système :** VisuaLigue
— **Acteur principal :** Entraîneur
— **Parties prenantes et intérêts :**
  — Entraîneur : Veut pouvoir créer rapidement des types d'obstacles pour les utiliser lors de l'édition d'une stratégie.
— **Prérequis :** Aucun.
— **Garanties en cas de succès :** L'obstacle est ajouté à la liste des obstacles existants et peut être utilisé lors de la création de stratégies.
— **Scénario principal :**
  1. L'entraîneur sélectionne l'outil d'édition d'obstacles.
  2. L'entraîneur crée un nouvel obstacle.
  3. L'application affiche les attributs de l'obstacle.
  4. L'entraîneur donne un nom à l'obstacle.
  5. L'entraîneur lui associe une image.
  6. L'entraîneur sauvegarde l'obstacle.
  7. L'application sauvegarde l'obstacle dans la liste des obstacles existants.
— **Autres situations :**
  — **2a. L'obstacle existe déjà :** L'entraîneur peut vouloir modifier un obstacle existant.

1. L'entraîneur sélectionne l'obstacle à modifier dans la liste des obstacles existants.
2. L'application affiche les attributs de l'obstacle.
3. L'entraîneur modifie le nom ou l'image de l'obstacle.
4. Le scénario principal reprend à l'étape 6.

— **Fréquence d'utilisation :** Ce cas d'utilisation est utilisé de temps à autre.
— **Diagramme séquence de système :** Voir E.5

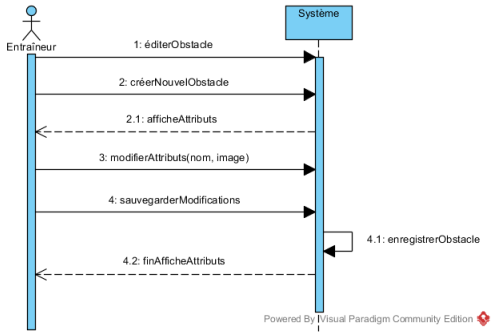

FIGURE E.5 – Diagramme de séquence de système pour Éditer un type d'obstacle

E.5 Éditer une stratégie

— **Cas d'utilisation :** Éditer une stratégie
— **Système :** VisuaLigue
— **Acteur principal :** Entraîneur
— **Parties prenantes et intérêts :**
  — Entraîneur : Veut pouvoir créer une stratégie composée de multiples déplacements de différents joueurs, ainsi que des passes entre ceux-ci, de manière intuitive et rapide.
— **Prérequis :** Aucun.
— **Garanties en cas de succès :** Les modifications peuvent être visualisées par les utilisateurs.
— **Scénario principal :**

1. L'entraîneur sélectionne la stratégie à éditer parmi la liste des stratégies existantes.
2. L'entraîneur ajoute les joueurs sur le terrain.
3. L'entraîneur assigne les rôles aux différents joueurs.
4. Ensuite, il peut sélectionner un joueur et tracer un mouvement ou interagir avec le projectile.

5. Pendant que le joueur est sélectionné, une simulation en temps réel des mouvements précédemment définis s'exécute.
6. La simulation recommence du début.
7. L'entraîneur répète les étapes 4 et 6 pour chaque joueur jusqu'à ce que la stratégie soit terminée.
8. L'entraîneur exécute *sauvergarder une stratégie* E.6.

— **Autres situations :**
  — **1a. La stratégie n'existe pas :** L'entraîneur peut décider de créer une nouvelle stratégie en sélectionnant nouvelle stratégie. Il devra alors donner un nom à celle-ci et lui assigner un sport.
  — **3a. Ajout d'obstacle :** En plus des joueurs, l'entraîneur peut ajouter des obstacles sur le terrain.
    1. L'entraîneur sélectionne un obstacle parmi la liste et le positionne sur le terrain.
    2. L'entraîneur ajuste les dimensions de l'obstacle.
    3. L'entraîneur répète les étapes 1 et 2 pour tous les obstacles qu'il veut ajouter.
    4. L'entraîneur continue le scénario principal à partir de l'étape 4.
  — **4a. Édition en mode image par image :** L'entraîneur édite la stratégie image par image...
    1. L'entraîneur place les joueurs sur le terrain.
    2. L'entraîneur clique sur un bouton de l'application et avance d'une image. Les joueurs deviennent transparents.
    3. L'entraîneur clique sur un joueur et le glisse vers sa nouvelle position. Le joueur sélectionné n'est plus transparent.
    4. L'entraîneur répète l'étape 3 pour tous les joueurs qu'il veut déplacer.
    5. L'entraîneur répète les étapes 2 à 4 jusqu'à ce qu'il ait terminé sa stratégie.
    6. L'entraîneur continue le scénario principal à partir de l'étape 6.
  — ***a. Visualiser la stratégie :** À tout moment de l'édition de la stratégie, l'entraîneur peut exécuter *visualiser la stratégie* E.8.
— **Fréquence d'utilisation :** Ce cas d'utilisation survient fréquemment.
— **Diagramme de séquence de système :** Voir E.6 et E.7.

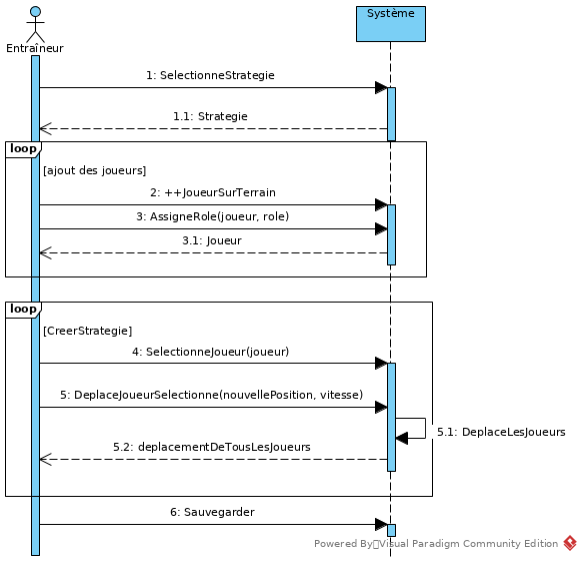

FIGURE E.6 – Diagramme de séquence de système pour le scénario principal d'Éditer une Stratégie

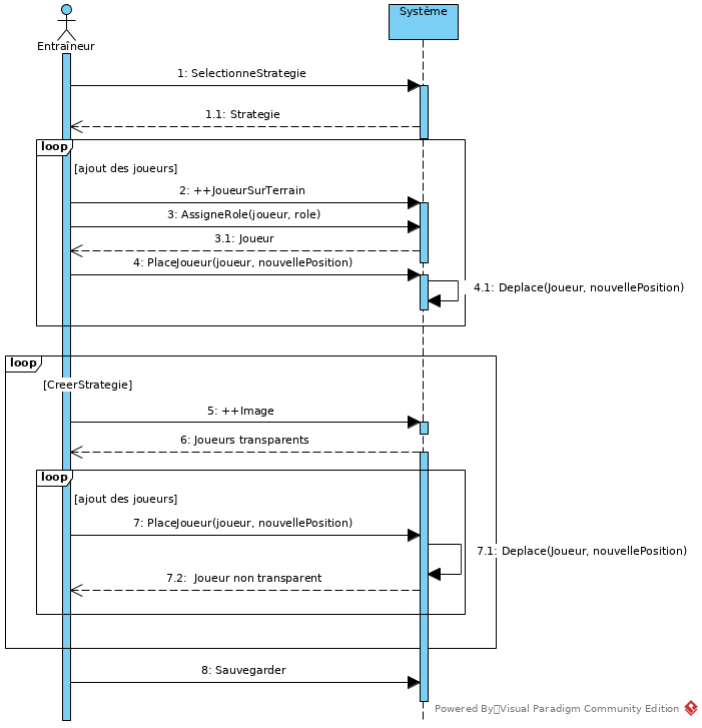

FIGURE E.7 – Diagramme de séquence de système pour le scénario alternatif d'Éditer une Stratégie en mode image par image

E.6 Sauvegarder une stratégie

- **Cas d'utilisation :** Sauvegarder une stratégie
- **Système :** VisuaLigue
- **Acteur principal :** Entraîneur
- **Parties prenantes et intérêts :**
  - Entraîneur : Veut sauvegarder une stratégie qu'il vient d'éditer afin de pouvoir la visionner ou l'éditer à nouveau plus tard.
- **Prérequis :** Une stratégie doit avoir été éditée.
- **Garanties en cas de succès :** La stratégie est sauvegardée dans la liste des stratégies existantes et peut être chargée à nouveau.
- **Scénario principal :**
  1. L'entraîneur a modifié une stratégie et la sauvegarde.
  2. L'application enregistre les éléments de la stratégie.
- **Autres situations :**
  - **Exportation dans un format d'image :**
    1. L'entraîneur souhaite plutôt exporter la stratégie dans un format de fichier.
    2. Il sélectionne le format de fichier et le nom pour l'exportation.
    3. L'application convertit la stratégie en une image et l'enregistre dans un fichier avec le bon nom.
- **Fréquence d'utilisation :** Ce cas d'utilisation survient fréquemment.
- **Diagramme de séquence de système :** Voir E.8

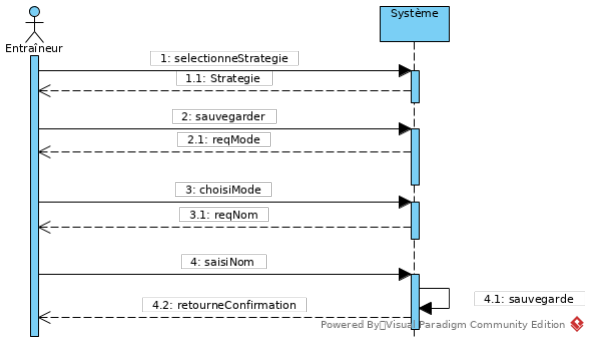

FIGURE E.8 – Diagramme de séquence de système pour Sauvegarder une Stratégie

E.7 Supprimer un élément

- **Cas d'utilisation :** Supprimer un élément
- **Système :** VisuaLigue
- **Acteur principal :** Entraîneur
- **Parties prenantes et intérêts :**
  - Entraîneur : Veut supprimer n'importe quel élément facilement en ayant la sécurité d'esprit de ne rien supprimer d'autre.
- **Prérequis :**Aucun.
- **Garanties en cas de succès :** L'élément sélectionné est supprimé.
- **Scénario principal :**
  1. L'entraîneur sélectionne l'élément à supprimer.
  2. L'entraîneur choisit l'option supprimer de l'application.
  3. Un message d'avertissement est affiché à l'écran.
  4. L'entraîneur confirme qu'il veut supprimer l'élément.
  5. L'élément est supprimé.
- **Autres situations :**
  - **4a. La suppression est annulée :** Si l'entraîneur change d'avis, il peut annuler la suppression. Dans ce cas-là, rien ne se passe.
  - **Erreur : Les données de l'élément sont corrompues :** Si les données de l'élément à supprimer sont corrompues, un message d'erreur est affiché et la suppression est annulée.
  - **Erreur : l'élément n'existe pas :** Si l'élément à supprimer n'existe pas, un message d'erreur est affiché et la suppression est annulée.
  - **Erreur : l'élément a été modifié ailleurs :** Si l'élément à supprimer a été modifié ailleurs, un message d'erreur est affiché et la suppression est annulée.
- **Fréquence d'utilisation :** Ce cas d'utilisation survient souvent.
- **Diagramme de séquence de système :** Voir E.9

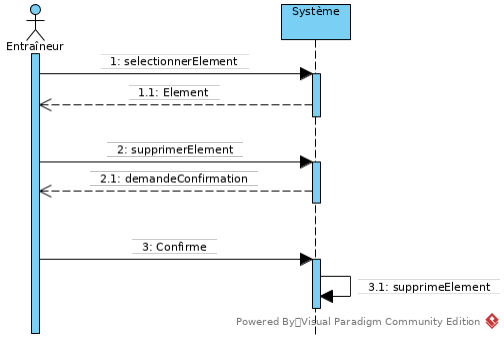

FIGURE E.9 – Diagramme de séquence de système pour Supprimer un élément

E.8 Visualiser une stratégie

- **Cas d'utilisation :** Visualiser une stratégie
- **Système :** VisuaLigue
- **Acteur principal :** Entraîneur ou joueur
- **Parties prenantes et intérêts :**
  - L'entraîneur : Veut pouvoir visualiser une stratégie pour déterminer si elle est adéquate. Veut pouvoir interrompre la visualisation à tout moment pour fournir des explications supplémentaires. Veut pouvoir reculer d'un temps fixe la visualisation pour montrer à nouveau un segment de la stratégie. Veut pouvoir redébuter la visualisation pour regarder à nouveau la stratégie.
  - Joueur : Veut pouvoir visualiser la stratégie pour l'étudier.
- **Prérequis :** Il faut que la stratégie soit enregistrée dans l'application pour que l'entraîneur puisse la visualiser.
- **Garanties en cas de succès :** À la fin de la visualisation, la stratégie n'a pas été modifiée.
- **Scénario principal :**
  1. L'utilisateur sélectionne la stratégie à visualiser.
  2. Il débute la visualisation.
  3. La visualisation prend fin lorsque toute la stratégie s'est déroulée.
- **Autres situations :**
  - **1a la stratégie est corrompue :**
    1. Lors de la sélection de la stratégie, celle-ci est corrompue.
    2. Un message d'erreur s'affiche.
  - **2a interrompre la visualisation :**
    1. L'utilisateur *pause* la visualisation.
    2. L'utilisateur redémarre la visualisation.

3. On reprend de l'étape 2.

— **2b redébuter la visualisation :**

1. L'utilisateur *arrête* la visualisation.
2. On reprend de l'étape 2.

— **2c reculer d'un temps donné :**

1. L'utilisateur *pause* la visualisation.
2. Il détermine un temps en secondes.
3. Il recule la visualisation.
4. Il redémarre la visualisation.
5. On reprend de l'étape 2.

— **2d avancer d'un temps donné :**

1. L'utilisateur *pause* la visualisation.
2. Il détermine un temps en secondes.
3. Il avance la visualisation.
4. Il redémarre la visualisation.
5. On reprend de l'étape 2.

— **Fréquence d'utilisation :** C'est le cas le plus fréquent d'utilisation.
— **Diagramme de séquence de système :** Voir E.10

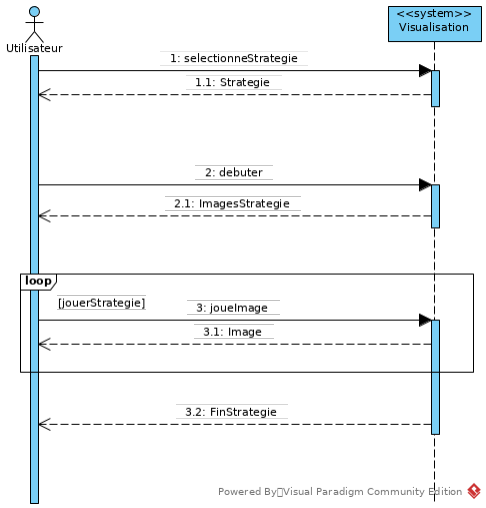

FIGURE E.10 – Diagramme de séquence de système pour Visualiser une stratégie

Annexe F

Diagrammes de séquence de conception

Ce chapitre présente les diagrammes de séquence de conception associée aux fonctionnalitées de l'application.

F.1 Convertir les coordonnées de la souris en coordonnées réelles

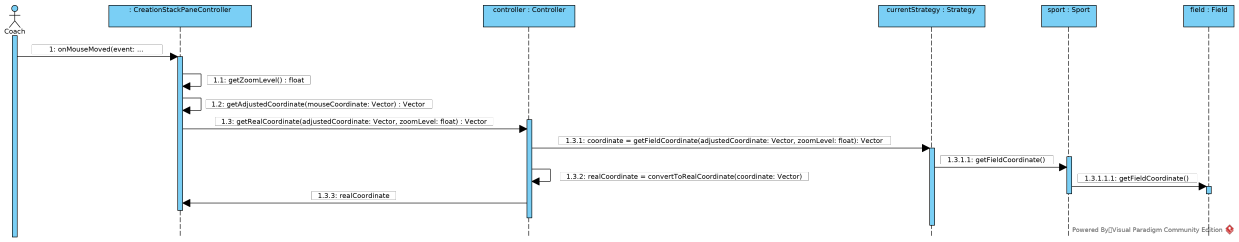

FIGURE F.1 – Diagramme de séquence de conception pour Convertir les coordonnées de la souris en coordonnées réelles

La figure F.1 est le diagramme de séquence pour obtenir les coordonées réelles à partir des coordonnées de la souris. La fenêtre reçoit un événement lors du déplacement de la souris. Elle récupère le niveau de zoom actuel et ajuste les coordonnées de la souris selon le déplacement horizontal et vertical du terrain. Le contrôleur est appelé, qui délègue l'appel pour obtenir les coordonées du terrain à la stratégie courante. Une fois les coordonnées du terrain obtenues, elles sont converties en coordonnées réelles (e.g : en mètre) et retournées.

F.2 Ajouter un joueur

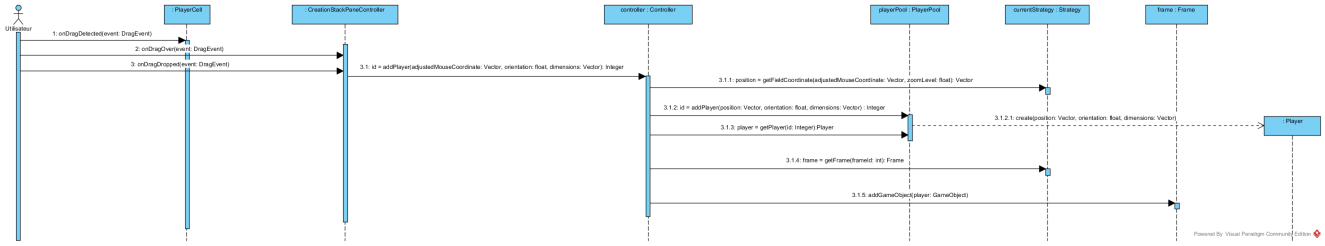

FIGURE F.2 – Diagramme de séquence de conception pour ajouter un joueur sur le terrain

La figure F.2 est le diagramme séquence lorsqu'un nouveau joueur est ajouté sur le terrain. La fenêtre reçoit un événement lorsque la souris est en mode *Glisser*. L'ajout d'un joueur se fait lorsque l'utilisateur dépose celui-ci sur le terrain. Le contrôleur est alors appelé et il crée un identifiant pour le nouveau joueur, Puis, ce joueur est ajouté dans le domaine, ainsi que dans l'image courante.

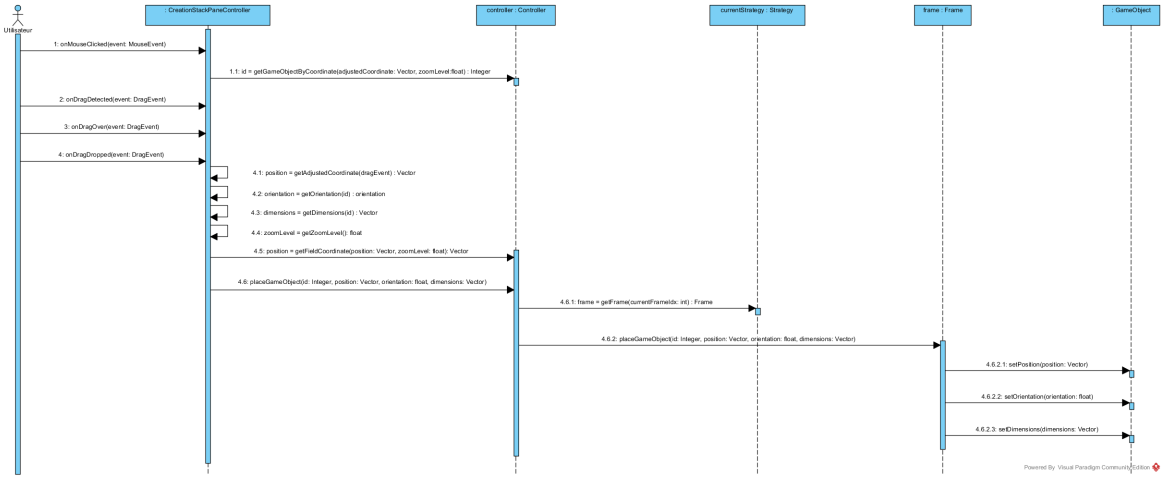

FIGURE F.3 – Diagramme de séquence de conception pour déplacer un joueur sur le terrain

La figure F.3 est le diagramme de séquence lorqu'un joueur est déplacé à un nouvel endroit sur le terrain. La sélection du joueur sur le terrain est expliqué en détail à la section F.3.

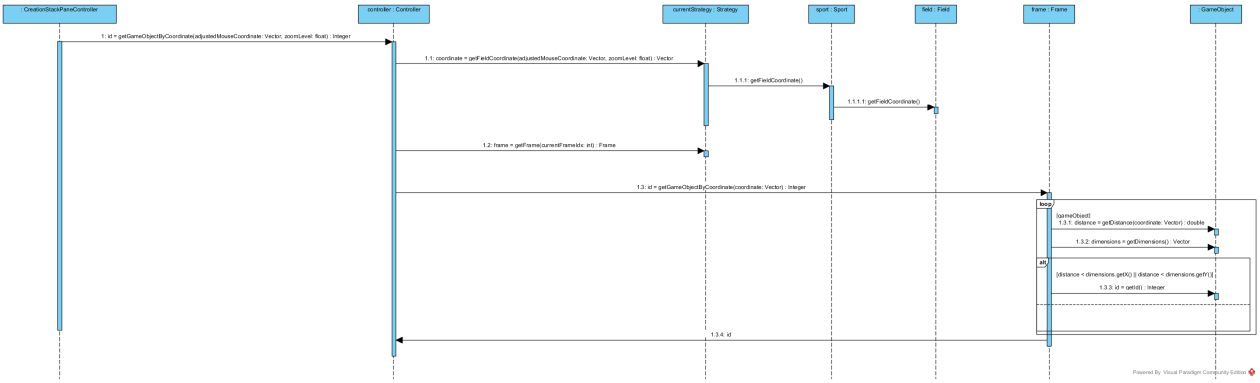

FIGURE F.4 – Diagramme de séquence de conception pour obtenir l'objet de jeu sous la souris après un clic

F.3 Sélectionner l'objet de jeu sous la souris après un clic

La figure F.4 est le diagramme de séquence pour sélectionner l'objet de jeu sous la souris. Les coordonnées du terrain sont obtenus en délégant l'appel à la stratégie courante. L'image actuelle de la stratégie est récupérée et l'objet *Frame* est appelée. Celui-ci parcours tous les objets de jeu qu'il contient, puis calcul la distance entre les coordonnées et l'objet de jeu. Si la distance est inférieur aux valeurs de dimensions en x ou en y, l'identifiant de cet objet de jeu est retourné. Autrement, un identifiant *null* est retourné.

F.4 Édition en mode image par image

La figure F.5 est le diagramme de séquence des différentes intéractions entre le système et l'utilisateur lors de l'édition d'une stratégie en mode **image par image**. Lorsque l'utilisateur change pour ce mode d'édition, la fenêtre principale s'occupe d'initialiser les éléments de la vue. Si l'utilisateur clique sur un joueur et le déplace, les scénarios pour obtenir la sélection sous la souris F.3 et déplacer un joueur F.2 sont exécutés. Par soucis de lisibilité, tous les appels ne sont pas reproduits.

Lorsque l'utilisateur clique sur «prochaine image», la fenêtre sauvegarde les informations actuelles, puis appelle le contrôleur pour obtenir la prochaine image. S'il n'y a pas de prochaine image, une nouvelle est créée en copiant les informations de l'image actuelle. La vue rend transparentes les informations de l'ancienne image et affiche la nouvelle image.

Lorsque l'utilisateur clique sur «image précédente», la fenêtre récupère à l'aide du contrô-

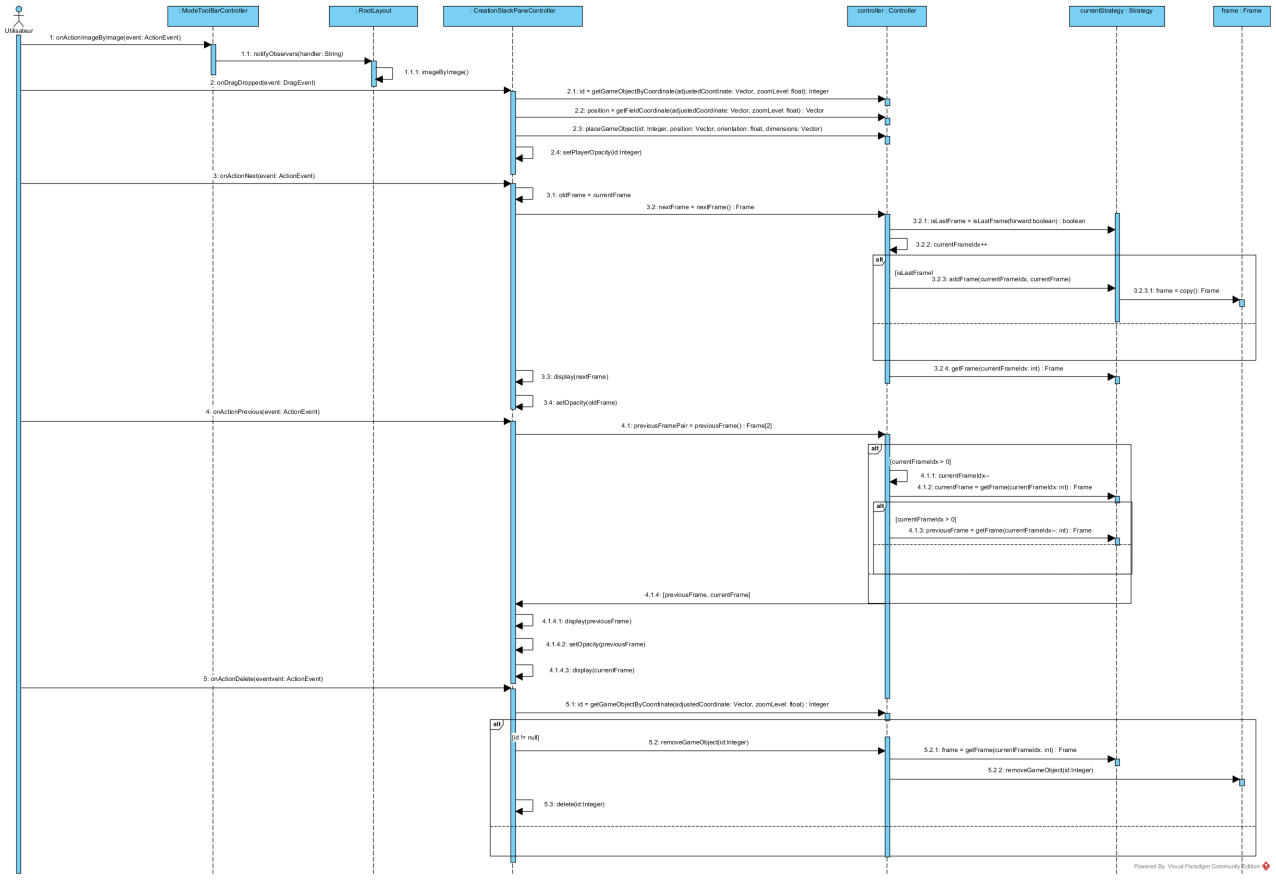

FIGURE F.5 – Diagramme de séquence de conception pour le mode d'édition image par image

leur les deux dernières images. Si l'une des deux images n'existe pas, null est retourné. La plus ancienne des deux images est affichée avec un niveau de transparence et l'image immédiatement précédente à celle en cours est affichée.

En mode « suppression », lorsque l'utilisateur clique sur un objet de jeu, le scénario pour obtenir la sélection, F.3, est exécuté. Ensuit, un appel au contrôleur est fait pour supprimer l'objet avec l'identifiant acquis. Cet appel est uniquement exécuté si l'identifiant n'est pas null.

F.5 Édition en mode temps réel

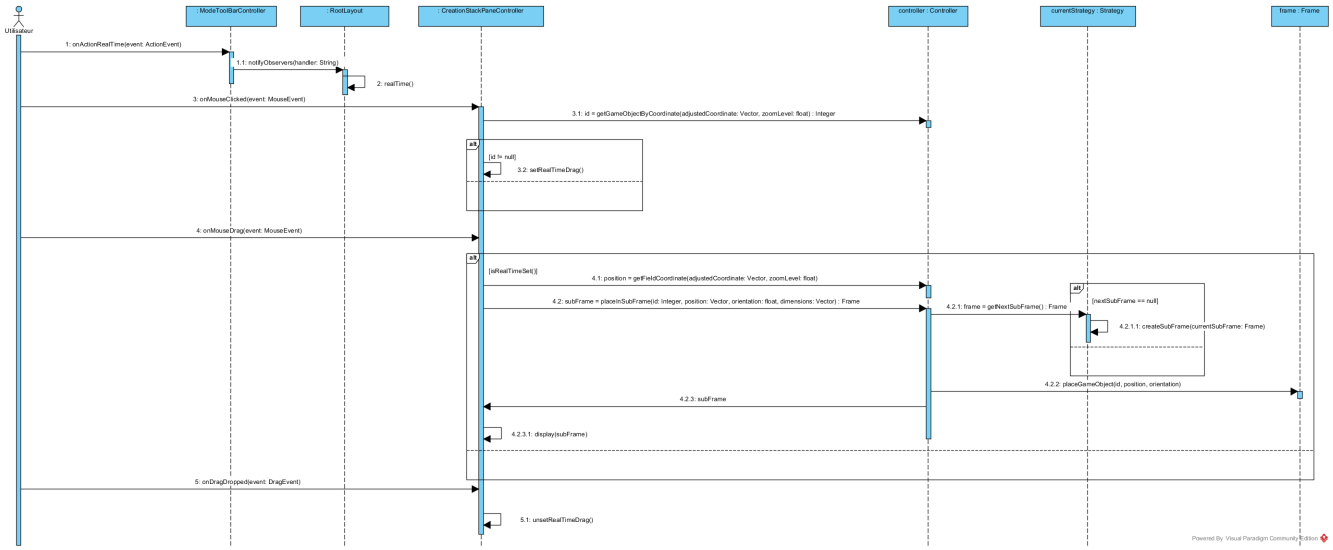

FIGURE F.6 – Diagramme de séquence de conception pour le mode d'édition en temps réel

La figure F.6 est le diagramme de séquence des différentes intéractions entre le système et l'utilisateur lors de l'édition d'une stratégie en mode **temps réel**. Ce mode est très similaire avec celui image par image (F.4). La différence principale est que lorsque l'utilisateur exécute le scénario pour déplacer un joueur (F.2), plusieurs appels supplémentaires doivent être faits. Lorsqu'un déplacement est débuté, un booléen est modifié à Vrai (*setRealTimeDrag()*).

À chaque événement de déplacement de la souris, les appels nécessaires pour positionner le joueur sont faits. Plutôt que de positionner dans une image, la position est mise dans une **sous-image**. Cette sous-image est retournée et affichée. Ceci permet de rejouer les mouvements depuis la dernière image des joueurs.

Lorsque le déplacement est terminé, le booléen est modifié à Faux (*unsetRealTimeDrag()*)

et la fenêtre met fin au mode de déplacement en temps réel. L'édition en temps réel peut se poursuivre.

F.6 Visionner le jeu

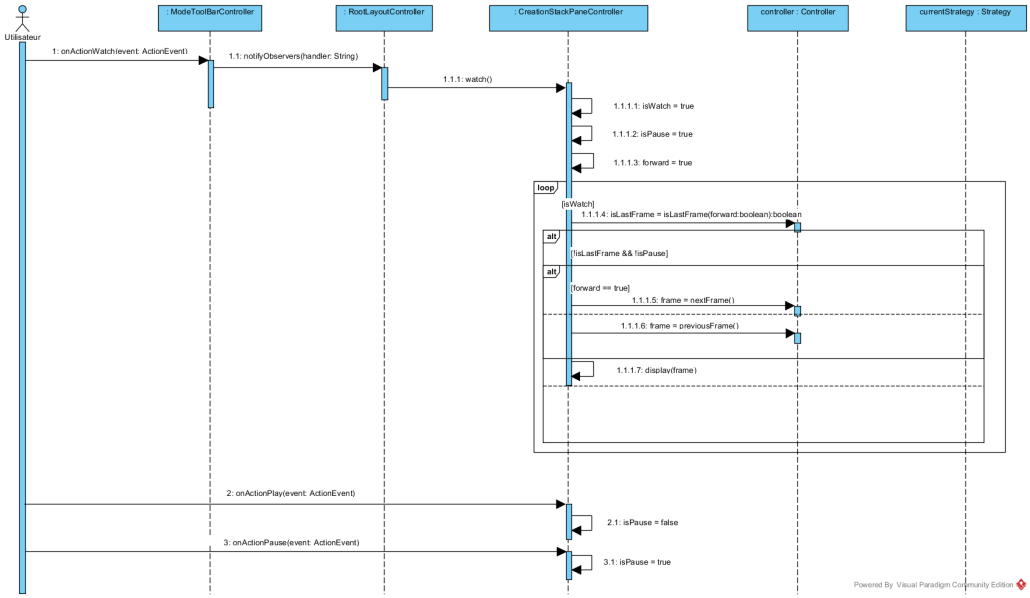

FIGURE F.7 – Diagramme de séquence de conception pour le mode de visionnement (partie principale)

La figure F.7 est le diagramme de séquence pour les principales intéractions lors du visionnement.

Lorsque l'utilisateur passe en mode visionnement, le système entre dans une boucle gouvernée par un booléen : **isWatch**. Par défaut, le visionnement est sur pause. Si l'utilisateur active le visionnement, le booléen : **isPause** est mis à Faux. Si l'utilisateur met sur pause, ce même booléen est mis à Vrai. À chaque itération de la boucle, le système récupère une image. Si le visionnement n'est pas sur pause ou si la dernière image affichée n'était pas la dernière image disponible, le système affiche l'image.

La figure F.8 est le diagramme de séquence lorsque l'utilisateur souhaite avancer ou reculer d'un temps ajustable. Lorsque le temps est saisi, le système appel le contrôleur pour ajuster le temps actuel. Le contrôleur est responsable de traduire le différentiel de temps en un différentiel d'images et il tente de modifier autant que possible l'image en cours. La nouvelle image actuelle est retournée et affichée.

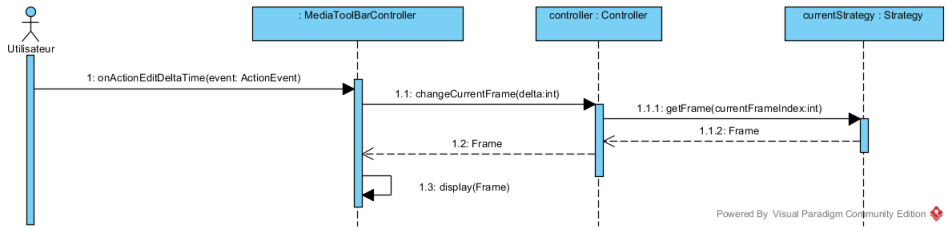

FIGURE F.8 – Diagramme de séquence de conception pour le mode de visionnement (modification du temps)

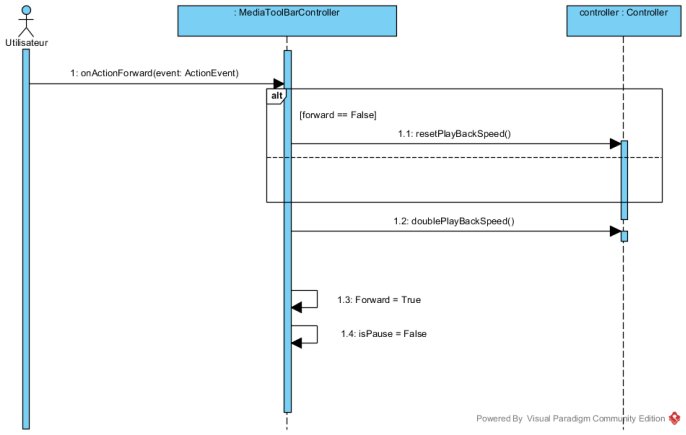

FIGURE F.9 – Diagramme de séquence de conception pour le mode de visionnement (avancement rapide)

La figure F.9 est le diagramme de séquence lorsque l'utilisateur souhaite avancer en accélérer la stratégie. Le sens est vérifié et si le visionnement se faisait en reculant, la vitesse de visionnement est remise à sa valeur par défaut. La vitesse est accélérée en appelant le contrôleur.

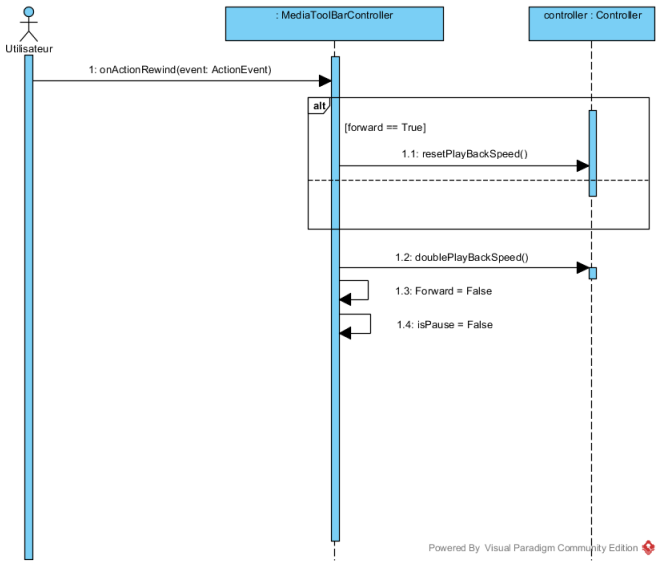

FIGURE F.10 – Diagramme de séquence de conception pour le mode de visionnement (reculuement rapide)

La figure F.10 est le diagramme de séquence lorsque l'utilisateur souhaite reculer en accélérer la stratégie. C'est la même logique que dans le cas de l'avancement rapide, mais modifié pour tenir compte du sens de visionnement renversé.

Annexe G

Glossaire

— Action : Déplacement ou interaction avec le projectile effectué joueur.

— Entraîneur : Utilisateur qui construit une stratégie et la présente à des joueurs.

— Joueur : Utilisateur qui exécute une action et qui possède un rôle.

— Obstacle : Objet statique qu'un joueur doit éviter lors d'un déplacement.

— Projectile : Objet principal d'un sport, *e.g :* balle, ballon, rondelle, etc.

— Rôle : Fonction d'un joueur lors d'une stratégie, *e.g :* ailier, centre, défenseur, etc.

— Stratégie : Ensemble des séquences des actions des joueurs.

Bibliographie

[1] Jonathan Gaudreault, Martin Savoie, 2016, *Énoncé de projet : VisuaLigue*, Département d'informatique et de génie logiciel de l'Université Laval, 4 p.