Antoine St-Laurent – 111 127 695, Antoine Hudon – 111 129 633,
Louis-Philippe Dubuc - 111 118 879, William Leclerc - 111 132 728

Analyse et conception de systèmes orientés objets
IFT-2007 – 91 229

Livrable 3

Travail présenté à
Martin Savoie

Département d'informatique et de génie logiciel
Automne 2016

Table des matières

Vision 4
- Opportunités d'affaires 4
- Énoncé du problème 4
- Énoncé de position sur le produit 4
- Alternatives et compétition 5
- Descriptions des parties prenantes 5
- Objectifs et problèmes des parties prenantes 6
- Objectifs utilisateurs 7
- Perspective du produit 7
- Résumé des avantages 8
- Le coût et la tarification 8
- Licences et installation 8
- Résumé des fonctionnalités du système 8

Cas d'utilisation 9

Domaine du modèle 23
- Explication du diagramme des classes et justification des cardinalités 24

Spécifications supplémentaires 27
- Fonctionnalité 27
- Journal 27
- Sécurité 27
- Facteur humain 27
- Fiabilité 27
- Recouvrabilité 27
- Performance 28
- Adaptabilité 28
- Configuration 28
- Contrainte d'implantation 28
- Composantes du logiciel libre 28
- Matériel 28
- Problème juridique 29
- Gestion des authentifications 29

Maquettes d'interface 30

Diagramme de classe de conception 36

Diagramme de packages 39

Diagrammes de séquence de conception 41
- Conversion des coordonnées 41
- Ajout d'un joueur à l'aide de la souris 43
- Déterminer si un joueur a été cliqué 44
- Interaction en mode image par image 45
- Interaction en mode temps réel 47
- Visionnement d'un jeu 48

Diagramme de classe de conception mis-à-jour 50

Diagrammes d'états 51
- Pour un joueur 51

Pour la rondelle **52**

Diagramme d'activité **53**

Créer une stratégie en mode image par image **53**

Diagramme de Gantt **54**

Glossaire **55**

Vision

Nous envisageons la création d'un nouveau système logiciel permettant un enseignement sportif simplifié par une interface graphique souple. L'interface simple et intuitive permettra son utilisation par différents clients dans le monde du sport et des loisirs.

Opportunités d'affaires

L'utilisation actuelle d'un tableau blanc magnétique pour la présentation de stratégies, tactiques et exercices sportifs est désuète. En effet, leurs durées de vie sont limitées, l'utilisation est peu pratique, ce n'est pas simple de comprendre et la visualisation des concepts peut être compliquée. Il a présentement des insatisfactions sur le marché et une demande importante pour corriger le problème.

Énoncé du problème

Il est difficile d'enseigner adéquatement les différents concepts sportifs avec les outils présentement disponibles ce qui entraine différents problèmes lors de la mise en œuvre des stratégies notamment un manque de coordination et une perte de temps. Les exercices doivent être expliqués à plusieurs reprises avant d'avoir le résultat escompté ce qui entraine un manque d'efficacité lors d'entrainement. Les personnes affectées sont les joueurs, les entraineurs et ultimement toute personne reliée à l'organisation sportive. Le mandat est de réaliser une application simplifiée pour les jeux d'entraineurs.

Énoncé de position sur le produit

Le système est conçu pour l'association des entraineurs mineurs du Québec. L'application *VisuaLigne* permet notamment la création de jeu et la sauvegarde de ceux-ci. Les enregistrements peuvent être visuellement accessibles dans l'application après le chargement.
L'application supportera deux modes de création. Le premier est un mode « image par image » qui permet de faire avancer les séquences une à une. Le deuxième mode est un mode « en temps réel » qui permet de faire des exercices plus efficacement. Ces deux modes sont des avantages concurrentiels face aux autres logiciels de ce type puisqu'il s'adapte facilement à l'environnement. Par exemple, on peut facilement décortiquer un nouveau jeu que les joueurs ne connaissent pas lors d'un entrainement avec le mode « image par image » pour faciliter la compréhension. D'un autre côté, l'entraineur peut utiliser le mode « en temps réel » pour aller directement au but lors d'un temps d'arrêt en pleine partie. L'entraineur peut aussi sélectionner un jeu enregistré préalablement et ainsi gagner du temps important sur un tableau blanc ordinaire. L'application possède aussi une fonctionnalité qui permet de faire pause lors de la démonstration ce qui permet de prendre le temps nécessaire pour l'explication des concepts qui ne sont pas bien compris par l'ensemble du groupe.
L'application est modélisée pour que son utilisation ne soit pas restreinte à un seul sport. En effet, il est possible d'utiliser un ballon, une rondelle ou une balle. Cet aspect de flexibilité permet d'aller chercher un auditoire plus général ce qui correspond mieux aux attentes de l'association des entraineurs mineurs du Québec. L'application permet notamment de créer un type de sport et ajouter un terrain correspondant avec celui-ci avec les dimensions en valeur réelle. Il est possible de définir le nombre de joueurs et leur catégorie. Les joueurs auront aussi une propriété permettant de leur définir un nom, un

numéro ou une position particulière. Cette propriété peut être activée ou désactivée afin de cacher les informations de façon à ne pas surcharger le jeu de texte. L'utilisateur pourra ajuster facilement l'orientation des joueurs en plus de pouvoir agrandir le focus sur le jeu à sa guise.
Afin de faciliter l'utilisation de l'application, plusieurs fonctionnalités seront développées pour permettre une interaction simplifiée avec elle. D'emblée, il est possible de faire un « annuler » et revenir à l'action précédente et ensuite d'avoir le choix de rétablir cette action. De plus, l'application permet d'exporter les jeux et stratégies en format image (JPEG, PNG, etc.). Les coordonnées de la souris seront affichées en tout temps afin de permettre une précision incomparable. Finalement, il est possible d'ajouter des obstacles de pratique comme des cônes. Les obstacles ne sont pas limités en nombre et il est possible d'en ajouter et d'en supprimer à volonté. Finalement, il est possible d'éditer l'image de l'obstacle.

Alternatives et compétition

Une des alternatives importantes pour notre application est *CoachNote : Sports Coach interactive whiteboard* disponible sur l'App Store d'Apple. C'est une des seules applications qui permet d'ajouter et d'enregistrer différents sports. Le problème majeur de cette application est la disponibilité en anglais uniquement. Cela peut facilement décourager certains entraineurs qui ne sont pas familiers avec la langue anglaise. Cependant, dans le cadre du développement de notre nouvelle application, *CoachNote* sera le principal compétiteur.
Les autres applications sur le marché ne sont pas bien conçues et leur utilisation compliquée rend le processus inefficace pour les entraineurs qui doivent apprendre à utiliser un nouvel outil rapidement.

Descriptions des parties prenantes

L'association des entraineurs mineurs du Québec donne le mandat de réaliser une application permettant de simplifier l'enseignement des stratégies de jeux pour les entraineurs.

Objectifs et problèmes des parties prenantes

Objectif de haut niveau	Priorité	Problèmes et préoccupation	Solution actuelle
Simplifier l'enseignement	haute	Perte de temps d'entrainement Manque de coordination dans les stratégies Concepts pas compris par les joueurs Difficulté des entraineurs à mettre en pratique de nouvelles stratégies (difficulté à progresser) Plus les stratégies sont complexes plus on passe de temps au tableau blanc	Le tableau blanc est limité par son utilisation physique sans aide technologique logiciel.
Nouvelles technologies	haute	Technologie désuète Manque d'outils pour travailler adéquatement Outils d'enseignement incomplets	Le tableau blanc traditionnel
Résultats sportifs positifs	basse	C'est l'objectif ultime de l'outil, mais pas nécessairement une priorité. Si les résultats ne se perçoivent pas sur le terrain ça ne veut pas dire que l'outil n'est pas utile lors d'entrainement et apprécié par les entraineurs Difficile de transiger les stratégies pratiquées vers des situations réelles de partie (ex : au hockey, beaucoup de temps est consacré aux avantages numériques, pourtant peu de buts y sont comptés)	Pratiquer le plus possible et revenir au tableau blanc, lorsque nécessaire

Capter l'attention	basse	Les joueurs ont tendance à perdre leur concentration lorsque l'entraineur explique des stratégies au tableau. Un outil technologique qui peut représenter les objets avec des images (les cônes sont représentés par des images, les joueurs par des joueurs, etc.) risque de capter l'attention des joueurs plus facilement et de les intéresser aux stratégies.	Demander la participation

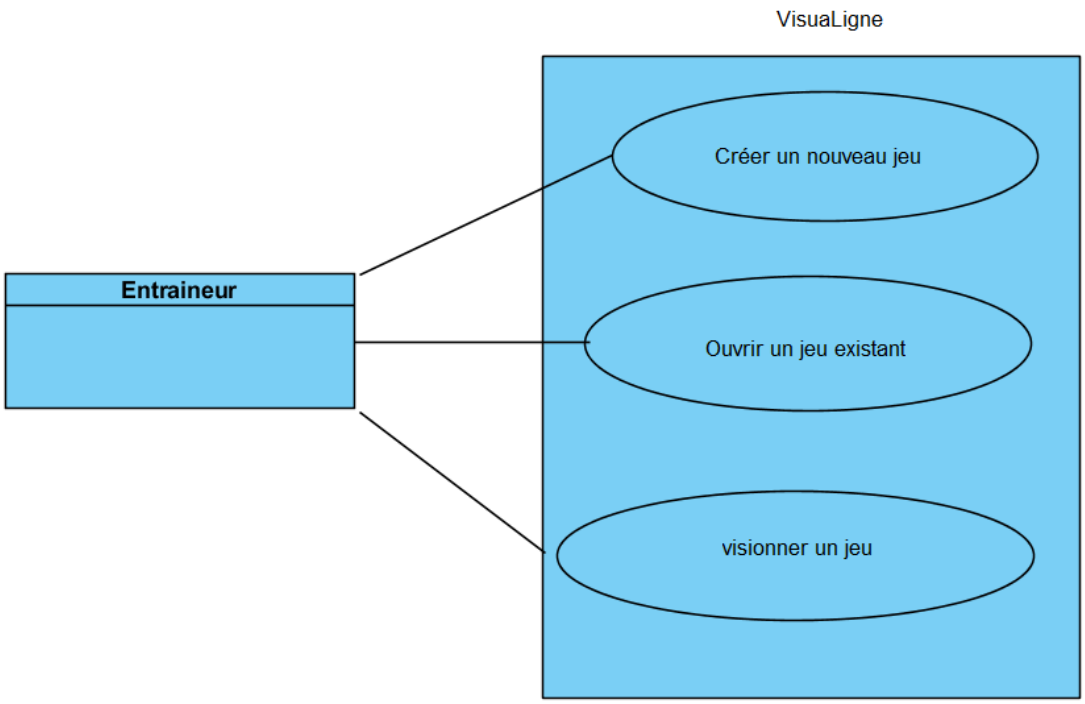
Objectifs utilisateurs

Entraineurs : Faciliter l'enseignement, le développement de stratégies et optimiser les temps d'entrainement.
Joueurs : s'intéresser davantage aux stratégies, les visualiser et les mettre en pratique.

Perspective du produit

Le logiciel *VisuaLigne* sera utilisé dans des stades, des arénas et des évènements sportifs autant pour les pratiques que pour les parties.

Diagramme de contexte :



Résumé des avantages

Fonctionnalités supportées	Bénéfices des parties prenantes
Logiciel permettant la création de jeux, de stratégies et de les enregistrer pour les réutiliser.	Optimiser les temps d'entrainement
Le logiciel permet d'annuler une action et de la rétablir au besoin et permet aussi de faire une pause sur la séquence en cours et, ensuite, de la reprendre.	Faciliter l'enseignement
Permets d'ajouter des images pour représenter les objets (cônes, ballons, rondelles, balles, etc.).	Développement et visualisation de stratégie

Le coût et la tarification

En se référant au temps consacré selon le plan de cours (1-2-6), on peut en déduire que deux des six heures de temps hors cours sont utilisées pour faire les ateliers et les lectures. Il reste donc 4 heures par étudiant par semaine pour la conception du projet durant les 15 semaines de la session. On peut estimer les coûts en utilisant les taux de traitement au 1er mai 2016 concernant les emplois étudiants de premier cycle universitaire qui est à 14,70 $. Le coût de conception du logiciel *VisuaLigne* est donc estimé à 3 528 $.
Avertissement : les tarifications sont basées sur les données disponibles par le plan de cours. Il faut considérer que le temps de 4 heures par semaine peut être largement dépassé pour la réalisation du projet.

Licences et installation

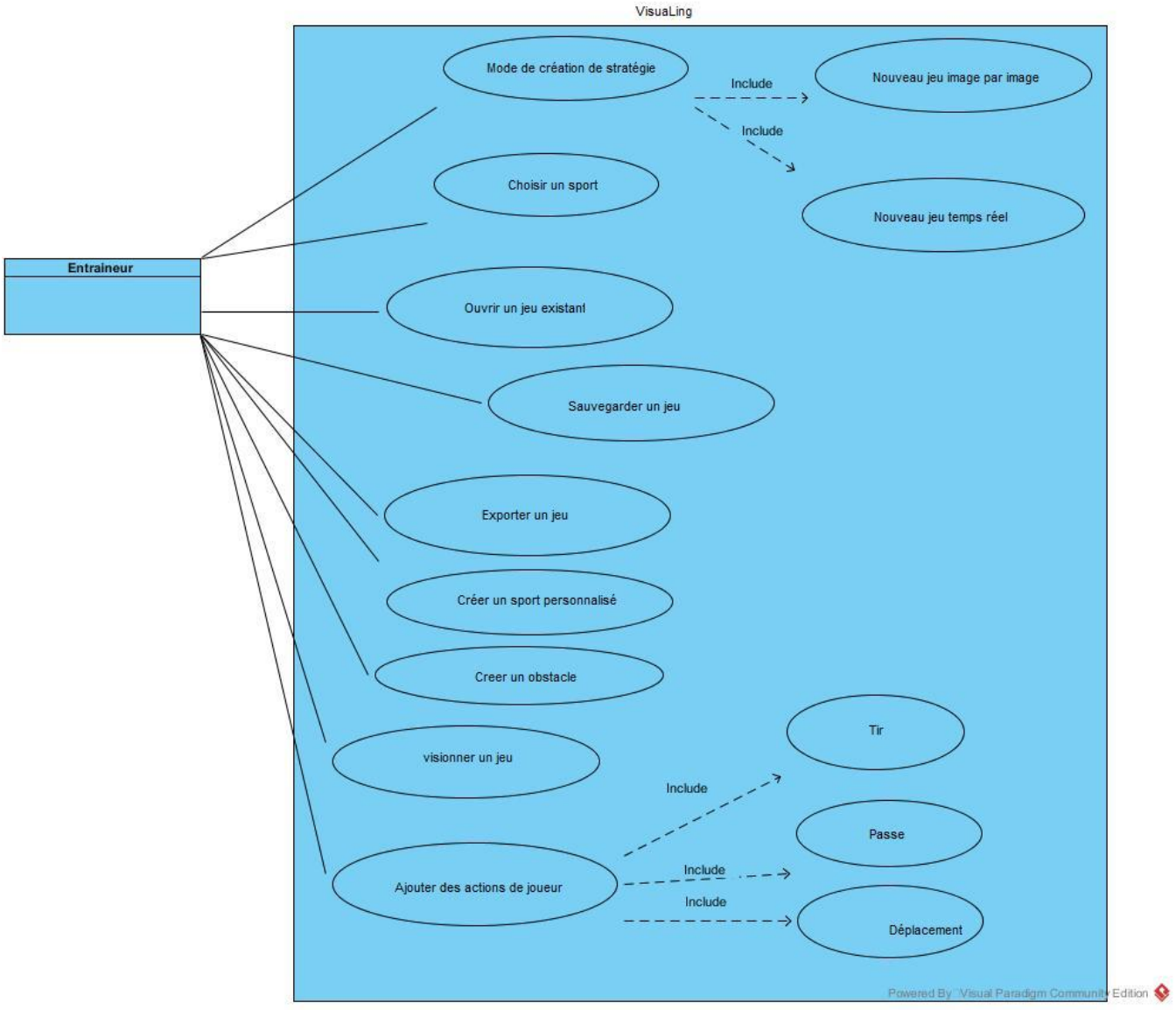
L'application est développée en Java ce qui demande seulement une installation simple pour l'utilisateur et c'est un langage qui est portable, c'est-à-dire qui peut être utilisé sous différents systèmes d'exploitation. Les licences sont celles appliquées par les règlements de l'Université Laval concernant les projets développés par des étudiants.

Résumé des fonctionnalités du système

- Création de jeu
- Sauvegarde de jeu
- Mode image par image
- Mode temps réel
- Déplacement interactif dans le temps (débuter, faire une pause, reculer)
- Sports personnalisés (ballon, balle, rondelle, etc.)
- Dimensions en valeur réelle

- Définir le nombre de joueurs, leur nom, leur position et leur numéro
- Ajouter des obstacles personnalisables (cônes, rondelles, etc.)

Cas d’utilisation



Cas d'utilisation :	Mode de création de stratégies
Système :	VisuaLigue
Acteurs :	Utilisateur
Parties prenantes et intérêts :	Utilisateur : Celui-ci désire créer une séquence de jeu
Pré conditions :	Le logiciel est ouvert et fonctionnel
Garantie en cas de succès :	Le nouveau jeu est créé
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur choisit l'option « Nouveau jeu » pour créer une nouvelle stratégie 4. Le logiciel initialise la fenêtre choix de mode. 5. Par la suite l'utilisateur va choisir son mode création. Il a le choix entre le mode image par image ou temps réel.
Scénario alternatif	L'utilisateur « Annule » l'action et retourne à l'interface d'accueil.

Cas d'utilisation :	Nouveau jeu image par image
Système :	VisuaLigue
Acteurs :	Utilisateur
Parties prenantes et intérêts :	Celui-ci désire créer une séquence de jeu image par image
Pré conditions :	Le bouton nouveau jeu a été sélectionné dans la fenêtre d'accueil.
Garantie en cas de succès :	Un mode de jeu séquence image par image a été créé
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre ouverte, l'utilisateur choisit l'option « Nouveau jeu » pour créer une nouvelle stratégie 4. Le logiciel initialise la fenêtre choix de mode. 5. Sur la fenêtre choix de mode, l'utilisateur choisit l'option « image par image ». 6. Le logiciel initialise la fenêtre choix du sport. 7. Une fois la séquence image par image créer, une nouvelle fenêtre va apparaitre pour que l'utilisateur puisse choisir le sport de sa séquence de jeu.
Scénario alternatif	L'utilisateur «Annule» l'opération et retourne au menu principal.
Variantes de données	Séquence image par image

Cas d'utilisation :	Nouveau jeu temps réel
Système :	VisuaLigue
Acteurs :	Utilisateur
Parties prenantes et intérêts :	L'utilisateur désire créer une séquence de jeu en temps réel
Pré conditions :	Le bouton nouveau jeu a été sélectionné dans la fenêtre d'accueil.
Garantie en cas de succès :	La séquence de jeu en temps réel a été créée
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre ouverte, l'utilisateur choisit l'option « Nouveau jeu » pour créer une nouvelle stratégie 4. Le logiciel initialise la fenêtre choix de mode. 5. Sur la fenêtre choix de mode, l'utilisateur choisit l'option « en temps réel». 6. Le logiciel initialise la fenêtre choix du sport. 7. Une fois la séquence en temps réel créée, une nouvelle fenêtre va apparaitre pour que l'utilisateur puisse choisir le sport de sa séquence de jeu.
Scénario alternatif	L'utilisateur «Annule» l'opération et retourne au menu principal.
Variantes de données	Séquence à temps réel

Cas d'utilisation :	Ouvrir un jeu existant
Système :	VisuaLigue
Acteurs :	Utilisateur
Parties prenantes et intérêts :	Celui-ci désire ouvrir une séquence de jeu déjà créée.
Pré conditions :	- Le logiciel est ouvert. - Il y a au minimum un jeu déjà créé.
Garantie en cas de succès :	Le jeu sélectionné par l'utilisateur est ouvert.
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne le jeu désiré avec un simple clic. 4. Le logiciel met le jeu sélectionné en surbrillance. 5. L'utilisateur sélectionne le bouton «Ouvrir». 6. Le logiciel vérifie que la sauvegarde est présente dans le dossier de sauvegarde. 7. Le logiciel ouvre l'interface de jeu sauvegardée étant sélectionné par l'utilisateur. 8. L'utilisateur peut travailler sur la séquence de jeu.
Scénario alternatif	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne le jeu désiré avec un simple clic. 4. Le logiciel met le jeu sélectionné en surbrillance. 5. L'utilisateur sélectionne le bouton «Ouvrir». 6. Le logiciel vérifie que la sauvegarde est présente dans le dossier de sauvegarde. 7. Un message d'erreur s'affiche comme quoi que le fichier de sauvegarde est corrompu ou n'est pas présent. 8. L'utilisateur ne peut pas ouvrir cette sauvegarde.
Variantes de données	Ouvrir un jeu

Cas d'utilisation :	Sauvegarder une séquence de jeu
Système :	VisuaLigue
Acteurs :	Utilisateur
Parties prenantes et intérêts :	Utilisateur : Enregistré la séquence de jeu créée
Pré conditions :	Une séquence de jeu a été créée.
Garantie en cas de succès :	Le jeu est enregistré et l'utilisateur peut ouvrir cette séquence où il l'a sauvegardée.
Scénario principal :	1. Sur l'interface principale, l'utilisateur sélectionne le bouton «Enregistré». 2. Le logiciel ouvre la fenêtre d'enregistrement. 3. L'utilisateur donne un nom à la séquence. 4. Le logiciel valide qu'il y a bien un seul document comportent ce nom dans le dossier de sauvegarde. 5. le logiciel enregistre une copie du document dans le dossier demandé. 6. L'utilisateur peut continuer à utiliser l'interface principale.
Scénario alternatif	1. Sur l'interface principale, l'utilisateur sélectionne le bouton «Enregistré». 2. Le logiciel ouvre la fenêtre d'enregistrement. 3. L'utilisateur donne un nom à la séquence. 4. Le logiciel valide qu'il y a bien un seul document comportent ce nom dans le dossier de sauvegarde. 5. Le logiciel affiche un message d'erreur comme quoi l'utilisateur doit sélectionner un autre nom parce qu'un fichier comportant le même nom se retrouve dans ce dossier. 6. L'utilisateur doit entrer un nom différent.
Variantes des données	Ce scénario s'applique de la même façon pour un mode d'utilisation de séquence image par image ou par séquence en temps réel.

Cas d'utilisation :	Exporter un jeu
Système :	VisuaLigue
Acteurs :	Utilisateur
Parties prenantes et intérêts :	Utilisateur : celui-ci désire exporter une séquence de jeu dans le format photo.
Pré conditions :	Une séquence de jeu est créée et sauvegardée
Garantie en cas de succès :	La séquence de jeu est exportée dans le format image.
Scénario principal :	1. Dans l'interface principale, l'utilisateur sélectionne le bouton «Exporter» 2. Le logiciel prend une capture d'écran en format JPEG de l'interface de jeu. 3. Le logiciel va placer la capture d'écran dans un dossier prédéfini. 4. L'utilisateur peut récupérer la séquence de jeu exportée en JPEG
Scénario alternatif	1. Dans l'interface principale, l'utilisateur sélectionne le bouton «Exporter» 2. Le logiciel prend une capture d'écran en format PNG de l'interface de jeu. 3. Le logiciel va placer la capture d'écran dans un dossier prédéfini. 4. L'utilisateur peut récupérer la séquence de jeu exportée en PNG

Cas d'utilisation :	Créer un sport personnalisé
Système :	VisuaLigue
Acteurs :	Utilisateur
Parties prenantes et intérêts :	Celui-ci désire créer un sport pour effectuer une séquence de jeu
Pré conditions :	Une séquence de jeu image par image ou en temps réel a été créée.
Garantie en cas de succès :	Le sport est créé et la surface de jeu s'affiche sur l'interface principale.
Scénario principal :	1. Dans l'interface de création d'un sport, l'utilisateur donne un nom à celui-ci. 2. L'utilisateur indique le nombre de joueurs. 3. L'utilisateur indique les dimensions du terrain (en mètre). 4. L'utilisateur ajoute les différentes positions qu'il désire. 5. L'utilisateur ajoute les différents rôles qu'il désire. 6. L'utilisateur clic sur le bouton «parcourir». 7. Le logiciel affiche la fenêtre de recherche dans le but que l'utilisateur affiche une image de format PNG ou JPEG. 8. L'utilisateur ajoute l'image de terrain désirée 9. Le logiciel valide que c'est bien un fichier de format PNG ou JPEG. 10. Le logiciel avertit l'utilisateur que l'image a été ajoutée à l'interface. 11. L'utilisateur sélectionne le bouton «valider» pour créer le sport. 12. Le logiciel valide que l'utilisateur n'a pas laissé d'espace d'information vide. 13. Le logiciel affiche l'interface principale.
Scénario alternatif	1.b L'utilisateur « Annule » l'action et retourne à l'interface d'accueil. 9.c Le logiciel voit que le fichier entré par l'utilisateur n'est pas un fichier de format PNG et JPEG. 10.d Le logiciel avertit l'utilisateur que le format entré n'est pas valide. 12.e Le logiciel avertit que l'utilisateur a omis d'entrer une information nécessaire. On lui précise quelle information est manquante. 13.e L'utilisateur entre l'information manquante.
Variantes de données	Nom du sport, nombre de joueurs, dimension du terrain X et Y, position, rôle, image du terrain.

Cas d'utilisation :	Créer un obstacle
Système :	VisuaLigue
Acteurs :	Entraineur
Parties prenantes et intérêts :	L'utilisateur désire ajouter un obstacle sur la stratégie.
Pré conditions :	L'utilisateur doit charger un jeu existant ou ouvrir un nouveau jeu.
Garantie en cas de succès :	Un obstacle est ajouté sur le terrain.
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne l'option de nouveau jeu ou de jeu existant. 4. Le logiciel ouvre et charge l'interface du nouveau jeu ou du jeu existant. 5. L'utilisateur sélectionne l'option d'ajouter des obstacles. 6. Le logiciel charge les obstacles disponibles pour l'insertion. 7. L'utilisateur peut ajouter les obstacles sur le terrain du jeu.
Scénario alternatif	L'utilisateur ajouter un obstacle à l'extérieur des limites du terrain et le système retourne un message indiquant que les limites sont dépassées. L'utilisateur ajoute un obstacle sur un autre obstacle avec les mêmes coordonnées et le système retourne un message indiquant que l'action est impossible.
Variantes des données	Les joueurs, les cônes, les rondelles, les ballons et les balles sont tous considérés comme des obstacles.

Cas d'utilisation :	Visionner un jeu
Système :	VisuaLigue
Acteurs :	Entraineur
Parties prenantes et intérêts :	L'utilisateur désiré visionner une stratégie sans pouvoir la modifier.
Pré conditions :	La stratégie doit avoir été créée et enregistrer.
Garantie en cas de succès :	La stratégie est ouverte en « lecture seule »
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne l'option de visionner un jeu. 4. Le logiciel ouvre et charge l'interface du jeu. 5. L'utilisateur a accès à la stratégie en Lecture seulement (il peut lire, reculer, avancer et faire pause sur la lecture de celui-ci).
Scénario alternatif	L'utilisateur sélectionne une action de modification et le système retourne un message indiquant que la stratégie est en mode de visionnement.
Variantes des données	Les options de modification sont désactivées.

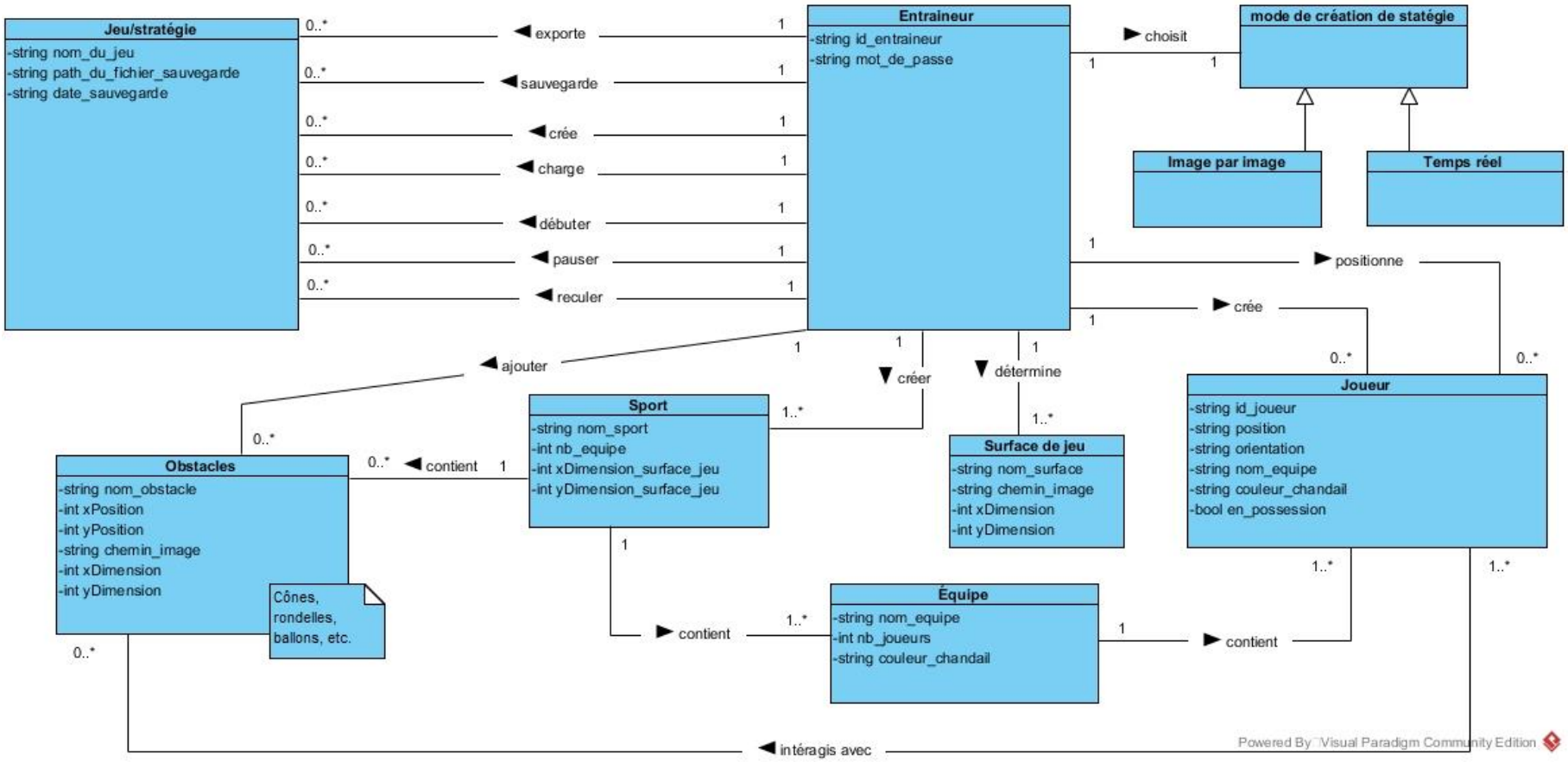
Cas d'utilisation :	Ajouter des actions au joueur
Système :	VisuaLigue
Acteurs :	Entraineur
Parties prenantes et intérêts :	Celui-ci désir désire ajouter une action de « tir » dans sa stratégie.
Pré conditions :	Un obstacle de type joueur doit être ajouté.
Garantie en cas de succès :	Des actions seront disponibles pour ajouter aux joueurs.
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne l'option de nouveau jeu ou de jeu existant. 4. Le logiciel ouvre et charge l'interface du nouveau jeu ou du jeu existant. 5. L'utilisateur sélectionne l'option des actions de joueurs. 6. Le système affiche le bouton comme actif/enfoncé et propose les différentes actions de joueurs.
Scénario alternatif	Les actions de joueurs ne s'affichent pas et le système retourne un message dans la barre d'état pour dire qu'aucun joueur n'a été créé.
Variantes des données	Les différents types d'actions sont : Tir, passe et déplacement.

Cas d'utilisation :	Insérer un Tir
Système :	VisuaLigue
Acteurs :	Entraineur
Parties prenantes et intérêts :	Celui-ci désir désire ajouter une action de « tir » dans sa stratégie.
Pré conditions :	L'acteur a sélectionné l'option pour ajouter des actions aux joueurs.
Garantie en cas de succès :	Une action de tir est ajoutée au joueur.
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne l'option de nouveau jeu ou de jeu existant. 4. Le logiciel ouvre et charge l'interface du nouveau jeu ou du jeu existant. 5. L'utilisateur sélectionne l'option des actions de joueurs. 6. Le système affiche le bouton comme actif/enfoncé et propose les différentes actions de joueurs. 7. L'utilisateur sélectionne l'option de tir et l'ajoute sur le jeu.
Scénario alternatif	L'utilisateur tente d'ajouter un Tir à l'extérieur des limites du terrain de jeu et le système retourne un message qui indique que les limites du terrain ont été dépassées.
Variantes des données	Tir

Cas d'utilisation :	Insérer une Passe
Système :	VisuaLigue
Acteurs :	Entraineur
Parties prenantes et intérêts :	Celui-ci désir désire ajouter une action de « passe » dans sa stratégie.
Pré conditions :	L'acteur a sélectionné l'option pour ajouter des actions aux joueurs.
Garantie en cas de succès :	Une action de passe est ajoutée au joueur.
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne l'option de nouveau jeu ou de jeu existant. 4. Le logiciel ouvre et charge l'interface du nouveau jeu ou du jeu existant. 5. L'utilisateur sélectionne l'option des actions de joueurs. 6. Le système affiche le bouton comme actif/enfoncé et propose les différentes actions de joueurs. 7. L'utilisateur sélectionne l'option de passe et l'ajoute sur le jeu.
Scénario alternatif	La « passe » dépasse la limite du terrain et le système informe que l'action est impossible.

Cas d'utilisation :	Déplacer un joueur sur l'interface
Système :	VisuaLigue
Acteurs :	Entraineur
Parties prenantes et intérêts :	Celui-ci désir désire déplacer un « joueur » dans sa stratégie.
Pré conditions :	L'acteur a sélectionné l'option pour ajouter des actions aux joueurs.
Garantie en cas de succès :	Une action de déplacement est ajoutée au joueur.
Scénario principal :	1. L'utilisateur ouvre l'application. 2. Le logiciel initialise la fenêtre d'accueil. 3. Sur la fenêtre d'accueil, l'utilisateur sélectionne l'option de nouveau jeu ou de jeu existant. 4. Le logiciel ouvre et charge l'interface du nouveau jeu ou du jeu existant. 5. L'utilisateur sélectionne l'option des actions de joueurs. 6. Le système affiche le bouton comme actif/enfoncé et propose les différentes actions de joueurs. 7. L'utilisateur sélectionne le joueur et le déplace sur le jeu, les coordonnées de départ et ceux d'arriver (X, Y) représenteront le déplacement du joueur
Scénario alternatif	Le joueur n'est pas déplacé et celui-ci conserve ses coordonnées (X, Y). Le joueur est déplacé en dehors des limites du terrain et le système retourne un message qui indique que les limites ont été dépassées.
Variante des données	Les données en X et en Y

Domaine du modèle



Explication du diagramme des classes et justification des cardinalités

La section qui suit est une suite d'explications et de justifications en ce qui concerne les cardinalités de notre diagramme de classes conceptuelles. Tout d'abord, dans notre architecture, un entraineur ne peut accéder à l'application qu'avec un nom d'utilisateur et un mot de passe fourni par l'AEMQ. De cette façon, un seul entraineur peut utiliser l'application à la fois. C'est donc pourquoi toutes les actions partant de l'acteur Entraineur ont une cardinalité de 1. Les principales actions pour lesquelles l'entraineur participe sont indiqués dans le diagramme de classes conceptuelles et sont listées ci-dessous :

Créer un jeu/stratégie
Un entraineur peut ne créer aucun jeu ou peut en créer plusieurs, ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action créer un jeu.

Sauvegarder un jeu/stratégie
Un entraineur peut ne pas sauvegarder son jeu ou peut le sauvegarder plusieurs fois, ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action sauvegarder un jeu.

Charger un jeu/stratégie
Un entraineur peut ne pas charger son jeu ou peut le charger plusieurs fois, ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action charger un jeu.

Exporter un jeu/stratégie en format jpg, png, etc.
Un entraineur peut ne pas exporter son jeu ou peut l'exporter plusieurs fois, ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action exporter un jeu.

Débuter un jeu/stratégie (déplacements)
L'action de débuter un jeu est l'action que peut faire l'entraineur lorsque celui-ci à déterminé tous les déplacements de ses joueurs. Il peut alors appuyer sur 'Débuter' et, selon le mode de création de stratégie choisit, les déplacements de ses joueurs se feront image par image ou en temps réel. L'entraineur peut ne pas appuyer sur 'Débuter', par exemple s'il ne veut que démontrer un positionnement à respecter à ses joueurs, ou il peut appuyer sur 'Débuter' plusieurs fois, ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action débuter jeu. Notons que lorsque la séquence de déplacements de joueurs touche à sa fin, la séquence se met ensuite automatiquement sur pause au temps 0 de la séquence. L'entraineur peut donc appuyer autant de fois qu'il le veut sur débuter pour recommencer la séquence.

Arrêter/pause un jeu/stratégie (déplacements)
Un entraineur peut ne pas arrêter un jeu comme il peut l'arrêter plusieurs fois. C'est pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action pauser un jeu.

Reculer un jeu/stratégie (déplacements)
Un entraineur peut ne pas reculer un jeu comme il peut le reculer plusieurs fois. C'est pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action reculer un jeu.

Créer un sport ainsi
Un entraineur peut créer aucun ou plusieurs sport(s), ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action créer un sport.

Déterminer surface de jeu
Lorsque l'entraineur créer un sport, celui-ci doit absolument choisir une surface de jeu. L'application fournit certaines images de surfaces de jeux de certains sports. La cardinalité est donc dans ce cas 1..* puisqu'il peut modifier à tout moment la surface de jeu.

Ajouter des obstacles dans le jeu
Un entraineur peut ajouter aucun obstacle, par exemple dans un exercices de patinage pour le hockey, ou plusieurs obstacle(s), ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action ajouter un obstacle.

Créer un ou des joueurs
Un entraineur peut ne pas créer de joueurs ou il peut en créer plusieurs joueur(s), ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action créer un joueur. Notons qu'un entraineur pourrait se servir de l'application que pour montrer l'emplacement des rondelles et des cônes à ses entraineurs adjoints après avoir montré aux joueurs leurs déplacements.

Modifier la position d'un ou de plusieurs joueurs
Un entraineur peut ne pas modifier la position d'un joueurs comme il peut modifier la position de plusieurs joueurs, ce pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action modifier position joueur. Notons qu'un entraineur peut ne pas avoir à déplacer ses joueurs dans le cas où celui-ci les a tous bien positionné au moment de leur création.

Choisir le mode de création de stratégie (image par image ou temps réel)
Un entraineur a deux choix en ce qui concerne le mode de création de stratégie : soit image par image, soit en temps réel. Puisque qu'il ne choisit un seul mode entre les deux, nous avons une cardinalité de 1 à l'arrivé de l'action choisir mode de création de stratégie.

Voici les autres associations présentent dans notre diagramme de classes conceptuelles.

Un sport contient un ou des obstacles
Un entraineur peut aussi bien mettre 50 rondelles en action dans un exercice que ne pas en mettre de tout, par exemple pour un exercice de patin pour le hockey. C'est pourquoi nous avons 0..* comme cardinalité d'arrivé de l'action sport contient obstacle. Puisque nous travaillons avec un seul sport à la fois, la cardinalité de départ de Sport est de 1.

Un sport contient un ou des équipes
Le Larousse définit le sport comme étant un ensemble d'exercices physiques se présentant sous forme de jeux individuels ou collectifs, donnant généralement lieu à compétition, pratiqués en observant certaines règles précises. Un sport contient donc nécessairement une ou plusieurs équipes constituées d'une seule personne ou de plusieurs personnes. C'est pourquoi la cardinalité d'arrivé de l'action sport contient équipe est 1..*. Puisque nous travaillons avec un seul sport à la fois, la cardinalité de départ de Sport est de 1.

Une équipe contient des joueurs

Comme définit ci-dessous, une équipe peut être constituée d'un ou de plusieurs individus, nous avons donc une cardinalité 1..* comme cardinalité d'arrivé de l'action équipe contient joueur. Puisqu'un joueur ne peut faire partie que d'une seule équipe à la fois, nous avons une cardinalité de départ de 1 pour sport.

Un joueur interagis avec des obstacles

Puisque les obstacles représentent les rondelles, ballons, cônes et autres, nous avons entre joueur et obstacles l'action d'interagir. Dépendamment du nombre d'obstacles ajouté par l'entraineur, le joueurs interagis avec 0 ou plusieurs obstacles, donc 0..*. Puisque les sports sont des sports d'équipes formés d'un seul ou de plusieurs individus, nous avons 1..* comme cardinalité de départ de joueur, ce qui veut dire que plusieurs joueurs peuvent interagir avec les obstacles ou qu'un seul joueur peut interagir avec les obstacles, comme par exemple dans un cours privé de hockey.

Spécifications supplémentaires

Cette section comprend toute les composantes nécessaires pour *VisuaLigue* qui ne sont pas incluses dans les cas d'utilisations.

Fonctionnalité

Fonctionnalités communes à plusieurs cas d'utilisations :

- Authentification
- Création d'un jeu
- Chargement d'un jeu

Journal

Toutes les erreurs seront gérées par le programme

Sécurité

Tous les utilisateurs seront authentifiés préalablement.
Le code logique du domaine est protégé par un contrôleur qui appel lui-même les fonctions nécessaires au fonctionnement de l'interface graphique (UI).

Facteur humain

Les joueurs doivent voir la police de caractères sur un écran large (le texte doit être visible à 5 mètres).
Avoir un logiciel convivial est primordial, si le logiciel est trop compliqué et devient un irritant pour l'utilisateur qui préfèrera la simplicité d'un tableau blanc traditionnel.

Fiabilité

L'application avertit l'utilisateur de sauvegarder le travail avant de fermer l'application.
L'application est développée de façon à ne pas générer des problèmes pouvant compromettre le travail en cours.

Recouvrabilité

Il est important de faire la gestion des risques du logiciel. Il est possible d'avoir un échec de fonctionnement du système qui est causé par différentes raisons dépendantes ou indépendantes du logiciel. Bref, peu importe la cause, on ne peut négliger un arrêt inattendu puisque plusieurs facteurs externes peuvent influencer les processus du logiciel.

Dans le cas d'un arrêt inattendu de l'application, nous assurons l'intégrité des dernières sauvegardes disponibles. Il est impossible de garantir là un travail qui n'est pas sauvegardé.

Performance

L'application orientée objet permet une bonne performance du logiciel. L'utilisateur ne doit pas trouver l'application lente pour atteindre notre objectif de convivialité. En effet, le but est qu'il n'y ait pas de lenteur qui pousserait le client à retourner vers une méthode traditionnelle.

Adaptabilité

L'utilisateur possède plusieurs scénarios possibles tels que différents types de sauvegarde, d'exportation, plusieurs types de sports et deux modes de création de jeux.

Configuration

L'utilisateur a plusieurs choix de configuration au niveau du terrain utiliser et des obstacles ajoutés sur celui-ci.

Contrainte d'implantation

On utilise le langage de programmation Java avec un design orienté objet. Le code d'interac sera séparé du code logique du domaine pour permettre une durée de vie à long terme et faciliter le développement futur.

Composantes du logiciel libre

- Java
- Netbeans
- Java JDK
- JavaFX
- Oracle JavaFX scene builder 2.0

Matériel

- Moniteur
- Micro-ordinateur ou autre appareil technologique avec un système d'exploitation supportant Java

Problème juridique

Notre application et les composantes sont développées avec le code source. N'est pas destiné pour la revente.

Gestion des authentifications

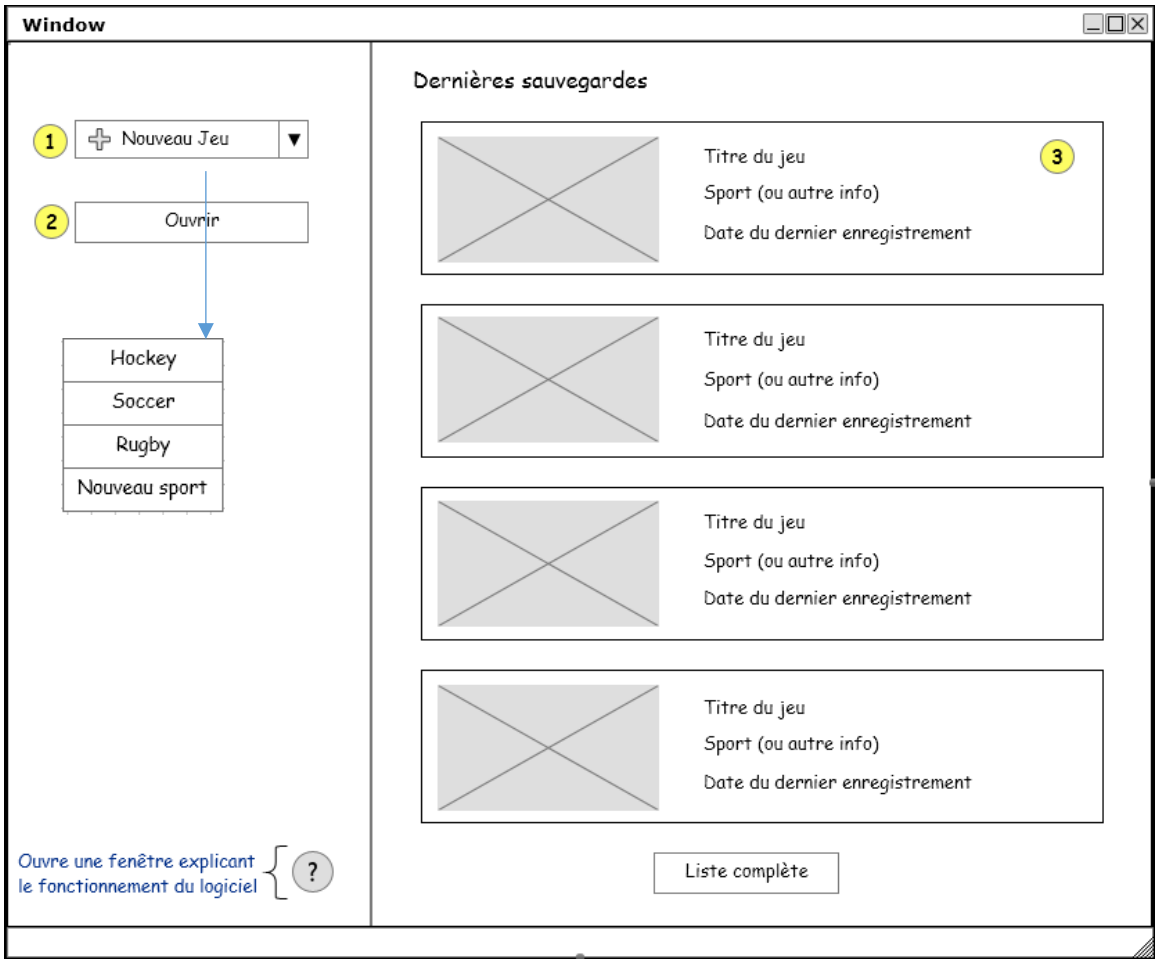
L'association des entraineurs mineurs du Québec fournira, au lancement du logiciel, un mot de passe ainsi qu'un identifiant utilisateur aux entraineurs détenteurs de l'application. Le nom d'utilisateur et le mot de passe sont stockés en local dans le logiciel. La vérification des authentifications se font donc elle aussi en local.

Maquettes d'interface

Connexion

Page de connexion qui s'affiche à l'ouverture du logiciel *VisuaLigue*. L'utilisateur doit entrer la bonne combinaison nom d'utilisateur/mot de passe pour pouvoir utiliser le logiciel.

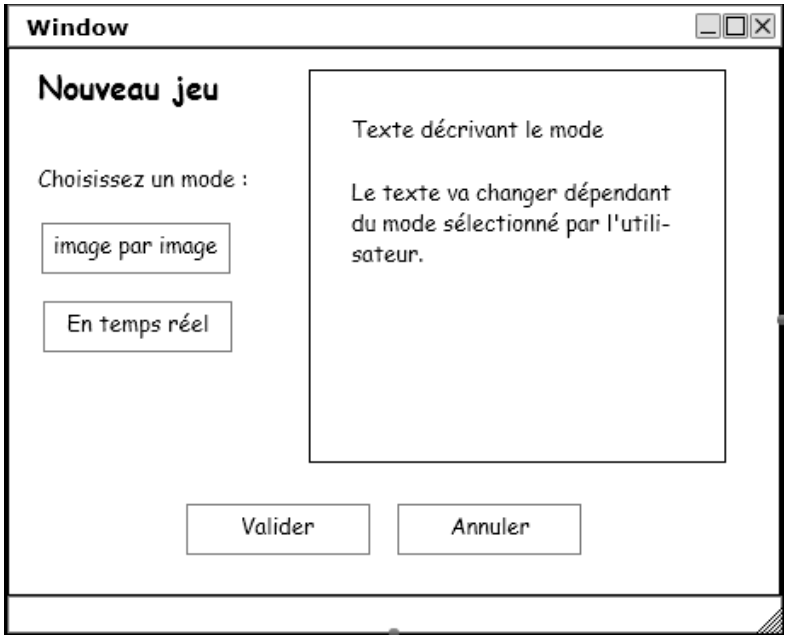
Accueil



L'accueil est la première fenêtre avec laquelle l'utilisateur pourra interagir après s'être connecté au logiciel. Il a le choix ici de créer un nouveau jeu ou d'ouvrir un jeu créé auparavant.

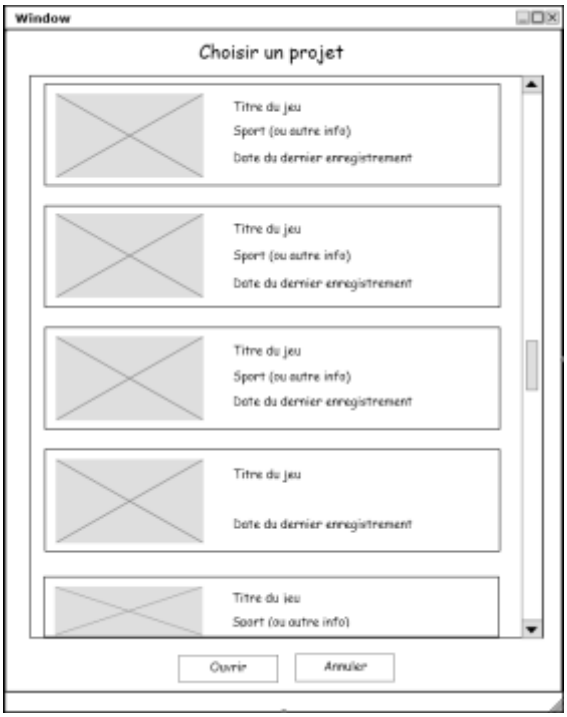
1. Menu déroulant permettant de créer un nouveau jeu en choisissant un sport déjà présent dans la liste ou de cliquer sur *nouveau sport* pour ouvrir la fenêtre de création de sport (voir Nouveau sport). Après avoir sélectionné un sport, la fenêtre de sélection de mode va s'afficher (voir Sélection du mode de jeu).
2. Ce bouton permet d'ouvrir une fenêtre système de sélection de fichiers pour ouvrir un fichier de jeu déjà créer auparavant, mais qui ne se retrouverait pas dans le dossier par défaut du logiciel.
3. Liste affichant les derniers jeux créés en plus de renseigner sur le titre du jeu, le sport et la date de création. Le bouton à la fin de la liste permet d'ouvrir une fenêtre affichant tous les jeux enregistrés et qui se retrouvent dans le dossier par défaut du logiciel (voir Liste des jeux enregistrés).

Sélection du mode de jeu



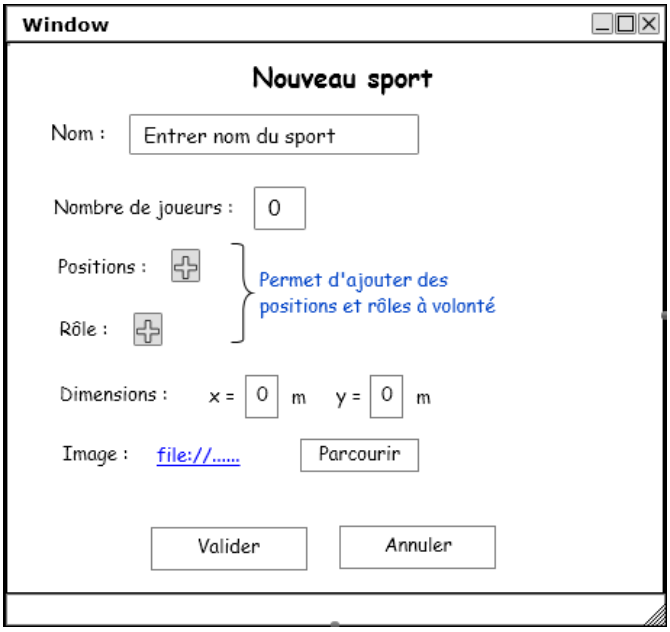
Fenêtre permettant de choisir le mode de jeu, soient « image par image » ou « en temps réel ». En cliquant sur un des deux choix, une courte description du mode apparait automatiquement dans le rectangle à droite.

Liste des jeux enregistrés



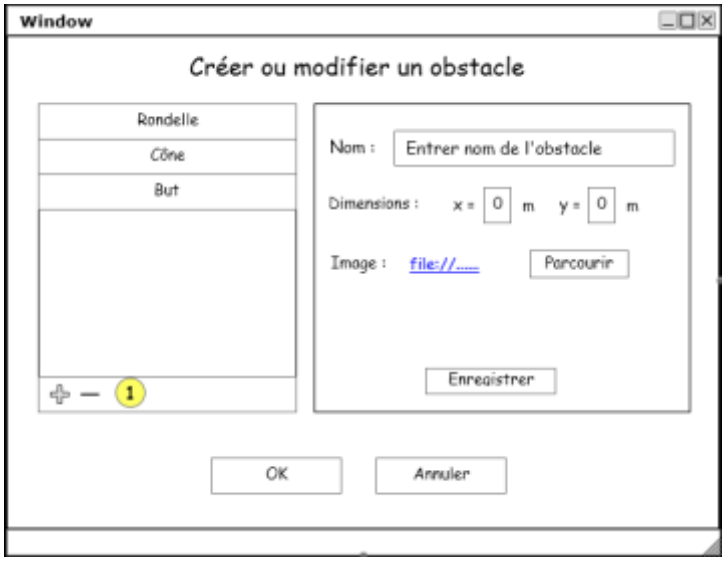
Fenêtre qui apparait après avoir cliqué sur le bouton « Liste complète » de la page d'accueil. On y retrouve la liste de tous les jeux enregistrés auparavant et qui sont placés dans le dossier par défaut choisi par les développeurs du logiciel. Pour chaque jeu, on y retrouve le nom de celui-ci, le sport, ainsi que la date du dernier enregistrement.

Nouveau sport



Cette fenêtre permet de créer un nouveau sport. Tous les champs, sauf « Rôle », sont obligatoires.

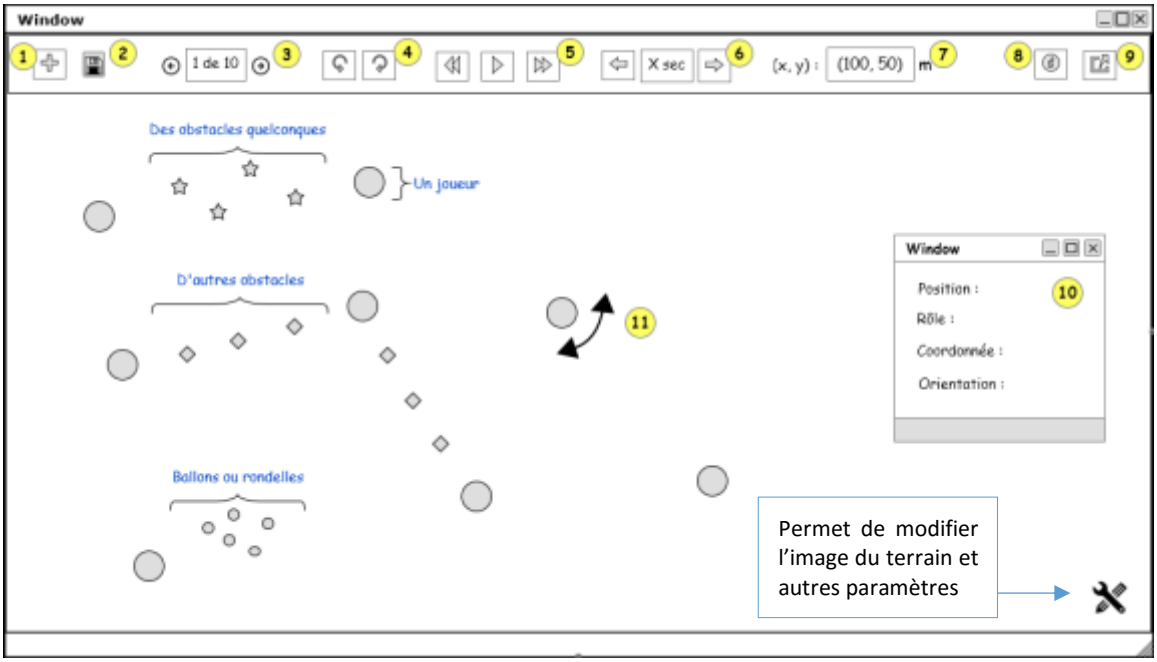
Créer ou modifier un obstacle



Cette page permet de modifier, créer et supprimer un obstacle. L'utilisateur peut y modifier le nom, les dimensions de l'obstacle et l'image affichée pour représenter l'obstacle.

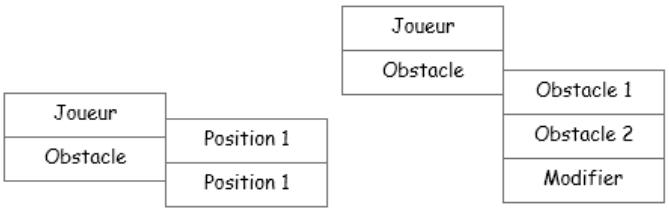
1. Les boutons « + » et « - » permettent d'ajouter un obstacle, ou d'en supprimer un lorsque l'un d'eux est sélectionné.

Mode image par image



1. Le bouton « + » ouvre un menu déroulant qui permet à l'utilisateur d'ajouter soit un joueur ou un obstacle. Ensuite, deux sous-menus s'offrent à l'utilisateur, lui permettant de choisir la position du joueur ou le type d'obstacle à ajouter. Pour les obstacles, une case à la fin de la liste permet de modifier

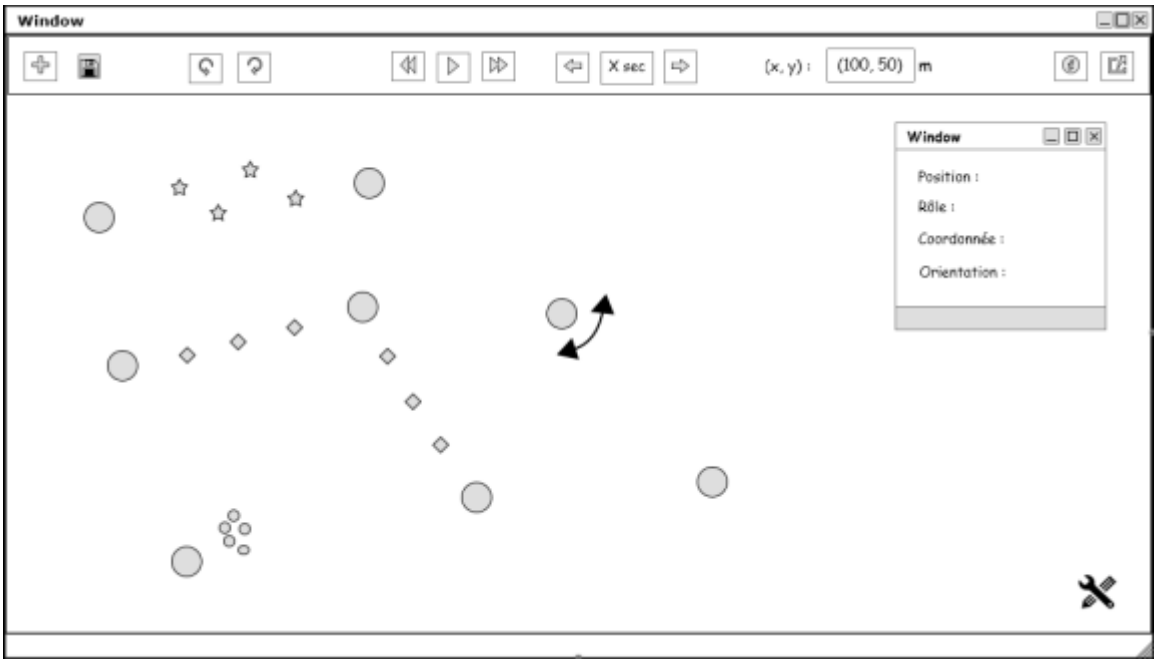
la liste, c'est-à-dire, d'ajouter de modifier ou de supprimer un des types d'obstacle. Exemple du menu déroulant :



2. Bouton permettant d'enregistrer le jeu en cours de création.
3. L'encadrer indique à quelle page l'utilisateur se trouve actuellement. Les flèches permettent de se déplacer de page en page.
4. Les boutons undo/redo permettent de rétablir ou d'annuler une action faite.
5. Les commandes de lecture permettent de démarrer la stratégie, de la reculer, de l'avancer et de la mettre sur pause.
6. L'utilisateur peut entrer un nombre de secondes dans l'encadrer et en cliquant sur les flèches, il pourra avancer ou reculer la stratégie du nombre de secondes qu'il a inscrit.
7. L'encadrer affiche les coordonnées en temps réel de la position de la souris. L'utilisateur ne peut pas entrer de nombre dans cet encadré.
8. Permet d'ouvrir la petite fenêtre décrite au point 10.
9. Bouton d'exportation qui ouvre un menu déroulant permettant à l'utilisateur de choisir sous quel type de fichier il veut exporter sa stratégie (ex : PNG, JPEG, etc.).
10. Cette fenêtre, qui peut être fermée et ouverte à tout moment, donne de l'information sur le joueur ou l'obstacle sélectionné, par exemple, en donnant les coordonnées.
11. En sélectionnant un joueur, un flèche courbe apparait autour du joueur permettant à l'utilisateur de changer son orientation.

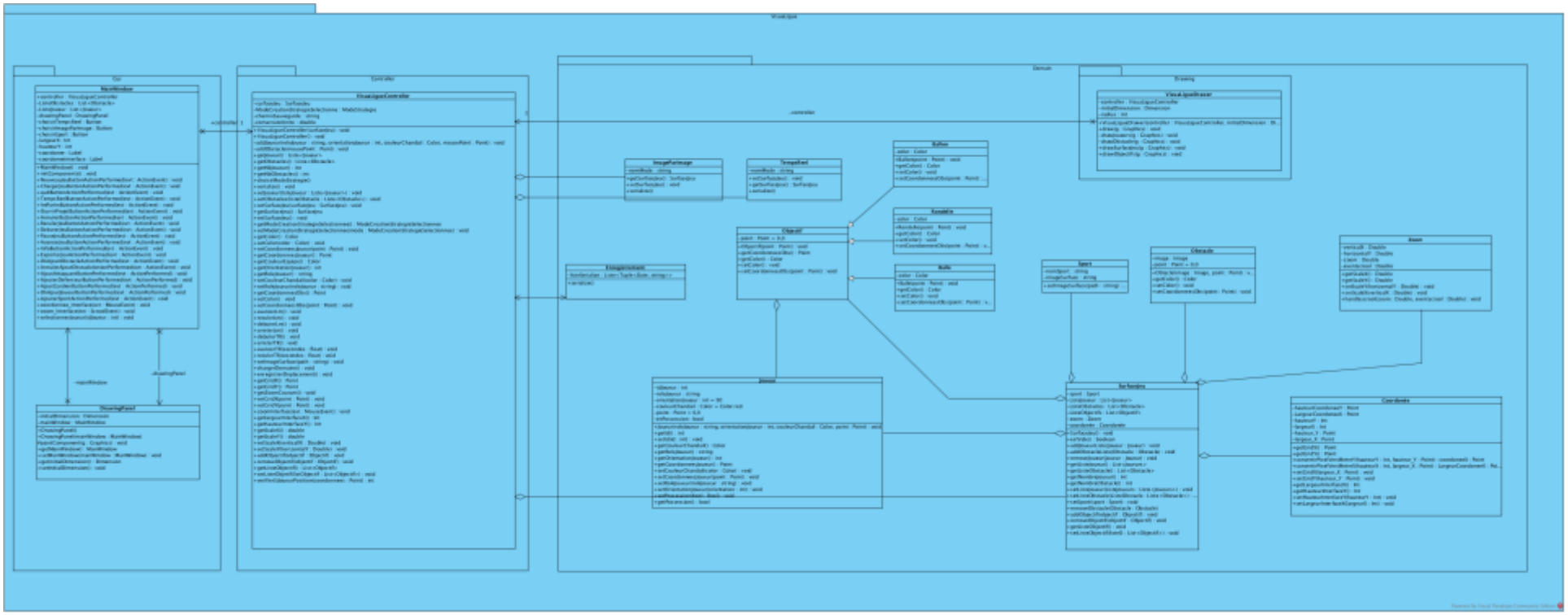
Note : les joueurs et/ou obstacles peuvent être effacés en cliquant dessus pour le sélectionner et en cliquant sur la touche *supprimer* ou *effacer*.

En temps réel



Le seul élément de l'interface qui change du mode « image par image », c'est l'encadrer avec ses flèches qui permettait de changer d'image. Ce composant est tout simplement inutile pour ce mode.

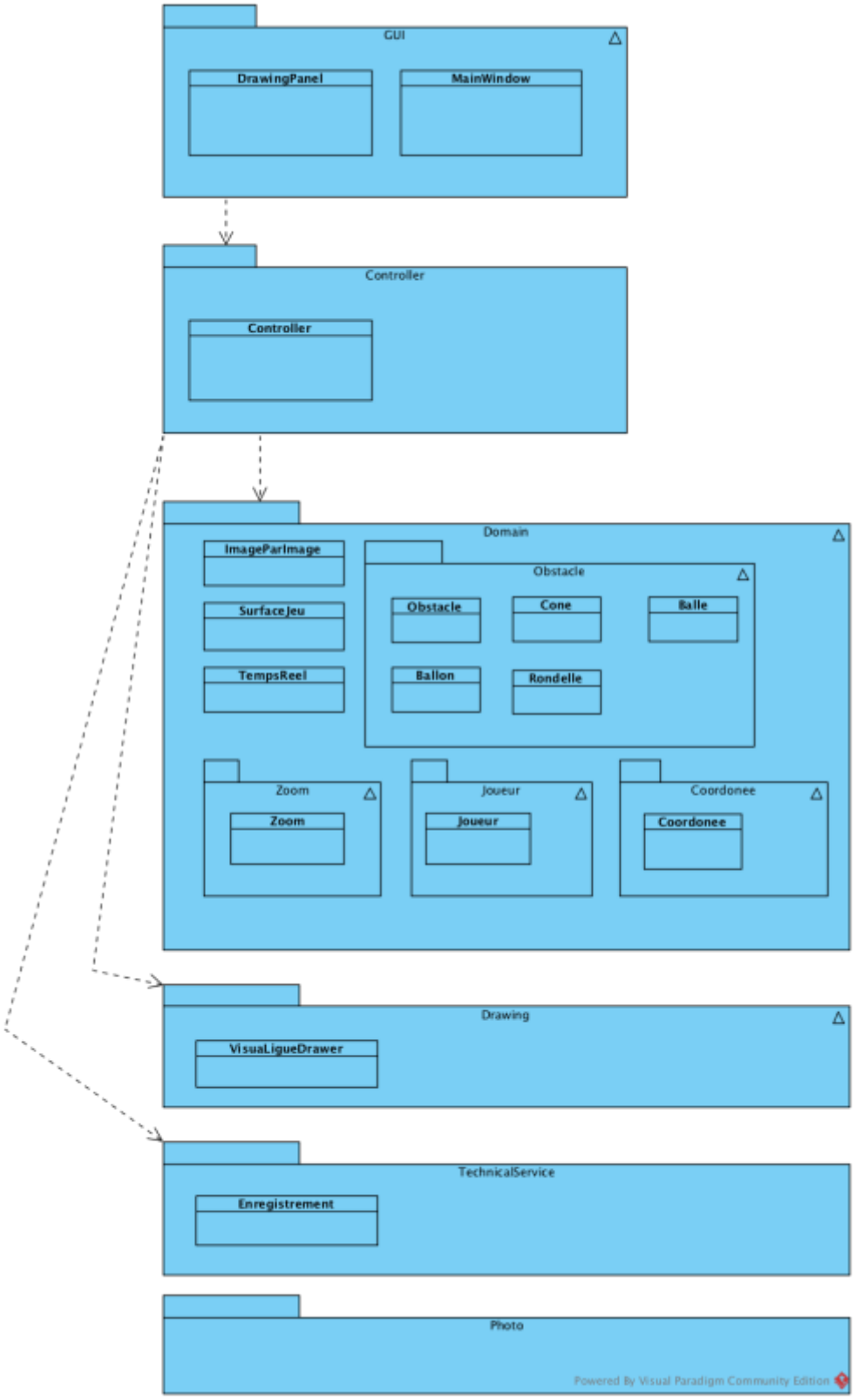
Diagramme de classe de conception



Notre diagramme de classe est séparé en 3 couches, soit le GUI, le contrôleur de Larman, et notre domaine. Nous avons aussi une classe externe, qui s'occupe de la sauvegarde Serialize de notre contrôleur.
Tout d'abord, notre GUI contient notre MainWindow, l'interface de notre logiciel, et du DrawingPanel, qui s'occupe de dessiner les éléments de notre MainWindow.
En ce qui concerne notre domaine, nous avons en grande partie récupéré les classes de notre diagramme de classes conceptuelles, mais avons aussi ajouté certains concepts. Un des grands changements à notre modélisation est l'arrivée de la classe SurfaceJeu. Pour faire un lien avec les ateliers du Wiki, la classe SurfaceJeu a plutôt la même fonctionnalité que la classe Basket de FruitBasket. La classe SurfaceJeu contient, d'un côté, un objet de type sport, qui lui, contient l'image représentant la surface de jeu, donc un string. L'objet SurfaceJeu contient aussi tous les éléments Joueurs, Objectifs et Obstacle présents sur celle-ci. Ses attributs sont donc un sport, une liste de Joueurs, une liste d'objectifs ainsi qu'une liste de Obstacles, ce qui justifie les agrégations entre SurfaceJeu, Sport, Obstacle, Objectif et Joueur, et ses méthodes sont principalement des Get et des Set pour travailler sur ces attributs. La classe Joueur elle, a été récupérée de notre diagramme de classe conceptuelle. Les attributs d'un joueur sont son orientation, sur 360 degrés, son rôle, sa couleur de chandail, ses coordonnées en X et en Y, un boolean qui indique si le joueur possède un objectif du sport (rondelle, ballon ou balle) et son identifiant qui lui permet d'être unique, géré par le logiciel. La classe Objectif a été rajoutée. Celle-ci contient comme attribut une couleur, pour représenter la balle/ballon/rondelle ainsi qu'un point, soit une coordonnée en X et Y. Les objets Balle, Ballon et Rondelle sont tous des objets de type Objectif. La classe Objectif est en lien avec la classe Joueur pour l'Attribut enPossession : bool de l'objet joueur. Nous avons aussi rajouté la classe Obstacle, qui représente un obstacle sur la surface de jeu. Celle-ci est en lien avec la classe SurfaceJeu qui contient comme attribut une liste d'Obstacles comme mentionné ci-précédemment. La classe Zoom a pour but grandir ou de réduire l'espace de jeu dans l'interface. Celle-ci prend en paramètre deux Doubles représentant les grandeurs X et Y de l'espace de jeu. Les opérations dans cette classe ont pour but d'effectuer les modifications sur la grandeur de l'espace de jeu selon les demandes de l'utilisateur. La classe coordonnée a pour but de déterminer l'emplacement d'un objet dans l'espace de jeu. Celle-ci prend en paramètre la hauteur et la largeur de l'espace de jeu en Int. De plus, elle prend la position X et Y sur l'espace de jeu de l'objet en question. Un objet peut être la souris, un objectif ou un cône. Les opérations de la classe permettent d'aller chercher les coordonnées X et Y et de faire la conversion en mètre. Finalement, parlons de la classe ImageParImage et de la classe TempsReel. Leur structure est strictement semblable, mais, dépendamment de l'objet qui est stocké dans le contrôleur, soit l'objet MondeCréationStratégieSelectionné, les méthodes AvancerJeu, PauserJeu, DébuterJeu, ReculerJeu ne feront pas exactement les mêmes tâches, de même pour la méthode Serialize. En mode image par image, la sauvegarde du contrôleur ne se fait qu'a à création d'une image par l'utilisateur, tandis qu'en mode temps réel, les mouvements de tous les joueurs sont sauvegardés à toutes les dixième seconde (ce que nous avons choisi).
Parlons maintenant du contrôleur de Larman. L'objet contrôler comprend bien sûr une surface de jeu, qui elle contient un sport, une liste de joueur et une liste d'obstacles, un mode de création de stratégie sélectionné, un chemin de sauvegarde et une unité de conversion qui permet de convertir des pixels à d'autres unités de mesure, comme des mètres par exemple. Les agrégations entre Contrôleur et ImageParImage, TempsReel et SurfaceJeu sont donc nécessaires. Les méthodes du contrôleur sont donc toutes les méthodes de toutes les classes du domaine plus le constructeur du contrôleur sans

paramètre et avec paramètre. Comme mentionné ci-précédemment, le contrôleur possède deux fonctions différentes pour Avancer, Reculer, Débuter et Pauser un jeu, puisque celle-ci sont définies différemment en fonction de l'objet MondeCréationStratégieSelectionné.
En ce qui concerne la sérialisation du contrôleur, nous avons une classe Enregistrement, qui est une classe du domaine très près du contrôleur. Celle-ci s'occupe de sérialiser le contrôleur. Son seul attribut est une liste d'enregistrement, soit une liste de tuples/paire de (fichier de sauvegarde, Date). Les méthodes avancer, reculer, débuter et pauser vont donc communiquer à plusieurs reprises avec la classe enregistrement. Notre relation entre enregistrement et contrôleur est donc navigable dans les deux sens.

Diagramme de packages



En ce qui concerne les packages, nous avons séparé ceux-ci en quatre packages principaux différents. Ces packages son GUI, Controller, Domain et Drawing.

Pour la représentation de nos packages, nous avons décidé de prendre une architecture en couche. Ce ci est dans le but de simplifié la compréhension et facilité le remplacement de certaines couches par de par de nouvelles implémentations en cas de changement.

Pour commencer, le package GUI contient l'entièreté de nos différentes interfaces. Nous retrouvons dans ce packages les différents fichiers fxml des interfaces et leur contrôleur respectif. Le package GUI se lie au Controller.

Le package Controller est composé d'un seul fichier java nommé Controller communément qui est notre contrôleur de Larman. Celui-ci fait le lien entre le GUI, le Domain et le Drawing. Le package contrôleur se lie au package Domain et Drawing.

Dans le packages Domain, nous avons décidé d'utiliser différent sous packages pour séparer nos classes. Ceci est dans le but de faciliter la compréhension de notre code et d'être en mesure de réutiliser des classes. Par exemple, nous utilisons les coordonnées à plusieurs reprises. Nous avons donc séparé la classe de coordonnée dans un sous packages dans le but d'être en mesure de le réutiliser pour les différentes utilisations.

Le package Domain contient les classes SurfaceJeu, ImageParImage, TesmpsReel, Cone sport et Enregistrement. Ces classes non pas besoin d'être disposé en sous packages dus au fait que nous n'avons pas à les réutilisé autre que les fonctions décrit dans les classes.

Le package objectif contient les classes objectives, Balle, Ballon et Rondelle. Ceux-ci représentent tous des objectifs que l'on retrouve dans une interface.

Le package Coordonee contient la classe coordonnée. Celui-ci se retrouve dans le package Domain. Il se peut que plus tard dans le développement du projet nous allions avoir à développer d'autre classe par rapport aux coordonnées. C'est pourquoi que nous avons créé un packages de ce type.

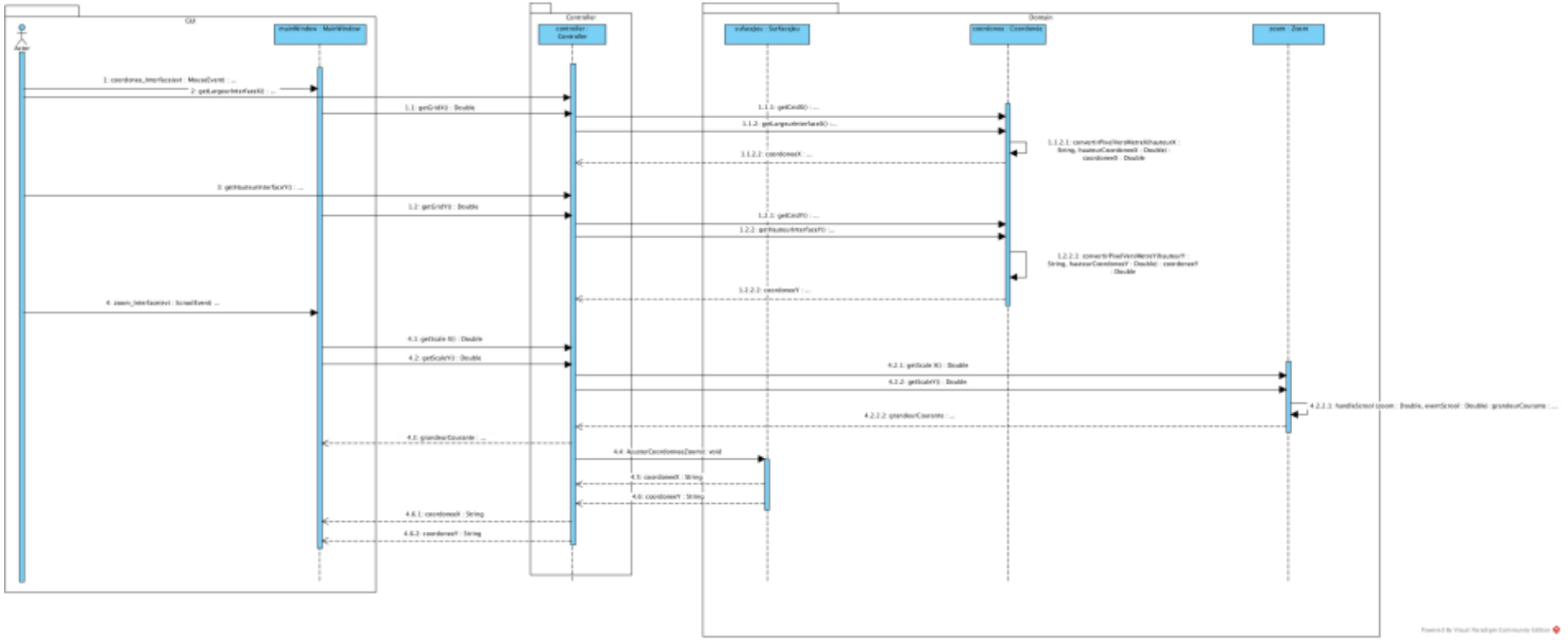
Le package Joueur contient la classe joueuse. Celui-ci se retrouve dans le package Domain. Ultérieurement, nous allons être en mesure d'ajouter des joueurs de différents types. D'autres classe vont être ajoutées.

Le package Zoom contient une classe intitulée zoom. Celui-ci se retrouve dans le package Domain. Lorsqu'un joueur va vouloir augmenter ou réduire sa vision de l'interface, c'est dans ce packages que l'on retrouvera le code lié au Domain.

Pour terminer, le package Drawing est lié au Controller. Nous retrouvons dans ce packages le code permettant l'utilisateur de dessiner, d'ajouter des joueurs ou des obstacles sur l'interface.

Diagrammes de séquence de conception

Conversion des coordonnées

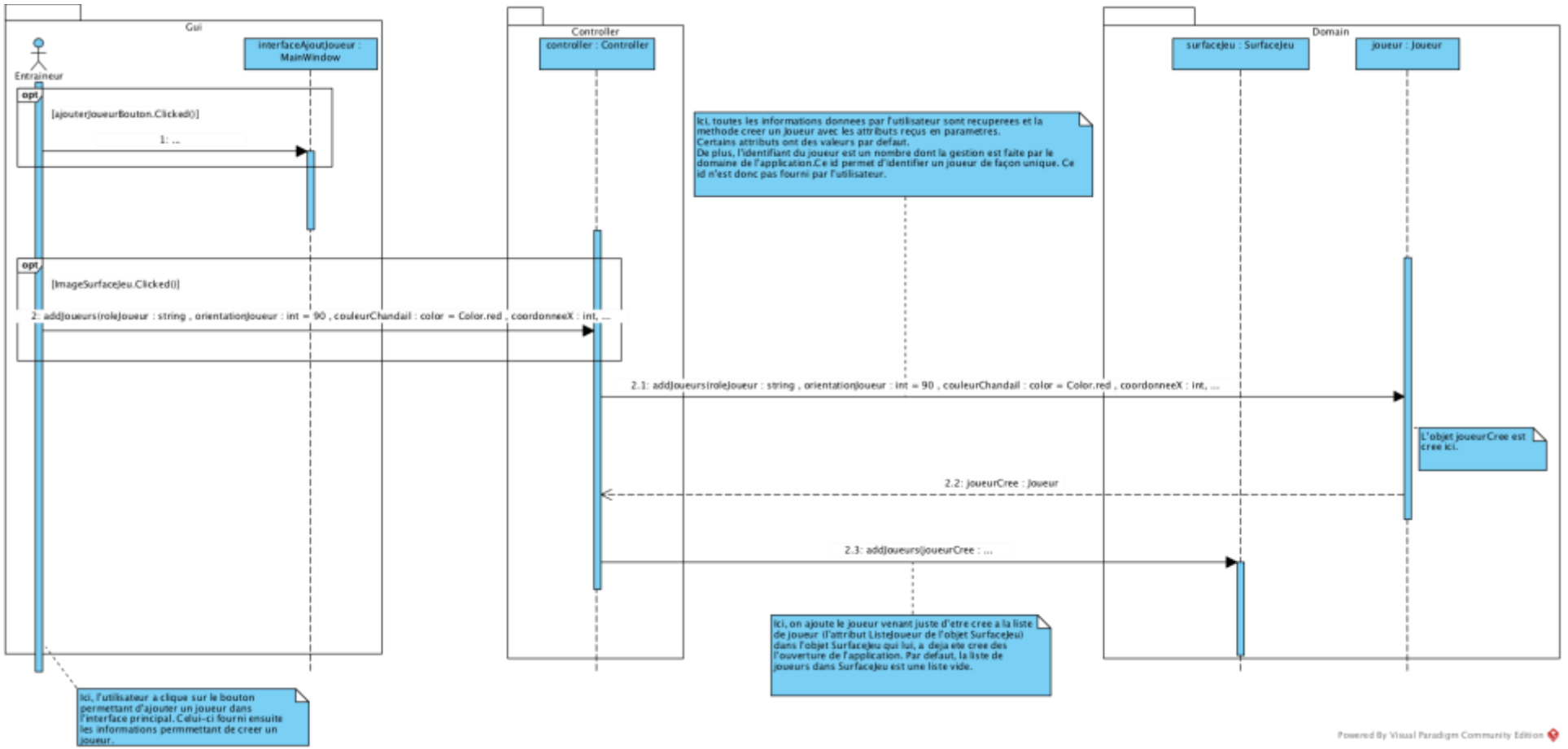


Pour commencer, cette opération se d'éclanche lorsque l'utilisateur effectue action, soit l'utilisateur bouge la souris sur l'interface et celui-ci agrandit ou rapetisse la grosseur de la surface de jeu avec la souris. Lorsque l'utilisateur effectue ces actions, deux opérations se déclenchent, soit coordonee_interface et zoom_interface. Pour aller chercher les coordonnées en X et en Y de la souris lorsque celle-ci se déplace sur l'image, nous faisons appelles aux opérations getGridX et getGridY. De plus, nous devons être en mesure de convertir les données de la position de la souris étant en pixel vers une unité de mesure en mètre. C'est pourquoi que l'on doit appeler deux opérations ayant pour but d'aller chercher des données de hauteur et largeur de la surface de jeu entré par l'utilisateur. Ces opérations se nomment getHauteurInterfaceY et getLargeurInterfaceX. C'est par le contrôleur que nous faisons la transition entre MainWindow et la classe coordonnée se retrouvant dans le domaine. Dans cette même classe, nous avons deux opérations nommées convertirPixelVersMetreX et convertirPixelVersMetreY. Ceux-ci sont en charge d'effectuer la transformation des coordonnées en mètre. De plus, chacune retourne leur coordonnée vers le contrôleur.

Pour la deuxième partie qui le zoom nous allons chercher la grandeur de l'interface en X et en Y. Pour ce faire, nous avons les opérations getScaleX et getScaleY. Encore une fois, le contrôleur est responsable de faire de pond entre l'interface et les opération dans le domaine relié. Dans la classe Zoom, c'est avec les grandeur X et Y de l'interface que l'opération handleScrool est en mesure de redimensionné la surface de jeu selon les demandes de l'utilisateur. Cette opération retourne la grandeur demandée par l'utilisateur vers l'interface.

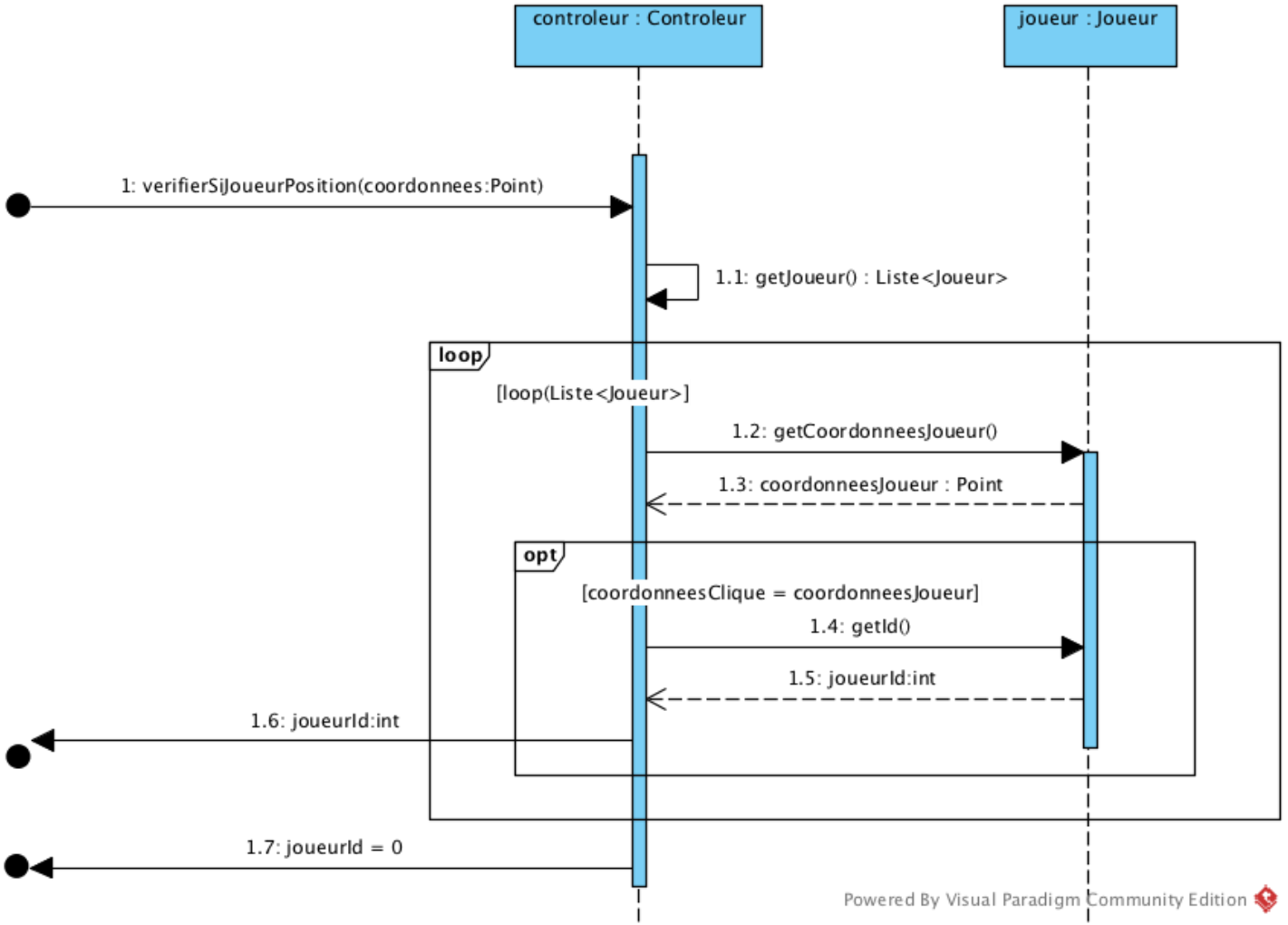
Pour terminer, il faut ajuster les coordonnées X et Y selon le zoom de l'écran. Pour cela, une opération AjusterCoordoneeZoom dans la classe SurfaceJeu permet l'ajustement. Cette opération retourne par la suite les coordonnées en X et en Y vers l'interface.

Ajout d'un joueur à l'aide de la souris



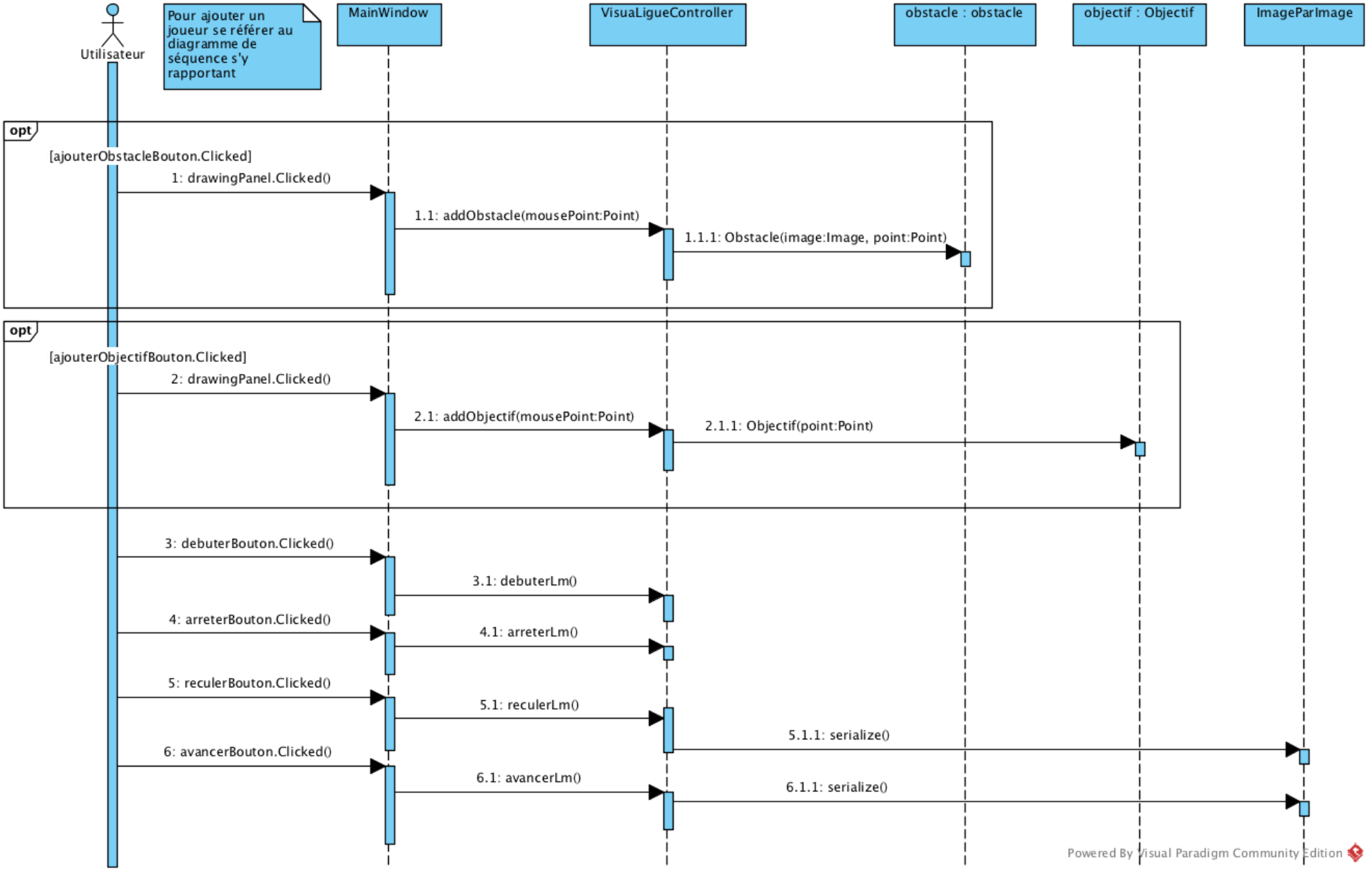
Pour ajouter un joueur au jeu, l'utilisateur clique sur le bouton Ajouter Joueur dans l'interface contenant la surface de jeu. À ce moment, l'interface Ajouter Joueur apparaît et on demande à l'utilisateur d'entrer les informations permettant la création d'un objet joueur : son rôle et sa couleur (équipe). Les paramètres sont maintenant enregistré, et l'utilisateur peut cliquer dans l'espace de la surface pour ajouter un joueur. La position en X en Y est récupérée des coordonnées en X et en Y du clique de l'utilisateur. À l'instant du clique, un create est appelé à la classe Joueur et le joueur créé est ensuite ajouté dans la liste de joueurs, l'attribut de l'objet de la classe SurfaceJeu.

Déterminer si un joueur a été cliqué



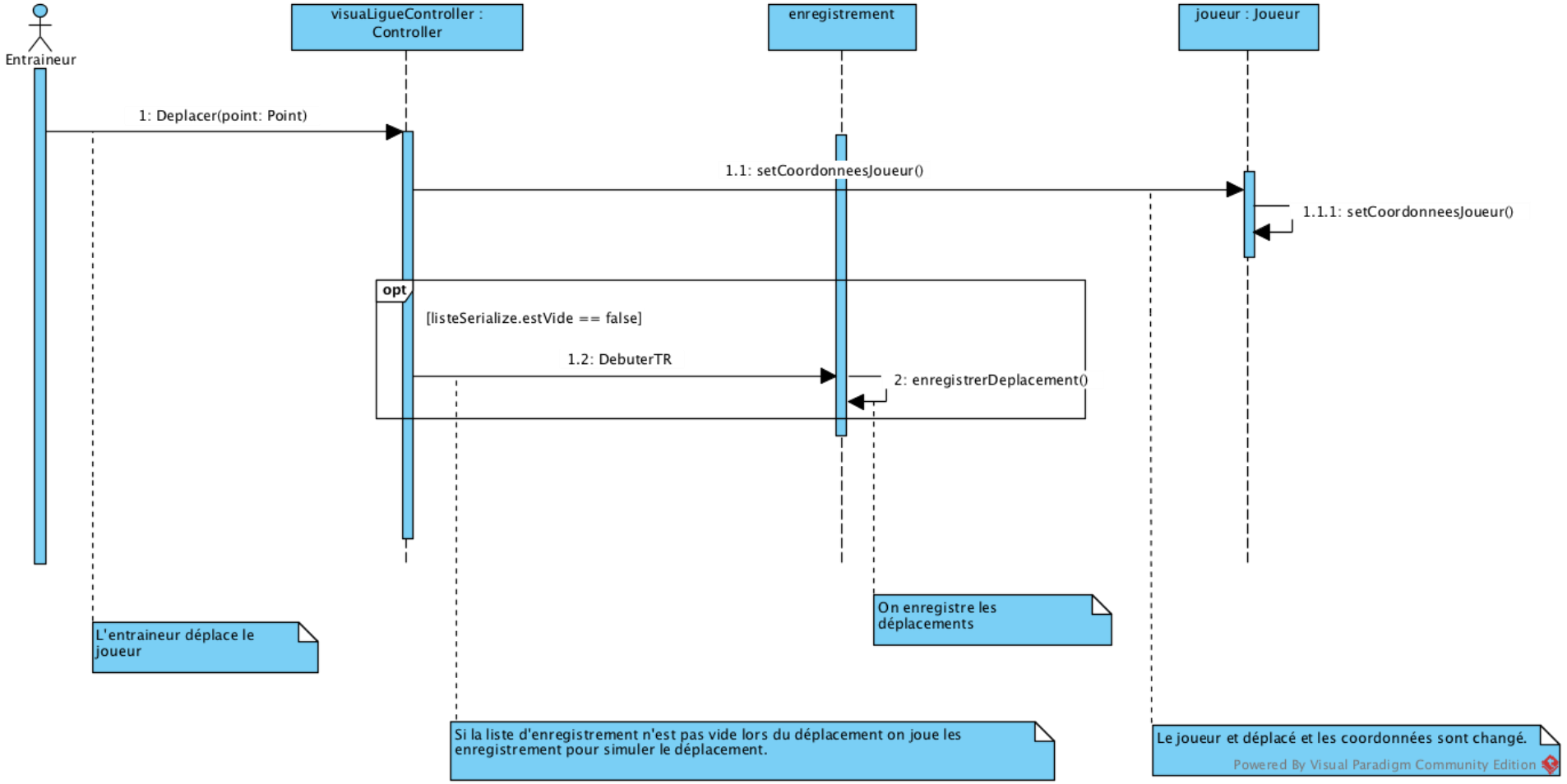
Lors d'un clic sur la surface de jeu, le contrôleur reçoit un appel à sa fonction verifierSiJoueurPosition avec les coordonnées du clic en paramètre. Ensuite le contrôleur appelle sa propre méthode getJoueur qui permet de récupérer la liste des joueurs sur la surface de jeu. Pour chacun des joueurs faisant partie de la liste, on récupère ses coordonnées. Par la suite, on compare ses coordonnées avec celles du clic et si elles sont les mêmes, on appelle la méthode getId de la classe Joueur et le contrôleur retourne l'id du joueur. Si, après avoir parcourue toute la boucle, aucun id n'a été renvoyé, on retourne le chiffre « 0 ».

Interaction en mode image par image



Dans le mode image par image l'utilisateur peut accomplir plusieurs actions. Premièrement, il peut ajouter soit, un joueur, un obstacle ou un objectif (rondelle, ballon, etc.). Pour cela, il doit premièrement appuyer sur un des trois boutons permettant un de ces ajouts et ensuite cliquer à l'endroit sur le terrain (drawingPanel) où il veut que cet élément soit ajouté. Un appel à la méthode addJoueur, addObstacle ou addObjectif du constructeur sera faite. Ce dernier fera ensuite appel au constructeur respectif à joueur, obstacle et objectif pour créer l'élément voulu.
Aussi, pour démarrer et arrêter la séquence, le joueur doit cliquer sur le bouton play ou pause. Par la suite, le MainWindow appellera soit la méthode debuterLm() pour le play ou arreterLm() pour le pause.
Pour reculer ou avancer la séquence, l'utilisateur doit cliquer sur le bouton reculer ou le bouton avancer. Le MainWindow appellera par la suite la méthode reculerLm() ou avanerLm() et le contrôleur appellera serialize() de la classe ImageParImage.

Interaction en mode temps réel

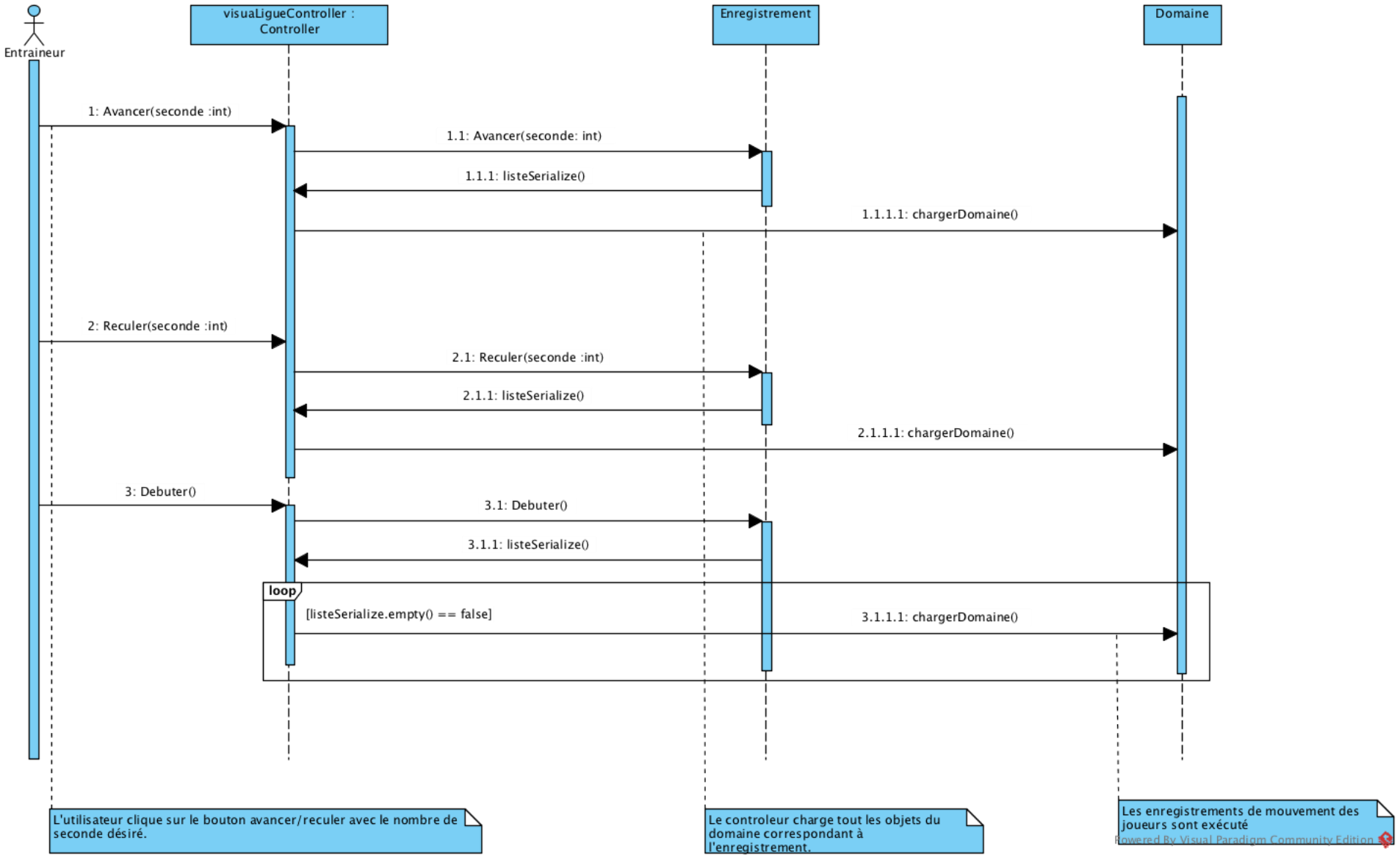


L'utilisateur déplace un joueur sur la surface de jeux et ceci provoque un enregistrement et un mouvement interactif des autres joueurs.

Lorsque le joueur a terminé son déplacement, le contrôleur change les coordonnées du joueur dans les priorités de l'objet.

De plus, lorsqu'on déplace un joueur, une opération est appelée au niveau du contrôleur et si la liste des enregistrements n'est pas vide, les déplacements seront exécutés pour rendre l'application interactive.

Visionnement d'un jeu



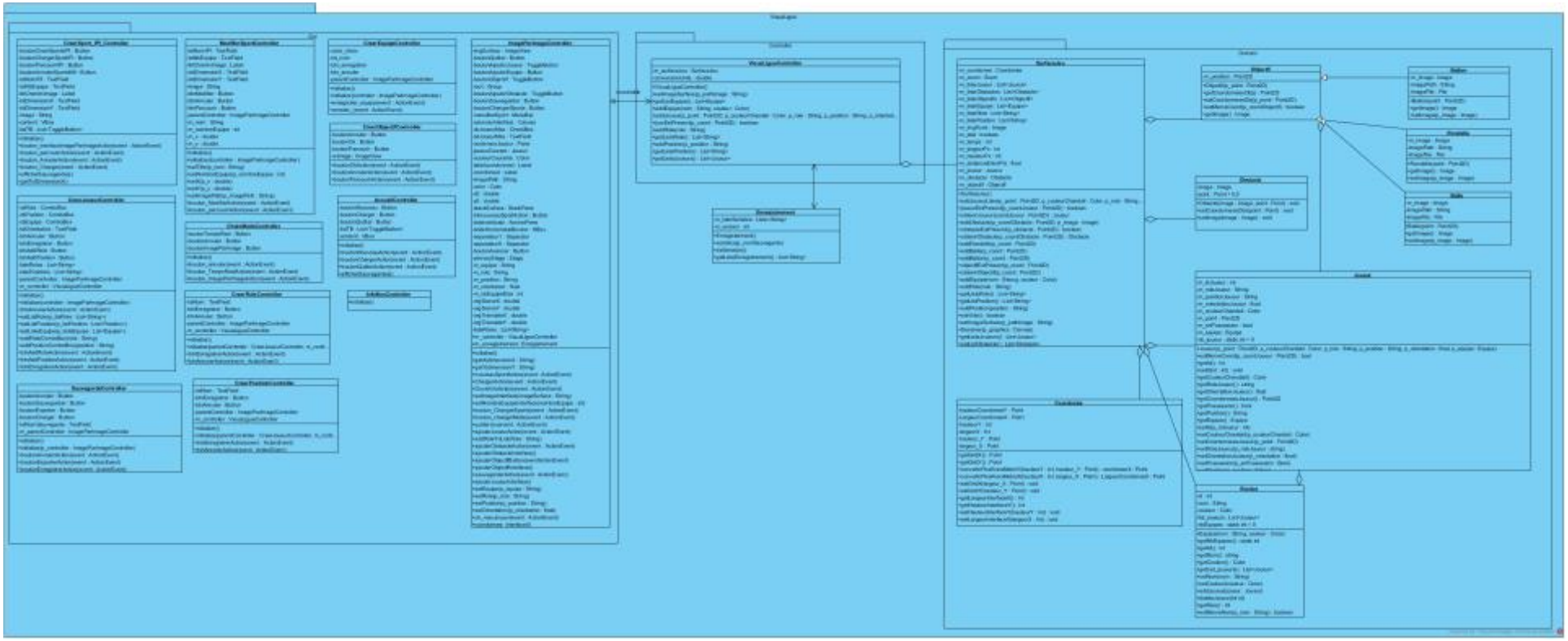
L'utilisateur clique sur le bouton avancer, reculer et/ou débuter le jeu. Le clic de l'utilisateur est la seule interaction nécessaire avec l'interface.

Lorsqu'on clique sur avancer, une opération est appelée au niveau du contrôleur qui lance une opération au niveau de notre classe enregistrement. La classe enregistrement est à l'extérieur du domaine et permet de stocker nos serialize dans une liste. La classe enregistrement retourne l'enregistrement (un lien vers un fichier serialize) vers le contrôleur. Lorsque le contrôleur reçoit l'enregistrement, il recharge le domaine.

La fonction pour reculer fonctionne de la même façon que celle pour avancer.

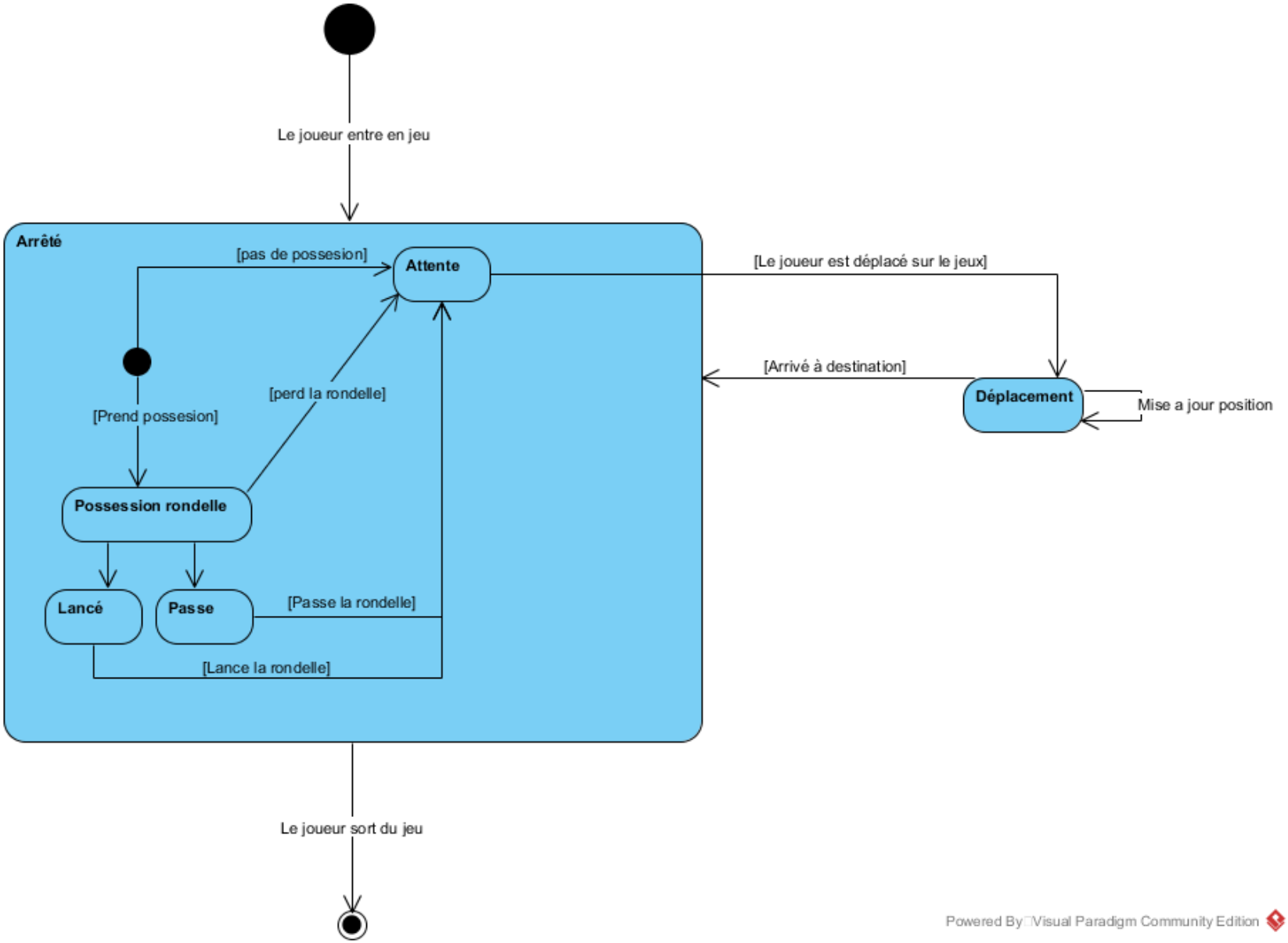
La fonction pour débuter et jouer un enregistrement est légèrement différente. En effet, le clic utilisateur lance la fonction « debuter » qui consulte la classe enregistrement et qui retourne l'ensemble des enregistrements dans une liste. Lorsque le contrôleur a reçu la liste, il parcourt et charge le domaine en boucle.

Diagramme de classe de conception mis-à-jour



Diagrammes d’états

Pour un joueur



Pour la rondelle

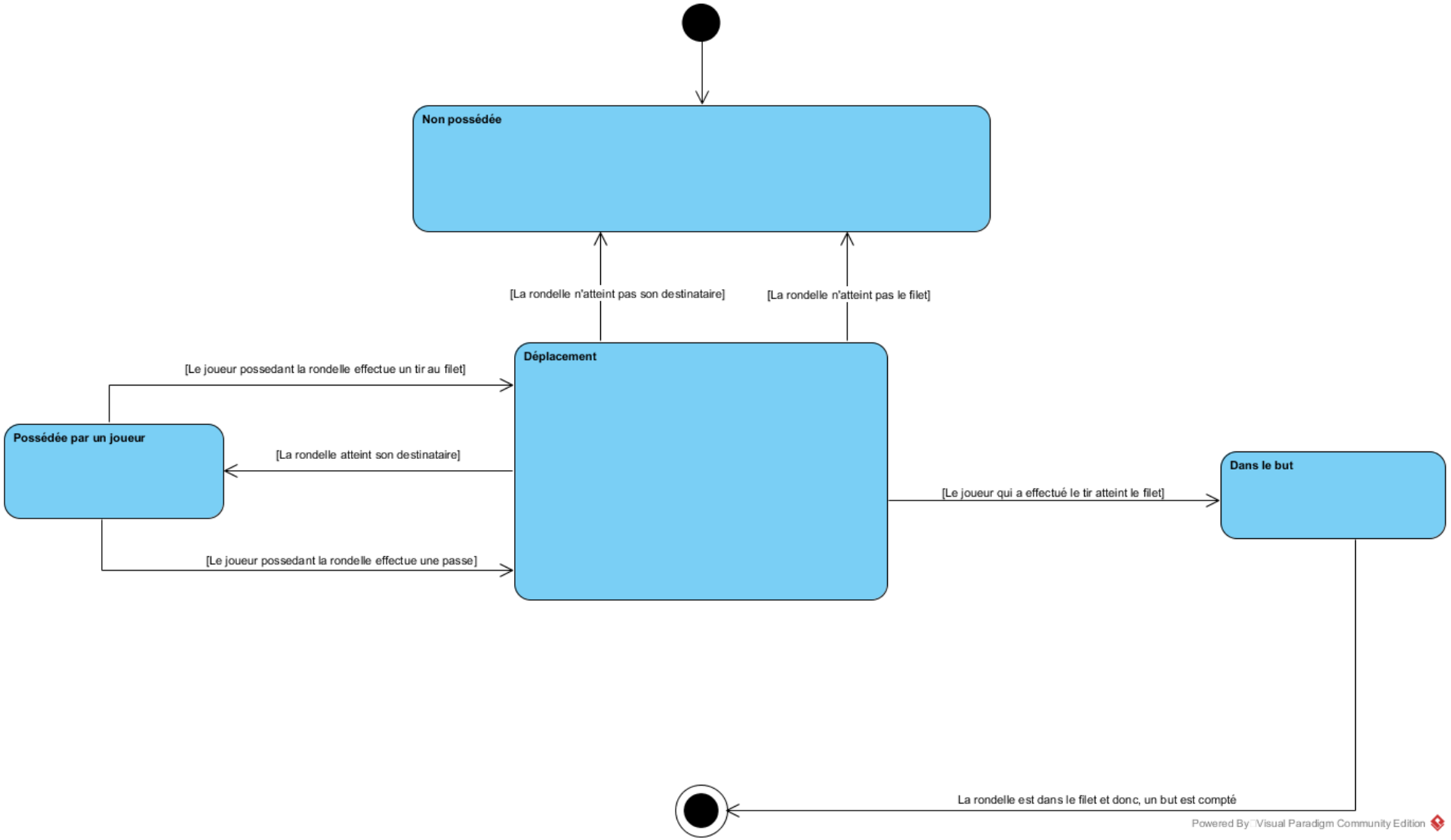


Diagramme d’activité

Créer une stratégie en mode image par image

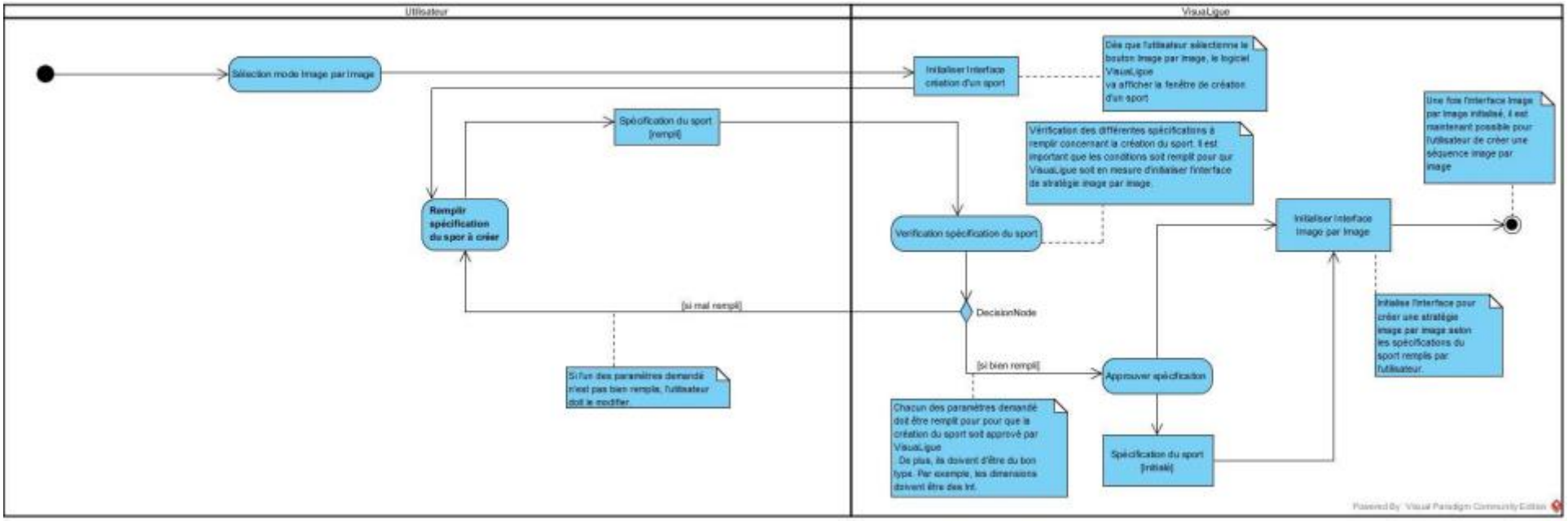
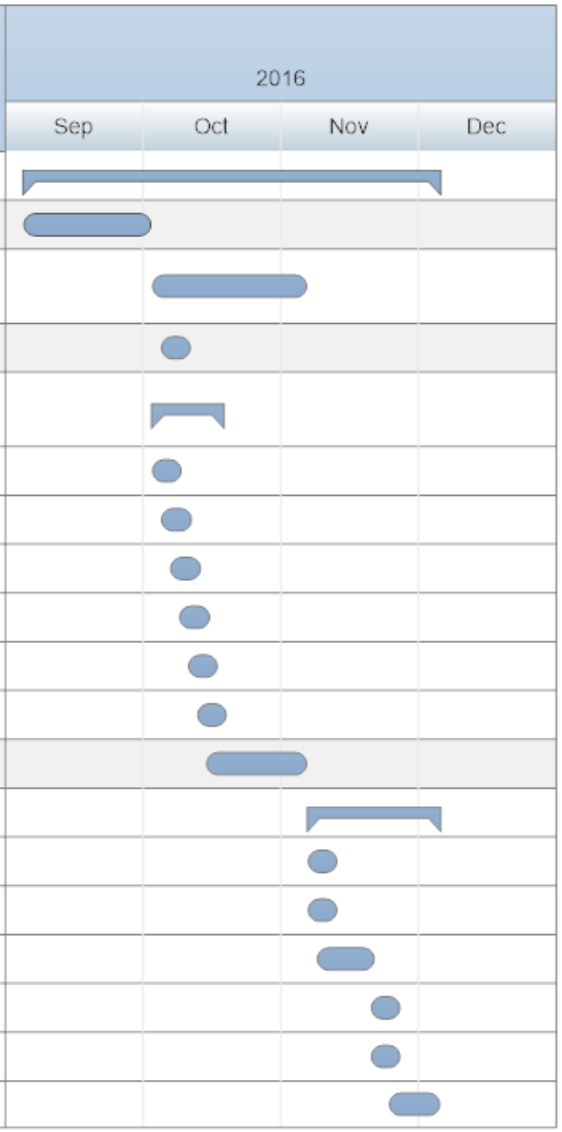


Diagramme de Gantt

	Task	Assigned To	Start	End	Dur
	VisuaLigue		9/5/16	12/5/16	76
1	Livrable 1		9/5/16	10/2/16	23
2	Diagramme de classe de conception	Antoine St-Laurent	10/3/16	11/6/16	29
3	Diagramme de packages	Louis-Philippe	10/5/16	10/8/16	4
4	Diagrammes de séquence de conception		10/3/16	10/18/16	13
4.1	Conversion coordonnées	Louis-Philippe	10/3/16	10/7/16	5
4.2	Ajout d'un joueur	Antoine St-Laurent	10/5/16	10/11/16	5
4.3	Position du clic	Antoine Hudon	10/7/16	10/13/16	5
4.4	Mode image par image	Antoine Hudon	10/9/16	10/15/16	5
4.5	Mode temps réel	William	10/11/16	10/15/16	5
4.6	Visionner les jeux	William	10/13/16	10/18/16	5
5	Programmation de l'interface	Tous	10/15/16	11/6/16	19
6	Programmation	À déterminer	11/7/16	12/5/16	24
6.1	Créer type de sport		11/7/16	11/8/16	2
6.2	Créer les stratégies		11/7/16	11/12/16	5
6.3	Créer un jeu image par image		11/9/16	11/21/16	10
6.4	Affichage du jeu		11/21/16	11/22/16	2
6.5	Visualiser le jeu		11/21/16	11/25/16	5
6.6	Autres fonctionnalités		11/25/16	12/5/16	9



À noter : Pour la partie « Programmation », chaque étape de réalisation comprend les tests et le déploiement de la fonctionnalité en question.

Glossaire

Application :
En informatique, programme ou ensemble de programmes destinés à aider l'utilisateur d'un ordinateur pour le traitement d'une tâche précise. SYNONYME : Logiciel (Source : Larousse.fr)

Avantage numérique :
Situation d'une équipe qui compte plus de joueurs sur la patinoire que l'équipe adverse, dont un ou plusieurs joueurs font l'objet d'une pénalité. (Source : oqlf.gouv.qc.ca)

Bouton :
En informatique, un bouton est un composant d'interface graphique opéré par pointer-et-cliquer. Un bouton porte un libellé qui décrit l'opération attachée au bouton. (Source : wikipedia.org)

Environnement de développement intégré (EDI) :
Un ensemble d'outils pour augmenter la productivité des programmeurs qui développent des logiciels. (Source : wikipedia.org)

Exporter :
Action de transférer des données depuis un programme vers un autre programme ou d'enregistrer des données dans un autre format de fichier. (Source : dicofr.com)

Fenêtre :
Une fenêtre est une zone rectangulaire qui apparaît à l'écran pour afficher le contenu d'un dossier par exemple ou même un logiciel. La fenêtre peut prendre tout l'espace (plein-écran) ou seulement une partie. Toutes les interfaces graphiques des systèmes utilisent des fenêtres. (Source : cours-informatique-gratuit.fr)

Interface graphique :
L'interface graphique désigne la manière dont est présenté un logiciel à l'écran pour l'utilisateur. C'est le positionnement des éléments : menus, boutons, fonctionnalités dans la fenêtre. (Source : cours-informatique-gratuit.fr)

Java :
Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour. (Source : java.com)

JPEG, JPG :
Acronyme de Joint Photographic Experts Group) est une norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe. (Source : wikipedia.org)

Menu déroulant :
En informatique, une liste déroulante est un élément d'interface graphique qui permet à l'utilisateur de sélectionner une ou plusieurs options. Il en existe deux types : celles qui affichent et permettent de choisir une seule option, et celles qui permettent de sélectionner plusieurs options et qui en montrent au moins deux. Cet élément est couramment utilisé par des sites Web et des logiciels. (Source : wikipedia.org)

Micro-ordinateur :
Petit ordinateur construit autour d'un microprocesseur auquel on adjoint l'environnement logiciel et matériel (écran, clavier, lecteurs de disquettes), nécessaire au traitement complet de l'information. (Source : larousse.fr)

Moniteur :
Écran de visualisation associé spécifiquement à un micro-ordinateur. (Source : larousse.fr)

Netbeans :
Environnement de développement intégré.

PNG :
Le Portable Network Graphics est un format ouvert d'images numériques.

Undo/redo :
Annuler ou rétablir une action fait par l'utilisateur dans le programme.