Projet Tutoré - Etude préalable

Conception et développement d'une simulation de Capture the Flag et mise en place d'agent autonome

HIRTZ - JEAN-BAPTISTE - JUNG - LE NALINEC - NAIGEON
RA-IL2

Tuteur : Amine BOUMAZA

Table des matières

Présentation du projet

Notre projet tutoré a pour ambition de concevoir et développer une simulation graphique d'un jeu de CTF (Capture The Flag). Dans un monde en 2D dimension, des agents se partagent la zone de jeu, et s'affrontent dans le but de ramener le drapeau de l'équipe adverse au sein de son territoire.
Notre but à terme est d'utiliser cette simulation pour trouver via des approches d'intelligence artificielle une politique optimale d'action dans le but d'avoir des agents autonomes les plus performants possible.

Vision du produit

1 Qui va acheter/utiliser le produit ? Qui est la cible ?

Le projet est destiné aux membres du groupe ainsi qu'aux personnes externes avec lesquelles nous voudrions partager la simulation (recruteurs, visiteurs de porte ouverte…) ainsi que les personnes trouvant notre github publique.

2. À quels besoins le produit va-t-il répondre ?

Le projet à pour but de nous initier aux techniques d'intelligence artificielle et de nous permettre d'apprendre à formaliser des problèmes pour y appliquer ces techniques. Nous allons notamment apprendre à utiliser des réseaux de neurones pour réaliser le produit final. Un autre besoin auquel le projet devra répondre est celui de pouvoir montrer les IA que nous aurons créé, cela implique donc d'avoir un moyen de visualiser les agents en train de jouer.

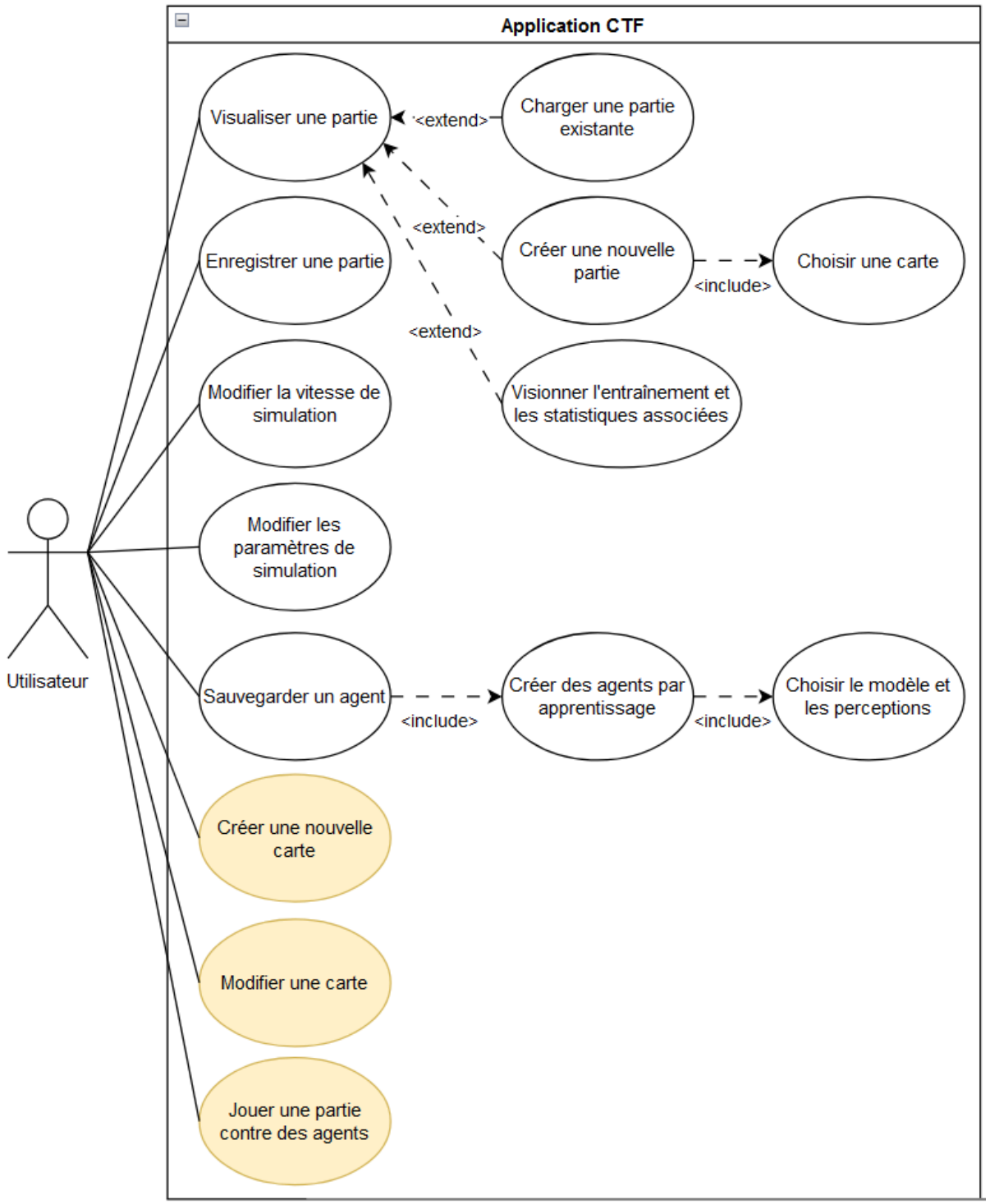
3. Quelles sont les fonctionnalités critiques pour répondre aux besoins de façon à avoir un produit réussi ?

- Le moteur de simulation
- L'interface de choix du modèle d'ia
- Le système d'évaluation des modèles

4. Comment le produit se situe-t-il par rapport aux produits existants sur le marché?

Il n'existe pas réellement d'équivalent à notre projet sur le marché, bien que DeepMind aie déjà mis en place des simulations similaires, elles ne sont cependant pas diffusées librement.

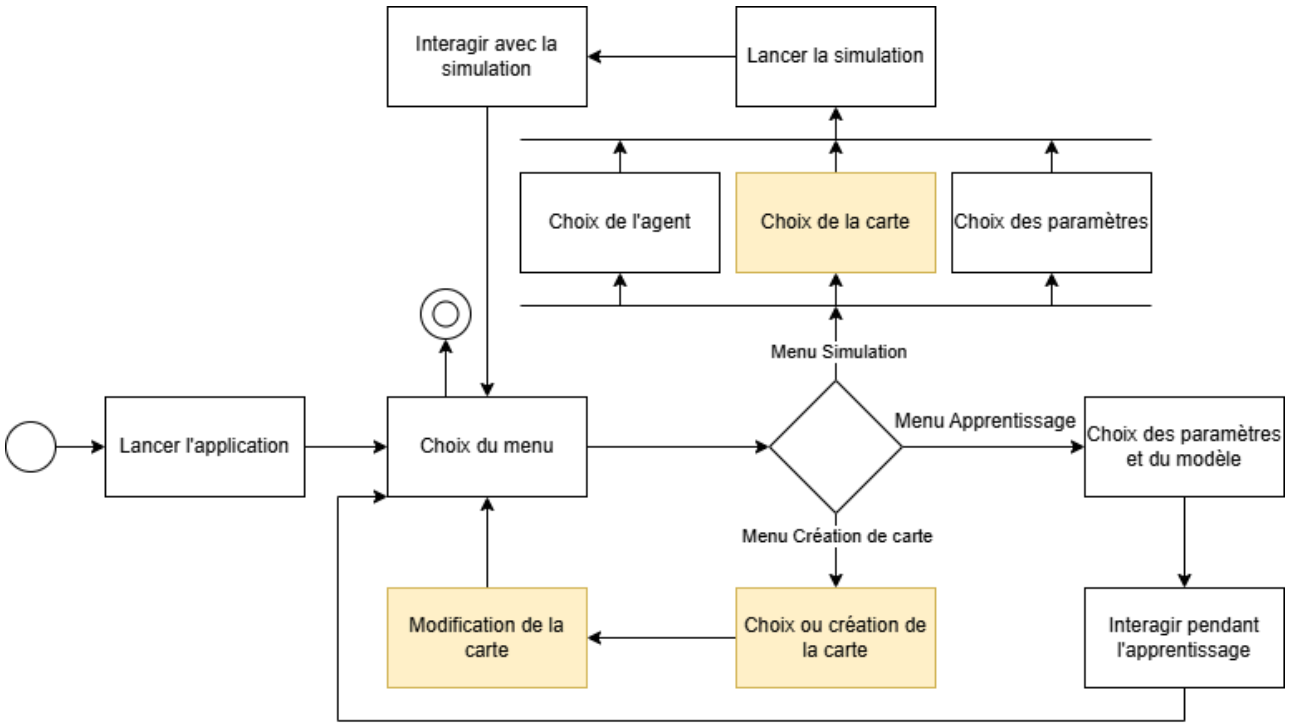
Cas d’utilisations



Les fonctionnalités en jaunes sont les cas d’utilisation bonus qui pourront être ajoutés plus tard.

Diagramme d'activité

Nous avons pour objectif de centraliser toutes nos fonctionnalités dans une même application. Les différentes parties du projet pouvant fonctionner en autonomie, elles seront d'abord développées de manière séparée avant d'être assemblées dans l'application globale.



Les fonctionnalités en jaunes sont les cas d'utilisation bonus qui pourront être ajoutés plus tard.

Scénario

Voici un scénario d'une utilisation lambda de l'application :

L'utilisateur lance l'application et ouvre le menu d'apprentissage d'agent.
L'utilisateur choisit les paramètres et entraîne l'IA. Il accélère la simulation pour réduire le temps d'apprentissage. L'apprentissage se termine.
L'utilisateur choisit le menu simulation et les paramètres de la simulation (Agent, carte et autres paramètres). Il observe la simulation et interagit avec pour comprendre les choix que prennent les agents lors du jeu.
Il recueille les résultats et conclut sur ses observations.

Diagramme de séquence système

Ce diagramme de séquence représente les étapes mises en place pour lancer un apprentissage d'agents : L'application lance un apprentissage pour X itérations, cet apprentissage va lui-même lancé des parties et recueillir leur résultats, pour ensuite mettre à jour le/les agent.s

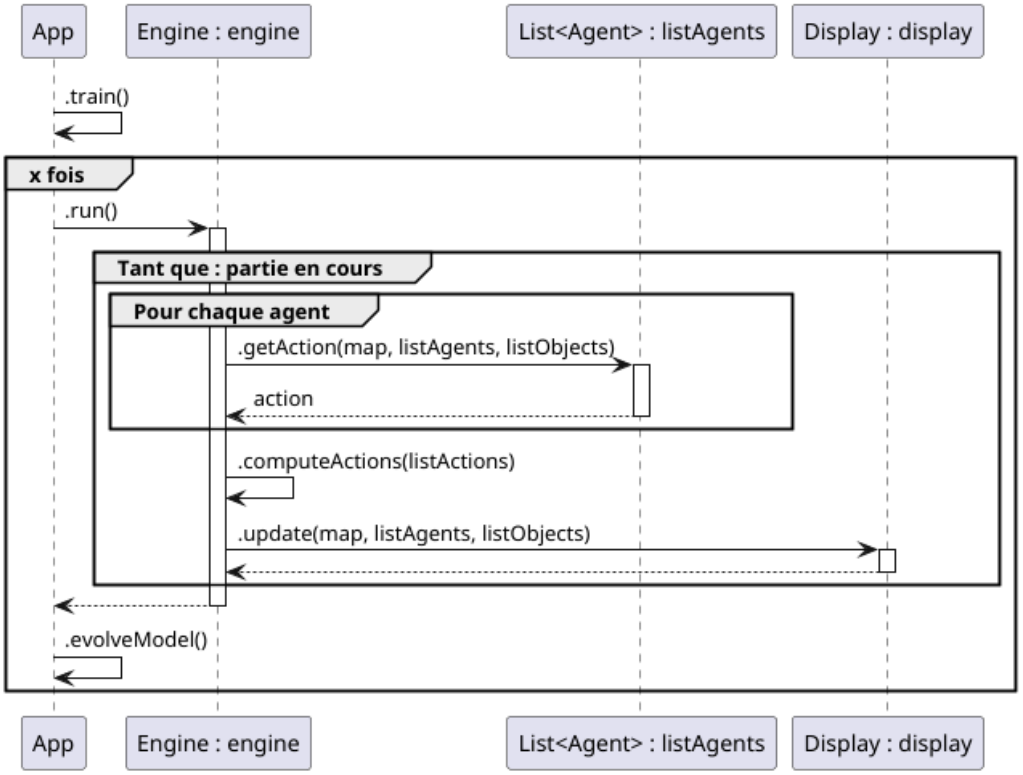
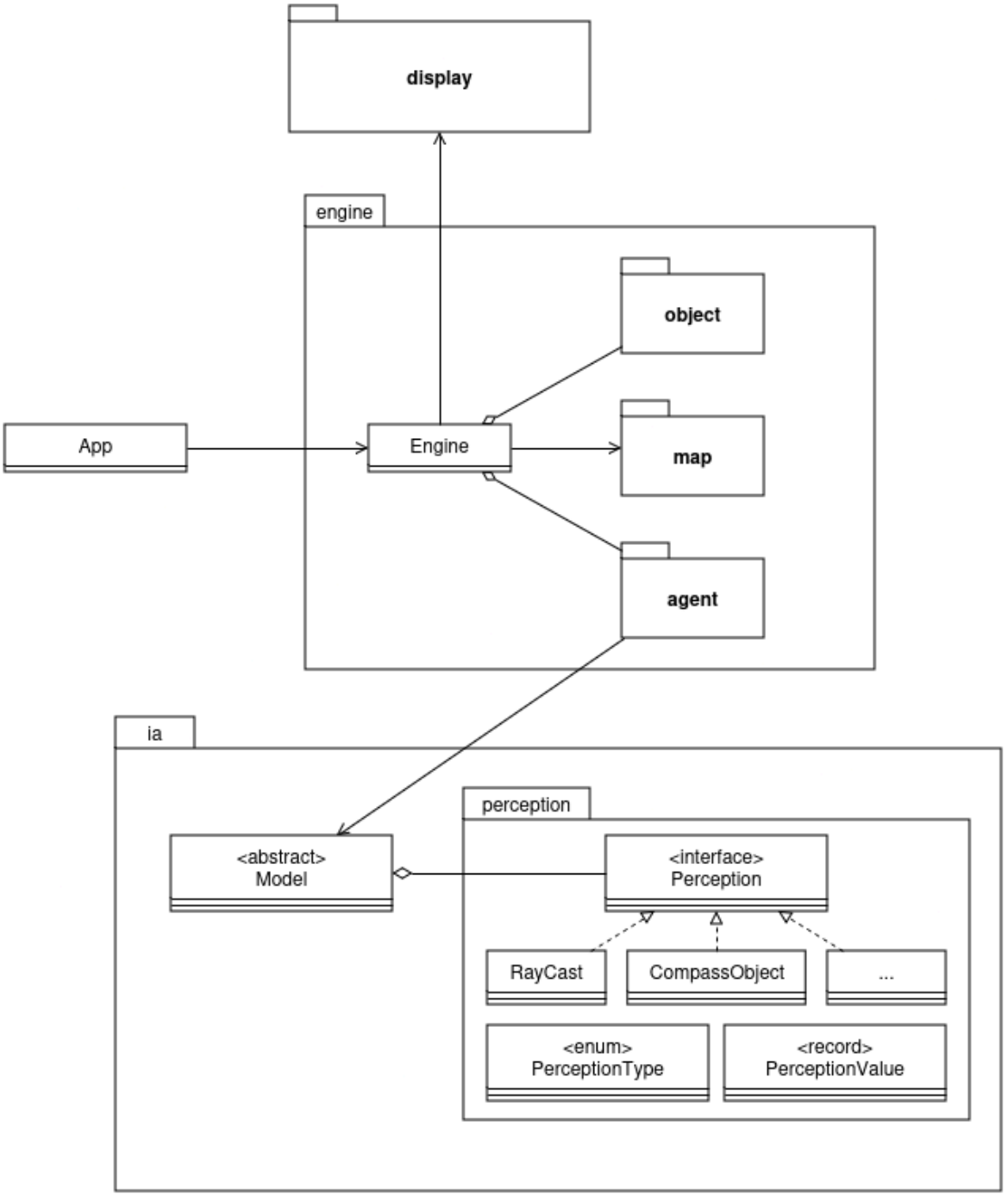


Diagramme de classes



Condition de validation

Le projet sera considéré comme validé s'il permet de visualiser et sauvegarder une partie avec des agents qui auront suivi un apprentissage (Génétique ou renforcement).

Recensement et évaluation des risques

	Gravité	Fréquence	Conséquences	Action préventive	Action corrective
Ne pas réussir à faire le moteur de jeu	Majeur	Faible chance	Impossibilité de poursuivre le projet	Se former à GDX	Utilisation d'un moteur déjà fait
Ne pas réussir à faire apprendre à des agents à jouer	Forte	Assez Faible Chance	Simplification ou changement des algorithmes	Étude préalable des algorithmes	X
Être en retard par rapport au planning	Mineur	Probable	Réduction des attentes du projet	Découper et répartir le travail	Réduire les attentes de chaque itération ou augmenter le temps de travail
Conflit entre les différents membres du groupe	Forte	Faible chance	Prendre du retard	Traiter les désaccords rapidement	Se réunir et régler le conflit
Les graphismes ne sont pas adaptés	Moyenne	Possible	Ralentissement dans la réalisation du projet	Préparer et vérifier les graphismes	Utiliser des assets temporaires

Planning

Voici le planning provisoire de notre projet. Nous avons laissé les deux dernières itérations pour l'implémentation des cas d'utilisations bonus, car nous pensons avoir fini l'application avec les cas de base à l'issue des quatre premières itérations. Cependant, ce planning n'est pas fixe, il tend à évoluer avec le temps. Si nous sommes en avance et que nous avons terminé l'itération un avant terme par exemple, nous pourrons commencer des tâches des itérations suivantes.

Itération 1 :
- Fenêtre pour visualiser une partie
- Charger une carte à partir d'un fichier
- Agent aléatoire
- Lancer une partie
- Modifier la vitesse de simulation
- Avancer la simulation d'un pas

Itération 2 :
- Agent à arbre de décision
- Sélection du modèle d'agent
- Sélection de la carte
- Sélection des paramètres de simulation
- Sauvegarder/charger une partie

Itération 3 :
- Lancer un apprentissage génétique
- Paramétrer le réseau de neurone à apprendre
- Stocker un agent entraîné
- Affichage spécifique aux entraînements

Itération 4 :
- Lancer un apprentissage par renforcement
- Choix du type d'apprentissage et des paramètres
- Éditeur de carte

Itération 5 :
- Agents avec mémoire
- Agents avec communication

Itération 6:
- Contrôler un agent
- Plusieurs drapeaux par équipe
- Bonus à ramasser par les agents

Essais GDX et py4j

Nous avons testé la librairie GDX. Un affichage basique d'une carte de CTF sous GDX a été réalisé pour tester la boucle GDX. Ce test nous a permis de voir que GDX fonctionne de manière assez fermée et très peu contrôlable ce qui pourrait nous poser des problèmes de complexité et rendre flou le fonctionnement du projet plus tard. Nous n'avons donc pas utilisé GDX mais JavaFX qui a été déjà vu et fortement utilisé par la majorité des membres du groupe ces deux dernières années.

Nous avons testé la librairie Py4j. Son utilisation est simple, il pourrait tout de même avoir des problèmes de sérialisation et de conversion de type java en type python.
Nous avons testé les performances entre du 100% java et du Java/Python avec Py4j sur la création de tableau de nombre flottant et l'appel de méthode en passant en paramètre le tableau ou en renvoyant le tableau.

Tests :
Exécution 100% java = 12 millisecondes.
Exécution java/python = 7 689 millisecondes.

La vitesse d'exécution est environ 600 fois plus lente avec du Java/Python que du 100% java sur cet exemple précis.
Nous nous orientons donc plutôt vers une application 100% java par souci de performances et également, car nous pensons ne pas avoir besoin des bibliothèques python pour créer nos modèles d'IA. Nous allons les créer en java.

Planification Tâche

Découpage des tâches

1-Apprentissage (Effectuer un apprentissage) :
- 1-Visualisation
  - 1-Gestion de la vitesse d'apprentissage
    - 1-Vitesse Maximale (pas d'affichage)
    - 2-Vitesse Saccadé (Saut d'images de rendu / itération)
    - 3-Vitesse Minimale (affichage de toutes les frames)
  - 2-Performances
    - 1-Rapport de performance en fin d'apprentissage
    - 2-Rapport de performance en fin de chaque itération/génération
- 2-Paramètres
  - 1-Choix d'un modèle
    - 1-Choix du type d'apprentissage
      - 1-Modèle Renforcement
      - 2-Modèle Algorithme génétique
    - 2-Gestion des paramètres
      - 1-Raycast
      - 2-Mémoire
      - 3-Communication
  - 2-Sauvegarde Modèle
    - 1-Exporter un modèle dans un stockage de modèle
    - 2-Stocker les modèles par type et garder les informations de paramètres
  - 3-Moteur d'apprentissage
    - 1-Boucle qui effectue un nombre de partie avec un agent donné
    - 2-Attribution des points à la fin d'une partie

2-Simulation
- 1-Paramètres de simulation
  - 1-Choix du modèle d'un agent
    - 1-Existence d'un modèle par défaut
    - 2-Sélecteur de modèle
  - 2-Choix de la carte
    - 1-Existence d'une carte par défaut
    - 2-Sélecteur de carte
  - 3- Paramètres
    - 1-Nombre d'agents
    - 2-Règles du jeu
    - 3-Nombre d'images par seconde
- 2-Visualiser la simulation
  - 1-Vitesse classique
  - 2-Vitesse rapide
  - 3-Vitesse Pas à Pas (Action par Action)
  - 4-Retour en arrière

1- Sauvegarde des étapes d’une simulation

3-Joueur Humain

1-Contrôle (ZQSD / Flèches directionnelles)

3-Éditeur de cartes

1-Création d’un nouvelle carte et édition existantes

2-Edition/Création de cartes

1-Visualiser une carte

2-Éditer une carte (Un clic permet de faire avancer une case à un état)

3-Sauvegarder une carte (sauvegarde en format texte dans un stockage)

3-Suppression de carte