Interaction distribuée. 3A SRI.

Arnaud GRODIN
Nelson SANCHEZ

Présentation du jeu.

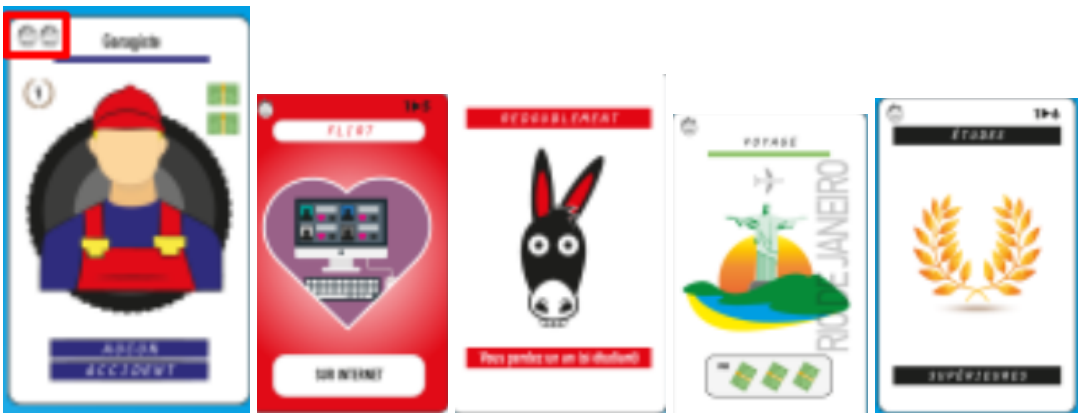
Smile life est un jeu où le but est de faire sa vie.
On peut faire sa vie en ayant :
Une vie pro: des études puis un métier et puis du salaire.
Une vie perso : Flirter puis pouvoir se marier et ensuite avoir des enfants

Le but du jeu est d'avoir le plus de smileys (points)  dans sa vie. Chaque carte donne (ou pas) un nombre établi de smiles au joueur. Les smiles, inscrits en haut à gauche des cartes posées, indiquent votre niveau de bonheur.

Commencement du jeu

Le jeu commence, le maître (machine) crée la pioche avec le nb de cartes dispo et la défausse (qui est initialisé à 0). Il crée également la table (whiteboard) où seront posées les cartes des joueurs, la pioche et la défausse.

Le maître distribue ensuite 5 cartes à chaque joueur disponible. Ce sont 5 cartes aléatoirement choisies. Les joueurs peuvent visualiser sa main en utilisant son application android.
/



Exemple de main pour joueur 1. Il a 5 smileys, les smileys peuvent être visualisés en haut à gauche de chaque carte. Cette carte est composée d'un métier, un flirt, une carte malus, une carte acquisition et une carte études.

Le joueur qui a plus de smileys dans sa main commence le jeu et est à lui de jouer.

Jouer un coup.

Lorsque c'est son tour, un joueur peut faire deux actions.

- **-Piocher une carte** de la pioche (le joueur a 6 cartes) et puis jouer une carte face visible pour l'ajouter dans son jeu (cartes études, flirts, métiers, marriages...). **Donner une carte** face visible à un autre joueur (carte malus) ou bien **défausser une carte** et passer son tour. (il finit toujours son tour avec 5 cartes en main)
- Prendre la toute dernière carte de la défausse si ce n'est pas une carte qui lui appartenait auparavant et ensuite la jouer directement (c'est à dire, il prend la carte de la défausse et l'ajoute dans son jeu s'il a la possibilité de la jouer directemment (cf. règles).

Ensuite, c'est le tour à la personne 2.

Fin du jeu.

Le jeu s'arrête quand la pioche est finie.

Chacun jette alors les cartes qu'il possède encore en main et compte les smileys sur les cartes posées dans son jeu. Le joueur qui a le plus de smileys remporte la partie.

Organisez votre vie.

(cf. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.play-in.com/pdf/rules_games/smile_life_regles_fr.pdf)

Logique du jeu.

Maître.

Avant de commencer la partie j'ai besoin d'un agent "maître du jeu" qui va s'occuper de la logique suivante:
Attributs :
-Nombre de joueurs
-Nombre de cartes dispo.
-Gérer la pioche et la défausse.
-Prendre la logique des cartes durant chaque tour.
-Dire qui est le prochain joueur qui va jouer.
-Est connecté avec les agents Joueur et Whiteboard

Le maitre:
Prépare la pioche, la défausse et les cartes des joueurs.
Maintient un suivi du tour des joueurs.

Interaction avec les joueurs :
Distribue les cartes au début du jeu.
Réagit aux actions des joueurs (piocher, jouer une carte).
Suit les scores des joueurs.

Interaction avec le whiteboard:
Afficher l'état du jeu (pioche restante, carte sur le dessus de la défausse).
Afficher les scores à la fin de la partie.

Execution:

1. **Initialisation** :

Le maître initialise le jeu avec un nombre défini de joueurs et de cartes.

Les cartes sont distribuées aux joueurs via setHand.

2. **Gestion des Tours** :

Le maître notifie le joueur actif via yourTurn.

Le joueur effectue une action (joue une carte ou pioche), et le maître réagit en appelant les services appropriés.

3. **Mise à Jour du Tableau Blanc** :

Après chaque action d'un joueur, le maître met à jour l'état du jeu (pioche restante, défausse).

4. **Calcul des Scores** :

Une fois la partie terminée (pioche vide), le maître demande les scores aux joueurs, les compile et affiche les résultats.

Callbacks (fichier main.py)
on_agent_event_callback : Gère les connexions des agents joueurs et whiteboard et enregistre les noms des agents connectés dans self.joueurs et self.whiteboard.

service_callback :Gère les services appelés par les joueurs, comme jouerCarte piocherCarte et finTour

Whiteboard

Joue le rôle d'interface graphique du jeu et affiche des informations pertinentes pour les joueurs et le maître. Ne prend pas de décisions dans le jeu, mais il est responsable de présenter l'état actuel et d'assurer une certaine interaction visuelle.

Attributs :
title: Le titre du tableau blanc.
background_color: La couleur d'arrière-plan du tableau.
labels_visible: Définit si les étiquettes (labels) sont visibles ou non.
chat_messages: Stocke les messages de chat échangés entre les joueurs ou envoyés par le maître.
game_state : Représente l'état du jeu (état de la pioche et la carte au desus de la défausse)
scores : Stocke les scores des joueurs une fois calculés par le maître.

Le Whiteboard:

Le programme configure l'agent avec un nom (agent_name) et un port réseau (port). Crée les inputs, outputs et services nécessaires pour que le tableau blanc puisse interagir avec le maître et les joueurs.

Interaction avec le maitre:
Reçoit des informations sur l'état du jeu
Affiche les scores des joueurs
Peut être réinitialisé

Interaction avec le joueur:
Affiche les messages de chat envoyés par les joueurs.
Peut afficher les cartes jouées ou d'autres informations en fonction des commandes reçues.

Execution:

Initialisation :

Le tableau blanc démarre et configure ses inputs, outputs, et services.

Mise à Jour de l'État du Jeu :

Le maître appelle le service updateGameState pour mettre à jour la pioche et la défausse.

Le tableau blanc met à jour ces informations dans self.game_state.

Affichage des Scores :

Le maître appelle le service showScores pour envoyer les scores des joueurs.

Le tableau blanc les affiche via self.add_text().

Réinitialisation :

Le tableau blanc peut être réinitialisé en appelant le service clear.

Callbacks (fichier main.py)

on_agent_event_callback : Gère les connexions avec d'autres agents.

on_service_callback : Gère les services appelés par le maître ou les joueurs.

Joueur

Représente un joueur individuel dans le jeu. Le joueur s'occupe de la logique suivante:

Attributs :
main : Une liste contenant les cartes en main du joueur.
whiteboard : La référence à l'agent Whiteboard pour afficher les informations.
maitre : La référence à l'agent Maître pour interagir avec le système de jeu.
score : Le score actuel du joueur, calculé à partir des cartes jouées.

Le joueur:

Le programme commence par configurer l'agent joueur.
Les services nécessaires (setHand, addCard, yourTurn, getScore) sont initialisés.
L'agent joueur reste en attente des instructions du maître et répond aux services.
Une fois le jeu terminé, l'agent s'arrête.

Interaction avec les maitre:
Reçoit des cartes
Informe le maître lorsqu'il joue une carte
Demande des cartes à piocher
Envoie son score au maître

Interaction avec le whiteboard:
Affiche les cartes en main ou les actions effectuées via des services (afficher_main).

Execution:

5. **Initialisation** :

Le maître distribue les cartes

6. **Tour** :

Si une carte est jouée, elle est envoyée au maître via jouer_carte.

Si aucune carte ne peut être jouée, le joueur demande une carte via piocher_carte.

7. **Fin Tour** :

Une fois son action terminée, le joueur appelle le service finTour pour informer le maître

8. **Calcul des Scores** :

À la fin de la partie, le maître appelle getScore pour obtenir les scores de tous les joueurs

Callbacks (fichier main.py)

on_agent_event_callback : Gère les connexions avec d'autres agents.
Si un maître ou un whiteboard se connecte, leurs références sont enregistrées dans self.maitre ou self.whiteboard.

on_service_callback : Répond aux services appelés par le maître. **setHand**, **addCard**, **yourTurn**, **getScore**.

Tests

Concernant les tests des différents scripts et fichiers .py, nous n’avons pas pu tester les différentes fonctionnalités des agents et nous n’avons pu les intégrer dans le Circle pour que chaque agent interagit les uns avec les autres (manque de temps et surtout comment utiliser le logiciel Circle / whiteboard).