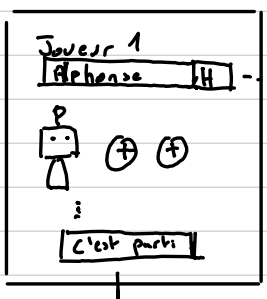
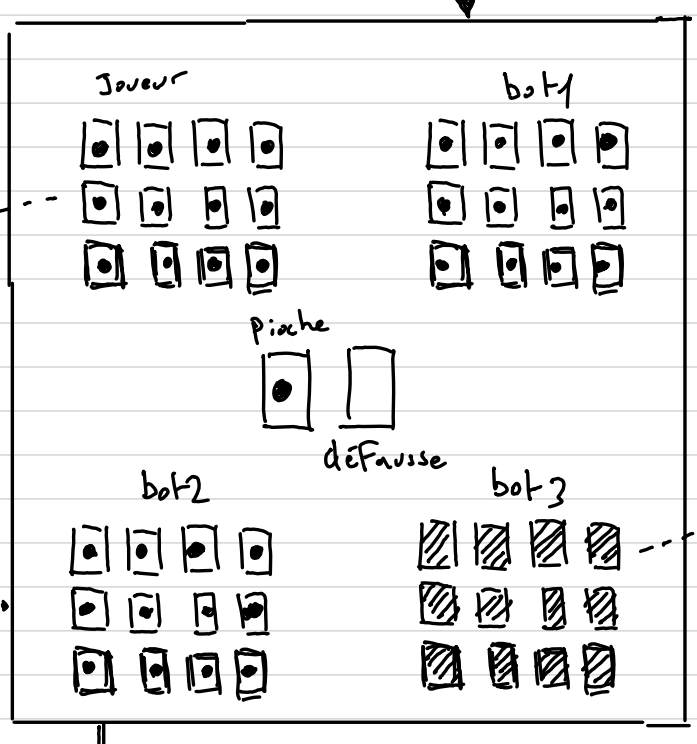
Définition API

Cmb de joueurs max ? 4 ?



→ humain / IA

PoC :
1 humain
x IA
(3 niveau de difficulté)



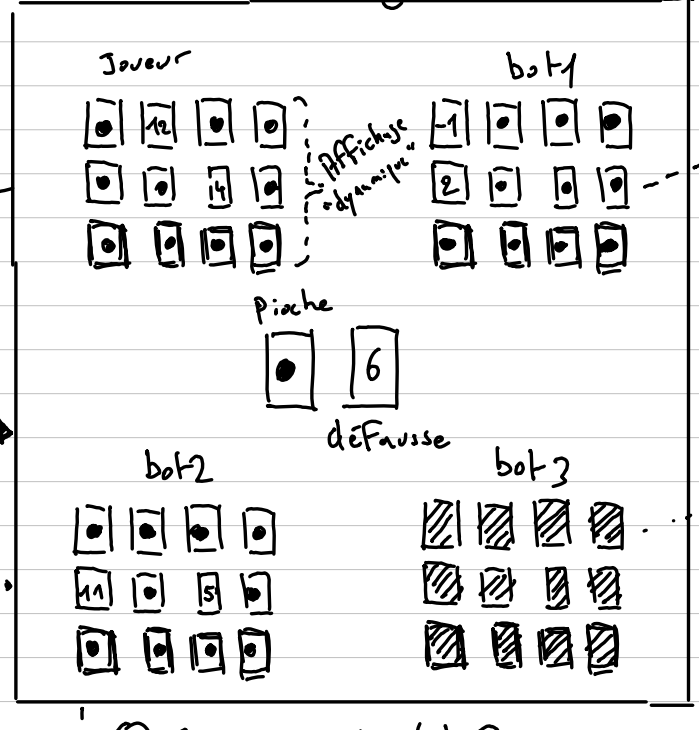
possible de zoomer ? <-

Cachée

Visible

Cartes grisées → si bot inactif

① Choisir 2 cartes



② Choisir pioche / défausse

- Remplacer une de ses carte
- Défausser la carte → Révéler une carte
- Remplacer une de ses cartes

③ Le(s) bot(s) joue(nt)

⇒ On envoie / récupère des requêtes POST avec l'état du board.

- Définir "clics" possibles en fonction de là où se trouve le joueur dans le flow

JSON pour créer l'écran principal :

```
Game_State: {
    game : {
        id : 1234,
        game_state : {
            discard : 5,
            draw : 12 | None
            player_boards : [
                {
                    player_name : Alphonse
                    player_board : [.... ], ← board representation here
                }, ...
            ]
        }
    }
```

→ Les réponses de l'API sont toujours ça.
→ Les appels sont des GET

boucle principale

- GET draw_card : récupère la carte de la pioche
- GET replace_card_with_draw (x,y) : remplace la carte à l'endroit (x,y) par la carte piochée
- GET discard_draw_card : met la carte piochée dans la défausse
- GET reveal_card (x,y) : révèle la carte en position (x,y)
- GET replace_card_with_discard (x,y) : remplace la carte en position (x,y) par la carte de la défausse.
- GET bot_moves (bot_name) : le résultat des décisions du bot (liste de game_state?)

- POST create_game : envoie les données pour mettre en place la partie.

```
data: {
    [player_name : "Alphonse",
     player_type : "human",
     ia_streaght : None],
    [player_name : "Bot 1",
     player_type : "bot",
     ia_streaght : 0],
    ...
}
```

Renvoie : game_state pour le début de partie.

persistance ? (score) → côté serveur.