




# Itération 2 : Bilan



Bernard Julien, Robineau Thomas, Rouyer  
Hugolin, Russo Nicolas  
S5 RA-IL1



# Sommaire

---

- I. Ce qui était prévu
- II. Répartition des tâches
- III. Détail des tâches
- IV. Prochaine itération

# Idées de départ

---

- Amélioration de **l'IA**:
- Ajout d'un système de **point** et de **temps de jeu**
- Prise en charge des **transformateurs** dans le jeu
- Tour par tour avec **deux joueurs**
- **Optimisation** du système de **recettes/états**
- **Réduction** de la classe **DonneesJeu**

# Répartition des tâches

---

Thomas : Amélioration de **l'interface graphique et prise en charge des transformateurs**

Hugolin : Prise en charge du mode 2 joueurs + optimisation du code

Nicolas : **Optimisation** du système de recettes, des classes, optimisation de la **vue**

Julien : Amélioration de **l'IA**

# Ce qui fonctionne - Système de transformation

---

- Les aliments peuvent changer d'états (couper, cuit ou les 2)
- Création de tests pour vérifier le bon fonctionnement du jeu
- Mise à jour des différentes règles de légalité de l'interaction du joueur avec les transformateurs
- Modification du système de données pour mieux correspondre aux attentes de l'IA

# Ce qui fonctionne - Interface Graphique



- Système de cuisson sur les transformateurs (Poêle et Planche) fonctionnel
- Mise à jour de l'affichage des aliments transformés (Foncé pour cuit et étoilé pour coupé)
- Affichage de l'empilement des aliments sur le joueurs (Cercles de plus en plus petits)

# Détails - Optimisation du système de plats

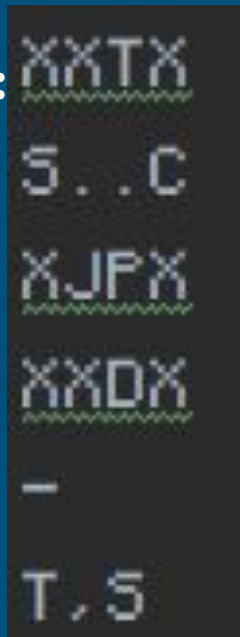
- **Aliment : Nom**
- **Plat** : Liste d'Aliments
- **Etat** : Plus de patron **décorateur**
  - Etat : **entier** (0 ; 1 ; 2 ; 3)
  - Pas de prévision d'ajout d'état
  - Optimise temps de calcul
- **Équivalences** entre différents Plats
  - Plus facile de comparer
  - Nom et état

```
public class Aliment extends Mouvable {  
  
    /**  
     * Etat de l'aliment  
     * 0 : cru  
     * 1 : cuit  
     * 2 : coupe  
     * 3 : cuit et coupe  
     */  
    16 usages  
    int etat;
```

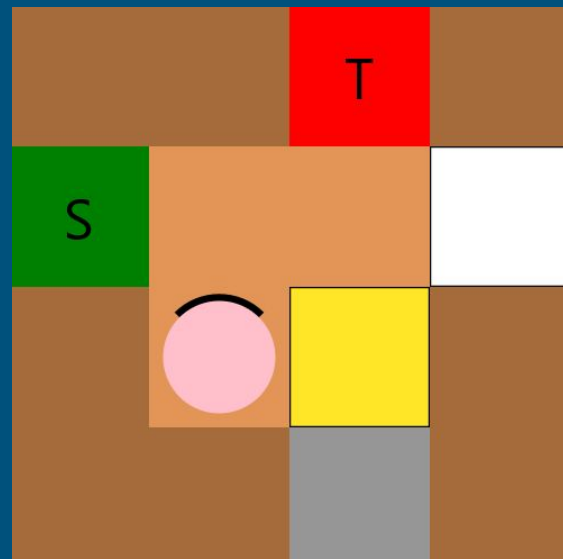
# Détails - Ajout de plat but au fichier texte

En plus de carte, intégration de **plats but** :

- **Séparateur** entre carte et plat ("-")
- Ajout de **l'état** (C, R) puis de **l'aliment**
  - Ex : (C R T : Tomate - Coupée - Cuite)
  - **Plat composé** possible
    - Salade-Coupée, Tomate = C S,T
- **Plusieurs plats but**
  - Chaque ligne
- Sera plus **optimisé** à l'avenir
  - Mettre **chiffres** pour les états
  - Mettre les Aliments avant les états



XXTX  
S..C  
XJFX  
XXDX  
-  
T.S

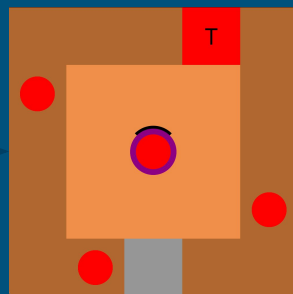
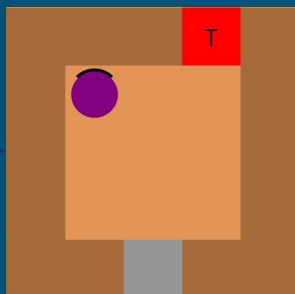




# Ce qui fonctionne - Gestion des données

- Le niveau est sous forme de **fichier texte**
- **Constructeur par copie** pour chaque objet
  - Évite les effets de bords
- **Vérification** si une instance **est identique** à une autre
- **Création d'un jeu de données à partir d'un fichier texte interprété par l'IA et l'interface**
- Tests vérifiant **le bon fonctionnement** du jeu

```
XXXTX
.....
XJ..X
X...X
X...X
X...X
XXDXX
```



# Prochaine itération

---

- Début de mise en place de l'IA centralisée
- Début de mise en place de **BDI** (IA)
- Ajout d'un système de **point** et de **temps de jeu**
- **Automatisation** du système de recette

Éventuellement :

- Simplification du système de plat but (lecture de fichier)
- Mise à jour de l'interface graphique