

Projet Logiciel Transversal

Hélène HU – Adrien SERMET

Table des matières

| | |
|--|----|
| 1 Objectif..... | 3 |
| 1.1 Présentation générale..... | 3 |
| 1.2 Règles du jeu..... | 3 |
| 1.3 Conception Logiciel..... | 3 |
| 2 Description et conception des états..... | 4 |
| 2.1 Description des états..... | 4 |
| 2.2 Conception logiciel..... | 4 |
| 2.3 Conception logiciel : extension pour le rendu..... | 4 |
| 2.4 Conception logiciel : extension pour le moteur de jeu..... | 4 |
| 2.5 Ressources..... | 4 |
| 3 Rendu : Stratégie et Conception..... | 6 |
| 3.1 Stratégie de rendu d'un état..... | 6 |
| 3.2 Conception logiciel..... | 6 |
| 3.3 Conception logiciel : extension pour les animations..... | 6 |
| 3.4 Ressources..... | 6 |
| 3.5 Exemple de rendu..... | 6 |
| 4 Règles de changement d'états et moteur de jeu..... | 8 |
| 4.1 Horloge globale..... | 8 |
| 4.2 Changements extérieurs..... | 8 |
| 4.3 Changements autonomes..... | 8 |
| 4.4 Conception logiciel..... | 8 |
| 4.5 Conception logiciel : extension pour l'IA..... | 8 |
| 4.6 Conception logiciel : extension pour la parallélisation..... | 8 |
| 5 Intelligence Artificielle..... | 10 |
| 5.1 Stratégies..... | 10 |
| 5.1.1 Intelligence minimale..... | 10 |
| 5.1.2 Intelligence basée sur des heuristiques..... | 10 |
| 5.1.3 Intelligence basée sur les arbres de recherche..... | 10 |
| 5.2 Conception logiciel..... | 10 |
| 5.3 Conception logiciel : extension pour l'IA composée..... | 10 |
| 5.4 Conception logiciel : extension pour IA avancée..... | 10 |
| 5.5 Conception logiciel : extension pour la parallélisation..... | 10 |
| 6 Modularisation..... | 11 |
| 6.1 Organisation des modules..... | 11 |
| 6.1.1 Répartition sur différents threads..... | 11 |
| 6.1.2 Répartition sur différentes machines..... | 11 |
| 6.2 Conception logiciel..... | 11 |
| 6.3 Conception logiciel : extension réseau..... | 11 |
| 6.4 Conception logiciel : client Android..... | 11 |

1 Objectif

1.1 Présentation générale

On se base sur le combat tour par tour du jeu Pokémon.

Deux Pokémon, A et B, s'affronte dans une arène. On a le choix de l'arène qui modifie le climat. On choisit une équipe de 6 Pokémon. Le choix des Pokémon se fait par une interface semblable à l'image ci-dessous :

The screenshot shows a Pokémon team management interface. At the top, there's a team slot for Haxorus, with other slots for Cradily, Serperior, Jellicent, Blissey, and Cresselia. Below this, the Haxorus details are shown, including its nickname, level (100), gender (—), happiness (255), and shiny status (No). It has the ability Mold Breaker and is holding a Lum Berry. Its moves are Iron Tail, Earthquake, Dragon Dance, and Outrage. The stats are: HP (green bar), Atk (252+), Def (yellow bar), SpA (—), SpD (36), and Spe (220). Below the stats, the EVs section shows a suggested spread: Fast Physical Sweeper: 220 Spe / 252 Atk / 36 SpD / (+Atk, -SpA). The EVs are distributed as follows: HP (0), Attack (252+), Defense (0), Sp. Atk (—), Sp. Def (36), and Speed (220). The IVs are all 31, resulting in base stats of: HP 76, Attack 147, Defense 90, Sp. Atk 60, Sp. Def 70, and Speed 97. The nature is Adamant (+Atk, -SpA).

| Base | EVs | IVs | Total |
|-------------|------|-----|-------|
| HP 76 | 0 | 31 | 293 |
| Attack 147 | 252+ | 31 | 432 |
| Defense 90 | 0 | 31 | 216 |
| Sp. Atk. 60 | — | 31 | 140 |
| Sp. Def. 70 | 36 | 31 | 185 |
| Speed 97 | 220 | 31 | 285 |

Remaining: 0

Nature: Adamant (+Atk, -SpA)

Ressources

images de fond pour les arènes, sprites de Pokémon, police d'écriture,

Illustration 1: Sprites de pokemon



1.2 Règles du jeu

Présenter ici une description des principales règles du jeu. Il doit y avoir suffisamment d'éléments pour pouvoir former entièrement le jeu, sans pour autant entrer dans les détails . Notez que c'est une description en « français » qui est demandé, il n'est pas question d'informatique et de programmation dans cette section.

Le joueur peut choisir d'attaquer ou changer de pokémon à chaque tour. Le but est de mettre KO tous les pokémon de l'adversaire. Il faut jouer sur les statistiques, forces et faiblesses des différents Pokémon.

1.3 Conception Logiciel

Présenter ici les packages de votre solution, ainsi que leurs dépendances.

2 Description et conception des états

L'objectif de cette section est une description très fine des états dans le projet. Plusieurs niveaux de descriptions sont attendus. Le premier doit être général, afin que le lecteur puisse comprendre les éléments et principes en jeux. Le niveau suivant est celui de la conception logiciel. Pour ce faire, on présente à la fois un diagramme des classes, ainsi qu'un commentaire détaillé de ce diagramme. Indiquer l'utilisation de patron de conception sera très apprécié. Notez bien que les règles de changement d'état ne sont pas attendues dans cette section, même s'il n'est pas interdit d'illustrer de temps à autre des états par leur possibles changements.

2.1 Description des états

Un état de jeu est formé par les deux équipe de 6 Pokémon et le terrain.

Pokémon A et B: Point de Vie restant, État Spécial, Position (Avant ou Arrière)

Chaque Pokémon possède un nom, un id, des points de vie (PV), 4 attaques, un ou deux types (Electrique, Psy, Posion, Spectre, Eau, Glace, Sol, Roche, Combat, Acier, Ténèbres, Feu, Plante, Insecte, Normal, Vol, Dragon, Fée), un état spécial (Paralysé, Brûlé, Endormi, Gelé, Empoisonné, Confus), un talent, un objet, une nature, des statistiques (PV, attaque, défense, attaque spéciale, défense spéciale, vitesse).

Chaque attaque a un nom, une puissance, une précision, des points pouvoir (PP), un type, une catégorie (physique ou spéciale), une priorité, une portée et un descriptif.

Terrains : Temps (Soleil, Pluie, Neige, Sable, Vent, Brouillard), Champs (Herbu, Electrique, Psychique, Brumeux).

Chaque terrain a un nom, une description et des effets.

2.2 Conception logiciel

2.3 Conception logiciel : extension pour le rendu

2.4 Conception logiciel : extension pour le moteur de jeu

2.5 Ressources

Illustration 2: Diagramme des classes d'état

3 Rendu : Stratégie et Conception

Présentez ici la stratégie générale que vous comptez suivre pour rendre un état. Cela doit tenir compte des problématiques de synchronisation entre les changements d'états et la vitesse d'affichage à l'écran. Puis, lorsque vous serez rendu à la partie client/serveur, expliquez comment vous aller gérer les problèmes liés à la latence. Après cette description, présentez la conception logicielle. Pour celle-ci, il est fortement recommandé de former une première partie indépendante de toute librairie graphique, puis de présenter d'autres parties qui l'implémente pour une librairie particulière. Enfin, toutes les classes de la première partie doivent avoir pour unique dépendance les classes d'état de la section précédente.

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

3.3 Conception logiciel : extension pour les animations

3.4 Ressources

3.5 Exemple de rendu

Illustration 3: Diagramme de classes pour le rendu

4 Règles de changement d'états et moteur de jeu

Dans cette section, il faut présenter les événements qui peuvent faire passer d'un état à un autre. Il faut également décrire les aspects liés au temps, comme la chronologie des événements et les aspects de synchronisation. Une fois ceci présenté, on propose une conception logiciel pour pouvoir mettre en œuvre ces règles, autrement dit le moteur de jeu.

4.1 Horloge globale

4.2 Changements extérieurs

4.3 Changements autonomes

4.4 Conception logiciel

4.5 Conception logiciel : extension pour l'IA

4.6 Conception logiciel : extension pour la parallélisation

Illustration 4: Diagrammes des classes pour le moteur de jeu

5 Intelligence Artificielle

Cette section est dédiée aux stratégies et outils développés pour créer un joueur artificiel. Ce robot doit utiliser les mêmes commandes qu'un joueur humain, ie utiliser les mêmes actions/ordres que ceux produit par le clavier ou la souris. Le robot ne doit pas avoir accès à plus information qu'un joueur humain. Comme pour les autres sections, commencez par présenter la stratégie, puis la conception logicielle.

5.1 Stratégies

5.1.1 Intelligence minimale

5.1.2 Intelligence basée sur des heuristiques

5.1.3 Intelligence basée sur les arbres de recherche

5.2 Conception logiciel

5.3 Conception logiciel : extension pour l'IA composée

5.4 Conception logiciel : extension pour IA avancée

5.5 Conception logiciel : extension pour la parallélisation

6 Modularisation

Cette section se concentre sur la répartition des différents modules du jeu dans différents processus. Deux niveaux doivent être considérés. Le premier est la répartition des modules sur différents threads. Notons bien que ce qui est attendu est une parallélisation maximale des traitements: il faut bien démontrer que l'intersection des processus communs ou bloquant est minimale. Le deuxième niveau est la répartition des modules sur différentes machines, via une interface réseau. Dans tous les cas, motivez vos choix, et indiquez également les latences qui en résulte.

6.1 Organisation des modules

6.1.1 Répartition sur différents threads

6.1.2 Répartition sur différentes machines

6.2 Conception logiciel

6.3 Conception logiciel : extension réseau

6.4 Conception logiciel : client Android

Illustration 5: Diagramme de classes pour la modularisation

