UNIVERSITÉ
CAEN
NORMANDIE

Développement d'un Jeu de Blackjack

Projet de l'UE – Méthodes de Conception

THOMAS Matthieu : 22304534

SIAGHI Massinissa : 22312276

TELLIER Basile : 22104032

4 décembre 2025

Table des matières

1 Introduction

1.1 Présentation générale du projet

Le projet s'inscrit dans le cadre de l'UE *Méthodes de conception*. Il consiste à développer un **jeu complet de Blackjack**, en plusieurs étapes :

- Développer un **jeu de cartes générique** : cartes, paquet, mélange, tirage...
- Développer un **jeu de Blackjack** reposant sur ce modèle de cartes.
- Implémenter les règles complètes : distribution, actions, gains, blackjack, bust, split, etc.
- Proposer une **version évoluée** du jeu adaptée au modèle MVC.

Le projet a donc suivi une progression logique structurée en deux grandes étapes :

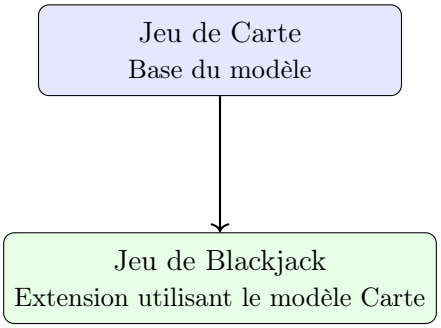

FIGURE 1 – Présentation des deux grandes étapes du projet de Blackjack

1.2 Problématique et points clés

< A MODIFIER

- **Comprendre et implémenter le langage RedCode** : Ce langage assembleur spécifique repose sur une sémantique simple en apparence, mais complexe dans ses effets. Chaque instruction (comme `MOV`, `ADD`, `CMP`, etc.) peut manipuler la mémoire avec des effets indirects, via des modes d'adressage variés (immédiat, direct, indirect, pré-décrémenté). Il fallait donc respecter avec rigueur la spécification ICWS-88 tout en permettant une exécution fluide et fidèle.
- **Modéliser la mémoire MARS et le comportement des guerriers** : La mémoire MARS fonctionne comme un tableau circulaire dans lequel les instructions peuvent être copiées, écrasées ou déplacées. Chaque guerrier y agit indépendamment, via un système de file d'instructions à exécuter. Il a fallu concevoir un système de mémoire robuste, capable de supporter cette logique concurrente sans erreurs de synchronisation.
- **Gérer l'exécution concurrente des guerriers** : Chaque guerrier possède une file FIFO d'instructions à exécuter. Une mauvaise gestion du multitraitement pouvait conduire à des déséquilibres ou à des comportements incohérents. Il fallait donc un système de file d'exécution stable, isolé et juste.
- **Générer automatiquement des guerriers efficaces** : Cette partie du projet relève de l'intelligence artificielle. Elle consiste à créer, faire évoluer et sélectionner automatiquement des programmes RedCode efficaces selon des critères de survie, d'agressivité ou de stratégie. Nous avons choisi d'exploiter un algorithme génétique simple pour générer et affiner les programmes au fil des générations. Cela demande un mécanisme de mutation, de croisement et de sélection bien intégré au reste de l'architecture.

— **Créer une interface utilisateur explicite et pédagogique** : L'interface graphique devait à la fois représenter l'état de la mémoire, l'avancement de chaque guerrier, et permettre d'observer l'évolution du match en temps réel. Cela implique une gestion précise des événements et des notifications entre le modèle et la vue (via le pattern Observer).
— **Maintenir la modularité du code** : Enfin, toute cette architecture devait rester claire, maintenable et évolutive. Le respect du modèle **MVC**, ainsi qu'une séparation stricte des responsabilités, ont été déterminants pour conserver une bonne lisibilité du projet.

</ **A MODIFIER**

1.3 Présentation du plan du rapport

Ce rapport a pour objectif de retracer de manière claire et progressive le développement du projet **Blackjack**. Il est structuré en septs sections, chacune correspondant à une phase essentielle du projet :

- **Section 1 – Introduction** : Pose le contexte du projet, ses objectifs pédagogiques et les motivations ayant guidé notre équipe.
- **Section 2 – Structuration du projet** : décrit les besoins fonctionnels et techniques, les fonctionnalités envisagées, ainsi que l'organisation du travail collaboratif.
- **Section 3 – Éléments techniques** : détaille les algorithmes et structures de données au cœur de la simulation, notamment le moteur d'exécution, la mémoire circulaire, et les opérateurs génétiques. **A MODIFIER**
- **Section 4 – Architecture du projet** : propose une vue d'ensemble de l'architecture logicielle, articulée autour du patron MVC. Cette partie s'appuie sur des diagrammes UML pour illustrer les relations entre les différentes classes et modules.
- **Section 5 – Tests du modele** : démontre les différents tests réalisés pour éprouver la robustesse des méthodes des modele de Cartes et de Blackjack.
- **Section 6 – Expérimentations et usages** : illustre l'utilisation concrète du simulateur à travers des cas d'usage commentés et des captures d'écran, en soulignant les résultats observés.
- **Section 7 – Conclusion** : dresse un bilan général du projet, récapitule les fonctionnalités implémentées et propose plusieurs axes d'amélioration pour la suite.

2 Structuration du projet

2.1 Analyse des besoins

2.2 Fonctionnalités implémentées

2.3 Organisation du travail

Le travail a été réparti entre les trois membres du groupe de manière à valoriser les compétences de chacun :

Membre	Responsabilités principales
Matthieu	Implémentation du `modèle` : — Conception de la logique modèle du Blackjack — Tests pour le modèle de Blackjack — UML des classes de Blackjack — Aide au rapport
Massinissa	Développement de la `vue` : — Interfaces graphiques (`Cartes`, `Blackjack`, etc.) — Modèle MVC Cartes & Blackjack — Mise en place du Contrôleur Cartes — Mise en place du Contrôleur Blackjack — Mise en place du Modèle & Vue Blackjack — UML Blackjack et rapport
Basile	Conception du `contrôleur` : — Conception du modèle de Cartes — Tests de Cartes — UML de Cartes et Blackjack — rédaction du rapport — Aide à la conception des tests Blackjack

TABLE 1 – Répartition des rôles au sein de l'équipe

Une collaboration régulière a été assurée via un dépôt Git, avec un suivi des commits et une validation croisée du code. Des séances hebdomadaires de synchronisation ont permis de maintenir une vision partagée de l'avancement.

2.4 Frise chronologique du projet

La planification et la progression du projet de Blackjack se sont organisées autour de jalons bien définis. La frise suivante illustre les grandes étapes de développement, depuis la compréhension du sujet jusqu'à la rédaction finale du rapport. Chaque étape a mobilisé les membres du groupe selon leur domaine d'expertise et les modules en charge.

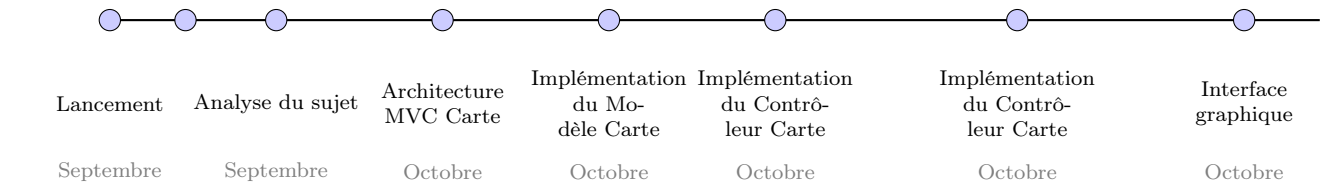

FIGURE 2 – Frise chronologique des grandes étapes du projet

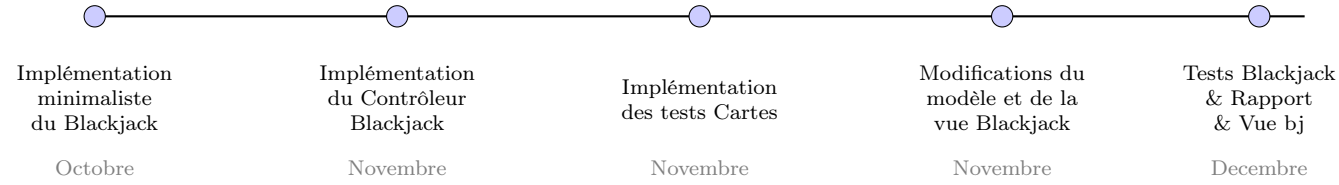

FIGURE 3 – Frise chronologique des grandes étapes du projet

3 Éléments techniques

3.1 Structures de données utilisées

< A REMPLIR PAR MASSI

1. Observer Pattern

2. Strategy Pattern

3. Factory method

4. Adaptator Pattern

A REMPLIR PAR MASSI />

5. Test Mock

4 Architecture du projet

4.1 Architecture de Carte

4.1.1 Contrôleur : orchestration de la simulation

4.1.2 Vue : interface graphique réactive

Communication modèle ↔ vue

4.1.3 Synthèse : une architecture M-V-C

4.2 Architecture de Blackjack

4.2.1 Contrôleur : orchestration de la simulation

4.2.2 Vue : interface graphique réactive

4.2.3 Synthèse : une architecture M-V-C

5 Tests du modele

5.1 Tests du modele Carte

5.2 Tests du modele Blackjack

5.2.1 Tests des actions

Des erreurs sont possibles au lancement des tests dû à la nature probabiliste de certains tests sur les actions.

5.2.2 Tests des joueurs

5.2.3 Tests du jeu

6 Expérimentations et usages

< A REMPLIR PAR MASSI

6.1 Cas d'utilisation :

6.2 Cas d'utilisation :

6.3 Cas d'utilisation :

6.4 Cas d'utilisation :

6.5 Cas d'utilisation :

6.6 Cas d'utilisation :

6.7 Cas d'utilisation :

6.8 Cas d'utilisation :

7 Conclusion

Récapitulatif des fonctionnalités principales

Propositions d'améliorations