Programmation : projet réussites

Une réussite est un jeu de cartes en solitaire. On fournit un logiciel permettant de programmer des réussites.

Le logiciel est organisé en plusieurs niveaux :

— au niveau le plus bas, on trouve les types, actions et fonctions permettant de manipuler des cartes et des tas de cartes sur un tableau de jeu.

— à un niveau intermédiaire, on trouve des actions permettant de décrire le déroulement d'une réussite en fonction des règles qui la définissent.

— enfin, on trouve les actions qui, pour chaque réussite, permettent de simuler plusieurs parties et de comptabiliser le nombre de parties gagnées à l'issue d'un nombre fixé de parties.

Les objectifs du projet sont :

— de s'approprier un logiciel permettant de *jouer* ces réussites. Cette phase peut se dérouler en plusieurs étapes :

1. observer le déroulement d'une réussite
2. étudier les chances de gain sur un grand nombre de parties
3. programmer une réussite

— de modifier et/ou compléter le logiciel fourni.

1 Objets et opérations de base

1.1 Cartes et jeux de cartes

Les objets de base sont les cartes. Une carte est un élément d'un jeu de 32 ou 52 cartes. On suppose ici que, dans une réussite, on ne manipule qu'un seul jeu. Une carte est caractérisée par :

— Sa couleur (par exemple pique ou carreau) et son rang (par exemple 7 ou valet).

— Sa visibilité : une carte est découverte (on voit sa face) ou cachée (on voit son dos). A toute carte sont associées des vues externes de face ou de dos. Le dos des cartes d'un jeu est le même pour toutes les cartes.

Trois relations d'ordre sont définies : sur les rangs, sur les couleurs et sur les cartes. Dans un jeu de 52 cartes, le plus petit rang est le deux ; dans un jeu de 32 cartes, c'est le sept.

1.2 Tas, séries

Sur le tableau de jeu d'une réussite, les cartes sont regroupées en tas, sous-ensembles du jeu de cartes. Dans un tas, une carte peut être découverte ou cachée. Pendant une réussite, l'ensemble des tas forme une partition du jeu de cartes. En particulier, une carte isolée est en fait un tas singleton.

Un tas a une localisation sur le tableau de jeu et un mode d'étalement : les cartes d'un tas peuvent être empilées (on ne voit que la carte située au dessus, de face ou de dos selon sa visibilité) ou étalées (on voit toutes les cartes du tas, de face ou de dos).

Une série est un tas particulier dans lequel les cartes sont rangées selon un ordre précis (par exemple, l'ordre croissant des rangs).

1.3 Tableau de jeu d'une réussite

Le tableau de jeu d'une réussite définit la disposition des tas de cartes sur la table, au départ et pendant le jeu. Le talon est l'ensemble des cartes qui restent une fois que la disposition initiale a été réalisée. Le rebut est l'ensemble des cartes qui n'ont pas pu être jouées.

Certaines opérations permettent d'initialiser une partie : construire un jeu neuf, battre les cartes, définir les tas jouant un rôle pour la partie, leur localisation et leur mode d'étalement, assurer la disposition initiale en extrayant certaines cartes pour initialiser des tas, etc.

1.4 Mouvements des cartes, modifications des tas

Pendant le déroulement d'une réussite, les opérations permises sont les suivantes :
— Une carte peut être déplacée mais uniquement d'un tas vers un autre.
— Une carte peut être retournée, ce qui change sa visibilité (mais ne la déplace pas).
— Le mode d'étalement des cartes d'un tas peut être modifié.
— Un tas peut être retourné, ce qui change la visibilité des cartes et l'ordre dans lequel elles se trouvent (mais ne le déplace pas).
— Un tas peut être posé sur un autre tas.
— Un tas peut être battu. Dans le tas résultant, les cartes se trouvent dans un ordre aléatoire. Battre un tas ne le déplace pas.

2 Spécification des objets de base

2.1 Couleurs

On considère l'ordre croissant sur les couleurs : trèfle, carreau, coeur, pique. On définit les types, constantes et fonctions suivants :
— **Couleur** : type [Trèfle, Carreau, Coeur, Pique]
— **PremièreCouleur** : constante Trèfle de type Couleur
— **DernièreCouleur** : constante Pique de type Couleur
— **CouleurSuivante** : fonction (C : Couleur) $\longrightarrow$ Couleur,
{*définie de manière circulaire, CouleurSuivante(C) : si C < DernièreCouleur alors C+1 sinon PremièreCouleur*}

2.2 Rangs

Dans le cas d'un jeu de 52 cartes, on considère l'ordre croissant sur les rangs : deux, ..., dix, valet, dame, roi, as. Un jeu de 32 cartes *commence* au rang sept.
On définit les types, constantes et fonctions suivants :
— **Rang** : type [Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix, Valet, Dame, Roi, As]
— **PremierRang** : Rang,
{*valeur constante initialisée par l'action CréerJeuNeuf, Deux pour un jeu de 52 cartes, Sept pour un jeu de 32 cartes*}
— **DernierRang** : constante As de type Rang
— **RangSuivant** : fonction (R : Rang) $\longrightarrow$ Rang,
{*définie de manière circulaire, RangSuivant(R) : si R < DernierRang alors R+1 sinon PremierRang*}

2.3 Cartes

On se place ici dans l'hypothèse où l'on ne joue qu'avec un seul jeu de 32 ou 52 cartes.
Les cartes font partie de tas de cartes. Elles sont construites au moment de construire un jeu de cartes. Une carte a une **couleur**, un **rang** et une **visibilité**.
On définit un type carte :
Carte : type
et les sélecteurs suivants :
— **LeRang** : fonction (C : Carte) $\longrightarrow$ Rang
— **LaCouleur** : fonction (C : Carte) $\longrightarrow$ Couleur

— **EstCachée** : fonction (C : Carte) ⟶ booléen,
{*vrai* ⟺ *la carte est cachée*}
— **EstDécouverte** : fonction (C : Carte) ⟶ booléen,
{*vrai* ⟺ *la carte est découverte*}

On définit une relation d'ordre sur les (faces de) cartes : C1 est avant C2 ⟺ la couleur de C1 précède celle de C2 ou C1 et C2 sont de même couleur et le rang de C1 précède celui de C2. On définit les fonctions de comparaisons suivantes :

— **CouleurInférieure** : fonction (C1, C2 : Carte) ⟶ booléen
— **MêmeCouleur** : fonction (C1, C2 : Carte) ⟶ booléen
— **RangInférieur** : fonction (C1, C2 : Carte) ⟶ booléen
— **MêmeRang** : fonction (C1, C2 : Carte) ⟶ booléen
— **EstCarteAvant** : fonction (C1, C2 : Carte) ⟶ booléen,
{*EstCarteAvant(C1, C2) est vrai* ⟺ *C1 précède C2 selon l'ordre sur les cartes.*}

2.4 Tas de cartes, jeu de cartes

La **hauteur** d'un tas est le nombre de cartes qu'il comporte. Sa **localisation** le situe sur le tableau de jeu. Un tas y apparaît dans le mode empilé ou étalé : on parle du **mode d'étalement** d'un tas. Un jeu de cartes est un tas particulier comportant toutes les cartes (32 ou 52). Un **tas vide** ne comporte aucune carte, sa hauteur est 0. Un tas vide est **actif** s'il joue un rôle sur le tableau de jeu : il a alors une localisation et un mode d'étalement et il apparaît lors de la visualisation du jeu. Un tas vide devient **inactif** lorsqu'il ne joue plus aucun rôle ; dans ce cas il n'apparaît plus lors de la visualisation du jeu. Tout tas non vide est nécessairement actif. L'ordre des cartes dans un tas non vide définit la relation **être au dessus**. Dans les spécifications qui suivent, et par convention, les cartes d'un tas sont numérotées du bas vers le haut, la carte du dessous ayant le numéro 1.

Types et constantes

— **Tas** : type
— **Rôle** : type [actif, inactif]
— **Mode** : type [empilé, étalé]
— **NbCartes** : entier dans {32, 52} {*Nombre de cartes du jeu (32 ou 52) : valeur constante initialisée par l'action CréerJeuNeuf*}
— **HmTas** : constante de type entier,
{*hauteur maximum d'un tas, lorsque l'on traite des réussites avec un seul jeu de cartes HmTas = NbCartes*}

Visualisation des tas

— **Localisation** : type,
{*caractérise un emplacement sur le tableau de jeu*}
— **InitialiserEcran** : action
— **AfficherTas** : action (donnée T : Tas, txt : texte),
{*affiche le tas T, en faisant apparaître le texte txt à proximité.*}

Constructeurs de tas

— **CréerJeuNeuf** : action (donnée N : entier dans {32, 52}, L : Localisation ; résultat T : Tas),
{*forme en L le tas empilé T avec l'ensemble des N cartes du jeu dans l'ordre des cartes et faces cachées. Donne leur valeur aux variables NbCartes et PremierRang.*
Pré-condition : l'emplacement L est disponible.}

— **CréerTasVide** : action (donnée L : Localisation, M : Mode ; résultat T : Tas),
{*associe à T un tas vide actif placé en L et de mode d'étalement M.*
Pré-condition : l'emplacement L est disponible.}
— **SupprimerTasVide** : action (donnée-résultat T : Tas),
{*rend le tas vide inactif. En particulier, la place qu'il occupait devient libre pour un autre tas.*
Pré-condition : le tas T est vide et actif.}

Testeur et sélecteurs de tas

Sauf pour la première ces fonctions ne sont définies que pour des tas actifs.
— **TasActif** : fonction (T : Tas) $\longrightarrow$ booléen,
{vrai $\Longleftrightarrow$ le tas est actif}
— **TasVide** : fonction (T : Tas) $\longrightarrow$ booléen,
{vrai $\Longleftrightarrow$ le tas est vide}
— **TasEmpilé** : fonction (T : Tas) $\longrightarrow$ booléen,
{vrai $\Longleftrightarrow$ le tas est en mode empilé}
— **TasEtalé** : fonction (T : Tas) $\longrightarrow$ booléen,
{vrai $\Longleftrightarrow$ le tas est en mode étalé}
— **LaHauteur** : fonction (T : Tas) $\longrightarrow$ entier sur [0 ... HmTas]
— **LaPlace** : fonction (T : Tas) $\longrightarrow$ Localisation

Consultation des cartes d'un tas

Ces fonctions ne déplacent pas la carte.
— **CarteSur** : fonction (T : Tas non vide) $\longrightarrow$ Carte,
{*carte située au dessus du tas* }
— **CarteSous** : fonction (T : Tas non vide) $\longrightarrow$ Carte,
{*carte située au dessous du tas* }
— **IèmeCarte** : fonction (T : Tas non vide, i : entier sur [1 ... HmTas]) $\longrightarrow$ Carte,
{*ième carte dans T (de bas en haut). Précondition : $i \leq$ LaHauteur(T)*}

Retournement d'une carte dans un tas

— **RetournerCarteSur** : action (donnée-résultat T : Tas),
{*retourne la carte située au dessus du tas T. Pré-condition : T non vide*}
— **RetournerCarteSous** : action (donnée-résultat T : Tas),
{*retourne la carte située au dessous du tas T. Pré-condition : T non vide.*}
— **RetournerIeme** : action (donnée-résultat T : Tas, i : entier sur [1 ... HmTas]),
{*retourne la ième carte du tas T. Pré-condition : T non vide et $i \leq$ LaHauteur(T).*}

Modifications d'un tas

— **BattreTas** : action (donnée-résultat T : Tas)
{*mélange les cartes du tas T*}
— **RetournerTas** : action (donnée-résultat T : Tas)
{*retourne le tas T : la première carte devient la dernière et la visibilité est inversée*}

Déplacements de cartes d'un tas sur un autre

Ces actions ne modifient ni la localisation des tas, ni leur mode d'étalement, ni la visibilité des cartes qu'ils contiennent.
— **DéplacerHautSur** : action (donnée-résultat T1, T2 : Tas),
{*enlève la carte située au dessus de T1 et la place au dessus de T2.*
Pré-condition : T1 n'est pas vide, T2 est actif.}

— **DéplacerHautSous** : action (donnée-résultat T1, T2 : Tas),
{*enlève la carte située au dessus de T1 et la place au dessous de T2.*
Pré-condition : T1 n'est pas vide, T2 est actif.}
— **DéplacerBasSur** : action (donnée-résultat T1, T2 : Tas),
{*enlève la carte située au dessous de T1 et la place au dessus de T2.*
Pré-condition : T1 n'est pas vide, T2 est actif.}
— **DéplacerBasSous** : action (donnée-résultat T1, T2 : Tas),
{*enlève la carte située au dessous de T1 et la place au dessous de T2.*
Pré-condition : T1 n'est pas vide, T2 est actif.}
— **DéplacerCarteSur** : action (donnée C, R : Couleur, Rang ; donnée-résultat T1, T2 : Tas),
{*enlève du tas T1, la carte de couleur C et de rang R et la place au dessus de T2.*
Pré-condition : T1 contient la carte et T2 est actif.}

Déplacement d'un tas sur un autre

— **PoserTasSurTas** : action (donnée-résultat T1, T2 : Tas),
{*pose le tas T1 sur le tas T2. Les deux tas doivent avoir le même mode d'étalement. A l'état final, le tas T1 est vide mais toujours actif, et le tas T2 comporte (de bas en haut) toutes les cartes qu'il avait au départ puis toutes les cartes de T1 dans l'ordre qu'elles avaient au départ dans chacun des tas. Cette opération ne modifie ni la visibilité des cartes, ni la localisation des tas T1 et T2, ni leur mode d'étalement.*}

3 Etude du relais des 7

On étudie la réussite *le relais des 7* La description de la réussite est donnée en annexe. Il s'agit d'étudier les algorithmes permettant de simuler la réussite, observer un programme la jouant et analyser les chances de gain sur un grand nombre de parties.

3.1 Simulation de la réussite R7 : spécifications

Représentation du tableau de jeu : il est constitué d'un talon (empilé, faces cachées), d'un rebut (empilé, faces découvertes), d'une ligne de 4 séries (étalées, faces découvertes), une dans chaque couleur, et d'un numéro de tour.
— **NumTourR7** : entier ≥ 0,
{*numéro de tour*}
— **SérieCouleurR7** : type Tas,
{*étalé, cartes découvertes, de même couleur, en rang croissant*}
— **LigneR7** : tableau sur [PremièreCouleur ... DernièreCouleur] de SérieCouleurR7
— **TalonR7** : Tas,
{*empilé, faces cachées*}
— **RebutR7** : Tas,
{*empilé, faces découvertes*}
Et pour la localisation des tas :
— **LocSériesR7** : tableau sur [PremièreCouleur ... DernièreCouleur] de Localisation
— **LocTalonR7**, **LocRebutR7** : Localisation

Formation du tableau de jeu initial :
— **SaisirLocTasR7** : action,
{*saisie des localisations LocSériesR7, LocTalonR7 et LocRebutR7.*}
— **CréerTableauInitialR7** : action,
{*création des tas du tableau de jeu et formation de l'état initial*}

— **ReformerTableauInitialR7** : action,
{*re-formation de l'état initial du tableau de jeu avec les tas de la réussite précédente.*}

Visualisation des états du jeu :

— **AfficherR7** : action,
{*Visualisation d'un état du tableau de jeu. En mode interactif, après affichage, l'exécution du programme est interrompue. La reprise survient après invite de l'utilisateur.*}

Jouer le relais des 7 :

— **ModeTrace** : type [SansTrace, AvecTrace]
— **ObserverR7** : action (donnée NP, NT : entier > 0),
{*joue NP parties du relais des 7 avec NT tours maximum. A l'état final, LigneR7, RebutR7 et NumTourR7 ont les valeurs atteintes à l'issue de la dernière réussite. Pendant le déroulement, les états successifs du tableau de jeu sont affichés.*}
— **JouerUneR7** : action (donnée NT : entier > 0, MT : ModeTrace),
{*joue un relais des 7 avec NT tours maximum. À l'état initial, le tableau initial est déjà formé. A l'état final, LigneR7, RebutR7 et NumTourR7 ont les valeurs atteintes à l'issue de la réussite. Les états successifs du jeu ne sont visualisés que si MT = AvecTrace.*}
— **JouerUnTourR7** : action (donnée MT : ModeTrace),
{*joue un tour du relais des 7 À l'état initial, le tableau initial est déjà formé. À l'état final, LigneR7 et RebutR7 ont les valeurs atteintes à l'issue du tour. Les états successifs du jeu ne sont visualisés que si MT = AvecTrace.*}
— **JouerTasR7** : action (donnée-résultat T : Tas ; résultat OK : booléen),
{*si la carte au dessus du tas T peut être posée (selon la règle du R7) sur une série, la carte est déplacée et OK a la valeur vrai. Sinon, OK a la valeur faux et le tableau de jeu est inchangé (en particulier le haut du tas T n'est pas déplacé).*}

Analyser le relais des 7

— **AnalyserR7** : action (donnée NP, NmaxT : entier > 0),
{*affiche l'histoire de NP parties de NmaxT tours maximum*}

4 Travail à faire

4.1 Les fichiers fournis pour la réalisation du projet

Les fichiers ci-dessous sont fournis :

— dans le sous-répertoire `assets` :
  — `cards.png` : images des cartes
  — `jackinput.ttf` : fonte utilisée pour les parties textuelles du graphique
— dans le sous-répertoire `include` :
  — les fichiers `.h` de tous les fichiers `.c` nécessaires.
  — `TypesConst.h` : définition des types de données et constantes utilisés globalement
— dans le sous-répertoire `lib` :
  — `Alea.c` : définition d'une action de tirage aléatoire
  — `Tas.c` : définition des primitives de gestion des cartes et des tas
  — `AfficherTas.c` : définition des primitives d'affichage des cartes et des tas
— dans le sous-répertoire `src` :
  — `R7.c` : réalisation de la simulation de la réussite *Relais des 7*
  — `InteractionR7.c` : primitives utilisées dans l'algorithme d'interaction avec l'utilisateur,
  — `ReussiteR7.c` : programme principal qui réalise l'interaction et appelle les actions de `R7.c`

— à la racine du répertoire projet :
  — `Makefile` : fichier qui permet la compilation des différents fichiers et des futures versions à réaliser.

Le `Makefile` fait référence aux répertoires `include`, `lib` et `src`.
La figure 1 donne le graphe de dépendance entre ces différents fichiers.

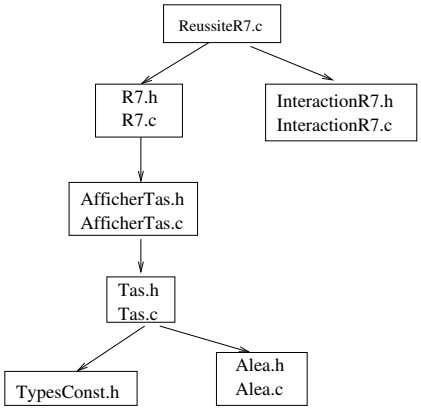

FIGURE 1 – Dépendances entre les fichiers C fournis

4.2 Etapes de travail

On vous suggère les étapes de travail énumérées ci-dessous. Certaines peuvent être traitées en parallèle. Par exemple, une fois intégré le mécanisme permettant de coder une réussite et de faire l'analyse d'une série de cette réussite avec l'exemple du *Relais des 7*, vous pouvez traiter une autre réussite et produire votre propre version des fonctions du fichier Tas.c.

1. Observer le programme codant la réussite *Relais des 7*, le comprendre et l'utiliser. Pour cela, compiler le fichier `ReussiteR7.c` qui fait appel à `R7.c` et `InteractionR7.c`, avec la commande `make ReussiteR7`.
2. Modifier le programme pour ajouter la possibilité de faire l'analyse (sans visualisation des parties jouées) d'une série de réussites *Relais des 7*. Il s'agit de compléter le fichier `R7.c` (un commentaire /* TODO */ apparaît dans le code et est visible via Visual Studio).
3. Choisir une autre réussite et réaliser les programmes la simulant. Si, par exemple, vous avez choisi la réussite *A la queue leu leu* :
   (a) Créer un fichier `ReussiteQLL.c`, un fichier `QLL.c` et un fichier `InteractionQLL.c` s'inspirant de `ReussiteR7.c`, `R7.c` et `InteractionR7`.c.
   (b) Compiler ces fichiers avec la commande : `make ReussiteQLL`
   (c) Elaborer des jeux d'essais et réaliser les tests correspondants.
4. Changer l'implantation du type **tas** :
   L'implémentation du type **tas** fournie dans **Tas.c** est basée sur une structure de listes doublement chaînées.
   (a) Choisir une autre représentation que des listes doublement chaînées pour représenter une séquence de cartes (listes chaînées dans un seul sens, tableaux, ...).
   (b) Modifier les fonctions de manipulation de tas (en ré-écrivant votre propre version du fichier **Tas.c**).
   (c) Tester et intégrer le tout.

5 ANNEXE : les réussites considérées

5.1 Le relais des 7 (R7)

Cartes : un jeu de 52 cartes.

Tableau de jeu : les quatre 7 sont placés sur une ligne horizontale face découverte, chacune constituant la première carte d'un tas. Il y a de plus un talon (empilé, faces cachées) et un rebut (empilé, faces découvertes), vide au départ.

But A partir des quatre sept de départ, former quatre séries croissantes (une dans chaque couleur) étalées.

Action : Les cartes du talon sont retournées une à une pour aller,

1. sur un tas de la ligne, si sa couleur est celle de la carte et si sa carte du dessus est de rang immédiatement inférieur à celui de la carte : un valet de coeur va sur un dix de coeur, un deux de carreau va sur un as de carreau, un as de trèfle va sur un roi de trèfle, etc.
2. sous un tas de la ligne, si sa couleur est celle de la carte et si sa carte du dessous est de rang immédiatement supérieur à celui de la carte : un sept de coeur va sous un huit de coeur, un roi de trèfle va sous un as de trèfle, un as de pique va sous un deux de pique etc.
3. sinon, sur le rebut, face découverte.
4. Si on a placé la carte donnée sur ou sous un tas de la ligne, les cartes du rebut peuvent être placées, l'une après l'autre, sur ou sous les tas de la ligne selon les règles (1) et (2).
5. Lorsque les cartes du talon sont épuisées, on recommence en prenant pour talon le rebut retourné (sans mélanger les cartes). La réussite est perdue si le but n'est pas atteint au bout de trois tentatives.

5.2 A la queue leu leu (QLL)

Cartes : un jeu de 32 ou de 52 cartes.

Tableau de jeu : une ligne de tas de cartes découvertes empilées. Le nombre de tas varie au cours de la partie. Au début, il y a deux tas, chacun comportant une carte découverte. Il y a de plus un talon (empilé, faces cachées).

But : Obtenir une ligne de deux tas, lorsque le talon est épuisé. Règle de déplacement des tas dans une ligne : deux tas de la ligne sont dits jouables, s'ils sont séparés par un tas intermédiaire et si leurs cartes du dessus sont de même couleur ou de même rang. On peut alors déplacer le tas intermédiaire sur celui placé à sa gauche. Une ligne est dite stable si aucun de ses tas n'est jouable.

Action : ayant formé les deux premiers tas avec les deux premières cartes du talon, les cartes du talon sont retournées une à une.

1. La carte retournée forme un nouveau tas placé à droite de la ligne. La ligne ainsi obtenue est rendue stable par applications répétées de la règle de déplacement : si le tas le plus à droite et le troisième tas en partant de la droite sont jouables, on déplace l'avant dernier tas et on répète le processus en cascade autant que possible. A la fin de ce processus la ligne est stable.
2. Fin de partie : lorsque le talon est vide, on essaye de jouer avec les deux premiers tas (les plus à gauche) en considérant que circulairement ils se trouvent à droite de la ligne.
3. La réussite est gagnée si le talon étant épuisé et tous les déplacements possibles ayant été faits, il ne reste que deux tas sur la ligne.
4. A l'issue d'une réussite, on reforme le talon pour la prochaine réussite de la manière suivante : les tas de la ligne sont posés les uns sur les autres de manière à obtenir un seul tas (empilé, faces découvertes) dans lequel on trouve, de haut en bas, les cartes du premier tas, puis celles du deuxième, et ainsi de suite jusqu'au dernier tas de la ligne. En retournant le tas ainsi obtenu, on dispose du prochain talon pour une nouvelle réussite.

5.3 Les quatre couleurs (C4)

Cartes : un jeu de 32 cartes.

Tableau de jeu : une ligne de quatre tas étalés. A chaque tas est associé une couleur (trèfle, carreau, coeur, pique).

But : obtenir quatre tas de cartes découvertes, chaque tas comportant toutes les cartes d'une même couleur.

Action : au départ, les cartes du talon sont distribuées, faces cachées dans les quatre tas de la ligne (8 cartes par tas).

1. La première carte d'un des tas est retournée face visible. Cette carte est glissée, face visible en dessous du paquet correspondant à sa couleur. On retourne la carte du dessus de ce paquet et on recommence l'opération.
2. La réussite est interrompue lorsque l'on place une carte sous un tas alors que toutes les autres cartes de ce tas sont découvertes.
3. On retourne alors toutes les cartes cachées, et si chacune se trouve dans le bon tas (a la couleur associée au tas où elle est), la réussite est gagnée. Sinon elle est perdue.

5.4 Montée-Descente (MD)

Cartes : un jeu de 32 cartes.

Tableau de jeu : une ligne horizontale de quatre tas de cartes découvertes empilées, destinés à former quatre séries croissantes. Quatre tas de stockage formés de cartes découvertes empilées.

But : former quatre séries croissantes (7, 8,. . .,V, D, R, A), une par couleur.

Action : Les cartes du talon sont retournées une à une pour,

1. "monter" sur l'une des séries de la ligne : la carte est placée au dessus d'une série. Elle doit être de la couleur associée à cette série. On ne peut le faire que si le rang de la carte est le successeur du rang de la carte du dessus de la série (un 8 sur un 7, un valet sur un 10, un as sur un roi, etc.), ou si la série est vide et la carte est un sept.
2. "descendre" sur l'un des tas de stockage : la carte est placée au dessus d'un tas de stockage. Pour cela il n'y a pas de contrainte sur la couleur. Mais on ne peut le faire que si le rang de la carte que l'on veut placer est inférieur ou égal au rang de la carte situé au dessus du tas. Toute carte peut être placée sur un tas de stockage vide.
3. Après avoir placé une carte sur l'une des séries de la ligne, on peut, avant de retourner la prochaine carte du talon, "remonter" une à une les cartes du tas de stockage vers les séries, en appliquant la règle (1).
4. La réussite est perdue, si l'on tire une carte du talon qui ne peut être placée ni dans une série, ni dans un tas de stockage.
5. La réussite est gagnée lorsque les quatre séries sont complètes.