Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Conception d'un générateur de jeux éducatifs

Lowette, Yves; Marchal, Jean-Christophe

Award date:
1993

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 20. Apr. 2024

Année Académique 1992-1993

"Conception d'un générateur de jeux éducatifs"

Yves Lowette
Jean-Christophe Marchal

Promoteur : **Monsieur Claude Cherton**

Mémoire présenté en vue
de l'obtention du grade de Licencié
et Maître en informatique

Résumé

Notre mémoire se propose de vous présenter la conception et l'implémentation partielle d'un générateur de jeux éducatifs. Cet outil, nous le destinons à des éducateurs de façon à ce qu'ils puissent créer des jeux sur mesure pour les personnes handicapées qu'ils ont sous leur responsabilité.

La première étape de notre étude a consisté en l'analyse d'un ensemble de jeux qui nous ont menés à la définition d'un paradigme unificateur. Les concepts constituant ce paradigme, organisés en un formulaire, nous ont fourni un outil fort simple pour réaliser la définition de jeux.

Cet outil de création permet de définir les trois éléments constituant la définition de jeux correspondant à la famille de jeux que nous nous sommes fixée. D'une part, il assure la création de l'ensemble des objets qui seront manipulés par le joueur. D'autre part, il se charge de la définition du plan de jeu pour lequel nous avons retenu une grille, un sabot et des zones dites de résultat, de message et d'attente. Enfin, il permet de définir les règles de jeu.

Nous avons tout lieu de croire que cet outil est efficace, car d'une part, il est capable de reconstruire les définitions des jeux choisis pour réaliser notre analyse, et d'autre part, il permet de réaliser des définitions donnant naissance à de nouveaux jeux.

Si cet outil apparaît à première vue efficace, il se révèle être également fort simple d'emploi. Cette simplicité, il la doit surtout au volet de la définition qui est consacré aux règles. En effet, l'utilisation du formulaire a pu être étendue à la définition des règles, ce qui a permis d'éviter la mise au point et l'utilisation d'un langage pour la définition de celles-ci.

Les étapes suivantes de notre travail, purement informatiques, ont consisté en l'élaboration d'une architecture et d'un ensemble de fonctions et de services, la mise au point des deux algorithmes principaux du logiciel, et finalement, la conception d'une interface et l'implémentation de la partie définition.

Abstract

This thesis describes the conception and the partial implementation of a generator of educational games. This implement is intended for educators so that they can create games to standard for disabled people.

The first step in our study consisted of analysing a set of games which brought us to the definition of an unifying paradigm. We organised the concepts constituating this paradigm in one document which provided a very simple tool to realise the definition of the games.

This creative toolbox allows to define the three elements generating the definitions of the games. These games correspond to the family of games we selected to carry out our analysis. First of all, the toolbox permits to create the group of objects that will be manipulated by the players. Secondly, it enables the definition of the plan of the game. Finally, it allows to define the rules of the games.

We believe that this toolbox is efficient for the following two reasons. On the one hand, it is capable of reconstructing the definitions of the games which were chosen to realise our analysis. On the other hand, it admits the possibility of realizing the definitions which give birth to new games.

At first sight, this implement not only appears to be efficient. Moreover it is easy to use. This simplicity is especially due to the definition section dealing with the rules. As a matter of fact, we were able to extend the application of the document to the definition of the rules. This avoided the perfecting and application of a language defining the games.

The following steps of our work were purely related to data processing. They consisted of elaborating an architecture and a set of functions and services, of the perfection of the principal algorithms of the software, and finally of the conception of an interface and implementation of the definition section.

Remerciements

C'est pour nous un plaisir d'exprimer nos remerciements sincères à Monsieur le Professeur Claude Cherton pour ses nombreux conseils, son écoute attentive, sa disponibilité et son ouverture d'esprit.

Table des matières

Introduction

partie 1 : Analyse et synthèse

1. Présentation et Analyse des jeux

2. Propriétés et concepts issus de l'analyse des jeux

3. Conception des jeux

4. Aspects statiques et dynamiques des concepts

partie 2 : Conception du logiciel

5. Choix pour la conception du logiciel

6. Architecture

7. Description des fonctions

8. Les services offerts par les concepts

9. Les structures de données

partie 3 : Les algorithmes

10. Présentation des algorithmes

11. Algorithme Partie Définition

12. Algorithme Partie Jeu

partie 4 : Interface

13. Présentation

14. Le logiciel

15. Conception d'une interface pour l'animateur

Conclusion

Bibliographie

Introduction

Présentation

Motivation

Notre mémoire se propose de vous présenter la conception et la réalisation d'un générateur de jeux éducatifs. Le logiciel que nous avons pu concevoir s'adresse plus particulièrement à des éducateurs encadrant des personnes handicapées.

L'intérêt de faire entrer la conception d'un tel outil dans le cadre de l'informatique est multiple. Tout d'abord, les moniteurs sont intéressés par l'informatique. Elle leur permet de concevoir un "entraîneur" qui est, face à l'enfant, patient et constant. Deuxièmement, les moniteurs conçoivent souvent de nombreux jeux sur toutes sortes de supports afin d'aider les enfants dont ils s'occupent à acquérir et maîtriser diverses notions. Ces jeux sont très variés de par l'étendue des disciplines abordées, mais également de par la variété des comportements des enfants, de leur progression personnelle et de leur familiarité avec tel ou tel monde de références. Une même matière sera ainsi présentée de manière différente à deux enfants différents. Enfin, les logiciels éducatifs sont malheureusement des ensembles fermés, qui ne peuvent évoluer que très légèrement en fonction des besoins. Les éducateurs doivent alors se constituer une logithèque très importante afin de répondre favorablement à la diversité des problèmes que rencontrent les personnes dont ils s'occupent.

On comprendra donc aisément le désir de vouloir offrir aux éducateurs un outil leur permettant de réaliser des jeux adaptés aux difficultés si variées des enfants.

Un générateur de jeux

On peut estimer qu'il existe une infinité de jeux car l'imagination n'a pas de limite. Un jeu pris individuellement peut d'ailleurs compter un très grand nombre de variantes car toute personne a la liberté d'apporter sa propre touche ou de l'adapter à ses besoins. Certains jeux, tels les échecs, exigent évidemment de respecter scrupuleusement des règles établies.

Cependant, la plus grande majorité accepte des variations qui sont reconnues ou mises au point et adoptées par des petits groupes de pratiquants. On trouve un bon exemple de ceci dans de nombreux jeux de cartes.

Si l'on considère qu'il existe une infinité de jeux, il faut aussi remarquer que l'on peut regrouper ceux-ci en familles. Par exemple les jeux se pratiquant avec une balle ou avec un bâton, les jeux de stratégie, les jeux de connaissances, les jeux de raisonnement ou les jeux de cartes que nous avons déjà cités.

Plutôt qu'une "boîte de jeux réunis" qui compterait certes plusieurs jeux, mais qui rencontrerait à l'arrivée le même type de problèmes que des jeux pris individuellement, c'est-à-dire la difficulté de s'adapter correctement aux désirs de l'éducateur, nous avons choisi de mettre au point un générateur de jeux. L'éducateur étant le mieux à même de définir ce dont il a besoin, nous désirons le laisser concevoir des jeux lui-même. C'est d'ailleurs ce qu'il fait bien souvent en dehors de l'informatique. A cette fin, nous voulons lui donner des "briques" de construction qu'il sera libre d'assembler à sa convenance. Il lui sera ainsi possible de constituer lui-même sa bibliothèque de jeux, qu'il pourra non seulement faire évoluer mais surtout dont il pourra modifier les éléments à volonté (comme si un enseignant pouvait réimprimer les passages des manuels que ses élèves assimilent le plus difficilement).

Nous avons donc voulu fournir un outil simple d'emploi, puissant, manipulant des concepts élémentaires qui, une fois définis et assemblés, permettront de définir une famille de jeux la plus importante possible.

Le paradigme unificateur

La première étape de notre travail fut de déterminer un paradigme unificateur des jeux qu'il allait être possible de créer. En fait, ce paradigme correspond à un ensemble de points communs que doivent posséder tous les jeux à générer. Il est à la fois réducteur de l'ensemble des caractéristiques qu'un jeu doit comporter, mais aussi suffisamment large pour offrir à l'animateur un outil de conception performant, intéressant et simple d'utilisation, lui assurant de définir selon un canevas unique, une famille de jeux qui soit la plus importante possible et qui présente les qualités attendues par le concepteur des jeux.

Pour définir ce paradigme, nous sommes partis de l'idée intuitive suivante : les jeux auraient à se pratiquer sur une grille divisée en cases et le joueur y manipulerait des jetons (ou pièces ou pions selon le vocabulaire employé). Cette idée nous est apparue fort intéressante. Elle nous a permis de limiter notre champ d'investigation tout en couvrant des jeux pour lesquels l'ordinateur semblait être un support puissant et bien adapté. En effet, la grille peut

être aisément représentée à l'écran et les jetons sont facilement manipulables au moyen de la souris (entre autres).

Comme vous le lirez dans l'étude qui suit, pour atteindre ces objectifs, nous sommes partis d'un ensemble de jeux ayant un certain nombre de caractéristiques communes, se jouant sur des supports semblables et correspondant, évidemment, à notre idée intuitive. En effet, la plupart de ces jeux se pratiquent sur une grille composée de cases de même taille et sur laquelle peuvent être disposés et éventuellement déplacés des pions, des pièces ou de jetons.

Objectifs principaux et démarche suivie

Les objectifs de notre travail sont clairs. Premièrement, nous devons fixer de façon précise le paradigme unificateur d'une famille de jeux. Il devra être à la fois suffisamment vaste et homogène. Deuxièmement, une fois cet objectif atteint, pour valider ce paradigme, chaque jeu de la famille devra pouvoir être relativement simplement décrit comme une actualisation du paradigme. Troisièmement, en tenant compte des résultats obtenus aux étapes précédentes, il conviendra de mettre au point un mode efficace et aisé de conception des jeux.

Une fois ces objectifs atteints, nous pourrons entamer la partie purement informatique de notre travail. Elle consistera en l'élaboration d'un logiciel supportant le générateur que nous voulons offrir, et ce indépendamment de l'interface avec le joueur.

Plan

Pour atteindre ces objectifs, nous avons réalisé une analyse, une conception et une implémentation partielle présentées dans les chapitres qui suivent. Notre étude se compose de quatre grandes étapes.

Dans la première étape, nous proposons l'étude d'un certain nombre de jeux, afin d'en retenir les caractéristiques communes. Pour nous conformer à l'idée intuitive de départ, nous avons sélectionné ces jeux car ils correspondent aux trois éléments de base du paradigme restant à fixer : la grille, les cases et les jetons. Notons déjà l'apparition d'un deuxième critère : les règles régissant les positionnements et les déplacements des jetons devaient être très simples. Ceci éliminait d'emblée des jeux tels que les échecs. Par contre, d'autres jeux pouvaient être retenus alors même qu'il n'était pas évident à priori qu'ils correspondent à notre idée intuitive. Par exemple, le jeu des dominos serait réalisable bien que l'on ne perçoive pas immédiatement la notion de case car elle n'est pas matériellement concrétisée.

De cette étude, nous avons pu dégager un outil d'analyse dont la structure s'est rapidement étoffée, fixant notre paradigme et, finalement, nous permettant de définir un outil de conception des jeux .

Dans l'étape suivante, nous abordons les aspects purement informatiques de notre travail avec la spécification et la conception détaillée du programme supportant le générateur.

La troisième étape propose les deux algorithmes principaux du logiciel. Signalons déjà que, faute de temps, il ne nous a pas été possible de terminer l'implémentation de celui-ci, mais qu'une des deux parties, celle consacrée à la création des jeux, est opérationnelle.

Dans la dernière étape, nous présentons L'interface de la partie implémentée du logiciel.

partie 1 : Analyse et synthèse

1. Présentation et analyse des jeux
2. Propriétés et concepts issus de l'analyse
3. Conception des jeux
4. Aspects statiques et dynamiques des concepts

1.
Présentation et Analyse des jeux

Comme précisé dans l'introduction, nous entamons notre étude par la description et l'analyse d'un ensemble de jeux dont les caractéristiques correspondent à l'idée intuitive du paradigme unificateur et rencontrent nos recommandations pour les règles de déplacement et de positionnement. Nous avons tenu à faire figurer cette analyse et cette description pour montrer que des jeux ne présentant, à première vue, que peu de similitudes pour des joueurs, peuvent, en fait, être regroupés dans une même famille, sur base d'une (plusieurs) caractéristique(s) fondamentale(s) commune(s), et nous mener à une définition complète et précise du paradigme unificateur.

La majorité des jeux sélectionnés dans cette première étape sont joués sur une grille composée d'un plus ou moins grand nombre de cases et emploient des pions, des jetons ou des pièces (selon le vocabulaire utilisé), possèdent des règles le plus souvent simples et, point important, ont un intérêt éducatif. Par ce choix, nous avons déjà réduit le nombre, parfois important, de caractéristiques qu'un jeu peut comporter.

Avant d'aborder la liste des jeux repris, il n'est peut-être pas inutile de faire une remarque concernant le vocabulaire utilisé dans ce chapitre. Nous emploierons les termes "grille", "sabot" et, "pièces" ou "pions" ou "jetons".

La **grille** correspond à l'espace où se déroule le jeu. Par exemple, pour le jeu d'échecs, l'échiquier constitue ce que nous appelons la grille. Pour un jeu de réseaux de tuyaux, l'espace dans lequel le réseau sera construit correspond à la grille. Celle-ci sera constituée de cases de tailles égales et de formes carrées. Ceci constitue une nouvelle limitation qui exclut des jeux se pratiquant sur des grilles dont les cases sont de forme triangulaire ou hexagonale (par exemple), ou de forme ou de taille irrégulière (nous pensons à certains jeux de stratégie).

Le **sabot** est la partie de l'espace de jeu (sorte de zone de stockage) où sont regroupées une partie ou toutes les pièces qui seront manipulables par le joueur dès le coup d'envoi de la partie active du jeu. Par exemple, dans certains jeux de cartes, le paquet des cartes non distribuées aux joueurs constitue ce que nous avons appelé le sabot. Notons également que

certains jeux n'en exigent pas car toutes les pièces sont placées sur la grille avant le début de la partie (par exemple le jeu de mémoire qui sera détaillé dans ce chapitre).

Enfin, il y a les **objets utilisés** dans le jeu, qui sont généralement appelés "jetons", "pièces" ou "pions". Dans cette première partie, nous emploieront indifféremment ces trois termes. Ces objets possèdent une face qui porte un motif particulier. Ce motif sera appelé contenu du jeton et le verso sera appelé son "dos".

Nous avons voulu présenter les jeux en répertoriant leurs caractéristiques selon un modèle général de façon à déjà faire ressortir des points communs. On trouvera donc pour chacun des jeux : le **but** que le joueur doit atteindre, les **objets** qui sont à sa disposition, les conditions qui entraînent la **fin du jeu**, la **présentation des pièces** au joueur et quelques **variantes** pour le jeu dont il est question. Remarquons que ce modèle est appliqué de façon souple pour permettre de présenter certains jeux n'ayant pas la même structure de description du fait qu'ils présentent des caractéristiques n'entrant pas dans ce moule.

1.1. Réseau de tuyaux

BUT : Joindre, au moyen d'un certain nombre de pièces représentant des morceaux de tuyaux, une entrée et une sortie de la grille à fin de constituer un réseau de façon telle que si on ouvrait un robinet à l'entrée, l'eau se répandrait dans l'ensemble du réseau et s'écoulerait par la sortie.

OBJETS : Ce sont des jetons qui représentent des morceaux de tuyaux (coudés, droits, en forme de T, ...).

FIN DU JEU : Lorsque les deux extrémités sont jointes.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci pourraient être toutes présentes dans le sabot ou bien n'arriver qu'une par une ou par groupes de taille définie.

VARIANTES :

- On pourrait admettre que le jeu se termine lorsque les deux extrémités sont jointes et ce quelque soit l'état du réseau, c'est-à-dire même si le réseau présente des fuites.

- Il pourrait y avoir dans la grille des points obligatoires à emprunter.

- Si les pièces arrivent les unes après les autres dans le sabot, le joueur est obligé d'employer la dernière pièce présentée. On pourrait imaginer que si elle ne correspond pas, le joueur puisse l'éliminer ou bien qu'il soit obligé de s'en servir. Dans ce dernier cas,

le morceau de tuyau contenu dans la pièce ne correspondant à aucun autre morceau de tuyau de pièces déjà placée sur la grille, le joueur sera malgré tout obligé de la placer sur la grille. Cette pièce constituera alors un point obligé de passage du réseau.

1.2. Jeu des associations

BUT : Sélectionner des pièces ayant des points communs entre elles. Le joueur doit appareiller les pièces proposées en fonction de la (des) caractéristique(s) commune(s) ou déterminée(s) par le concepteur. Par exemple, si les pièces choisies pour un jeu sont des formes géométriques, on pourrait regrouper les pièces par type de forme, ou par taille, ou par couleur, ou par un ensemble de caractéristiques.

OBJETS : Le joueur dispose d'une grille et d'un ensemble de pièces disposées sur celle-ci. Le contenu des pièces peut être très varié. On imagine facilement des formes géométriques, des couleurs, des objets usuels, les lettres de l'alphabet, ...

FIN DU JEU : Lorsque toutes les pièces de la grille ont été appareillées avec au moins une autre de la grille (de façon à constituer des couples, des triplets, ... de pièces ayant une (plusieurs) caractéristique(s) commune(s)) et que cet assortiment correspond à ce qui était exigé.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci pourraient être toutes présentes dans le sabot ou bien n'arriver qu'une par une ou par groupes de taille définie.

VARIANTE : La grille est vide au départ et les pièces se présentent au fur et à mesure dans le sabot. Le joueur doit alors placer les pièces dans la grille en les regroupant par paires, triplets, ...

1.3. Dominos

Ce jeu étant un classique connu de tout le monde, nous n'en faisons pas la description ici. Nous soulignons simplement le fait que le contenu des pièces peut-être étendu au-delà des séries de points que l'on trouve habituellement sur les pièces de ce jeu. On pourrait en effet prendre les pièces géométriques déjà citées plus haut, des classes d'objets, ... Remarquons enfin que le jeu des dominos peut être vu comme un cas particulier du jeu des associations.

1.4. Puzzle

Le principe du puzzle nous a donné l'idée de quelques jeux.

1°) Assez classiquement, on aurait un dessin réalisé par le concepteur qui serait découpé. Les pièces, de la taille des cases de la grille seraient mélangées et présentées au joueur qui aurait à recomposer le dessin de départ.

2°) Jeu de "taquet" : sur le même principe que le puzzle, on imagine un dessin découpé dont les pièces sont mélangées. Ensuite, on ôte une des pièces, et le dessin doit être recomposé en faisant glisser les pièces et ce grâce au trou laissé par la pièce manquante.

3°) Le principe du puzzle peut être étendu à un jeu de coloriage : ce sont alors des pièces de couleur qu'il faut placer aux bons endroits.

4°) Le puzzle négatif. Il faut faire disparaître certaines pièces de la grille pour faire apparaître un dessin.

5°) Enfin, on pourrait imaginer que l'on propose au joueur des pièces et une définition, et ce dernier doit réaliser le dessin demandé à partir des pièces à sa disposition. Par exemple, dessiner une maison, une voiture, ...

FIN DE JEU : Ces jeux se terminent lorsque le dessin initial ou demandé est (re)trouvé.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci seront normalement toutes présentes dans le sabot ou sur la grille (par exemple pour le jeu de "taquet").

1.5. Jeu de mémoire

BUT : Le but de ce jeu est de retrouver des paires (de pièces généralement identiques) dans une grille dont les pièces apparaissent en début de jeu puis se retournent. Chaque fois que le joueur désigne une pièce, celle-ci se retourne faisant apparaître son contenu. Le joueur doit alors choisir parmi les autres pièces toujours retournées, celle qui devrait constituer une paire avec la première choisie. Ce jeu constitue un cas particulier du jeu des associations.

OBJETS : Mêmes types de pièces que pour le jeu des associations.

FIN DU JEU : Lorsqu'il n'y a plus de pièces dans l'aire de jeu et que toutes les paires ont été retrouvées.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci seront toutes présentes, dos tourné vers le joueur.

1.6. Arithmétique et français

BUT : Il s'agit dans ce jeu de compléter des calculs simples ou de compléter des mots ayant des lettres manquantes.

OBJETS : Les lettres de l'alphabet, les nombres et les opérateurs +, - ,* , / .

PRINCIPE : Il vaut pour le jeu de mathématique et de français. Le joueur découvre un mot dans lequel il manque une lettre, il doit retrouver celle-ci pour compléter le mot. De même on lui propose une opération mathématique simple, dans laquelle il manque un terme, et il doit découvrir celui-ci.

EXEMPLES : AV . ON ☞ il manque I

M . ISON ☞ il manque A

4 + . = 7 ☞ il manque 3

FIN DU JEU : Lorsque toutes les expressions sont complétées et correctes.

PRESENTATION DES PIECES AUX JOUEURS : Les pièces manipulables par le joueur pour compléter les expressions seront normalement toutes présentes dans le sabot.

1.7. Remplir les trous

BUT : Le joueur voit apparaître une grille dans laquelle il y a des trous, il dispose de pièces qui vont lui permettre de boucher ceux-ci.

OBJETS : Le joueur reçoit des pièces de formes différentes.

FIN DU JEU : Lorsque tous les trous sont bouchés.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci pourraient être toutes présentes dans le sabot ou bien n'arriver qu'une par une ou par groupes de taille définie.

VARIANTES :

- Les formes sont mélangées avec des formes incorrectes, il faut simplement sélectionner celles qui sont correctes.

- Pour la manipulation des pièces, on peut faire l'hypothèse que le joueur n'a qu'à cliquer sur la pièce pour la sélectionner. Si la pièce choisie est correcte, le trou est bouché, sinon il lui faut continuer à chercher la pièce. On peut aussi imaginer une variante qui consiste à exiger du joueur qu'il choisisse et qu'il déplace la pièce jusqu'au trou, au moyen de la souris.

- Enfin, on pourrait laisser l'initiative au joueur de créer les pièces adéquates. C'est-à-dire qu'en imaginant que les pièces sont constituées de carré de taille identique aux cases de la grille, on laisse la possibilité au joueur de construire à partir de carrés la pièce qui lui paraît appropriée pour boucher le trou.

1.8. Jeu des liens logiques

BUT : On propose au joueur une suite d'objets, d'actions ou de représentations d'événements. Le joueur doit alors choisir parmi un ensemble de pièces, celle qui logiquement vient compléter la liste.

EXEMPLES :

1 2 3 4 il faut 5

a A b B il faut c

🕐 🕑 🕒 🕓 il faut 🕔

OBJETS : Comme on le voit, le contenu des pièces peut être très varié.

FIN DU JEU : Lorsque toutes les suites sont correctement complétées.

PRESENTATION DES PIECES AUX JOUEURS : Les pièces devraient être toutes présentes dans le sabot.

VARIANTES :

- Plutôt que de donner la cinquième pièce logiquement, on pourrait imaginer que le joueur doit trouver une pièce qui est la représentation du point commun entre les quatre pièces présentées.

- On pourrait également imaginer que ce soit non pas la cinquième mais l'une quelconque des cinq pièces qu'il faut retrouver (exemple : 1 2 ... 4 5 et il faut 3).

1.9. Jeux moteurs

BUT : Il s'agit de quelques jeux simples qui ont pour objectif de familiariser le joueur avec le maniement de la souris ou avec le déplacement dans l'espace de l'écran au moyen des flèches du clavier.

JEUX :

1°) Circuit à parcourir avec la souris en partant d'un point A vers un point B.

2°) Un petit mobile se déplace à l'écran et le joueur doit le suivre avec le pointeur de la souris.

3°) L'écran reprend un ensemble de cases, ces cases clignotent les unes après les autres dans un ordre quelconque. Dès qu'une case clignote, le joueur doit se rendre le plus rapidement possible sur celle-ci et doit la sélectionner.

1.10. Mots croisés

BUT : Il existe des jeux de mots croisés élémentaires dans lesquels la définition de chacun des mots est donnée par un dessin. Les mots sont alors constitués par des lettres reprises chacune dans un jeton.

OBJETS : Les lettres de l'alphabet.

FIN DU JEU : Lorsque tous les mots ont été correctement trouvés par le joueur.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci devraient être toutes présentes dans le sabot.

1.11. Jeu des familles

BUT : Ce jeu se rapproche très fort du jeu des associations. La différence est que, pour le jeu des familles, l'on doit regrouper des pièces ayant un point commun déterminé, dans une partie de la grille. Ce jeu a pour objectif de faire distinguer au joueur des classes d'objets.

EXEMPLES : Ce jeu permet, par exemple, de faire distinguer les chiffres et les lettres. On imagine 20 pièces présentées au joueur, 10 comportant des chiffres et 10 autres des lettres. Le joueur doit placer les dix lettres dans une moitié de la grille et les dix chiffres dans l'autre moitié. On peut distinguer les notions de féminin et masculin, les couleurs, les tailles, les objets des différentes pièces de la maison ... . Le nombre des familles est évidemment déterminable et pas seulement limité à deux.

OBJETS : Une infinité de possibilités est offerte en fonction des choix de l'animateur.

FIN DU JEU : Le jeu se termine lorsque les familles sont correctement reconstituées.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci pourraient être toutes présentes dans le sabot ou bien n'arriver qu'une par une ou par groupes de taille définie.

1.12. Suites d'objets a répéter

BUT : Il s'agit d'un jeu assez simple qui fait appel à la mémoire. Il faut répéter une suite d'objets à laquelle vient s'ajouter à chaque fois un nouvel élément.

EXEMPLE :

1 apparaît puis disparaît et il faut répéter 1

1 2 apparaît puis disparaît et il faut répéter 1 2

1 2 3 apparaît puis disparaît et il faut répéter 1 2 3

1 2 3 4 apparaît puis disparaît et il faut répéter 1 2 3 4

OBJETS : Ici aussi le choix des pièces des suites ne sera limité que par l'imagination du concepteur du jeu.

FIN DU JEU : Le jeu se termine lorsque la suite complète à mémoriser est atteinte.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci devraient être toutes présentes dans le sabot.

REMARQUE : Ce jeu peut, par exemple, permettre à un éducateur d'apprendre l'alphabet ou les chiffres à ses élèves. Mais compte tenu de la grande diversité des contenus que les pièces peuvent prendre, on peut imaginer que l'éducateur, par ce jeu de répétitions, puisse apprendre aux enfants des suites d'actions se suivant logiquement et que ceux-ci retiennent difficilement.

1.13. Prendre n éléments parmi x

BUT : On présente un plus ou moins grand nombre de pièces au joueur et celui-ci doit retrouver parmi celles-ci, celles qui appartiennent à l'ensemble voulu par l'éducateur.

EXEMPLE : L'animateur peut, par exemple, noyer toute une série d'ustensiles de cuisine parmi des objets divers et demander à son élève d'en retrouver cinq ou dix.

OBJETS : Ici aussi, les possibilités sont infinies.

FIN DU JEU : Le jeu se termine quand toutes les pièces ou un nombre suffisant (fixé par l'animateur) d'entre elles ont été retrouvées.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci devraient être toutes présentes dans le sabot.

1.14. Souci du détail

BUT : On présente deux dessins desquels on extrait dix détails et il faut retrouver à quel dessin appartient chaque détail.

EXEMPLE : On présente les dessins de deux maisons et on présente dans le sabot des pièces représentant des fenêtres, des cheminées, des morceaux de mur, ...

OBJETS : Les pièces dépendent des dessins de départ.

FIN DU JEU : Le jeu se termine lorsque les détails ont été attribués correctement à chacun des deux dessins.

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci devraient être toutes présentes dans le sabot.

1.15. Portraits robots

BUT : On présente au joueur un visage pour lequel il manque les yeux, les oreilles, le nez, ... On propose également toute une série de pièces représentant des nez, des bouches, des oreilles, ... Le joueur doit placer correctement les attributs du visage qu'il choisi et doit essayer de sélectionner ceux-ci de façon cohérente par rapport au caractère masculin ou féminin du sujet.

OBJETS : Les pièces dépendent du thème choisi par l'éducateur.

FIN DU JEU : Celui-ci se termine avec la reconstitution correcte (emplacement des attributs) et cohérente (par rapport au caractère féminin ou masculin).

PRESENTATION DES PIECES AUX JOUEURS : Celles-ci devraient être toutes présentes dans le sabot.

VARIANTE : Le thème peut être quelconque, on peut imaginer que l'animateur propose la forme d'une maison et que l'enfant soit invité à replacer les portes, les fenêtres, ...

REMARQUE : Ce jeu se distingue du précédent par le fait que le joueur doit faire preuve d'imagination et non plus d'esprit d'observation.

2.
Propriétés et concepts issus de l'analyse des jeux

2.1. Présentation

2.1.1. Les concepts

Dans le chapitre précédent, nous décrivons des jeux retenus pour leur caractère plus ou moins éducatif et pour leurs caractéristiques relativement communes : l'emploi d'une grille et parfois d'un sabot et la manipulation de pièces. Tous ces jeux, malgré leurs points communs, possèdent un grand nombre de particularités. Certains utilisent des pièces ayant la même taille que les cases de la grille, d'autres pas. Ces mêmes pièces sont parfois retournables, superposables, éliminables, ... Certains jeux utilisent une grille divisée en plusieurs parties. La liste est assez longue.

Outre les concepts de pièce, de grille et de sabot, déjà retenus, l'analyse menée, sur base de la description de chacun des jeux, nous a permis de mettre en évidence de nouveaux concepts tels que le mode d'association des pièces, la validation des actions du joueur et les conditions de fin du jeu. Ces nouveaux concepts ont un rapport direct avec les règles de déroulement et de fin des jeux, c'est à dire l'aspect plus dynamique des jeux.

Ce deuxième chapitre présente et définit l'ensemble de ces concepts.

2.1.2. Démarche

Dans un premier temps, nous avons répertorié et étudié, lors de l'analyse, l'ensemble de toutes les caractéristiques des jeux. Notre objectif fut de déterminer l'utilité et l'emploi possible de chacune de ces caractéristiques avec constamment à l'esprit la nécessité de déterminer le paradigme. Ensuite, nous avons réalisé une synthèse de tous ces éléments pour aboutir au contenu final du paradigme unificateur. Cette synthèse a été menée en ne reprenant que les

éléments nécessaires et suffisants pour permettre une définition complète de chacun des jeux retenus pour l'analyse.

2.2. Les objets manipulés (pièces, jetons, pions, ...)

Cette section reprend toutes les caractéristiques pouvant être assignées aux objets manipulés dans les jeux. Elles peuvent être ou non présentes dans certains jeux et peuvent également être combinées.

Nous avons fait remarquer que d'un jeu à l'autre, le vocabulaire employé pour désigner les objets manipulés par le joueur peut varier. Certains jeux emploierons le terme "jeton", d'autres le terme "pièce", et d'autres encore le terme "pion". Nous avons décidé de fixer le vocabulaire.

La grille est divisée en cases carrées de taille identique. Dans notre travail, nous appellerons "jeton" tout objet dont la taille est égale à la taille d'une case. Par exemple, dans le jeu des tuyaux, les objets manipulés sont des jetons. Nous appellerons "pièce" tout objet dont la taille est égale à au moins deux cases. C'est le cas, par exemple, des objets employés dans le jeu des dominos.

2.2.1. Retournable

Cette caractéristique apparaît pour certains objets qu'il est possible de retourner face contre grille, de façon à en dissimuler le contenu tout en conservant à l'écran une trace visible de leur présence : leur dos. C'est le cas du jeu de mémoire où le dos des jetons est visible jusqu'à ce que le joueur clique dessus.

2.2.2. Superposable

Il peut être intéressant de pouvoir superposer des objets. Des objets possédant cette caractéristique peuvent recouvrir ou être recouverts par un (plusieurs) autre(s), dissimulant de cette façon tout ou une partie seulement du contenu du (des) objet(s) recouvert(s).

2.2.3. Interchangeable

Il s'agit d'un objet qui, bien qu'ayant un contenu différent d'un autre objet, peut jouer, dans le cours d'un jeu, le même rôle que ce dernier. Ces deux objets peuvent alors prendre chacun la place de l'autre sans modifier le résultat de la validation.

2.2.4. Peut tourner sur lui-même (pivotable)

Objet sur lequel le joueur peut appliquer une rotation d'un certain nombre de quarts de tours.

2.2.5. Mobile dans la grille

Objet pouvant se déplacer dans la grille selon un parcours fixé par l'éducateur ou par le joueur, si l'éducateur lui en laisse la liberté. Cette possibilité est utile pour un jeu de type moteur consistant à proposer à l'enfant un petit mobile se déplaçant sur l'écran et qu'il doit suivre avec le curseur de la souris.

2.2.6. Peut être repris et déplacé dans la grille

Objet qui, bien qu'ayant déjà été joué, peut être repris par le joueur et placé ailleurs sur la grille. C'est utile si, par exemple, le joueur se rend compte d'une erreur et qu'il veut la corriger.

2.2.7. Réalisable par le joueur

Objet qui peut être créé par le joueur. Ceci permet, par exemple, de réaliser une des variantes du jeu "remplir les trous" dans laquelle le joueur doit construire les pièces dont il a besoin.

2.2.8. En contact avec au moins un autre

Un tel objet ne peut être positionné dans la grille que s'il est mis en contact avec au moins un objet déjà présent dans celle-ci.

2.3. La grille

La grille constitue l'espace de jeu. Elle sera donc obligatoirement présente dans tous les jeux. Nous avons retenu deux possibilités de découpes qui sont mutuellement exclusives.

2.3.1. Découpe en zones libres

La grille peut être découpée en zones de telle sorte qu'il soit possible d'établir une distinction entre les différentes associations qui y seront réalisées. Le jeu des familles est une

illustration de cette option. Chaque famille à retrouver est regroupée dans une zone distincte de la grille. Le joueur est libre de choisir les zones.

Si on désire un contrôle automatique du contenu des zones, il faut tenir compte des trois situations suivantes.

Tout d'abord supposons que le joueur place des représentants de plusieurs familles dans une même zone, il faut choisir la famille dont les objets seront conservés.

Supposons une deuxième situation où le joueur place des objets dans une zone de sorte que deux familles sont les seules représentées et qu'elles possèdent un même nombre de représentants. Il faut éliminer l'une des deux.

Une dernière situation doit être évitée. Il s'agit du cas où les représentants d'une même famille apparaissent un nombre de fois maximum mais dans plusieurs zones. Il faut se résoudre à ne les conserver que dans une seule des zones.

Ces situations imposent de poser trois hypothèses fortes. La première est que la ou les caractéristique(s) qui apparai(ssen)t en majorité dans une zone de l'écran correspond(ent) à la famille de jetons que le joueur désire associer dans celle-ci. Cette hypothèse est cruciale si l'on désire une vérification automatique suivie d'une élimination automatique des jetons non correctement associés.

La deuxième hypothèse permet de départager des familles de jetons dont la (les) caractéristique(s) apparai(ssen)t simultanément un nombre de fois maximum et égal dans une zone. Il faut se résoudre à en éliminer une des deux. Arbitrairement, nous avons choisi de conserver la (les) caractéristique(s) apparue(s) le plus tôt dans la zone en partant du coin supérieur gauche de la zone, en se déplaçant vers la droite, ligne par ligne dans la zone.

La troisième hypothèse impose que si une caractéristique apparaît un nombre de fois maximum dans plusieurs zones différentes, les objets de la zone vérifiée en premier seront conservés. Les objets des autres zones seront éliminés au profit des objets correspondant à la (les) caractéristique(s) apparue(s) un nombre de fois juste inférieur à la première caractéristique citée.

2.3.2. Découpe en zones fixées

Cette option se rapproche, par son objectif, de la découpe précédente. Elle en diffère par le fait qu'il y a au moins, dans chaque zone de la grille, une caractéristique pré définie par le concepteur au moment de la définition du jeu. Toujours pour le jeu des familles, le fait de

regrouper les pièces dans telle ou telle zone de l'écran est déterminé par la présence avant le début du jeu d'une pièce de cette famille dans chacune des zones.

La différence avec le point précédent est marquée par le fait que dans le premier regroupement, c'est le joueur qui fixe la caractéristique que les pièces devront avoir dans cette zone, à partir du moment où il dépose la première pièce dans la zone. Par contre, dans cette deuxième découpe, c'est le concepteur du jeu qui fixe cette caractéristique.

2.3.3. Contenu

Dans certains jeux, il est intéressant, avant le début du jeu, de pouvoir couvrir tout ou une partie de la grille par un dessin ou par quelques objets.

2.4. Le sabot

Nous avons déjà donné une définition du sabot. Soulignons qu'il est optionnel. Dès lors, il est nécessaire de pouvoir déterminer sa présence ou non, car certains jeux, tel le jeu de mémoire, n'en requièrent pas.

Il faut également en définir l'organisation. On peut imaginer que tous les objets soient présentés dans le sabot dès le début du jeu, ou qu'une partie seulement soit présente. On peut également imaginer que tous les objets seront présentés à plat ou qu'il seront organisés en piles.

2.5. La zone d'attente

Cette partie de l'écran permet au joueur de sélectionner et donc de stocker un objet du sabot ou de la grille dans une partie autre que l'une de ces deux dernières.

L'intérêt d'une telle zone intermédiaire apparaît si, par exemple, la grille et le sabot sont remplis et que le joueur désire voir apparaître un nouvel objet dans le sabot. Il peut alors sélectionner un objet dans le sabot et le placer dans cette zone d'attente. Il libère ainsi une place pour l'arrivée d'un nouveau jeton. Cette zone du jeu est vide au début de la partie.

Le concepteur du jeu devra en déterminer l'emplacement et la taille s'il désire en utiliser une dans son jeu.

2.6. La zone de résultat

Cette zone de l'écran permet de stocker les objets correctement joués par le joueur et validés comme tels.

Si le concepteur désire que de tels objets disparaissent de la grille pour libérer de la place pour les mouvements futurs du joueur, il peut décider de l'élimination de ceux-ci. Plutôt que de les voir disparaître définitivement de l'écran, le concepteur peut décider que ces objets seront stockés (temporairement ou jusqu'à la fin du jeu) dans cette zone pour signaler au joueur les objets correctement joués jusqu'à ce point du jeu. Cette zone du jeu est vide au début de la partie.

Le concepteur du jeu devra en déterminer l'emplacement et la taille s'il désire en employer une dans son jeu.

2.7. La zone de message

Cette zone reprendra les messages que le concepteur voudra communiquer de façon permanente au joueur.

Cette partie de l'écran comprendra, par exemple, un taux d'erreur, un temps écoulé, des directives, ...

Remarquons que les messages en début (par exemple pour présenter ou rappeler les règles au joueur), en cours (différents messages d'encouragement ou de félicitations) et en fin de jeu pourront être communiqués via les boîtes de message offertes par l'interface "Windows". Ces messages étant liés à l'évolution du jeu, il n'est pas nécessaire qu'ils apparaissent dans la zone de message.

Le concepteur du jeu devra en déterminer l'emplacement, la taille et le contenu, s'il désire en utiliser une dans son jeu.

2.8. Les règles de jeux

Dans cette section, nous traitons l'aspect dynamique des jeux : leurs règles.

Tout jeu se compose globalement de pièces ou de jetons, d'un espace de jeu, d'objectifs à atteindre et de règles qui régissent le déroulement de toute partie, de son coup d'envoi à sa fin (objectif(s) atteint(s) ou non). Les règles constituent donc l'élément principal conditionnant le

déroulement du jeu. L'intérêt que tout joueur trouvera dans un jeu dépendra du contenu de ces règles.

Dans la réalité, avant de débuter la partie, les joueurs doivent, si ce n'est déjà fait, prendre connaissance des règles et doivent les appliquer pendant le cours du jeu.

Pour notre travail, une transposition d'un tel principe au domaine des personnes handicapées est difficilement envisageable. En raison des difficultés qu'elle rencontre, cette personne aura toutes les peines du monde à se conformer exactement aux règles dans leur ensemble (oubli de certaines règles, mauvaise compréhension ou application de celles-ci, mauvaises manipulations, ...). Le jeu va rapidement devenir inintéressant et l'objectif que le concepteur lui aura assigné ne sera certainement jamais atteint, à moins que ce dernier ne soit présent (ce qui vide l'outil informatique de tout intérêt).

Il faut donc offrir à l'éducateur un outil de définition de règles qui seront prises en charge par l'ordinateur et qui lui assureront le contrôle des comportements du joueur. De cette façon, le jeu conservera tout son intérêt, le concepteur ne devra pas être systématiquement présent pour contrôler l'application et le respect des règles, et l'objectif pédagogique attendu sera certainement atteint dans une beaucoup plus grande mesure.

A première vue, la définition de règles et leur application par l'ordinateur semblent être une tâche plutôt complexe, qui nous obligerait à passer par la définition d'un langage que le concepteur aurait à assimiler et maîtriser pour définir les règles des jeux. La détermination d'un langage, fut-il élémentaire, met en évidence un certain nombre de difficultés : sa définition, une certaine complexité à l'emploi, un apprentissage inévitable de la part du concepteur, ... Dans notre souci de vouloir offrir un outil simple d'utilisation, le passage par la définition d'un pseudo-langage n'est donc pas chose évidente car il faut définir une syntaxe, une sémantique, une structure de programmation, ... qui devront être assimilées par le concepteur des jeux.

En réponse aux difficultés qu'une telle entreprise pourrait entraîner, nous sommes partis de l'idée qu'il serait possible de déterminer un mode de définition des règles qui soit aussi simple que celui qui permettra de définir les caractéristiques des objets. Si on imagine que la définition des objets peut se faire au moyen d'un formulaire dans lequel il suffit de cocher des choix, on pourra inclure dans celui-ci une section où les règles pourront être définies avec une égale simplicité.

2.8.1. Les règles d'association

Comparé à un pseudo-langage, un tel mode de définition des règles peut apparaître simpliste. Cependant, si l'on a à l'esprit les caractéristiques des jetons et des pièces, et si l'on regarde attentivement la famille de jeux que nous avons considérée, on constate que les objets manipulés sont la plupart du temps associés, soit en fonction des caractéristiques de leur contenu (association, dominos, ...), soit en fonction de leur position dans la grille (puzzle, mots croisés, ...), soit en fonction de leurs côtés (réseau de tuyaux). Les règles du jeu peuvent alors se résumer en un premier groupe de trois règles, que nous avons intitulées règles d'association : **association par caractéristiques** (en fonction du contenu des jetons ou des pièces); **association par côtés**; **association par position** (par rapport aux coordonnées dans la grille)

■ **Association par côtés** : Il est possible de définir le (les) côté(s) d'un objet qui peut (peuvent) être accolé(s) à un (plusieurs) autre(s). Par exemple, un jeu tel les réseaux de tuyaux met en oeuvre des jetons qui doivent être accolés de façon à ce que les dessins contenus forment un réseau.

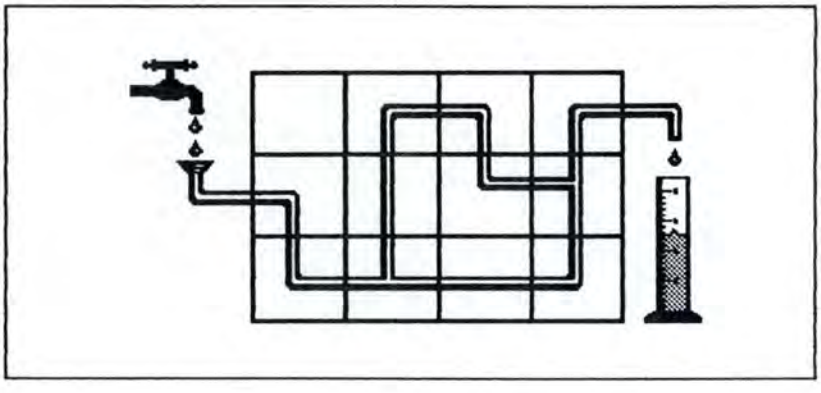


figure 2.1.

Dans notre exemple, le réseau est composé de douze jetons. Ils doivent être accolés de façon à ce que le morceaux de tuyaux qu'ils représentent soient joints. Il faut éviter que les côtés communs de deux jetons accolés ne soient pas compatibles. C'est le cas quand l'un présente un morceau de tuyau et l'autre pas.

Ci-dessous, les deux premiers exemples sont valides, le troisième ne l'est pas.

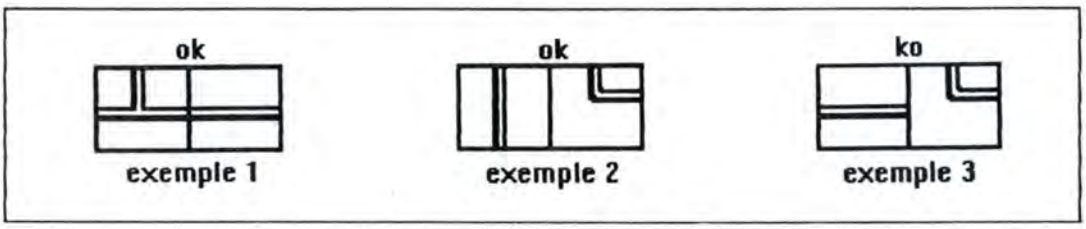


figure 2.2.

Si on peut désigner les quatre côtés d'un jeton et leur associer une valeur, il suffit d'assigner la valeur "ok" à un côté où aboutit un morceau de tuyau et "ko" dans le cas contraire. Lorsque deux côtés simultanément "ok" ou "ko" seront mis en communs, l'association sera valide. Dans le cas contraire, elle sera invalide

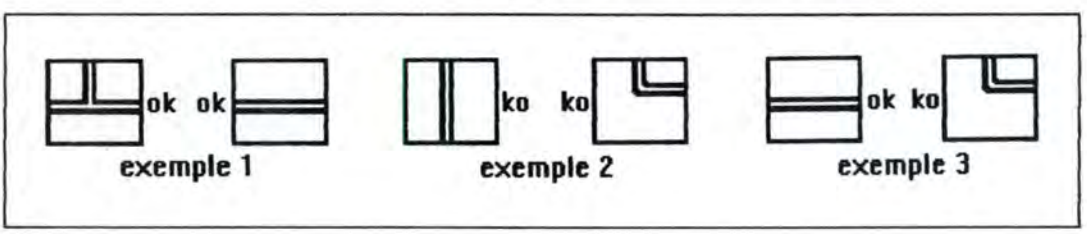


figure 2.3.

■ **Association par caractéristiques** : A chaque jeton peut être associé une liste de caractéristiques le décrivant. Cette liste est constituée par l'éducateur selon sa perception du jeton et de l'usage qu'il veut en faire. Les caractéristiques sont par exemple la couleur, la forme, la taille, la place dans un ordre, la famille. Une fois la liste déterminée, l'éducateur doit choisir sur base de quelle(s) caractéristique(s) les associations s'établiront.

Exemples :

- liste de caractéristiques associées à un jeton contenant le dessin d'une fourchette : [bleue, fourchette, grande, premier, table]

  - jeu d'associations : il faut appareiller les jetons de couleur bleue, peu importe leur contenu;

  caractéristique choisie : la première.

  - jeu des familles : il faut regrouper les jetons représentant les objets qui se trouvent sur la table lors des repas;

  caractéristique choisie : la dernière.

  - jeu des liens logiques : même principe que le jeu précédent mais, en plus, il faut replacer l'assiette, le couteau, la fourchette et la cuillère dans leur ordre

de disposition sur la table.
caractéristiques choisies : la dernière pour la sélection et la quatrième avec une relation d'ordre.

■ **Association par position** : Un jeton peut être associé à au moins (au moins pour assurer l'interchangeabilité) une position sur la grille. Il ne pourra dès lors occuper qu'une seule des places qui lui sont assignées sur celle-ci. Toute autre place occupée sur la grille sera non valide.

Par exemple, pour le jeu du puzzle, une fois le dessin réalisé, on peut imaginer une découpe automatique qui fournirait autant de jetons qu'il y a de cases composant la grille. A chacun de ces jetons peuvent être assignées les coordonnées de la case où il se trouvait suite à la découpe. Le joueur doit recomposer le dessin. Pour valider ses actions, il suffit de vérifier que les jetons sont replacés dans la grille de telle façon que les coordonnées leur avaient été assignées au moment de la découpe correspondent aux coordonnées des cases où ils ont été placés par le joueur.

Ces règles doivent s'accompagner de deux autres types de règles : la **règle de validation** et les **règles de déplacement**.

2.8.2. La règle de validation

Elle permet au concepteur de déterminer les moments où des vérifications de la correction des associations des objets par le joueur seront réalisées, et de définir les actions qui accompagneront cette vérification.

Les paramètres de cette règle sont les suivants.

■ **moment** : Le concepteur doit déterminer à quels instants du déroulement du jeu, la validation doit être effectuée; elle peut être activée après x coups joués, après x secondes ou minutes, à la demande du joueur, lorsqu'il n'y a plus d'objet dans le sabot, ...

■ **actions liées** : La vérification terminée, certains événements peuvent survenir pour signaler au joueur l'état d'avancement du jeu et le résultat de ses actions depuis la dernière validation, ou pour réorganiser les pièces ou depuis le début du jeu (si c'est la première validation). Quelques exemples d'événements : message d'encouragement ou de félicitations, retrait des pièces correctement jouées accompagné d'un message de félicitations, retour des pièces incorrectement jouées vers le sabot avec message d'explication, ...

■ **taux d'erreur** : Après chaque validation, on peut modifier la valeur d'un compteur qui prend en considération les erreurs commises depuis le début de la partie. Au-delà d'un nombre d'erreurs fixé le jeu pourrait se terminer, un message pourrait avertir le joueur, ... Le joueur pourrait d'ailleurs être averti de l'état de ce compteur après chaque validation.

2.8.3. Les règles de déplacement

Les règles de déplacement déterminent les mouvements (des pièces ou des jetons lors de leur manipulation) autorisés entre ou au sein des zones de l'écran. Ces règles correspondent à la liberté offerte au joueur de sélectionner une pièce ou un jeton dans une zone de l'écran et de la placer dans une autre.

Le graphe qui suit représente les déplacements de pièces que peuvent réaliser les joueurs.

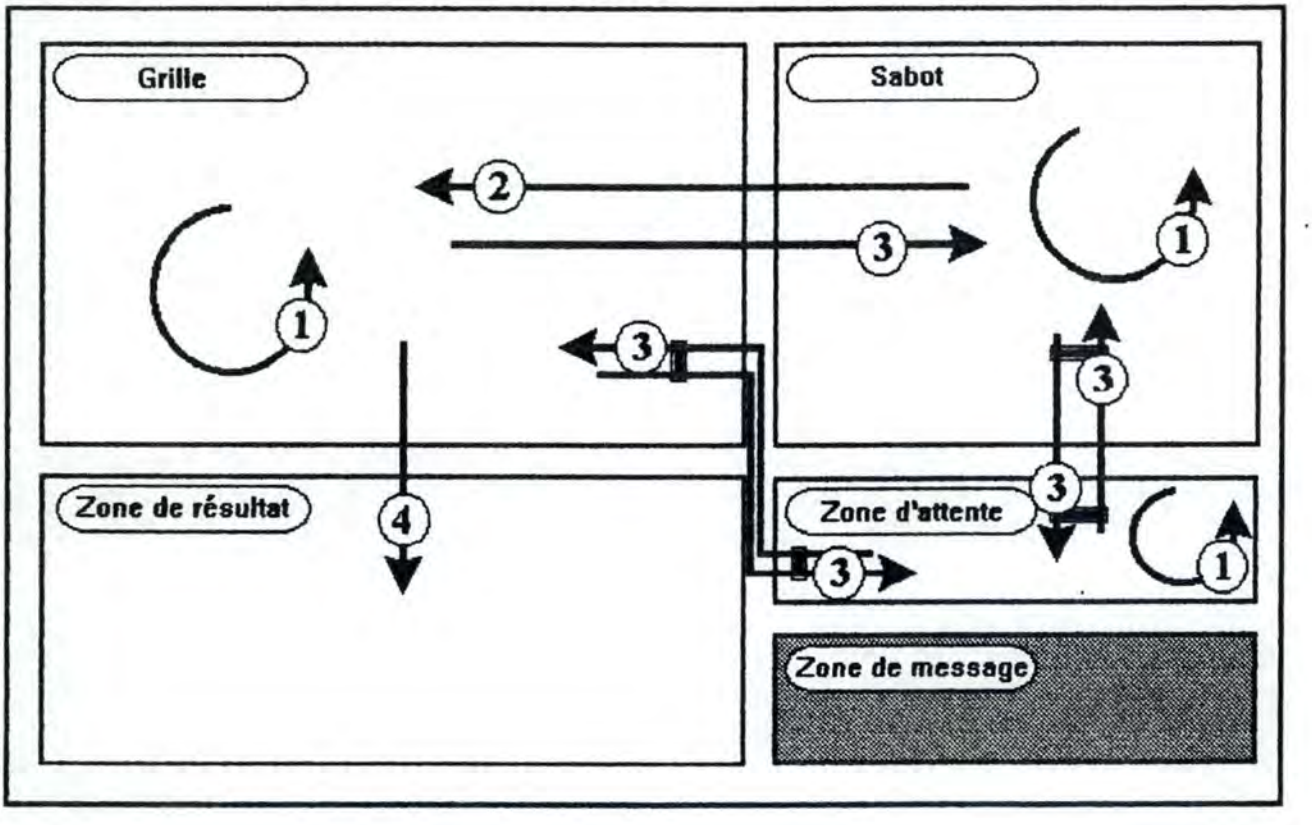


figure 2.4.

■ ① Il s'agit du déplacement d'une pièce à l'intérieur d'une même zone. Cette possibilité n'apparaît pas dans la zone de résultat où il n'est pas possible de sélectionner les pièces du simple fait qu'il s'agit de pièces mises hors du jeu, car elles ont déjà été validées et déclarées correctement associées. Le concepteur a la

liberté d'accorder ou non ces déplacements ou d'en accorder certains et pas d'autres.

■ ② C'est le seul déplacement qui est obligatoire dans le cas où un sabot est défini. En effet, si ce déplacement était optionnel et que le concepteur ne le choisissait pas, alors qu'il a déterminé l'existence d'un sabot, il serait impossible au joueur de sélectionner les jetons dans le sabot et de les positionner sur la grille.

■ ③ Ce sont des déplacements que le concepteur peut ou non autoriser.

Pour éviter une incohérence qui résulterait du choix du déplacement de la grille et du sabot vers la zone d'attente sans sélectionner la possibilité d'en sortir, (entraînant que le joueur pourrait placer ses jetons ou pièces dans la zone d'attente où ils seraient perdus faute de pouvoir les récupérer), les déplacements concernant la zone d'attente seront automatiquement définis dans les deux sens, ce qui est représenté graphiquement par le lien unissant les deux flèches.

■ ④ Ce déplacement est optionnel. Le concepteur peut en effet décider que les jetons valides pourront rester sur la grille ou devront disparaître de l'écran. Ce déplacement, s'il est choisi, se fera de façon automatique après chaque validation pour les pièces jugées correctement associées. Le joueur n'interviendra donc jamais. S'il tente de placer lui-même une pièce dans la zone de résultat, il faudra le prévenir de cette interdiction et replacer la pièce à sa position d'origine. Remarquons que la même interdiction vaut pour la zone de message où le joueur ne pourra pas plus que dans la zone de résultat placer des pions.

On obtient donc les situations suivantes, parmi lesquelles le concepteur des jeux pourra choisir : dans la grille, dans le sabot, dans la zone d'attente, de la grille vers le sabot, entre la grille et la zone d'attente, entre le sabot et la zone d'attente.

Pour le déplacement au sein de la grille, une contrainte pourra être imposée par le concepteur sur l'importance du déplacement. Le joueur ne pourra déplacer sa pièce au-delà d'une distance déterminée en nombre de cases.

Remarquons que le déplacement de la grille vers la zone de résultat n'apparaît pas ici car il ne s'agit pas d'une règle de déplacement à offrir au joueur. Nous l'avons repris comme l'un des événements possibles accompagnant la fin de la validation.

Remarquons, pour terminer, que les déplacements d'une zone à l'autre n'ont de sens que si les zones sont définies par le concepteur. Il faudra donc veiller, lorsque l'animateur définira

un jeu, à éviter des incohérences du genre "déplacement de la zone d'attente à la grille autorisé" alors qu'une zone d'attente n'a pas été définie.

2.9. Règles de fin de jeu

Généralement, pour la famille de jeux qui nous intéresse, on constate que le jeu se termine lorsque toute la grille est remplie d'objets bien associés, ou lorsque celle-ci est vide, ou lorsque la grille et le sabot sont vidés de leur contenu, ... Cela revient donc à vérifier que certaines zones de l'écran sont dans un certain état et que les objets qu'elles contiennent (si elles en contiennent) sont toutes bien associées.

Mais ce n'est pas tout. Le concepteur du jeu peut vouloir limiter la durée de la partie. Nous lui offrons donc la possibilité de fixer un temps limite ou un nombre d'actions limité.

Le concepteur d'un jeu aura donc à définir l'état final qu'une ou plusieurs zone(s) de l'écran devront posséder et une limite de temps ou de coups.

2.9.1. Grille, sabot, zone d'attente

Pour ces trois zones, le concepteur doit déterminer si elles doivent être vides ou remplies à la fin du jeu (et ce indépendamment les unes des autres), ou si cela n'a aucune importance.

2.9.2. Temps limite et nombre de coups limite

En déterminant ces valeurs, le concepteur du jeu pourra limiter la durée de jeu.

2.10. La manipulation et la sélection des objets

2.10.1. Sélections et déplacements

Nous avons pris en compte deux modes pour la manipulation des objets : les sélections (par exemple, au moyen de la souris : par un simple clic de celle-ci sur un objet) et les déplacements (toujours au moyen de la souris : clic, bouton enfoncé sur un objet, mouvement de la souris qui est imprimé à l'objet, relâchement du bouton de la souris pour désigner le point d'encrage de l'objet).

A partir de ces deux modes de manipulation, on peut envisager les trois possibilités suivantes : seules des **sélections** sont autorisées; seuls les **déplacements** sont permis; les **sélections** et les **déplacements** sont acceptés.

2.10.2. Notions d'action valide et de coup joué

Une action valide correspond à une action autorisée par rapport au mode de manipulation et aux règles de déplacement. Par exemple, déplacer un jeton de la grille vers le sabot alors que ce déplacement est autorisé et que le mode de manipulation est du type "déplacements" constitue une action valide.

Un coup joué correspond à une action valide qui en plus permet d'atteindre l'objectif final du jeu. Par exemple, dans le jeu des associations, prendre un jeton dans le sabot ou dans la zone d'attente et le placer dans la grille constitue un coup joué.

2.10.3. En résumé ...

En croisant les modes de manipulation et les notions d'action valide et de coup joué, on obtient : (nous ne considérons, ici aussi, que des manipulations réalisées au moyen de la souris)

- **si sélections seules :** un coup joué est réalisé par un simple clic au moyen de la souris sur un jeton non encore sélectionné de la grille; les déplacements sont interdits; cliquer sur un objet déjà sélectionné le désélectionne, on obtient donc le retrait d'un coup joué; toute pression de la souris en dehors de la grille n'a aucun effet.

- **si déplacements seuls :** un coup joué est réalisé en prenant un objet dans la zone d'attente ou le sabot et en le plaçant dans la grille; la sélection n'a aucun effet; le fait de prendre un objet dans la grille et de le replacer à un autre endroit de la grille ne constitue pas un coup joué; le fait de sortir un objet de la grille correspond au retrait d'un coup joué.

- **si les deux modes sont autorisés :** un coup joué est réalisé soit en cliquant sur un objet de la grille non encore sélectionné, soit en prenant un objet en dehors de la grille et en le posant dessus ; les commentaires des deux points ci-dessus sont valables.

3. Conception des jeux

3.1. Le paradigme

L'analyse qui a précédé nous a permis de dégager un ensemble de concepts et de caractéristiques attachées à ces concepts. Ceux-ci constituent notre paradigme unificateur.

3.2. L'outil de conception

Notre étude nous conduit à déterminer un outil permettant à tout éducateur de concevoir des jeux, avec la volonté que cet outil soit simple d'emploi et efficace. En dressant un tableau à partir de tous ces concepts, nous pouvons constituer un tel outil. Il se présente comme une sorte de formulaire dans lequel le concepteur peut introduire les valeurs qu'il désire en fonction du jeu à définir. Nous considérons qu'il s'agit là d'un outil simple, de taille raisonnable, apparaissant complet. De plus, le concepteur de jeux n'aura, nous semble-t-il, que peu de choses à assimiler, et l'apprentissage et la maîtrise de ce moyen de définition devraient être rapides.

Le tableau qui suit, à la page suivante, synthétise tous les concepts et leurs caractéristiques. La première colonne reprend les intitulés des concepts, la seconde liste les caractéristiques correspondant à ceux-ci et la troisième colonne présente le type de valeurs qu'il faudra introduire pour donner une définition aux concepts.

CONCEPTS	CARACTERISTIQUES	VALEURS
Objets	- pièces ou jetons : ... - caractéristiques : ... - côtés associables : ... - position départ: ... - position(s) d'arrivée: ... - retournables : ... - représentation: ... - superposables : ... - interchangeables : ... - pivotables : ... - mobiles dans la grille : ... - reprise(s) et déplacée(s) dans la grille : ... - réalisables par le joueur : ... - en contact avec au moins un (une) autre : ...	- pièces ou jetons - définition des caractér. - désignation des côtés - couple de coordonnée - couple(s) de coord. - oui ou non - face cachée, ombrée, ... - oui ou non - oui ou non - oui ou non - oui ou non - oui ou non - oui ou non - oui ou non
Règles d'association des pièces	- en fonction de leurs côtés : ... - par rapport à leurs caractéristiques : ... - par positions : ...	- oui ou non - oui ou non - oui ou non
Règle de validation	- quand ? : ... - événement fin de la validation : ... - taux d'erreur accepté : ...	- moment/ valeur entière - choix dans une liste - valeur entière
Règles de déplacement	- dans la grille : ... - dans le sabot : ... - dans la zone d'attente : ... - de la grille vers le sabot : ... - entre la grille et la zone d'attente : ... - entre le sabot et la zone d'attente : ...	- oui ou non + valeur ent. - oui ou non - oui ou non - oui ou non - oui ou non - oui ou non
Manipulation	- mode : ...	- sel/depl/sel-depl
Fin du jeu	- grille : ... - sabot : ... - zone d'attente : ... - temps limite : ... - nombre de coups limite : ...	- vide/pleine/non pertinente - vide/plein/non pertinent - vide/pleine/non pertinente - valeur entière - valeur entière
Grille	- découpée en zones libres : ... - découpée en zones fixées : ...	- liste de zones - liste de zones avec caractéristique(s)
Sabot	- présent ou non présent : ... - caractéristiques et organisation : ...	- oui ou non - une des possibilités
Zone de résultat	- présente ou non présente : ...	- oui ou non
Zone d'attente	- présente ou non présente : ...	- oui ou non
Zone de message	- présente ou non présente : ... - contenu : ...	- oui ou non - message/score

3.3. Commentaires

▪ Pour chaque **objet**, le concepteur devra définir s'il s'agit d'une pièce ou d'un jeton. Il devra également déterminer au moins une caractéristique, ou les côtés associables, ou une ou plusieurs position(s) que l'objet devra occuper dans la grille, en fonction du type d'association choisi.

La représentation concerne l'aspect du jeton. Il peut clignoter, être caché, ... dans telle ou telle situation.

Les autres caractéristiques des objets sont choisies par une simple affirmation ou négation se rapportant à leur emploi. Ces dernières sont cumulables.

▪ Les **règles d'association** sont déclarées actives (ou non). Remarquons que les deux premières sont conjugables.

▪ Pour la **règle de validation**, voici la liste des événements dans laquelle le concepteur peut choisir :

- si incorrect :
  - retour vers le sabot des objets mal joués;
  - retour vers la zone d'attente des objets mal joués;
  - les objets ne bougent pas, mais leur représentation les distinguent des objets correctement associés (ils se retournent ou clignotent, ...);
  - les objets incorrectement associés sont éliminés;
  - un message est communiqué;
  - un score est mis à jour;
  - un taux d'erreur est mis à jour;

- si correct:
  - déplacement des objets correctement associés vers la zone de résultat;
  - les objets ne bougent pas, mais leur représentation les distinguent des objets non encore correctement associés(ils se retournent ou clignotent, ...);
  - les objets correctement associés sont éliminés;
  - un message est communiqué;
  - un score est mis à jour;
  - un taux d'erreur est mis à jour;

Remarquons que plusieurs événements pourront être combinés.

▪ Les **règles de déplacement** sont conjugables. Le déplacement au sein de la grille possède un attribut qui permet de limiter le déplacement de l'objet en nombre de cases.

▪ La **manipulation** sera soit du type sélection (sel), soit du type déplacement (depl), soit du type sélection et déplacement (sel-depl).

▪ La définition des zones pour **la fin du jeu** est réalisée en les déclarant vides, pleines ou non pertinentes. "Vide" signifie que la zone devra être vidée de tout objet pour que la fin du jeu soit vérifiée. "Plein(e)" joue le rôle inverse. "Non pertinent(e)" signifie que cette zone n'entre pas en ligne de compte pour vérifier que la fin du jeu est atteinte. Remarquons qu'il n'y a aucun lien entre les définitions des états des trois zones.

▪ L'organisation du **sabot** revient à définir s'il y a des piles ou non, si tous les objets sont présents ou non au départ (pour cause de taille du sabot), la fréquence et la taille des séquences d'arrivée des objets s'il ne sont pas tous présents sur le sabot au départ ... .

3.4. Retour aux jeux

L'outil de conception que nous venons de définir pouvant paraître simpliste, nous proposons de vous présenter la définition, au moyen de cet outil, de quelques jeux pris parmi ceux que nous avons retenus au chapitre deux. Nous avons choisi le jeu de mémoire, le jeu de "taquet" (une des variantes du jeu de puzzle) et un jeu de familles ou les caractéristiques des zones de la grille sont fixées par le concepteur. Remarquons que tous les autres jeux sont également définissables avec autant de succès et de simplicité.

Dans les tableaux qui suivent, nous n'avons repris que les éléments nécessaires à la définition de chacun de ces jeux.

3.4.1. Jeu de mémoire

CONCEPTS	CARACTERISTIQUES	VALEURS
Objets	- pièces ou jetons : ... - caractéristiques : ... - retournables : ...	- jetons - définition des caractér. - oui
Règles d'association des pièces	- par rapport à leurs caractéristiques : ...	- oui

Règle de validation	- quand ? : ... - événement fin de la validation : ... - taux d'erreur accepté : ...	- toutes les deux sélections - ▪ si ok élimination des jetons sélectionnés ▪ si ko les jetons se retournent
Règles de déplacement	- aucune	
Manipulation	- mode : ...	- sel
Fin du jeu	- grille : ...	- vide
Grille	- une zone unique	
Sabot	- présent ou non présent : ... - caractéristiques et organisation : ...	- non - non pertinent
Zone de résultat	- présente ou non présente : ...	- oui ou non (selon que l'on veuille ou non conserver une trace des jetons correctement joués)
Zone d'attente	- présente ou non présente : ...	- non
Zone de message	- présente ou non présente : ... - contenu : ...	- oui - message, score

3.4.2. Jeu de taquet

CONCEPTS	CARACTERISTIQUES	VALEURS
Objets	- pièces ou jetons : ... - position : ... - reprises et déplacées dans la grille : ...	- jetons - une coordonnée par jeton - oui ou non
Règles d'association des pièces	- par positions : ...	- oui
Règle de validation	- quand ? : ...	- après chaque action
Règles de déplacement	- dans la grille : ...	- oui; d'une seule case à la fois
Manipulation	- mode : ...	- déplacement
Fin du jeu	- grille : ...	- pleine (à une case vide près)
Grille	- une zone unique	
Sabot	- présent ou non présent : ...	- non
Zone de résultat	- présente ou non présente : ...	- non
Zone d'attente	- présente ou non présente : ...	- non
Zone de message	- présente ou non présente : ... - contenu : ...	- oui - message

3.4.3. Jeu des familles en zones fixées

CONCEPTS	CARACTERISTIQUES	VALEURS
Objets	- pièces ou jetons : ... - caractéristiques : ... - reprises et déplacées dans la grille : ...	- jetons - définition des caractér. - oui ou non
Règles d'association des pièces	- par rapport à leurs caractéristiques : ...	- oui
Règle de validation	- quand ? : ... - événement fin de la validation : ...	- après quelques coups - ▪ si ok rien ▪ si ko retour des jetons vers la zone d'attente (uniquement car le sabot est organisé en piles)
Règles de déplacement	- dans la grille : ... - dans le sabot : ... - dans la zone d'attente : ... - de la grille vers le sabot : ... - entre la grille et la zone d'attente : ... - entre le sabot et la zone d'attente : ...	- oui - oui - oui - oui - oui - oui
Manipulation	- mode : ...	- déplacement
Fin du jeu	- grille : ... - sabot : ... - zone d'attente : ... - temps limite : ... - nombre de coups limite : ...	- pleine - vide - vide - valeur entière - valeur entière
Grille	- découpée en zones fixées : ...	- liste de zones avec caractéristique(s)
Sabot	- présent ou non présent : ... - caractéristiques et organisation : ...	- oui - organisé en piles
Zone de résultat	- présente ou non présente : ...	- non
Zone d'attente	- présente ou non présente : ...	- oui
Zone de message	- présente ou non présente : ... - contenu : ...	- oui - message

4.
Aspects statiques et dynamiques des concepts

Avant d'aborder la partie informatique de notre travail, nous avons repris de façon systématique l'ensemble des concepts, afin d'en dégager leurs aspects dynamiques et statiques, tant du point de vue du joueur que de celui du concepteur.

4.1. Introduction

4.1.1. Place du joueur

Nous avons pu déterminer un outil simple pour la définition de jeux. Au cours de la conception de celui-ci, nous avons mis en évidence un ensemble de concepts possédant un plus ou moins grand nombre de caractéristiques que les jeux envisageables doivent partager. Nous nous sommes concentrés dans cette partie du travail sur les éléments à offrir au concepteur sans réellement aborder ce dont le joueur dispose.

Comme nous l'avons déjà souligné, les jeux définissables s'adresserons à des personnes handicapées qui rencontrent diverses difficultés. Pour cette raison, nous avons laissé le soin au concepteur de définir les règles du jeu. Poursuivant dans cette idée, nous avons décidé que le concepteur serait le seul acteur ayant accès aux fonctions de définition de tous les concepts d'un jeu. Le joueur ne pourra donc en aucun cas décider du contenu des jeux (ceci dit, rien n'empêche que le joueur, s'il est présent lors de la définition du jeu, puisse guider celle-ci).

4.1.2. Les concepts

Compte tenu des points de vue différents qu'ont le joueur et le concepteur, les concepts ne seront pas perçus de la même façon par ces deux acteurs. De plus, ces concepts possèdent des caractéristiques statiques et dynamiques. Les caractéristiques statiques sont celles qui sont

visibles ou perceptibles soit par le joueur, soit par le concepteur, soit par les deux. Les caractéristiques dynamiques regroupent les actions admises sur les concepts.

A partir de ces distinctions, il est possible de dégager une structure partagée par tous les concepts, selon laquelle ceux-ci vont être présentés :

- identifiant du concept : permet de désigner de manière univoque les concepts;
- caractéristiques disponibles pour le concepteur-éducateur;
- caractéristiques disponibles pour l'utilisateur-joueur;
- aspects statiques : leurs caractéristiques ou leur attributs;
- aspects dynamiques : fonctions qui seront offertes pour définir ou manipuler les concepts;

La présentation des concepts pourra se faire selon le modèle suivant :

identifiant	**Statique**	**Dynamique**
Concepteur		
Utilisateur		

Nous ne tiendrons compte dans notre étude que de deux catégories d'acteurs. Les joueurs (les personnes handicapées) et les concepteurs (les éducateurs qui réalisent des définitions de jeux). Nous ne tiendrons donc pas compte de la possibilité qu'un éducateur ne participe pas à la conception d'un jeu et qu'il reçoive des définitions qu'il applique sans les modifier.

4.2. Le type de jeton

4.2.1. Utilité du concept

Il s'agit d'un nouveau concept. Il a pour objectif d'alléger la tâche de définition des jetons. Le concepteur d'un jeu doit déterminer un plus ou moins grand nombre de jetons qui seront manipulés par le joueur. S'il doit les créer, il doit, pour chacun d'entre-eux, fournir une liste des éléments les caractérisant : les caractéristiques, les côtés associables, le contenu, ...

Supposons que le concepteur désire créer plusieurs jetons ayant le même contenu, les mêmes caractéristiques et les mêmes côtés associables, mais différant les uns des autres par le

fait que certains sont retournables ou clignotants ou ... et les autres pas. Selon le mode de définition que nous avons élaboré, le concepteur est obligé de réaliser autant de définitions complètes de jetons qu'il y a de jetons.

Pour éviter de répéter un grand nombre de fois une même définition à quelques éléments près, nous avons choisi de créer un nouveau concept, le type de jetons, qui reprendra le contenu, les caractéristiques, les côtés associables et un nombre d'occurrences (limitant le nombre de jetons pouvant être créés à partir de cette définition). Chacun des jetons qui sera créé à partir de cette définition héritera des caractéristiques soulignées et possédera une définition propre pour les autres caractéristiques (empilable, superposable, ...).

Par exemple, voici le type de jeton "Marteau" et les attributs qui définiront une partie (excepté le nombre d'occurrences) des caractéristiques de l'ensemble des jetons qui seront créés à partir de cette définition :

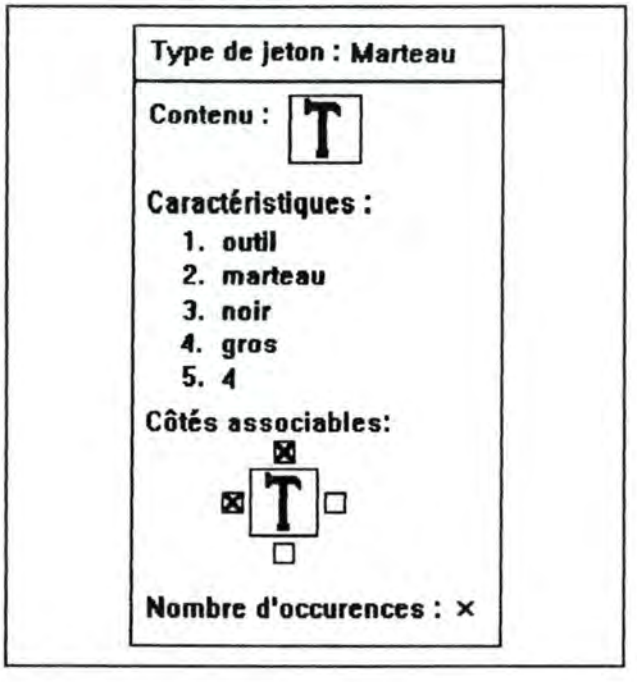


figure 4.1.

Voici à présent trois jetons qui pourraient être créés à partir de cette définition:

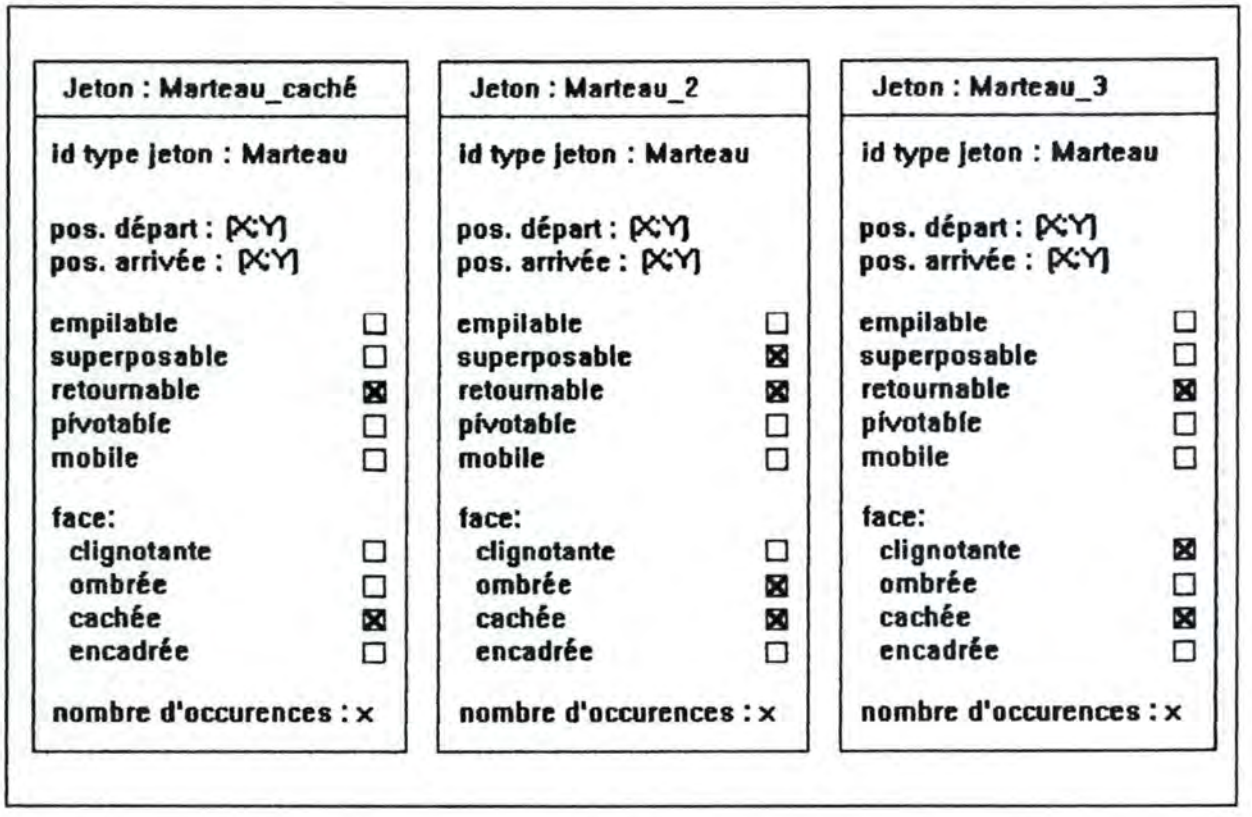


figure 4.2.

Le premier pourrait être un jeton du jeu de mémoire car il est, retournable et sa face est cachée. Le deuxième pourrait être employé dans un jeu d'association.

Le travail du concepteur est allégé car il ne doit spécifier que quelques caractéristiques pour créer plusieurs jetons. De plus, la définition du type de jeton accroît la stabilité des copies des jetons. En effet, en recopiant dix fois la définition d'un même jeton, le concepteur risquerait de commettre des erreurs. C'est évité grâce à cette définition commune.

Remarque : le mode de définition des types de jeton et des jetons employés pour cet exemple n'a été choisi que pour illustrer l'utilité du concept "type de jeton". Lors de conception du logiciel, nous avons défini une interface différente pour la définition de ces concepts.

4.2.2. Structure

Si ce nouveau concept semble apporter un plus au concepteur, le joueur, de son côté, ne perçoit toutefois pas cette réalité. Celui-ci voit seulement les jetons qu'il manipule. Pour cette raison, seul le point de vue du concepteur est pris ici en considération.

tj-(identifiant)	**Statique**	**Dynamique**
Concepteur	- identifiant du contenu - liste des caractéristiques - côtés associables - limite du nombre d'occurrences de jetons appartenant à ce type	- créer, modifier, copier et supprimer une définition

4.2.3. Commentaires :

■ Nous limiterons le contenu des types de jetons à des dessins. On pourrait imaginer qu'ils puissent également contenir des sons qui seraient produits lors de leur sélection.

■ La liste de caractéristiques et la désignation des côtés associables permettent de définir les associations dont peut faire l'objet le type de jeton.

4.3. Le jeton

4.3.1. Attributs

Le jeton est l'un des deux objets manipulables (avec la pièce) par le joueur.

j-(identifiant)	Statique	Dynamique
Concepteur	- identifiant d'un type de jeton - position de départ - position(s) d'arrivée - retournable - pivotable - mobile - reprenable et déplaçable - superposable - représentation de la face : clignotante, ombrée, cachée, encadrée - réalisable par le joueur - en contact avec au moins une autre	- créer, modifier, copier et supprimer une définition - définir les évolutions de représentation
Utilisateur	- position de départ - position(s) d'arrivée - retournable - pivotable - mobile - reprenable et déplaçable - superposable - réalisable par le joueur - en contact avec au moins une autre - représentation de la face : clignotante, ombrée, cachée, encadrée	- sélectionner - déplacer - faire tourner - éliminer - réalisable

4.3.2. Commentaires :

■ Chaque jeton réfère un et un seul type de jeton, désignant la classe à laquelle il appartient.

■ Un jeton a une position de départ, généralement le sabot, et éventuellement une position d'arrivée prédéterminée dans le cas d'une association par position. Ces positions peuvent s'exprimer sous forme de couples de coordonnées (x,y) ou de noms de zones de l'écran (sabot, zone de la grille, ...). Les objets peuvent posséder plusieurs positions d'arrivée, ce qui assure l'interchangeabilité.

4.4. La pièce

4.4.1. Structure

La pièce est une construction basée sur au moins deux types de jetons. Des types constitutifs, on ne gardera que le contenu et les caractéristiques d'associations.

p-(identifiant)	Statique	Dynamique
Concepteur	■ Eléments hérités : - identifiants des contenus - listes des caractéristiques - côtés associables ■ Eléments propres : - description de la forme de la pièce - position de départ - position(s) d'arrivée - retournable - pivotable - mobile - reprenable et déplaçable - superposable - représentation de la face : clignotante, ombrée, cachée, encadrée - réalisable par le joueur - en contact avec au moins une autre	- créer, modifier, copier et supprimer une définition - définir les évolutions de représentations

Utilisateur	■ Eléments propres : - forme de la pièce - position de départ - position(s) d'arrivée - retournable - pivotable - mobile - reprenable et déplaçable - superposable - représentation de la face : clignotante, ombrée, cachée, encadrée - réalisable par le joueur - en contact avec au moins une autre	- sélectionner - déplacer - faire tourner - éliminer - réaliser

4.4.2. Commentaires :

■ La définition d'une pièce comprend la détermination de sa forme. Celle-ci pourrait se faire par le marquage, au sein d'une matrice 3x3, des cellules délimitant le contour de la pièce.

■ Les caractéristiques s'entendent comme dans le cas des jetons et types de jetons.

4.5. Les zones de l'écran

L'écran peut être divisé en différentes zones. Chaque zone est destinée à recevoir des objets à un moment donné du jeu et correspond à des fonctions distinctes du jeu. Une zone doit occuper un emplacement précis sur l'écran. Il faut donc en définir la position et la taille. Les zones de l'écran ne sont pas manipulables par l'utilisateur. Il en percevra donc uniquement les aspects statiques.

4.5.1. La grille

La grille est la partie de l'écran sur laquelle le jeu se déroule.

g-(identifiant)	Statique	Dynamique
Concepteur	■ La grille - taille et place - un dessin contenu - identifiant(s) d'un (plusieurs) jeton(s) contenu(s) - place du contenu - nombre de zones de la grille - ordre d'utilisation des zones - marque de la zone courante ■ Pour chaque zone de la grille - taille et place de la zone - un dessin contenu - identifiant(s) d'un (plusieurs) jeton(s) contenu(s) - place du contenu dans la zone	- créer, modifier, copier et supprimer une définition
Utilisateur	■ La grille - taille et place - dessin contenu - un (plusieurs) jeton(s) contenu(s) - position du contenu - nombre de zones de la grille - marque de la zone courante ■ Pour chaque zone de la grille - taille et place de la zone - dessin contenu - un (plusieurs) jeton(s) contenu(s) - position du contenu dans la zone	

Commentaires :

■ La grille est la seule zone de l'écran qui est indispensable.

■ La grille peut être recouverte totalement ou partiellement par un dessin.

■ La grille peut être découpée en zones. Chaque zone peut avoir une taille propre (exprimée en nombre de cases), c'est-à-dire qu'il n'est pas obligatoire de diviser la grille en zones de tailles identiques. Cependant, nous nous sommes cantonnés à des zones de tailles rectangulaires ou carrées. Comme la grille, chaque zone peut contenir un dessin.

■ Un ordre d'utilisation des zones peut être imposé. Dans ce cas, il est nécessaire d'indiquer quelle est la zone courante.

4.5.2. Le sabot

Le sabot est la partie de l'écran où sont disposés les jetons et les pièces mis à la disposition du joueur au cours du jeu.

s-(identifiant)	Statique	Dynamique
Concepteur	- taille et place - présentation des objets : nombre d'objets présents, taille des "wagons", type de "wagon" - disposition des objets (à plat ou en piles) - ordre de présentation et de disposition - affichage du nombre d'objets	- créer, modifier, copier et supprimer une définition
Utilisateur	- taille et place - présentation des objets : nombre d'objets présents, taille des "wagons', type de "wagon' - disposition des objets (à plat ou en piles) - ordre de présentation et de disposition - affichage du nombre d'objets	

Commentaires :

■ Les objets peuvent se présenter sur le sabot de différentes manières. Ils peuvent être tous présents dès le départ ou arriver par groupe de n éléments, ce que nous appelons des "wagons" de taille n, ou encore se présenter à la fois dès le départ pour un certain nombre, et

arriver par "wagons" pour le reste. De plus, les "wagons" peuvent arriver à la demande, ou automatiquement, en remplacement des objets utilisés.

■ Les objets peuvent également être disposés de deux manières : soit empilés (par exemple, par type de jetons), soit disposés à plat les uns à côté des autres.

■ L'ordre dans lequel les objets sont présentés et disposés peut être aléatoire ou déterminé selon certains critères (les caractéristiques des types de jetons par exemple).

■ Le nombre d'objets présents dans le sabot est affiché soit en marge du sabot si les jetons sont disposés à plat, soit sous chaque pile dans le cas contraire.

4.5.3. La zone de résultat

Cette zone contiendra les jetons ou les pièces vérifiées et jugées valides. Elle est particulièrement utile pour dégager la grille des jetons correctement manipulés, tout en conservant une trace de ce qui a déjà été fait.

v-(identifiant)	**Statique**	**Dynamique**
Concepteur	- taille et place	- créer, modifier, copier et supprimer une définition
Utilisateur	- taille et place	

4.5.4. La zone d'attente

La zone d'attente sert à mettre en réserve des objets à l'écart de ceux disposés sur le sabot ou la grille. Ceci peut être utile pour les mettre en évidence ou, au contraire, les éliminer temporairement.

a-(identifiant)	**Statique**	**Dynamique**
Concepteur	- taille et place	- créer, modifier, copier et supprimer une définition
Utilisateur	- taille et place	

4.5.5. La zone de message

La zone de message permet d'afficher des informations utiles pour l'utilisateur, pendant toute la durée du jeu, comme guidage ou pour signaler l'état d'avancement du jeu.

v-(identifiant)	Statique	Dynamique
Concepteur	- taille et place - contenu	- créer, modifier, copier et supprimer une définition
Utilisateur	- taille et place - contenu	

4.6. Les règles du jeu

4.6.1. Les règles d'association

Le concepteur doit définir sur quelle base les associations vont s'établir. L'utilisateur voit son action décrite et limitée par cette définition.

ra-(identifiant)	Statique	Dynamique
Concepteur	- choix de (des) forme(s) d'association - choix éventuel de la (des) caractéristique(s) pertinente(s) et de la relation portant sur celle(s)-ci	- créer, modifier, copier et supprimer une définition
Utilisateur	- forme(s) d'association - la (les) caractéristique(s) pertinente(s) et la relation portant sur celle(s)-ci	

Commentaires :

■ Les associations par côtés et par caractéristiques sont combinables. L'association par position ne cohabite avec aucun autre type d'association.

■ Les relations sur base des caractéristiques peuvent s'établir sur une relation d'égalité (tous les objets rouges) ou sur base d'une relation d'ordre (du plus petit au plus grand).

4.6.2. Les règles de validation

La validation est utilisée pour "contrôler" et "juger" les actions de l'utilisateur. Elle est toujours réalisée en vérifiant la correction des associations.

L'utilisateur ne perçoit ces règles que par l'effet de leur mise en oeuvre.

rv-(identifiant)	**Statique**	**Dynamique**
Concepteur	- moment : après x pièces jouées ou après un certain temps - actions liées au résultat de la validation - taux d'erreur et action associée à son dépassement	- définition des caractéristiques pour chaque jeu
Utilisateur	- moment : après x pièces jouées ou après un certain temps - actions liées au résultat de la validation - taux d'erreur et action associée à son dépassement	

Commentaires :

■ Les actions liées au résultat de la validation peuvent être l'affichage d'un message, la suppression des associations valides ou le déplacement des jetons ou des pièces correctement joués vers la zone de résultat, la modification de la présentation des jetons ou des pièces constituant les associations valides, le retour au sabot des objets intervenant dans des associations non-valides, ... Les actions ont été répertoriées dans le chapitre précédent (voir point 4.3.).

■ Pour la définition des événements, le concepteur aura à sa disposition la liste de ces actions dans laquelle il pourra sélectionner un ou plusieurs événements.

4.6.3. Les règles de déplacement

Les déplacements d'objets réalisables par le joueur sont définis de manière unique pour tous les éléments d'un jeu.

rd-(identifiant)	Statique	Dynamique
Concepteur	- dans la grille - dans le sabot - dans la zone d'attente - de la grille vers le sabot - entre la grille et la zone d'attente - entre le sabot et la zone d'attente	- créer, modifier, copier et supprimer une définition
Utilisateur	- dans la grille - dans le sabot - dans la zone d'attente - de la grille vers le sabot - entre la grille et la zone d'attente - entre le sabot et la zone d'attente	

Commentaires :

■ Rappelons que le déplacement du sabot vers la grille est implicite si un sabot est défini pour le jeu et que le déplacement des jetons vers la zone de résultat est laissé à l'initiative du jeu et non du joueur. Il ne figure pas ici car il est un des événements possibles de l'après-validation.

4.7. Les règles de fin de jeu

fj-(identifiant)	Statique	Dynamique
Concepteur	- état final de la grille - état final du sabot - état final de la zone d'attente - taux d'erreur - temps limite	- créer, modifier, copier et supprimer une définition
Utilisateur	- état final de la grille - état final du sabot - état final de la zone d'attente - taux d'erreur - temps limite	

4.8. La règle de sélection et de manipulation

sm-(identifiant)	Statique	Dynamique
Concepteur	- le mode	- créer, modifier, copier et supprimer une définition
Utilisateur	- le mode	

4.9. Le jeu

Type regroupant pour chacune de ses occurrences l'ensemble des règles choisies, des occurrences des types de jetons, de jetons ou/et des pièces et des zones de l'écran.

x-(identifiant)	Statique	Dynamique
Concepteur	- identifiant de la grille - identifiant du sabot - identifiant de la zone d'attente - identifiant de la zone de message - identifiant des règles d'associations - identifiant de la règle de validation - identifiant des règles de déplacement - identifiant des règles de fin de jeu - identifiant de la règle de manipulation - nombre de jetons ou pièces - identifiants des jetons ou pièces	- définition des caractéristiques pour chaque jeu
Utilisateur	- une grille - peut-être un sabot - peut-être une zone d'attente - peut-être une zone de message - peut-être une zone de résultat - des jetons ou des pièces - des règles qui lui sont communiquées par le concepteur	- manipulation des concepts inclus dans le jeu selon leurs caractéristiques propres

partie 2 : Conception du logiciel

5. Choix pour la conception du logiciel
6. Architecture
7. Description des fonctions
8. Services offerts par les concepts
9. Les structures de données

5.
Choix pour la conception du logiciel

Ce sixième chapitre entame la deuxième partie, consacrée à la conception informatique du logiciel. Après avoir décrit et analysé un grand nombre de jeux, après avoir réalisé une synthèse ayant produit un ensemble de concepts permettant la définition des jeux, nous entamons ici une démarche de développement d'un logiciel qui est purement informatique.

5.1. Ensemble Minimal

Avant d'entamer cette nouvelle partie, nous avons dû nous résoudre à éliminer le concept de pièce, les caractéristiques "superposable", "pivotable", "réalisable par le joueur" et "mobile" des objets pour l'implémentation. En effet, celle-ci risquant d'être complexe et le temps imparti pour la réalisation du logiciel étant limité, il était préférable de mettre de côté ces deux caractéristiques et ce concept pour éviter de compromettre l'entièreté du travail.

Bien que nous n'ayons pas pu mener l'implémentation jusqu'à un produit complètement opérationnel, nous avons tenu à faire figurer cette remarque. En effet, sans celle-ci, le lecteur aurait pu être étonné de ne pas retrouver ces éléments dans la suite de notre travail.

Nous considérerons donc les concepts de jeu, de jetons, de types de jetons, de plan de jeu et de règles de jeu. Ceux-ci sont déjà suffisants pour envisager un grand nombre de jeux. Remarquons également qu'un de nos objectifs pour l'implémentation sera de réaliser un logiciel possédant une architecture modulaire, afin d'offrir la possibilité de reprendre et d'étoffer ultérieurement notre travail le plus aisément possible.

5.2. Contenu du logiciel

5.2.1. Les fonctions

Le logiciel comprendra deux types de fonctions. Un premier groupe de fonctions aura la responsabilité de la définition des éléments constituant les jeux. Ces fonctions (création, modification, suppression, ... des définitions des concepts) sont mises à la disposition du concepteur et il est le seul à y avoir accès.

Le deuxième groupe de fonctions se charge de la gestion du déroulement des jeux : gestion de la manipulation des jetons, application des règles d'association, ... Ces fonctions ne seront accessibles ni par le concepteur, ni par le joueur. Cependant, il est évident que leur portée et leur application seront dépendantes des définitions du concepteur, et donc des valeurs attribuées aux caractéristiques des concepts grâce aux fonctions du premier groupe.

5.2.2. Différents points de vue

- **concepteur :** il a à sa disposition toutes les fonctions pour la définition des concepts et une fonction lui permettant d'activer les jeux;

- **joueur :** aucune fonction du logiciel ne lui est accessible; il peut manipuler les jetons, en cours de jeu, au moyen de la souris; les règles lui sont communiquées soit par le concepteur, soit par un message présent dans la zone de message pendant toute la partie.

- **programme :** il a la responsabilité de la gestion de la définition; il gère le déroulement de la partie et l'application des règles.

5.3. Choix pour la spécification

La spécification se décompose en quatre étapes. En premier, nous proposons et justifions l'architecture que nous avons adoptée. Deuxièmement, nous recensons l'ensemble des fonctions que le logiciel doit présenter pour permettre à l'éducateur de créer des jeux offrant les possibilités résumées aux chapitres précédents. Troisièmement, nous listons et spécifions pour chacun de ces concepts les services (cette notion sera expliquée ultérieurement) qu'ils offrent pour que les fonctions de la deuxième étape puissent être réalisées. Enfin, nous présentons les structures de données des objets manipulés dans le logiciel, c'est-à-dire la spécification formelle des concepts vus dans les chapitres précédents.

Avant d'entamer cette partie de notre étude, voici un certain nombre de remarques concernant les concepts et la procédure à suivre pour la définition des jeux.

5.3.1. Quelques précisions sur l'utilisation des concepts

▪ **Concept et occurrence de concept :** Chaque concept définit un ensemble de caractéristiques ou de type d'actions. C'est une structure vide, un moule, qu'il convient de remplir pour créer un jeu. Lorsque l'animateur utilise un concept et qu'il attribue des valeurs aux caractéristiques, il crée une occurrence de ce concept. L'ensemble de ces occurrences définit le jeu.

▪ **Concept obligatoire ou facultatif, occurrence multiple ou unique :** Si tous les concepts sont utiles, tous ne sont pas nécessaires. Par exemple, les concepts de sabot, de zone d'attente, de zone de résultat et de zone de messages peuvent ne pas être employés pour un jeu.

Parallèlement, remarquons que le concept de type de jetons peut donner plusieurs occurrences pour un même jeu. A l'inverse, une seule instance des concepts définissant les zones de l'écran (grille, sabot, zone d'attente, zone de message) et les règles (de déplacement, d'association et de validation) peut entrer dans la description d'un jeu.

5.3.2. Ordre de définition d'un jeu

Les différents concepts n'étant pas indépendants les uns des autres, il est nécessaire de respecter une certaine chronologie lors de la création d'un jeu. Sans cela, certaines définitions porteraient sur des éléments indéfinis.

Le graphe suivant montre l'enchaînement des étapes de la création. Chaque flèche signifie que la définition du concept-destination (de la flèche) se déroule au plus tôt à la fin de la définition du concept-origine (de la flèche). Deux concepts fictifs ont été introduits (représentés par deux demi-cercles) afin de permettre à plusieurs concepts de former une seule origine.

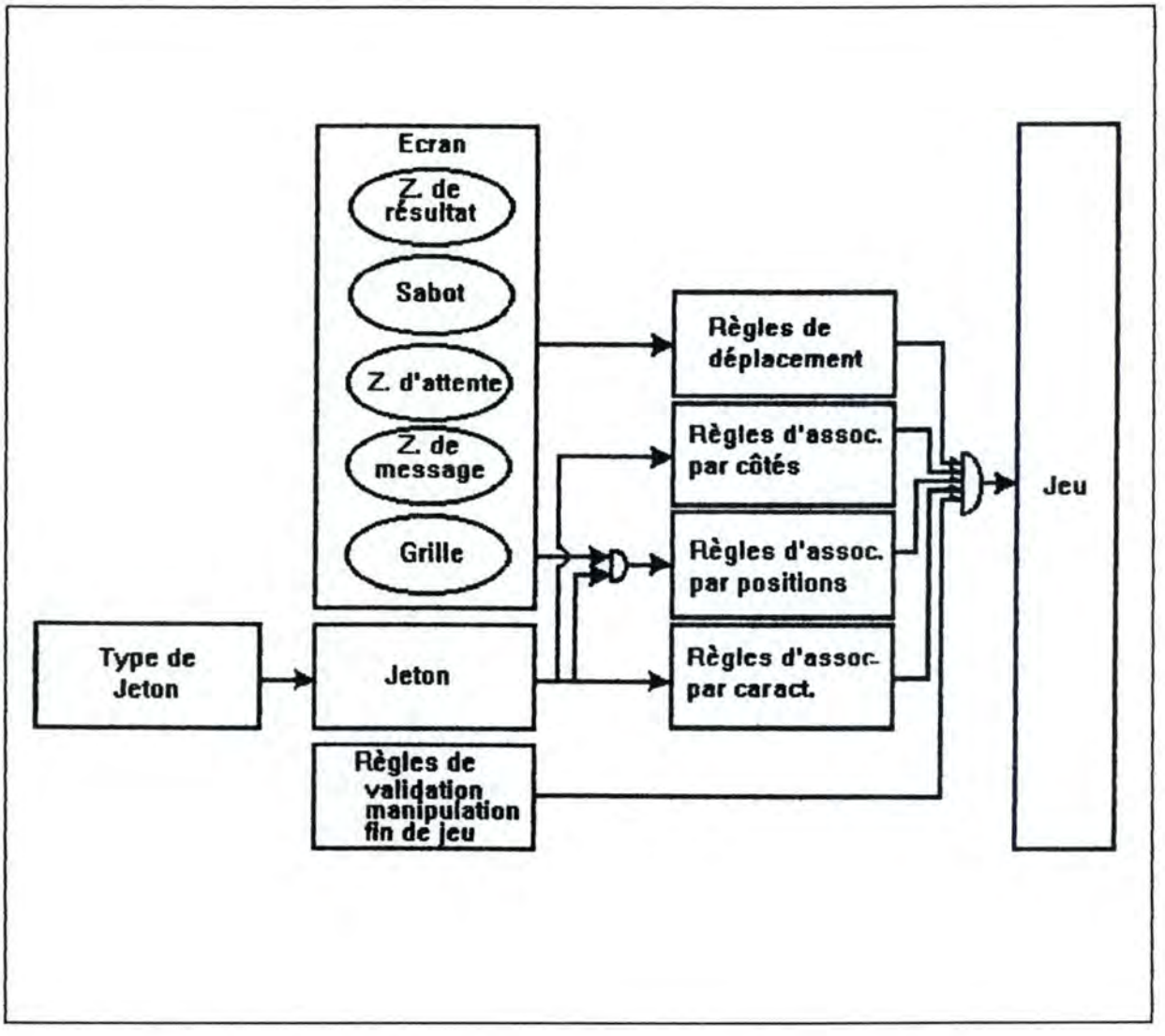


Figure 5.1 : Graphe d'enchaînement des étapes de la conception d'un jeu

Les types de jeton ne font pas directement partie des jeux et peuvent donc être définis indépendamment de ceux-ci. La définition d'un jeton dépend de l'existence d'un type de jetons dont il hérite des caractéristiques.

Les règles de validation sont indépendantes de tout autre concept entrant dans la définition d'un jeu, elles peuvent être définies à tout moment sans qu'il y ait d'effets sur les autres concepts. Les règles de déplacement dépendent des zones de l'écran qui sont définies, elles doivent attendre que ces dernières soient elles-mêmes définies. Les règles d'association dépendent de la définition des jetons. De plus, les règles d'association par position dépendent également de la définition de la grille.

Le jeu est défini une fois les règles d'association (et donc les jetons), de déplacement (et donc l'écran) et de validation définies.

5.3.3. Constitution d'un jeu

En conclusion, la constitution d'un jeu comprend :
- la définition des jetons avec lesquels on jouera,
- la définition de l'écran de jeu,
- la définition des règles du jeu.

▪ **La définition des jetons :** Un jeton doit être défini à partir de la définition d'un type de jeton. Le concepteur commencera donc par définir ce dernier si ce n'est déjà fait. Il déterminera ensuite les caractéristiques du jeton.

▪ **La définition de l'écran de jeu :** L'écran se compose obligatoirement d'une grille. Il peut également contenir un sabot, une zone de message, une zone de résultat et une zone d'attente. Ces éléments peuvent être ou non définis ensemble.

▪ **La définition des règles :** Les règles seront définies en dernier lieu.

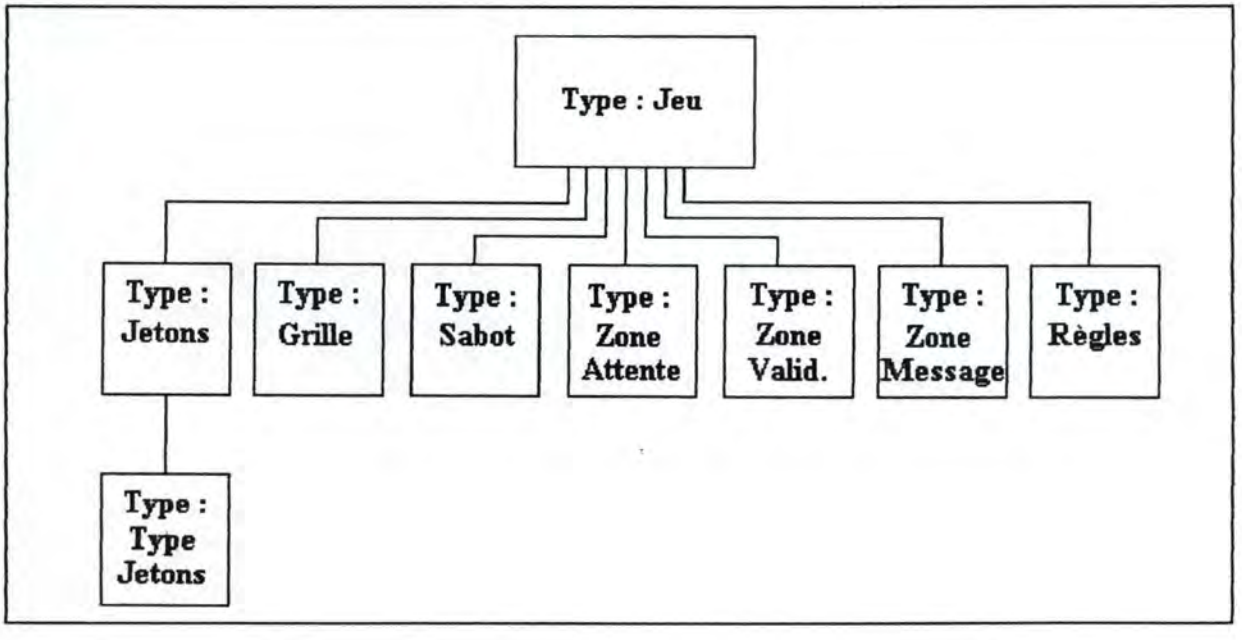
6. Architecture

6.1. Architecture

Nous avons adopté une architecture basée sur les concepts que nous avons définis. Nous emploierons à partir de ce chapitre le terme "type abstrait" pour les désigner. Par exemple, le concept "jeton" est à présent renommé type abstrait "jeton".

Il existe des relations entre ces types. Nous avons déjà souligné que pour créer un jeton, il était nécessaire d'employer la définition d'un type de jetons déjà défini. De même, la définition d'un jeu passe par la définition de tous ses composants. Certains types ont donc besoin de services rendus par d'autres types.

Si nous définissons qu'un type abstrait utilisant un service d'un autre type abstrait se situe à un niveau supérieur, nous pouvons alors établir une hiérarchie qui peut être représentée par la figure suivante.



Ce graphe représente clairement une hiérarchie en niveaux. On traduira le lien qui peut unir deux types de niveaux différents (par exemple "jeu" au niveau supérieur et "zone d'attente" au niveau inférieur) par "un type abstrait d'un niveau n utilise un ou plusieurs service(s) offert(s) par un type abstrait du niveau n-1".

Au niveau le plus élevé apparaît le type abstrait "jeu". Les fonctions qu'il propose sont offertes uniquement au concepteur. Par contre, les autres types abstraits offrent leurs services au type jeu et par ce fait, se trouvent à un niveau inférieur. Le type abstrait "type jeton" occupe le niveau le plus bas car il offre des services au type "jeton'.

remarque : le type "type jetons" occupe une position particulière car il offre à l'utilisateurs des fonctions du même niveau que les fonctions du type "jeu" et des services du plus bas niveau aux fonctions du type "jetons".

6.2. Les services et les fonctions

On constate que les fonctions, vues du logiciel, sont appelées "services", tandis que vues par le concepteur, elles conservent leur intitulé "fonction". Par exemple, "création_grille" sera vue par le concepteur comme la fonction qui lui permet de définir de la grille. Pour le logiciel, "création_grille" sera un service, défini et implémenté au niveau du type abstrait "grille" et qui sera employé, entre autres, par la fonction "création_jeu" du type abstrait "jeu".

Remarquons que les fonctions du type abstrait "jeu" conservent dans toutes les circonstances leur intitulé "fonction" car du point de vue du logiciel, ces fonctions sont du plus haut niveau et ne sont employées par aucun autre type abstrait.

7.
Description des fonctions

Dans ce chapitre, nous décrivons et spécifions de manière informelle les fonctions que le logiciel doit posséder du point de vue du concepteur. La plupart de celles-ci ont été citées dans la description dynamique des concepts, au chapitre quatre. Il ne sera pas question ici, des services inter-types abstraits.

Pour chacune des fonctions citées, nous proposons un énoncé de l'objectif que celle-ci doit remplir, nous citons la liste des informations qu'elle doit recevoir (Entrée) afin de s'exécuter correctement, la liste des résultats obtenus (Sortie) et les règles de traitement dirigeant le déroulement interne de la fonction.

Remarque : pour les concepts obligatoires nous n'avons pas défini de fonction de suppression.

7.1. Fonctions relatives au jeu

7.1.1. Créer_jeu

Objectif	réaliser le lien entre les éléments constitutifs d'un jeu, à savoir les zones de l'écran, les règles et les jetons.
Entrée	- un identifiant d'une grille - un identifiant d'un sabot - un identifiant d'une zone d'attente - un identifiant d'une zone de message - un identifiant d'une zone de résultat - un identifiant des règles de déplacements - un identifiant de la règle de validation - un identifiant des règles de fin de jeu - un identifiant des règles d'associations - un identifiant de la règle de manipulation - les identifiants des jetons
Sortie	- l'identifiant du jeu, en l'occurrence son nom
Règles	- les éléments liés entre eux doivent être cohérents (par exemple les règles de déplacement doivent correspondre à des zones de l'écran définies) - il y a obligatoirement un et un seul identifiant de grille ainsi qu'un identifiant de règles de manipulation, de fin de jeu, de validation et d'association - il y a au moins un identifiant de jeton - les autres identifiants sont facultatifs

7.1.2. Modifier_jeu

Objectif	remplacer, supprimer, modifier, ajouter certains éléments constitutifs d'un jeu existant
Entrée	- le nom du jeu - l' (les) identifiant(s) de(s) l'élément(s) à remplacer
Sortie	- /
Règles	- idem "créer_jeu"

7.1.3. Supprimer_jeu

Objectif	supprimer la définition d'un jeu
Entrée	- le nom du jeu
Sortie	- un message indiquant la suppression
Règles	- /

7.1.4. Résumer_jeu

Objectif	donner un résumé de l'ensemble des éléments et caractéristiques constitutifs d'un jeu
Entrée	- le nom du jeu
Sortie	- une description des zones de l'écran, des règles et des jetons du jeu
Règles	- /

7.1.5. Jouer_jeu

Objectif	activer un jeu
Entrée	- le nom du jeu
Sortie	- la présentation et l'organisation de tous les éléments constitutifs d'un jeu; le jeu est alors prêt à être joué
Règles	- /

7.2. Le type de jeton

7.2.1. Créer_type_jeton

Objectif	créer une définition de type de jeton
Entrée	- l'identifiant du contenu - les caractéristiques du type de jeton - les côtés associables - nombre d'occurrences
Sortie	- l'identifiant du type de jeton
Règles	- il y a un et un seul contenu par type de jeton - il n'y a, au plus, qu'une seule valeur par caractéristique

7.2.2. Modifier_type_jeton

Objectif	modifier les éléments de la définition d'un type de jeton
Entrée	- l'identifiant du type de jeton - les éléments modifiés
Sortie	- /
Règles	- voir "créer_type_jeton"

7.2.3. Supprimer_type_jeton

Objectif	supprimer une définition de type de jeton
Entrée	- l'identifiant du type de jeton
Sortie	- un message indiquant la suppression
Règles	- /

7.3. Le jeton

7.3.1. Créer_jeton

Objectif	créer une définition de jeton
Entrée	- un identifiant de type de jeton - les caractéristiques du jeton - les évolutions de représentations
Sortie	- l'identifiant du jeton

Règles	- l'identifiant du type de jeton est obligatoire et est unique - il y a, pour chaque jeton, au plus une valeur par caractéristique et par représentation

7.3.2. Modifier_jeton

Objectif	modifier les caractéristiques dans la définition d'un jeton
Entrée	- l'identifiant du jeton - les caractéristiques modifiées
Sortie	- /
Règles	- voir création_jeton

7.3.3. Supprimer_jeton

Objectif	supprimer une définition de jeton
Entrée	- l'identifiant du jeton
Sortie	- un message indiquant la suppression
Règles	- /

7.4. Les règles d'association

7.4.1. Créer_règles_association

Objectif	créer une définition de règles d'association
Entrée	- le(s) mode(s)
Sortie	- l'identifiant des règles
Règles	- l'association par position n'est pas combinable avec une des deux autres

7.4.2. Modifier_règles_association

Objectif	modifier la définition des règles d'association
Entrée	- l'identifiant des règles - la modification à apporter
Sortie	- /
Règles	- voir créer_règles_association

7.5. Les règles de déplacement

7.5.1. Créer_règles_déplacement

Objectif	créer une définition de règles de déplacement
Entrée	- le(s) mode(s)
Sortie	- l'identifiant des règles
Règles	- cohérence avec les zones définies pour l'écran

7.5.2. Modifier_règles_déplacement

Objectif	modifier une définition des règles de déplacement
Entrée	- l'identifiant des règles - la modification à apporter
Sortie	- /
Règles	- voir créer_règles_déplacement

7.5.3. Supprimer_règles_déplacement

Objectif	supprimer une définition des règles de déplacement
Entrée	- l'identifiant des règles
Sortie	- un message indiquant la suppression
Règles	- /

7.6. La règle de validation

7.6.1. Créer_règle_validation

Objectif	créer une définition de la règle de validation
Entrée	- moment - événements de fin de validation - taux
Sortie	- l'identifiant de la règle
Règles	- événements cohérents avec les zones définies dans l'écran

7.6.2. Modifier_règle_validation

Objectif	modifier une définition de la règle de validation
Entrée	- l'identifiant de la règle - la modification à apporter
Sortie	- /
Règle	- voir créer_règle_validation

7.7. La règle de manipulation

7.7.1. Créer_règle_manipulation

Objectif	créer une définition de la règle de manipulation
Entrée	- le mode
Sortie	- l'identifiant de la règle
Règle	- cohérence avec les règles de déplacement

7.7.2. *Modifier_règle_manipulation*

Objectif	modifier une définition de la règle de manipulation
Entrée	- l'identifiant de la règle - la modification à apporter
Sortie	- /
Règle	- voir créer_règle_manipulation

7.8. Les règles de fin du jeu

7.8.1. Créer_règles_fin_jeu

Objectif	créer une définition des règles de fin de jeu
Entrée	- état des zones de l'écran - temps limite - nombre de coups limite
Sortie	- l'identifiant des règles
Règles	- cohérence avec les zones de l'écran définies

7.8.2. Modifier_règles_fin_jeu

Objectif	modifier une définition des règles de fin de jeu
Entrée	- l'identifiant des règles - la modification à apporter
Sortie	- /
Règles	- voir créer_règles_fin_jeu

7.9. La grille

7.9.1. Créer_grille

Objectif	créer une définition de grille
Entrée	- le mode de découpe et les attributs de la découpe
Sortie	- l'identifiant de la grille
Règles	- /

7.9.2. Modifier_grille

Objectif	modifier une définition de grille
Entrée	- l'identifiant de la grille - la modification à apporter
Sortie	- /
Règle	- /

7.10. Le sabot

7.10.1. Créer_sabot

Objectif	créer une définition de sabot
Entrée	- les attributs du sabot
Sortie	- l'identifiant du sabot
Règle	- /

7.10.2. Modifier_sabot

Objectif	modifier une définition de sabot
Entrée	- l'identifiant du sabot - la modification à apporter
Sortie	- /
Règle	- /

7.10.3. Supprimer_sabot

Objectif	supprimer une définition de sabot
Entrée	- l'identifiant du sabot - la modification à apporter
Sortie	- un message indiquant la suppression
Règle	- /

7.11. La zone d'attente

7.11.1. Créer_zone_attente

Objectif	créer une définition d'une zone d'attente
Entrée	- les attributs de la zone d'attente
Sortie	- l'identifiant de la zone d'attente
Règle	- /

7.11.2. Modifier_zone_attente

Objectif	modifier une définition d'une zone d'attente
Entrée	- l'identifiant de la zone d'attente - la modification à apporter
Sortie	- /
Règle	- /

7.11.3. Supprimer_zone_attente

Objectif	supprimer une définition d'une zone d'attente
Entrée	- l'identifiant de la zone d'attente
Sortie	- un message indiquant la suppression
Règle	- /

7.12. La zone de résultat

7.12.1. Créer_zone_résultat

Objectif	créer une définition d'une zone de résultat
Entrée	- les attributs de la zone de résultat
Sortie	- l'identifiant de la zone de résultat
Règle	- /

7.12.2. Modifier_zone_résultat

Objectif	modifier une définition d'une zone de résultat
Entrée	- l'identifiant d'une zone de résultat - la modification à apporter
Sortie	- /
Règle	- /

7.12.3. Supprimer_zone_résultat

Objectif	supprimer une définition d'une zone de résultat
Entrée	- l'identifiant de la zone de résultat
Sortie	- un message indiquant la suppression
Règle	- /

7.13. La zone de message

7.13.1. Créer_zone_message

Objectif	créer une définition d'une zone de message
Entrée	- les attributs de la zone de message
Sortie	- l'identifiant de la zone de message
Règle	- /

7.13.2. Modifier_zone_message

Objectif	modifier une définition d'une zone de message
Entrée	- l'identifiant de la zone de message - la modification à apporter
Sortie	- /
Règle	- /

7.13.3. Supprimer_zone_message

Objectif	supprimer une définition d'une zone de message
Entrée	- l'identifiant de la zone de message - la modification à apporter
Sortie	- un message indiquant la suppression
Règle	- /

8.
Les services offerts par les concepts

Dans ce chapitre, nous détaillons les services offerts par chacun des types abstraits. Il s'agit, pour chacun de ceux-ci, de la liste des services offerts aux types du niveau qui leur est directement supérieur pour que ces derniers puissent réaliser leurs fonctions. Par exemple, les services "créer_jeton" ou "modifier_jeton" seront utilisés par les fonctions "créer_jeu" ou "modifier_jeu" du type abstrait "jeu".

Remarquons que les premiers services décrits pour chacun des types ont déjà été présentés lors de l'étude des fonctions au chapitre précédent. Nous n'en faisons donc pas l'étude dans ce huitième chapitre.

8.1. Le type de jeton

Nom du service	Données en entrée	Données en sortie
créer_type_jeton		
modifier_type_jeton		
supprimer_type_jeton		
recherche_information	identifiant d'un type de jeton	rend toutes les caractéristiques du type de jeton
affiche_contenu	identifiant d'un type de jeton	affiche le contenu du type de jeton

8.2. Le jeton

Nom du service	Données en entrée	Résultat
créer_jeton		
modifier_jeton		
supprimer_jeton		
rechercher_info	identifiant du jeton	rend toutes les caractéristiques du jeton
afficher_jeton	identifiant du jeton	affiche le jeton à sa position courante
modifier_position	identifiant du jeton coordonnée horizontale coordonnée verticale	modifie la valeur de la position courante du jeton
modifier_présentation	identifiant du jeton	modifie la valeur de l'état de représentation du jeton
éliminer_jeton	identifiant du jeton	fait disparaître le jeton
recherche_position_xy	coordonnée horizontale coordonnée verticale	rend l'identifiant du jeton se trouvant à la position indiquée
recherche_positio_id	identifiant du jeton	rend les identifiants des jetons se trouvant autour du jeton indiqué
valide_position	identifiant de jeton	rend vrai ou faux suivant que le jeton indiqué se trouve à sa position d'arrivée ou pas
valide_côté	deux identifiants de jetons	rend vrai ou faux suivant que les deux jetons sont correctement associés par côtés
valide_caractéristique	deux identifiants de jetons	rend vrai ou faux suivant que les deux jetons sont correctement associés par caractéristiques

8.3. Les règles

Toutes les règles fournissent les mêmes services. Certaines ne fournissent pas le service "supprimer".

Nom du service	Données en entrée	Données en sortie
créer_règle_...		
modifier_règle_...		
supprimer_règle_...		
appliquer_règles	/	applique une règle particulière en fonction du contexte, de l'évolution du jeu, d'un moment particulier, ...

8.4. La grille

Nom du service	Données en entrée	Données en sortie
créer_grille		
modifier_grille		
afficher_grille	identifiant de la grille	affiche la grille à sa position

8.5. Le sabot

Nom du service	Données en entrée	Données en sortie
créer_sabot		
modifier_sabot		
supprimer_sabot		
afficher_sabot	identifiant du sabot	affiche le sabot à sa position
afficher-nombre	/	affiche le nombre de jetons selon les caractéristiques du sabot
wagon	/	fait apparaître un wagon de jetons selon les caractéristiques du sabot

8.6. La zone de résultat et de message

Ces deux zones fournissent les mêmes services.

Nom du service	Données en entrée	Données en sortie
créer_zone_...		
modifier_zone_...		
supprimer_zone_...		
afficher_zone_...	/	affiche la zone à sa position

8.7. La zone de message

Nom du service	Données en entrée	Données en sortie
créer_zone_...		
modifier_zone_...		
supprimer_zone_...		
afficher_zone_...	/	affiche la zone, garnie de son contenu, à sa position

9.
Les structures de données

Ce chapitre décrit les structures de données utilisées pour représenter les concepts que nous avons définis précédemment.

Les structures reprennent d'une part l'ensemble des éléments utiles à la description des concepts et d'autre part les informations nécessaires à leur mise en oeuvre.

Les éléments sont décrits à l'aide des types élémentaires suivants :
- string : une chaîne de caractères;
- boolean : un booléen;
- integer : un entier;

et des constructeurs de types suivants :
- set : un ensemble d'éléments;
- cp : un élément constitué de plusieurs autres éléments.

9.1. Le jeton

Structure	Signification
id-j : string	l'identifiant du jeton
id-tj : string	l'identifiant du type de jeton
face : set (cp (	la représentation de la face
cligne : boolean	face clignotante
ombre : boolean	face ombrée
cache : boolean	face cachée
encadre : boolean	face encadrée
))	
position : cp (	les positions du jeton
départ,	la position de départ
courante,	la position courante (pendant le jeu)
arrivée : cp (	la position d'arrivée
zone : integer	la position exprimée en terme de zone de l'écran
coord : cp (	la position exprimée en terme de coordonnée
x,	la coordonnée horizontale
y : integer	la coordonnée verticale
)	
)	
)	
état : integer	l'état du jeton

9.2. Le type de jeton

Structure	Signification
id-tj : string	l'identifiant du type de jeton
id-contenu : string	l'identifiant du contenu du jeton
caractéristiques : cp (	les caractéristiques du type de jeton
c1,	première caractéristique
c2,	deuxième caractéristique
c3,	troisième caractéristique
c4,	quatrième caractéristique
c5 : cp (	cinquième caractéristique
pré,	préfixe de la caractéristique
post : string	post fixe de la caractéristique
)	
)	
côtés : cp (	les côtés du type de jeton
cote_supérieur,	le côté supérieur
cote_droit,	le côté droit
cote_inférieur,	le côté inférieur
cote_gauche : boolean	le côté gauche
)	
limite : integer	la limite du nombre d'occurrence

9.3. La grille

Structure	Signification
id-g : string	l'identifiant de la grille
coordonnées : cp (	la position du coin supérieur gauche de la grille
x,	la coordonnée horizontale
y : integer	la coordonnée verticale
)	
cellules : cp (	le nombre de cellules
longueur,	le nombre de cellules en longueur
largeur : integer	le nombre de cellules en largeur
)	
contenu : cp (	le dessin contenu par la grille
id-contenu : string	l'identifiant du contenu
position : cp (	la position du contenu
x,	la coordonnée horizontale
y : integer	la coordonnée verticale
)	
)	
zone : integer	le nombre de zone de la grille
liste-de-zone : set (cp (	la liste des zones
id-zone : integer	l'identifiant de la zone
position : cp (	la position du coin supérieur de la zone
x,	la coordonnée horizontale
y : integer	la coordonnée verticale
)	
cellules : cp (	le nombre de cellules
longueur,	le nombre de cellules en longueur
largeur : integer	le nombre de cellules en largeur
)	

Structure (suite)	Signification
contenu : cp (	le dessin contenu par la grille
id-contenu : string	l'identifiant du contenu
position : cp (	la position du contenu
x,	la coordonnée horizontale
y : integer	la coordonnée verticale
)	
)	
)	
)	

9.4. Le sabot

Structure	Signification
id-s : string	l'identifiant du sabot
coordonnées : cp (	la position du coin supérieur gauche de la grille
x,	la coordonnée horizontale
y : integer	la coordonnée verticale
)	
cellules : cp (	le nombre de cellules
longueur,	le nombre de cellules en longueur
largeur : integer	le nombre de cellules en largeur
)	
nombre-initial : integer	nombre de jetons ou pièces présentent initialement sur le sabot
wagon : integer	la taille des wagons
arrivée-wagon : boolean	le type d'arrivée des wagons
disposition : integer	la taille des piles
ordre-présentation : boolean	la présentation est ordonnée
ordre-disposition : boolean	la disposition est ordonnée
affichage-nombre : boolean	le nombre de jetons est affiché

9.5. Les zone de résultat, d'attente et de message

En dehors de leurs identifiants respectifs, les zones de résultat, d'attente et de message partagent la même structure.

Structure	Signification
id-(v,a,r) : string	l'identifiant de la zone
coordonnées : cp (	la position du coin supérieur gauche de la grille
x,	la coordonnée horizontale
y : integer	la coordonnée verticale
)	
cellules : cp (	le nombre de cellules
longueur,	le nombre de cellules en longueur
largeur : integer	le nombre de cellules en largeur
)	

9.6. Les règles d'association

Structure	Signification
id-ra : string	l'identifiant des règles d'association
choix-forme : cp (	la forme des associations
caractéristiques,	association par caractéristiques
position,	association par position
côté : boolean	association par position
)	
choix-caractéristiques : cp (	les caractéristiques sur lesquelles se basent les associations
car1,	première caractéristique
car2,	deuxième caractéristique
car3,	troisième caractéristique
car4,	quatrième caractéristique
car5 : cp (	cinquième caractéristique
choix : string	préfixe de la caractéristique
comp : string	type de comparaison (égalité, ...)
)	
)	

9.7. Les règles de validation

Structure	Signification
id-rv : string	l'identifiant des règles de validation
moment : integer	le moment de la validation
action-validation : cp (	l'action liée au résultat de la validation
ok,	l'identifiant de l'action liée à une validation correcte
ko : integer	l'identifiant de l'action liée à une validation incorrecte
)	
taux-erreur : integer	le taux d'erreur limite
action-taux-erreur : integer	l'identifiant de l'action liée au dépassement du taux d'erreur

9.8. Les règles de déplacement

Structure	Signification
id-rd : string	l'identifiant des règles de déplacement
grille,	dans la grille
sabot,	dans le sabot
zone-attente,	dans la zone d'attente
grille-sabot,	de la grille vers le sabot
grille-zone-attente,	entre la grille et la zone d'attente
sabot-zone-attente : boolean	entre le sabot et la zone d'attente

9.9. Les règles de fin du jeu

Structure	Signification
id-fj : string	l'identifiant de la règle de fin de jeu
limite-temps : integer	la limite de temps de jeu
action-limite-temps : integer	l'identifiant de l'action liée au dépassement du temps imparti au jeu
etat-grille	état de la grille à la fin du jeu : pleine, vide ou sans importance
etat-sabot	état du sabot à la fin du jeu : plein, vide ou sans importance

etat-zone-attente	état de la zone d'attente à la fin du jeu : pleine, vide ou sans importance

9.10. La règle de manipulation

Structure	Signification
id-sm : string	l'identifiant de règle de manipulation
sel,	sélections seules
dépl,	déplacements seuls
sel-depl	sélections et déplacements

9.11. Le jeu

Structure	Signification
id-x,	l'identifiant du jeu
id-g,	l'identifiant de la grille
id-s,	l'identifiant du sabot
id-v,	l'identifiant de la zone de résultat
id-a,	l'identifiant de la zone d'attente
id-m,	l'identifiant de la zone de message
id-ra,	l'identifiant des règles d'association
id-rv,	l'identifiant des règles de validation
id-rd : string	l'identifiant des règles de déplacement
id-fj : string	l'identifiant des règles de fin de jeu
id-sm : string	l'identifiant de la règle de manipulation
jetons : set (id-j : string)	les identifiants des jetons

partie 3 : Les algorithmes

10. Présentation des algorithmes

10.1. Présentation

Dans les deux chapitres qui suivent, nous allons présenter les deux algorithmes principaux utilisés par notre logiciel. Ils correspondent à deux des fonctions mises en évidence dans la deuxième partie : "créer_jeu" et "jouer_jeu".

10.2. Notion de procédure

10.2.1. Présentation

Au cours de notre étude, nous employons systématiquement le terme "procédure" pour désigner une partie d'un des deux algorithmes principaux. Comme vous vous en rendrez compte, il nous aurait été impossible de présenter en une fois l'intégralité de l'un des deux programmes principaux. Nous avons donc décidé de présenter les algorithmes par degré croissant de raffinement du coeur de leur résolution. Le terme "fonction" a été conservé pour désigner les fonctions.

L'emploi d'une architecture basée sur un ensemble de procédures rencontre l'objectif de modularité que nous nous sommes fixé pour l'implémentation du logiciel.

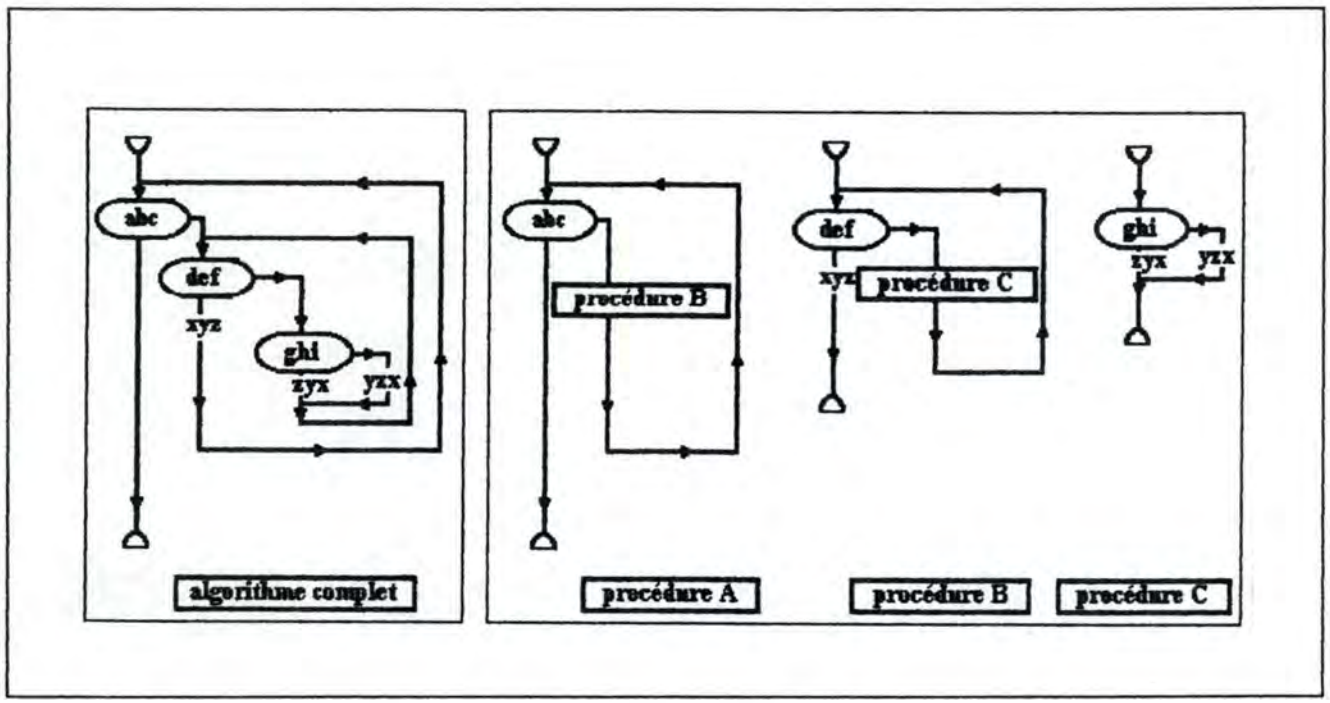


figure 10.1

La figure 10.1. illustre, par un exemple simple, notre choix de représentation. Nous indiquerons, pour chaque procédure, la (les) procédure(s) qui l'appelle (appellent) et la (les) procédure(s) qu'elle appelle. De plus, pour clairement la présenter et comprendre son fonctionnement, chaque procédure est accompagnée par une figure et un commentaire.

10.2.2. Situation des procédures

Nous proposerons également une vue d'ensemble des deux algorithmes principaux afin de bien replacer chaque procédure dans leur architecture. Il s'agit dans les deux cas d'une présentation en niveaux construite au moyen de deux outils : une boîte représentant une fonction ou une procédure et un lien unissant deux boîtes de niveaux n et n+1. Ce lien se traduira de la façon suivante : "la procédure du niveau supérieur (n) appelle la procédure du niveau qui lui est directement inférieur (n+1)".

Pour illustrer cette notion, voici ce que cela donne pour les procédures de l'exemple précédent :

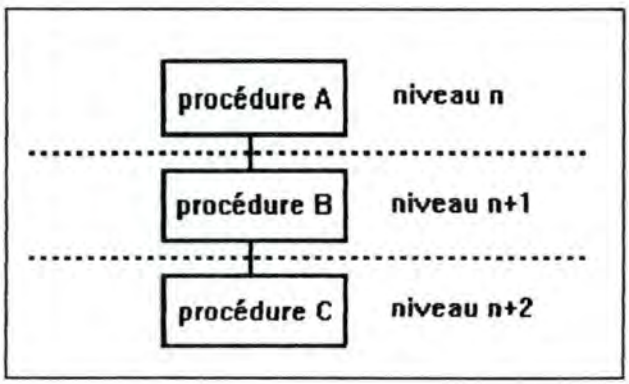


figure 10.2.

10.2.3. Conventions

Chaque procédure est présentée de la façon suivante :

- **situation :** Cette notion permet de présenter la place occupée par la procédure en cours d'étude (la procédure courante) parmi les autres procédures. "est appelée" désigne la (les) procédure(s) qui appelle(nt) la procédure courante, tandis que "appelle" désigne la (les) procédure(s) appelée(s) par la procédure en cours.

Pour l'exemple que nous avons présenté à la page précédente, si on se situe au niveau de la procédure B, pour "est appelé" on aura "A" et pour "appelle" on aura "C'.

Sous cet intitulé, le symbole "/", signifie qu'il n'y a pas de procédure qui appelle ou qui est appelée par la procédure courante.

- **présentation :** Ce point propose généralement une explication de l'objectif de la procédure et de la manière choisie pour l'atteindre.

- **spécifications :** Cette section, présente les objets utilisés par la procédure, les préconditions et les postconditions qui portent sur son exécution.

remarques :

Certains types abstraits, tel le jeton, possèdent une structure très lourde et donc de nombreux attributs. Pour les chapitres qui suivent, nous ne ferons figurer dans chaque procédure que les caractéristiques qu'elle utilise directement.

Nous avons décidé de désigner les attributs des objets de la façon suivante : nom_objet.nom_attribut. Pour un jeton nous aurons donc : jeton.position, jeton.caractéristique, ...

▪ **déroulement interne :** Sous cet intitulé, nous proposons toutes les explications nécessaires à la bonne compréhension du déroulement de la procédure.

11. Algorithme partie définition

Nous présentons, dans ce chapitre, l'algorithme gérant la définition des jeux. Cet algorithme est présenté selon les conventions définies au chapitre précédent. Nous ne présenterons selon ce canevas que les procédures que nous avons jugées nécessaires à la compréhension de l'ensemble.

La figure 11.1. présente l'architecture globale de cet algorithme :

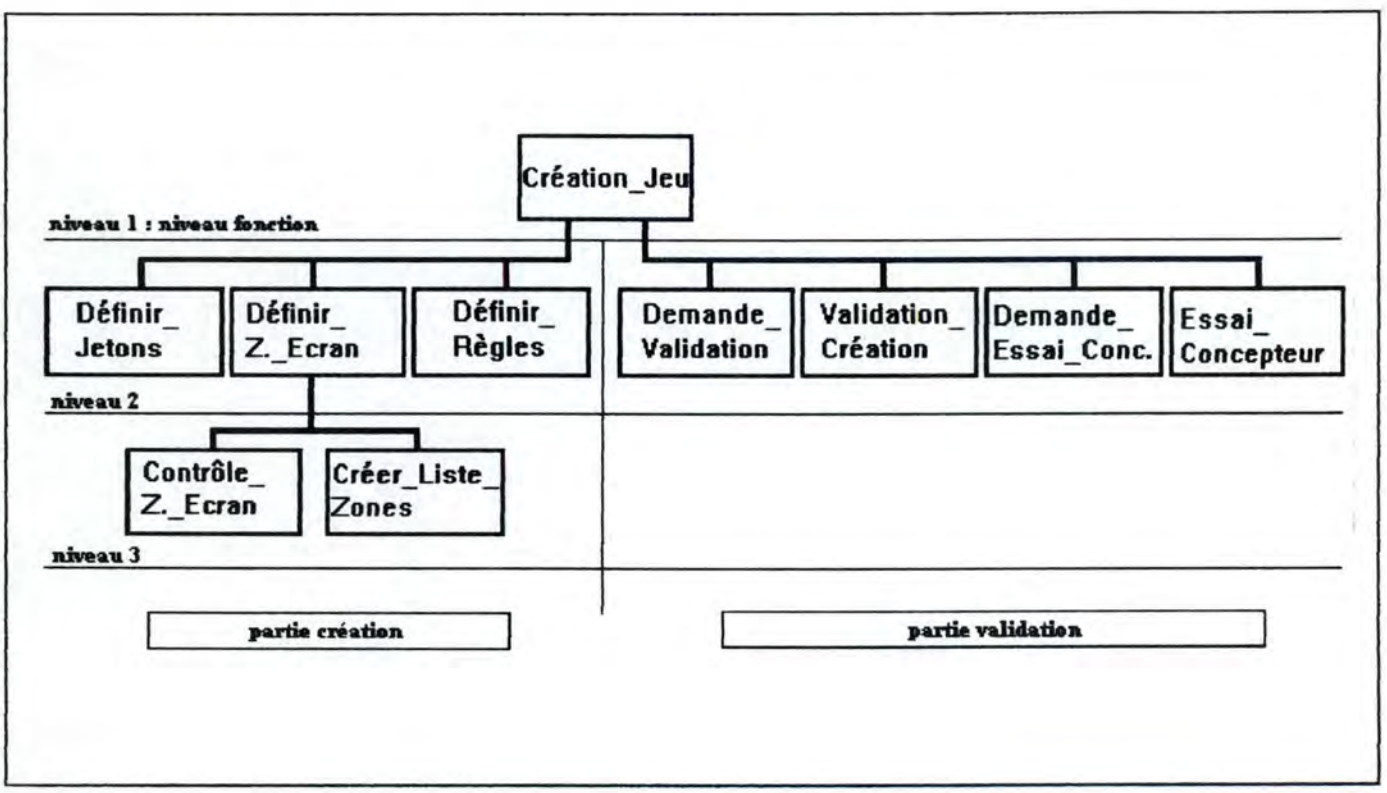


figure 11.1.

11.1. fonction Création Jeu

11.1.1. situation

est appelée par : /

appelle : "Définir_Jeu", "Demande_Validation", "Validation_Création", "Demande_Test", "Test_Concepteur".

11.1.2. présentation

Cette fonction a pour objectif de conduire et de vérifier la procédure de création d'un jeu. La définition du jeu, constitue la première partie de celle-ci. Elle fait appel, elle-même aux procédures de définition des jetons, des zones de l'écran et des règles.

Compte tenu de la qualité de l'interface offerte sous Windows, il est possible d'offrir au concepteur un outil de définition dans lequel il peut passer d'une définition à l'autre, dans l'ordre qu'il veut, de quitter un module de définition particulier alors que la définition n'est pas achevée et de la reprendre plus tard, ... Une telle souplesse sera offerte par les trois procédures de définition. Ceci a l'avantage de rendre la définition d'un jeu plus souple mais, il y a le risque d'oublis de la part du concepteur.

C'est pour cette dernière raison que lorsque l'on quitte la première partie de cette procédure, le concepteur peut demander l'exécution d'une procédure ayant pour objectif de valider la définition. Il s'agit d'un test contrôlant la cohérence de la définition par rapport aux exigences définies pour qu'un jeu soit jouable. Ce contrôle ne teste en aucun cas la qualité du jeu, il constate simplement si les éléments minimaux devant constituer une définition pour un jeu sont présents. Remarquons que le concepteur a la liberté de quitter la définition sans contrôler cette cohérence, car il sait, par exemple, que la définition est inachevée et qu'il désire la poursuivre ultérieurement.

Si le concepteur choisit de valider sa définition il peut, après le contrôle de validité et si celle-ci s'avère correcte, demander à essayer le jeu. Ce deuxième test lui permettra de déterminer si le jeu qu'il vient de définir a tout l'intérêt et toutes les qualités qu'il désirait lui donner.

11.1.3. spécifications

objets utilisés

- **validation :** variable booléenne qui prend sa valeur dans la procédure "Demande_ Validation"; elle permet de constater si le concepteur veut quitter directement le module de création des jeux ou s'il désire tester la qualité de sa définition;
- **valide :** variable booléenne prenant sa valeur dans la procédure "Validation_ Création"; elle permet de constater si la définition comporte l'ensemble minimal d'éléments obligatoires pour qu'une définition donne un jeu activable;
- **dtest :** variable booléenne prenant sa valeur dans la procédure "Demande_Essai_ Concepteur"; elle permet de déterminer si le concepteur veut ou non essayer le jeu;
- **test :** variable booléenne prenant sa valeur dans la procédure "Essai_Concepteur"; elle permet de déterminer si le concepteur est satisfait de sa définition;

préconditions

- /

postconditions

- On possède une définition correcte ou non, complète ou partielle, d'un nouveau jeu. La qualité de celle-ci aura pu être déterminée grâce aux tests qui auront été effectués.

11.1.4. déroulement interne

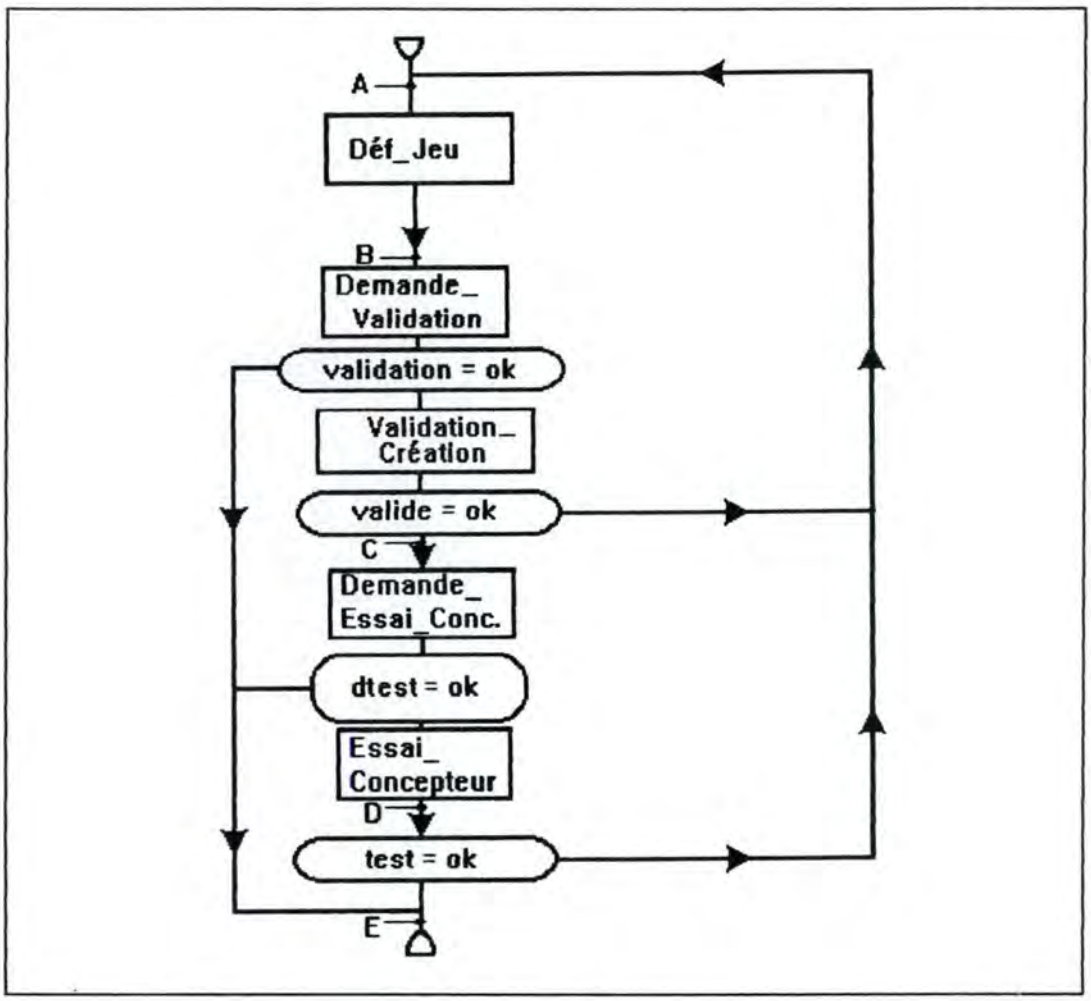


figure 11.2.

■ On entre au point **A** et on réalise la définition.

■ Quand on quitte la partie définition, on demande au concepteur s'il désire valider sa définition. S'il ne le désire pas, on quitte la fonction. Sinon, on appèle la procédure de validation de la création. Si la validation fournit une valeur négative à la variable "valide", on retourne au point **A**, sinon on passe par **C**.

■ Si on est passé par **C**, on demande si le concepteur veut tester sa définition. S'il ne le désire pas, on quitte la procédure. Par contre, s'il désire la tester, le concepteur passe par la procédure "Essai_Concepteur". A la sortie de cette dernière, il pourra décider de retravailler ou non sa définition. Dans le premier cas, il retourne en **A**. Dans le second, il quitte la fonction "Création_Jeu".

11.2. procédure Définir_Jeu

11.2.1. situation

est appelée par : "Création_Jeu"

appelle : "Définir_Jetons", "Définir_Règles", "Définir_Ecran"

11.2.2. présentation

Cette procédure offre toutes les fonctions pour la définition des éléments définissant un jeu.

11.3. procédure Définir_Jetons

11.3.1. situation

est appelée par : "Définir_Jeu"

appelle : /

11.3.2. présentation

Cette procédure offre toutes les fonctions pour la définition des jetons pour un jeu. Il s'agit en fait des services "création_jetons", "modification_jetons" et "supprimer_jetons" que nous avons étudiés précédemment.

Dans le chapitre consacré à l'interface, nous présentons la boîte de dialogue destinée à la création et à la modification. La suppression sera réalisée par une simple sélection du jeton à éliminer suivi d'une suppression.

11.4. procédure Définir_Règles

11.4.1. situation

est appelée par : "Définir_Jeu"

appelle : /

11.4.2. présentation

Cette procédure offre toutes les fonctions pour la définition et la modification des règles d'un jeu. Il s'agit en fait des différents services "création_règles" et "modification_règles" que nous avons présentés lors de l'étude des types abstraits. Cette procédure concerne les règles de fin de jeu, de validation, de déplacement, d'association et de manipulation. Remarquons qu'il est également possible de supprimer les règles de déplacement.

Les services offerts pour ces règles seront gérés par des boîtes de dialogue. Elles sont présentées dans le chapitre consacré à l'interface.

11.5. procédure Définir_Zones_Ecran

11.5.1. situation

est appelée par : "Définir_Jeu"

appelle : "Contrôle_Zone_Ecran"

11.5.2. présentation

Cette procédure gère la définition des zones de l'écran. Elle sera donc accessible lorsque le concepteur demandera à définir un jeu et qu'il voudra définir les zones que l'écran contiendra. Rappelons que l'écran doit au moins contenir une grille et que les autres zones sont optionnelles.

Cette procédure, telle que nous l'avons construite, reprend les services de création, de modification et de suppression (uniquement pour les zones optionnelles) des définitions de chacune des cinq zones de l'écran. La définition des zones est réalisée en spécifiant pour chacune d'entre elles une position et une taille en nombre de cases. La position sera simplement donnée en déterminant les coordonnées de la case du coin supérieur gauche de la zone. La taille sera définie en donnant la hauteur et la largeur en nombre de cases.

La grille peut encore recevoir des caractéristiques complémentaires telles que le nombre de zones qui la composent. Nous avons défini deux modes de définition pour les zones de la grille : zones libres et fixées. Il faut en tenir compte lors de la définition de celle-ci. Dans les deux cas, il faut déterminer les zones qui composeront la grille et, dans le deuxième, il faut encore assigner une ou plusieurs caractéristiques (pour un jeton) à chacune des zones.

Dans le cas où un sabot est choisi pour le jeu, il faut en déterminer l'organisation. Les jetons seront-ils présentés en piles ?; seront-ils présentés à plat ?; dans le deuxième cas, seront-ils tous présents au début de la partie ?; ... .

Si une zone de message est choisie, il faudra en déterminer le contenu.

11.5.3. spécifications

objets utilisés :

- **tabdef :** variable structurée dont chaque élément est du type "déf";
- **déf :** variable structurée qui reprend dans chacun de ses éléments les éléments constitutifs d'une zone;
  - **déf.ind :** champ prenant la valeur "1" si la zone a reçu une définition pour le jeu et "0" si elle n'est pas utilisée dans le jeu;
  - **déf.nom :** champ prenant une valeur de type chaîne de caractères; il reprend le nom de la zone;
  - **déf.coordl :** champ reprenant la première coordonnée du couple des coordonnées désignant la case du coin supérieur gauche de la zone; cette coordonnée désigne la ligne contenant cette case; il s'agit d'une variable entière;
  - **déf.coordh :** champ reprenant la seconde coordonnée du couple des coordonnées désignant la case du coin supérieur gauche de la zone; cette coordonnée désigne la colonne contenant cette case; il s'agit d'une variable entière;
  - **déf.larg :** ce champ reprend la largeur en nombre de cases de la zone;
  - **déf.haut :** ce champ reprend la hauteur en nombre de cases de la zone;
  - **déf.car :** variable structurée; ce champ reprend les caractéristiques particulières que doit posséder la zone;
- **place :** variable booléenne prenant sa valeur dans la procédure "Contrôle_Zones_Ecran"; elle désigne le fait, quand elle est vraie, qu'aucune zone ne dépasse de l'écran ou ne recouvre une autre zone;
- **i :** variable entière jouant le rôle de compteur des éléments de la variable "tabdef";

préconditions

- /

postconditions

- On dispose d'une liste contenant les emplacements, les attributs et les tailles des zones définies pour un jeu particulier.

11.5.4. déroulement interne

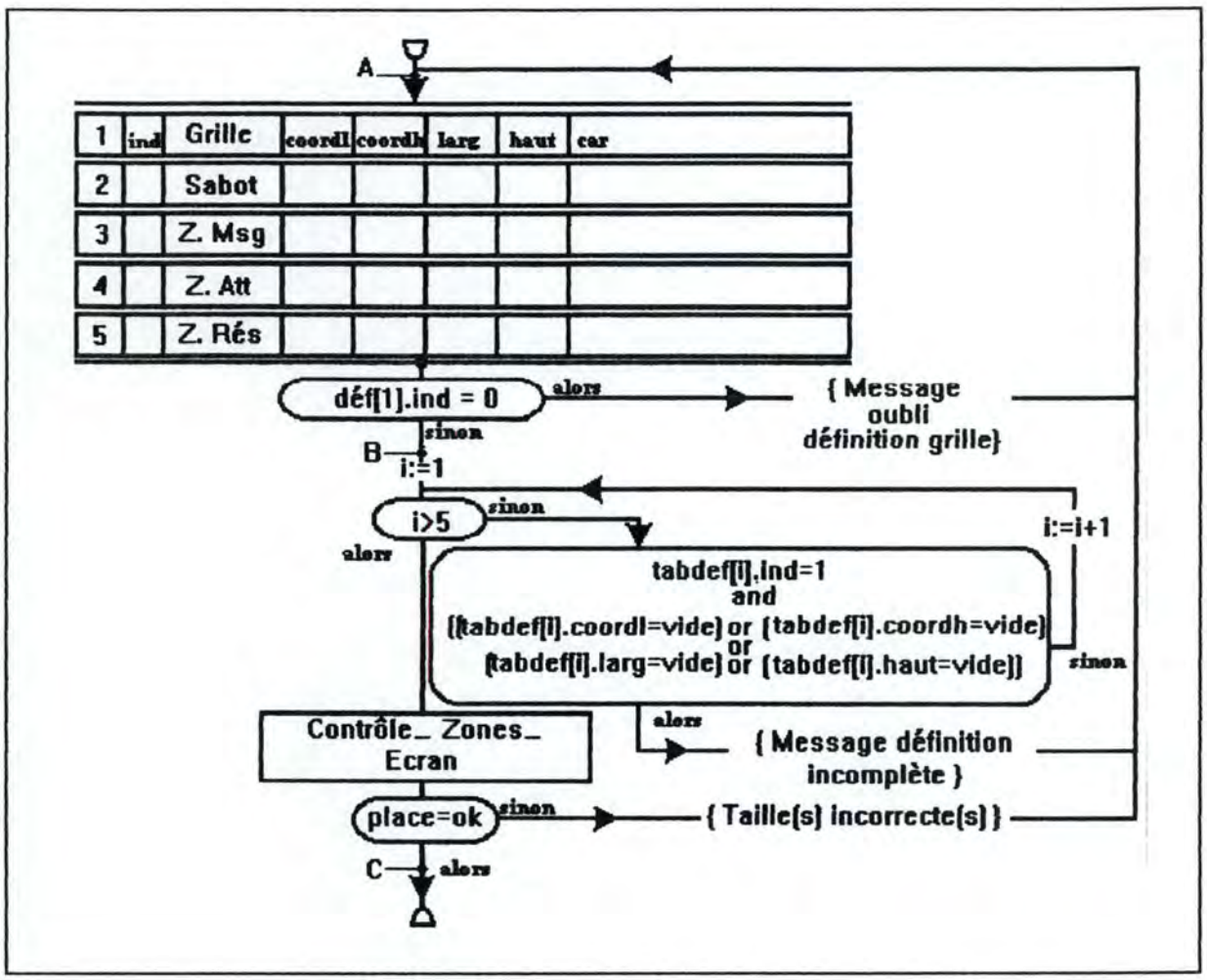


figure 11.3.

■ Après être passé en **A**, on réalise la définition des zones de l'écran. Une fois que le concepteur a décidé que celles-ci le satisfont, il quitte la zone de définition.

■ Dès qu'il a quitté cette partie consacrée à la définition des zones de l'écran, on teste qu'au moins une grille a été définie en contrôlant que le champ ".ind" de la définition de la grille a pris la valeur "1". Si c'est le cas, on passe à la suite en se dirigeant vers **B**. Dans le cas contraire, on renvoie le concepteur à la partie définition du module. Remarquons que l'indicateur d'une zone est initialisé à "0" et garni automatiquement de la valeur "1" dès que le concepteur commence à définir celle-ci. Pour supprimer une zone, il suffit de supprimer les éléments inscrits dans les champs de celle-ci. La valeur de "ind" est alors automatiquement remise à "0".

■ Après être passé en **B**, on teste, pour chacune des définitions que si elle a été choisie, elle possède bien les quatre coordonnées nécessaires à sa définition. Si c'est le cas on passe à la suite. Si ce n'est pas le cas, à la première erreur, on retourne à la partie définition afin que le concepteur complète ou puisse modifier la définition incorrecte.

11.6. procédure Contrôle_Zones_Ecran

11.6.1. situation

est appelée par : "Définir_Zones_Ecran"

appelle : /

11.6.2. présentation

Cette procédure intervient après la définition des zones de l'écran pour vérifier que celles-ci ne débordent pas de l'écran ou ne se chevauchent pas.

11.6.3. spécifications

objets utilisés

■ **tabdef :** variable structurée reprenant par élément une définition de zone;
■ **déf :** variable structurée qui reprend dans chacun de ses éléments les éléments constitutifs d'une zone;
déf.ind : champ prenant la valeur "1" si la zone a reçu une définition pour le jeu et "0" si elle n'est pas utilisée dans le jeu;
déf.nom : champ prenant une valeur de type chaîne de caractères; il reprend le nom de la zone;
déf.coordl : champ reprenant la première coordonnée du couple des coordonnées désignant la case du coin supérieur gauche de la zone; cette coordonnée désigne la ligne contenant cette case; il s'agit d'une variable entière;
déf.coordh : champ reprenant la seconde coordonnée du couple des coordonnées désignant la case du coin supérieur gauche de la zone; cette coordonnée désigne la colonne contenant cette case; il s'agit d'une variable entière;
déf.larg : ce champ reprend la largeur en nombre de cases de la zone;
déf.haut : ce champ reprend la hauteur en nombre de cases de la zone;
déf.car : variable structurée; ce champ reprend les caractéristiques particulières que doit posséder la zone;
■ **place :** variable booléenne qui prend la valeur vraie dans le cas où aucune zone ne chevauche une autre zone ou ne sort des limites de l'écran;
■ **hecran :** constante égale à la hauteur de l'écran calculée en taille de case; elle est égale à la hauteur de l'écran divisée par la taille d'une case;

■ **lecran** : constante égale à la largeur de l'écran calculée en taille de case; elle est égale à la largeur de l'écran divisée par la taille d'une case;
■ **i, j** : variables entières jouant le rôle de compteur;

préconditions

■ On dispose d'au moins une définition pour la grille.

postconditions

■ Les définitions sont inchangées et la variable "place" a acquis une valeur.

11.6.4. déroulement interne

■ Nous représentons le déroulement interne de cette procédure au moyen des deux figures suivantes. La première inclut la seconde.

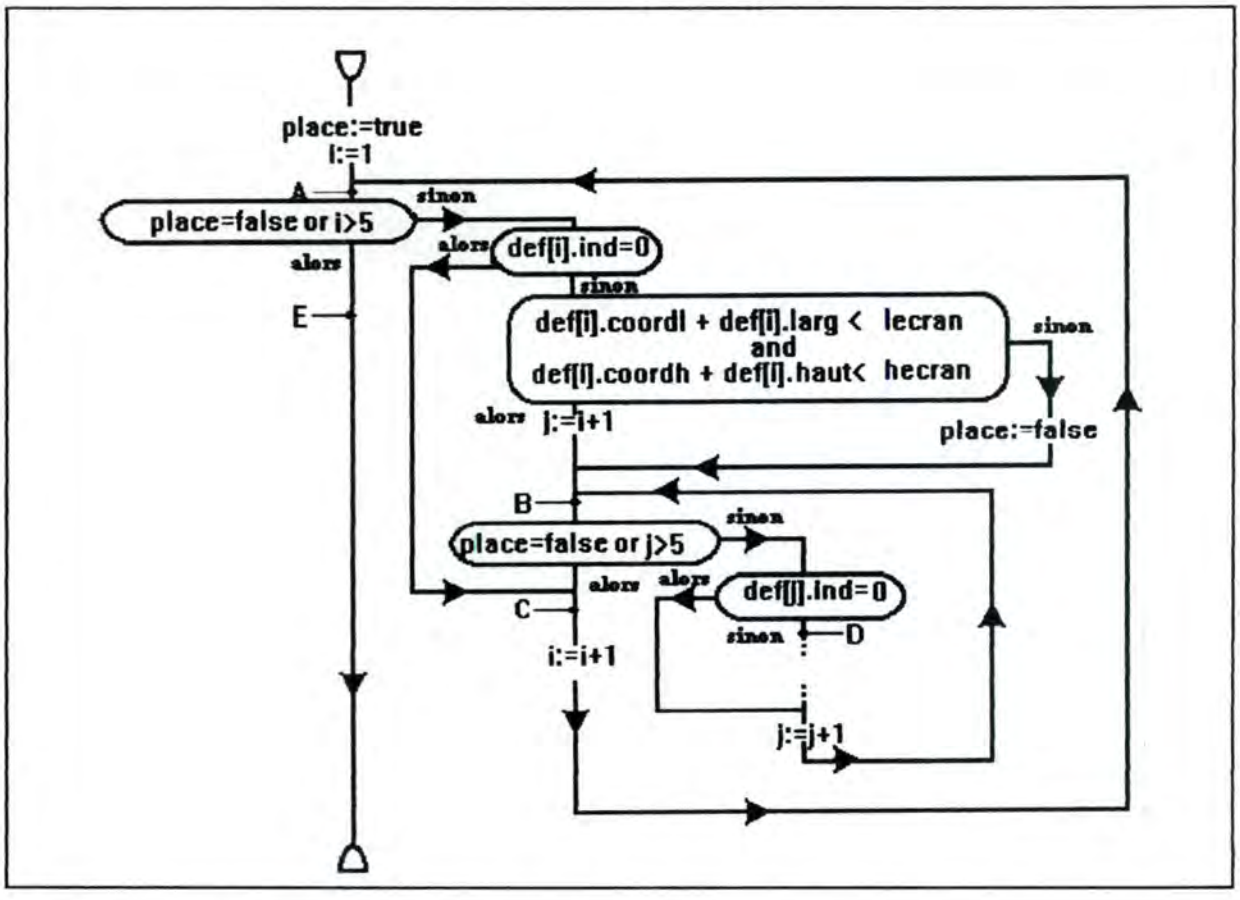


figure 11.4

■ Après passage en **A**, on entame l'entrée en boucle tant que "i" n'a pas dépassé la valeur cinq, ce qui correspond à traiter la définition de chacune des zones. Dans un premier temps, on teste si la ième zone a reçu une définition. Si ce n'est pas le cas, on passe

directement à l'incrémentation de "i". Si c'est le cas on teste ensuite que la zone "i" est bien dans l'écran. Si ce test se révèle positif, on poursuit, si ce n'est pas le cas, "place" prend la valeur false.

■ Le compteur "j" indice la définition des zones qui seront comparées avec la ième. On remarque que "j" est systématiquement initialisée avec la valeur de "i". C'est simplement dû au fait que nous allons comparer les définitions des zones de la façon suivante : 1 avec 2,3,4 et 5; 2 avec 3,4 et 5; 3 avec 4 et 5; 4 avec 5.

■ Une fois que "j" a dépassé cinq, on sort de la boucle imbriquée, en passant par C, pour augmenter "i" d'une unité. Tant que "j" est inférieur à cinq, on traie la ième définition avec la jième.

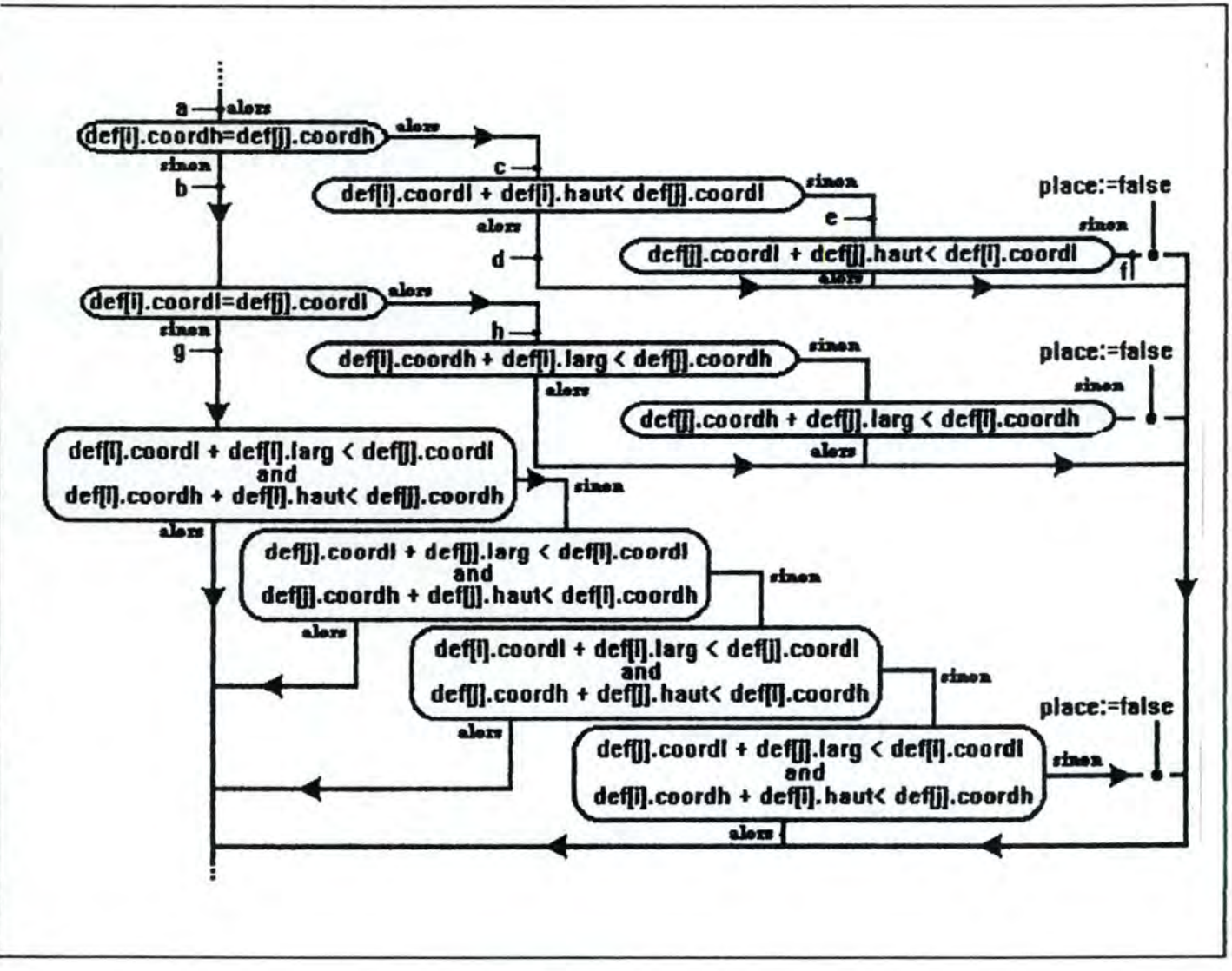


figure 11.5.

■ la figure 11.5. illustre le coeur du traitement.

■ On entre en **a**, on teste alors si les deux zones sont ancrées sur la même colonne. Si c'est le cas, on passe par **c**. Si la zone courante est au-dessus de la jième zone, il faut que la coordonnée en ligne augmentée de la largeur de la zone "i" soit inférieure à la coordonnée en ligne de la zone "j". Si c'est le cas, on passe en **d**, puis on quitte vers la partie de la figure 11.4.. Si ce n'est pas le cas, on passe par **e**, où l'on va tester la situation inverse où la zone "i" serait cette fois sous la zone "j". Si ce test s'avère positif, on quitte et la variable "place" est inchangée. Si ce n'est pas le cas, les deux zones se chevauchent et "place" prend une valeur négative. Dans les deux situations, on atteint ensuite la sortie de cette partie du module.

■ Si le test qui suit **a**, s'était conclut de façon négative, on passe alors par **b**, où selon le même principe que ci-dessus, on teste les zones "j" et "i" dans le cas où elles se trouvent sur la même ligne. Si c'est le cas, on passe par **h** au-delà duquel on teste les deux zones par rapport à leur position sur la droite ou la gauche l'une de l'autre. Si ce n'est pas le cas, on se dirige par **g**.

■ Au-delà de ce point on effectue une série de tests tenant compte du fait que les deux zones ne possèdent pas une coordonnée en ligne ou en colonne qui soit commune. Le premier test vérifie que si la zone "i" est au-dessus et sur la gauche de la zone "j", alors sa coordonnée en ligne augmentée de sa largeur doit être inférieure ou égale à la coordonnée en ligne de la zone "j" et de même pour sa coordonnée en colonne. Les trois autres tests vérifient les autres possibilités de placement de ces deux zones l'une par rapport à l'autre.

11.7. procédure Validation_Création

11.7.1. situation

est appelée par : "Création_Jeu"

appelle : /

11.7.2. présentation

Cette vérification est dite "syntaxique" car elle ne juge en aucun cas l'aspect éducatif ou qualitatif du jeu. Si la définition passe positivement au travers des tests composants cette procédure, on peut alors affirmer que le jeu est jouable.

Elle réalise deux types de vérifications sur la définition d'un jeu. Premièrement, elle vérifie si les éléments obligatoires qui doivent, au minimum, constituer la définition sont présents : un ensemble de jetons, une définition pour la règle de validation, des règles d'association, des règles de fin de jeu, une définition pour la règle de manipulation et une grille.

Si ces six éléments sont présents, on peut passer à la deuxième partie de la vérification qui consiste à déterminer si les définitions forment un ensemble cohérent.

Voici quelques uns des tests que cette procédure contient :

- Vérifier que tous les jetons définis ont reçu une définition compatible avec le mode de validation d'association choisi. Par exemple, si on a choisi l'association par caractéristiques, il faut que tous les jetons possèdent au moins une caractéristique.

- Il faut contrôler que les événements choisis pour la fin de la validation sont compatibles avec la définition des zones de l'écran. Par exemple, si on ne définit pas de zone d'attente, il est interdit de choisir "réorganiser la zone d'attente" parmi les événements de l'après validation.

- Il faut vérifier que les déplacements autorisés sont cohérents avec les zones définies. Il est en effet erroné d'accorder le déplacement entre la grille et la zone d'attente si cette dernière n'a pas été définie.

- Il faut contrôler que la règle de validation est compatible avec les zones de l'écran. Il serait, par exemple, irréaliste de ne choisir que la seule "sélection" comme mode de manipulation alors qu'un sabot est défini et contient tous les jetons au coup d'envoi de la partie.

En conclusion de cette batterie de tests, on sait si la définition fournit un jeu qui est (n'est pas) jouable. On remarque également que si la réponse est négative, deux types de problèmes peuvent être mis en évidence : des situations erronées et des situations qui ne sont pas erronées mais irréalistes. Par exemple, le troisième test conclut à une situation erronée tandis que le quatrième conclut à une situation irréaliste.

Nous interdiront les situations erronées. Pour les situations irréalistes, nous laisseront le concepteur poursuivre en l'avertissant simplement que la situation est irréaliste. Il peut, en effet, trouver un intérêt à produire une définition du deuxième type.

11.8. procédure Demande_Validation

11.8.1. situation

est appelée par : "Création_Jeu"

appelle : /

11.8.2. présentation

Cette procédure gère simplement un dialogue avec le concepteur. Elle lui demande s'il désire que la définition qu'il vient de réaliser soit ou non validée.

11.9. procédure Demande_Essai_Concepteur

11.9.1. situation

est appelée par : "Création_Jeu"

appelle : /

11.9.2. présentation

Cette procédure gère, elle aussi, un dialogue avec le concepteur. Elle lui demande s'il désire ou non tester la définition qui vient d'être validée.

11.10. procédure Essai_Concepteur

11.10.1. situation

est appelée par : "Création_Jeu"

appelle : /

11.10.2. présentation

Cette procédure permet au concepteur de tester si la définition qu'il vient de réaliser, et qui a été validée positivement, lui fournit un jeu répondant à ses attentes. Nous ne détaillons pas cette procédure ici car le chapitre suivant est consacré à cette partie dynamique du jeu.

12. Algorithme Partie Jeu

12.1. fonction Joue_Jeu

Voici l'architecture générale de cette fonction :

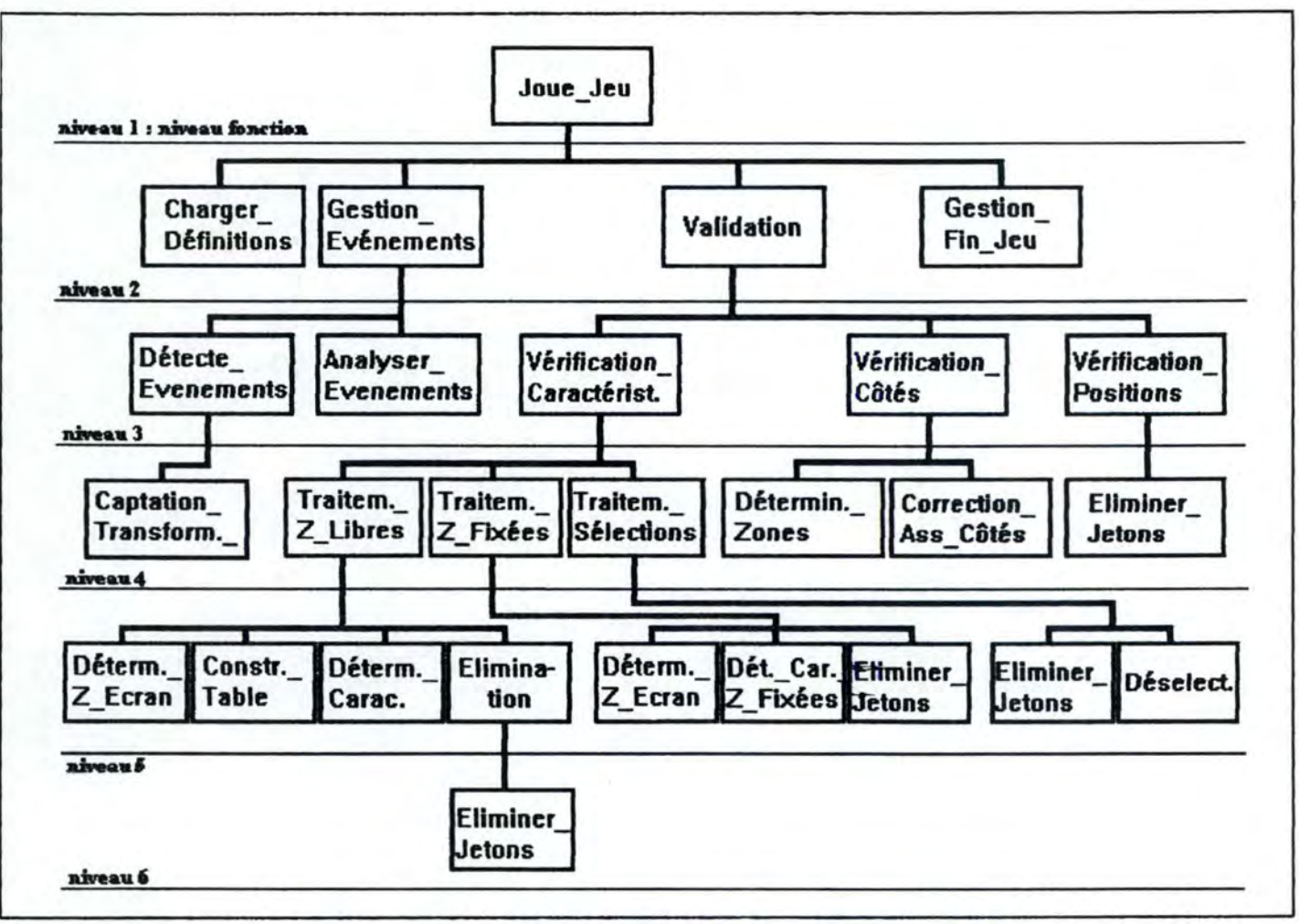


figure 12.1.

12.1.1. situation

est appelée par : /

appelle : "Charger_Définitions", "Gestion_Evénements", "Validation" et "Gestion_Fin _Jeu".

12.1.2. présentation

Cette fonction gère toute la partie active des jeux. Elle nécessite l'emploi de quatre procédures : "Charger_Définitions", "Gestion_Evénements", "Validation" et "Gestion_Fin_ Jeu" .

Lorsque l'on demande à jouer avec un jeu, on réalise, comme première opération, un test sur la valeur de la variable "valide". Cette variable a acquis une valeur dans la procédure "Validation_Création". Si la valeur est "vraie", cela veut dire que la définition est syntaxiquement correcte. On peut alors procéder au chargement des définitions de tous éléments constitutifs de la définition du jeu. Ensuite, tant que la condition de fin de jeu n'est pas atteinte, on doit (dans la boucle) détecter et analyser les événements produits par le joueur et les valider.

12.1.3. spécifications

objets utilisés

- **valide :** variable booléenne qui accompagne la définition du jeu et qui doit posséder une valeur positive afin de signaler que la définition a été contrôlée et qu'elle est au moins correcte d'un point de vue syntaxique; elle a acquis sa valeur dans la procédure "Validation_Création".

préconditions

- /

postconditions

- /

12.1.4. déroulement interne

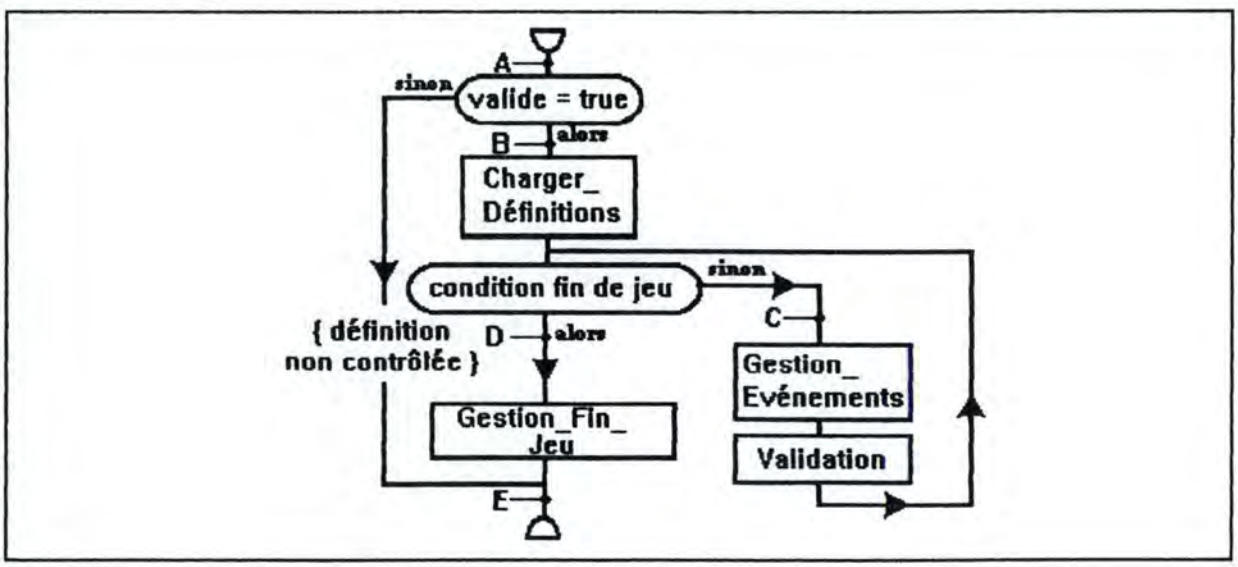


figure 12.2.

■ Après être entré en **A**, on teste la validité de la définition du jeu. Si celle-ci s'avère être correcte, on peut passer à la suite de la procédure, sinon on quitte la fonction.

■ On passe par **B**. On atteint le test qui contrôle si la condition de fin du jeu est atteinte. Tant qu'elle n'est pas vérifiée on entre dans la boucle et on passe par **C**.

■ Dans le corps de la boucle, on exécute systématiquement deux modules : la gestion (détection et analyse) des événements réalisés par le joueur suivie de la validation des actions qui découlent de ces événements. On repasse par le test. Si le test est vérifié, on quitte la boucle, on passe par **D**, et on effectue la gestion de la fin du jeu. Ensuite on sort et le jeu est terminé.

12.2. procédure Charger_Définitions

12.2.1. situation

est appelée par : "Joue_Jeu"

appelle : /

12.2.2. présentation

Cette procédure gère le chargement des éléments constitutifs de la définition d'un jeu. Lors de l'appel de la fonction "Joue_Jeu", ce sera la première procédure exécutée. Elle permet de disposer des définitions des jetons, des zones de l'écran et des règles. Ensuite, elle organise l'espace de jeu et le présente au joueur.

12.3. procédure Gestion_Fin_Jeu

12.3.1. situation

est appelée par : "Joue_Jeu"

appelle : /

12.3.2. présentation

Cette procédure gère les actions accompagnant la fin du jeu.

12.4. procédure Gestion_Evénements

12.4.1. situation

est appelée par : "Joue_Jeu"

appelle : "Détecter_Evénements", "Analyser_Evénements"

12.4.2. définitions

Un **événement** est une manipulation réalisée par le joueur.

Une **action** est un événement qui a été réalisé dans ou entre les zones de l'écran, où le joueur est autorisé à manipuler les jetons : la grille, le sabot et la zone d'attente.

Un **coup** est une action qui revient à déplacer un jeton vers la grille ou à sélectionner un jeton dans celle-ci. C'est donc une action significative pour l'atteinte de l'objectif du jeu.

12.4.3. présentation

Cette procédure a pour objectif de gérer les événements réalisés par le joueur. Elle se situe, dans la boucle de la fonction "Joue_Jeu", avant la procédure de validation. Elle gère les événements tant qu'un délai n'est pas dépassé ou que le joueur n'a pas réalisé le nombre de coups exigé. Tant que l'une de ces deux conditions n'est pas remplie, on exécute les procédures de la boucle.

12.4.4. spécifications

objets utilisés

- **i :** variable entière jouant le rôle de compteur des éléments d'une liste de jetons sélectionnés;
- **timer, oldtime :** variables prenant comme valeur le temps horloge en entrée dans le module; elles seront employées pour vérifier que le délai accordé au joueur entre deux validations n'est pas dépassé;
- **coup, oldcoup :** variables entières prenant pour valeur le nombre de coups joués depuis la dernière validation;
- **action :** variable booléenne prenant sa valeur dans les procédures de la boucle et signalant si l'événement produit par le joueur est ou non une action;
- **delai :** variable prenant sa valeur lors de la définition des règles de validation; elle détermine le délai maximum accordé au joueur entre deux validations;
- **coupaccord :** idem délai, mais pour le nombre de coups entre deux validations;

préconditions

- La condition de fin de jeu n'est pas atteinte.

postconditions

- Soit le délai est dépassé, soit le nombre de coups est atteint. Les événements ont été analysés.

12.4.5. déroulement interne

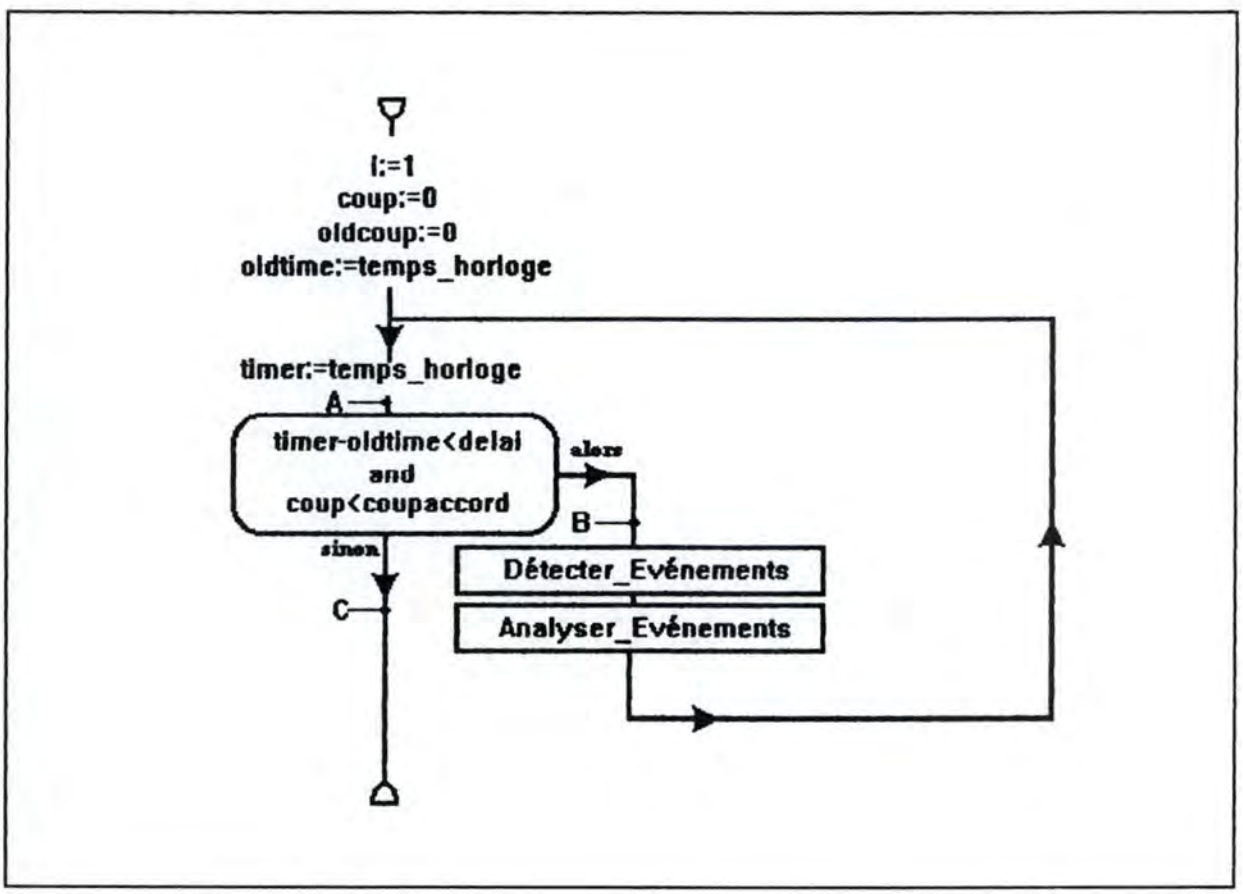


figure 12.3.

■ Après l'initialisation, on atteint le point **A**. On remarque que la variable "timer" prend sa valeur après le retour de la boucle, juste avant le test du délai.

■ Tant que le test n'est pas vérifié, on passe par **B**, où on réalise l'exécution des différents modules.

■ Si le test est vérifié, on peut alors passer vers la sortie.

12.5. procédure Détecter Evénement

12.5.1. situation

est appelée par : "Gestion_Evénements"

appelle : "Captation_&_Transformation_Coordonnées_Souris"

12.5.2. présentation

Cette procédure a pour objectif d'attendre et de détecter un événement du joueur. Il peut s'agir d'une sélection ou d'un déplacement. Elle fournit la nature de l'événement et ses coordonnées au module "Analyse_ Evénements".

L'étude d'une interface appropriée constituant un sujet indépendant, nous ne considérerons pas de moyen particulier pour la réalisation des deux types de manipulation que nous avons retenus. Cependant, nous citerons l'emploi de la souris pour donner un exemple de traitement des événements produit au moyen d'un objet interactif.

12.5.3. spécifications

objets utilisés

- **sel :** variable prenant les coordonnées de l'endroit où une sélection a eu lieu;
  sel.coordl : dans le cas où l'événement est une sélection, ce champ prend la valeur de la coordonnée, exprimée en cases, par rapport aux lignes de l'écran;
  sel.coordh : dans le cas où l'événement est une sélection, ce champ prend la valeur de la coordonnée, exprimée en cases, par rapport aux colonnes de l'écran;
- **depl :** elle prend les coordonnées des points de départ et d'arrivée du déplacement;
  depl.codepl : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux lignes de l'écran, exprimée en cases, du point de départ du déplacement;
  depl.codeph : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux colonnes de l'écran, exprimée en cases, du point de départ du déplacement;
  depl.coarrl : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux lignes de l'écran, exprimée en cases, du point d'arrivée du déplacement;
  depl.coarrh : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux colonnes de l'écran, exprimée en cases, du point d'arrivée du déplacement;
- **manip :** cette variable prend la valeur dep ou sel selon la nature de l'événement ;

préconditions

■ On n'a pas atteint le délai ni le nombre de coups à jouer; "timer" a une valeur de temps horloge supérieure à celle de "oldtime".

postconditions

■ On a détecté une manipulation du joueur, on connaît le type de l'événement et les coordonnées de la souris, exprimées en taille de case, qui s'y rapportent.

12.5.4. déroulement interne

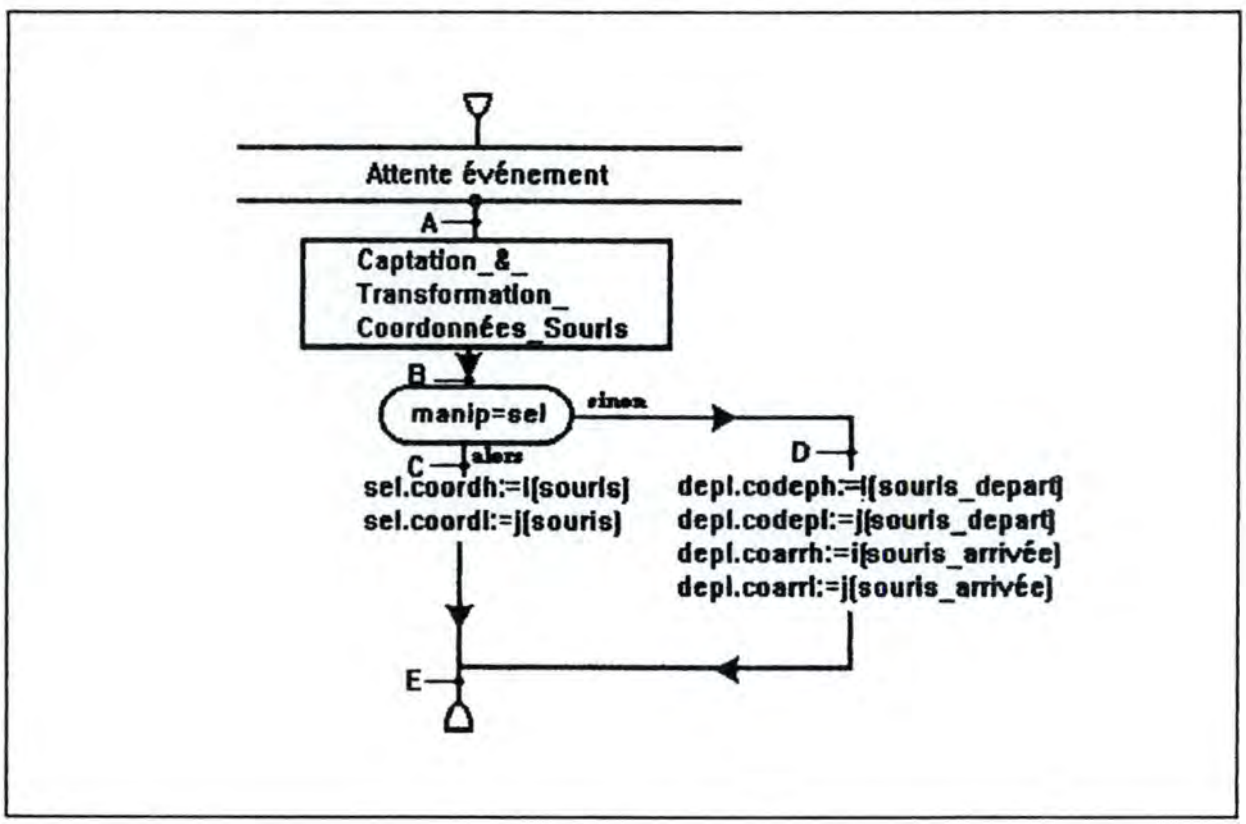


figure 12.4

■ Dès qu'un événement survient, on passe par **A**, on capte et on transforme les coordonnées de la souris au moyen de la procédure "Captation_&_Transformation_ Coordonnées_Souris".

■ Une fois ce traitement réalisé, on connaît les coordonnées de la souris et le type de manipulation. On passe par **B** et on teste la valeur de "manip". S'il s'agit d'une sélection, on passe par **C**, on attribue les valeurs transformées dans la procédure

"Captation_&_Transformation_Coordonnées_Souris" à la variable "sel". Dans le cas contraire, on passe par **D**, on attribue les valeurs à la variable "depl".

■ On quitte dans les deux cas en passant par **E**.

12.6. procédure Captation & Transformation Coordonnées Souris

12.6.1. situation

est appelée par : "Détecter_Evénements"

appelle : /

12.6.2. présentation

Cette procédure permet, pour chaque événement réalisé par le joueur, de capter les coordonnées de la souris, et de les transformer en coordonnées correspondant à une unité de mesure égale à la taille des cases de la grille.

12.7. procédure Analyse Evénement

12.7.1. situation

est appelée par : "Gestion_Evénements"

appelle : "Gestion_Liste_Sélection"

12.7.2. présentation

Cette procédure analyse la manipulation réalisée par le joueur et détermine si celle-ci constitue ou non une action.

Les situations à contrôler sont donc les suivantes :

■ Si le concepteur a choisi la sélection seule, il faut que les coordonnées de la souris tombent dans la grille sur une case non vide. Si ce n'est pas le cas, on ne peut pas considérer l'événement comme étant une action.

■ Si le concepteur a choisi les déplacements seuls, il faut constater si chaque déplacement s'est effectué entre les trois zones de l'écran (grille, sabot et zone d'attente), que celles-ci existent et qu'en plus de tels déplacements sont autorisés.

12.7.3. spécifications

objets utilisés

■ **sel :** variable prenant les coordonnées de l'endroit où une sélection a eu lieu;
sel.coordl : dans le cas où l'événement est une sélection, ce champ prend la valeur de la coordonnée, exprimée en cases, par rapport aux lignes de l'écran;
sel.coordh : dans le cas où l'événement est une sélection, ce champ prend la valeur de la coordonnée, exprimée en cases, par rapport aux colonnes de l'écran;
■ **depl :** variable prenant les coordonnées de l'endroit de départ et d'arrivée du déplacement;
depl.codepl : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux lignes de l'écran, exprimée en cases, du point de départ du déplacement;
depl.codeph : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux colonnes de l'écran, exprimée en cases, du point de départ du déplacement;
depl.coarrl : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux lignes de l'écran, exprimée en cases, du point d'arrivée du déplacement;
depl.coarrh : dans le cas où l'événement est un déplacement, ce champ prend la valeur de la coordonnée par rapport aux colonnes de l'écran, exprimée en cases, du point d'arrivée du déplacement;
■ **manip :** cette variable prend la valeur dep ou sel selon que l'événement soit un déplacement ou une sélection;
■ **coup, oldcoup :** variables entières, compteur du nombre de coups valides depuis la dernière validation;
■ **etat :** variable qui prend les valeurs "select" ou "deselect", en fonction de l'action du joueur; elle est employée par la procédure "Gestion_Liste_sélection";.
■ **choix :** variable reprenant le choix réalisé pour la règle de manipulation;
■ **tabdef :** cette variable structurée reprend les valeurs des définitions des zones de l'écran;

préconditions

■ Un événement a été détecté par la procédure précédente; on connaît les coordonnées qui s'y rapportent.

postconditions

■ Cet événement est analysé.

12.7.4. déroulement interne

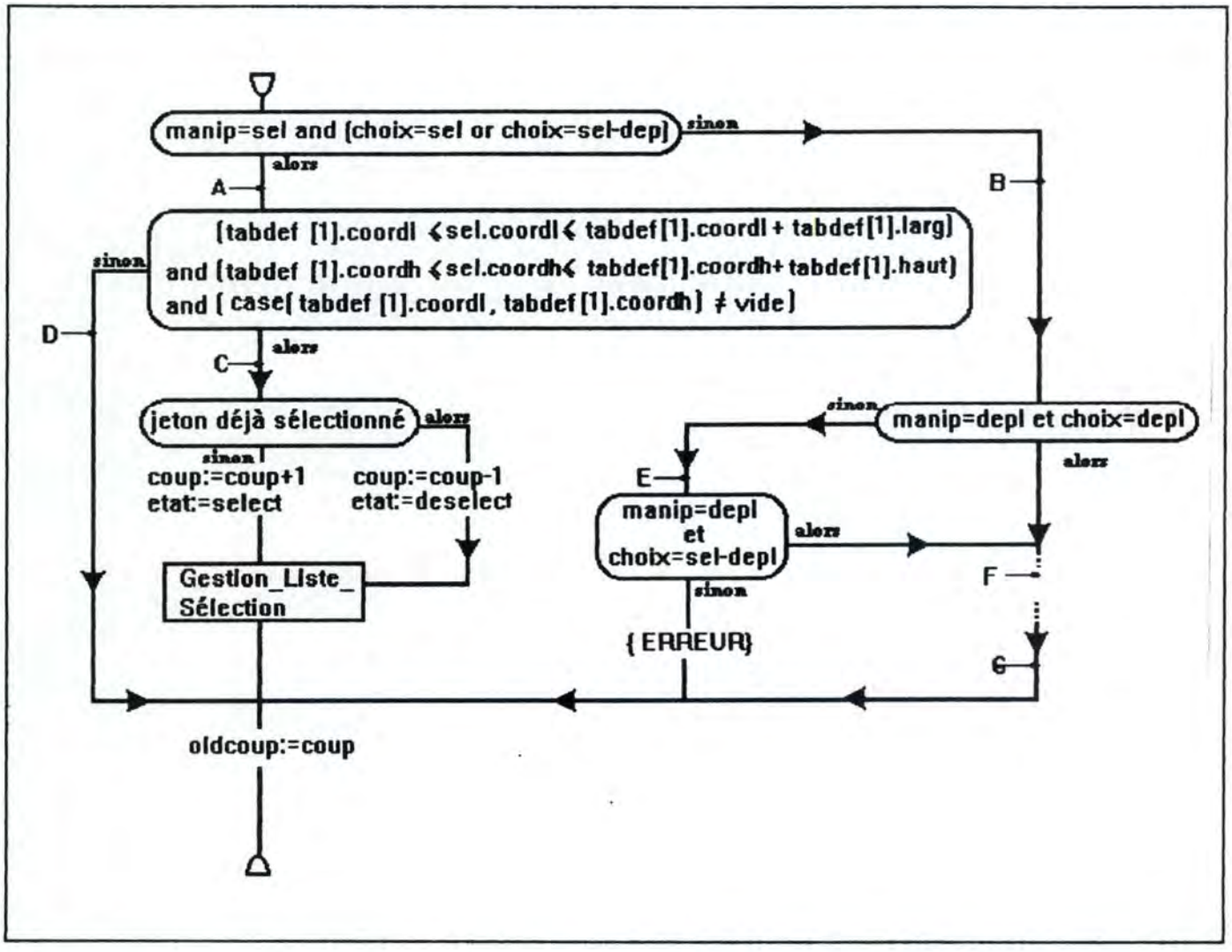


figure 12.5.

■ En entrant dans la procédure (figure 12.5.), on teste si l'événement était une sélection et si le choix de la règle de manipulation réalisé lors de la définition du jeu portait bien sur la sélection. Dans le premier cas, on passe par **A**. Dans le second cas, on passe par **B**.

■ Etant passé par **A**, on doit comparer les coordonnées de l'événement avec celles de la grille. En effet, nous avons fait l'hypothèse que la sélection ne pouvait s'effectuer que dans la grille. De plus, il faut que la case, si on est dans la grille, soit non vide. Si les trois conditions du test sont vérifiées, la manipulation du joueur constitue une action valide. Dans le cas contraire, il ne s'agit pas d'une action.

■ Si le test est vérifié, on passe par **C**. On rencontre alors un test. Si le jeton était déjà sélectionné, on doit en fait procéder à sa désélection. Ce qui veut dire que la variable "coup" doit être diminuée d'une unité. Dans le cas contraire, on augmente "coup" d'une unité car l'action valide peut-être considérée comme étant un coup joué. La variable "etat" sera employée par la procédure "Gestion_Liste_Sélection".

■ Si on était passé par **B**, la manipulation du joueur était donc du type déplacement. Au-delà de ce point, on teste la cohérence entre la manipulation et le mode de manipulation. La situation où le joueur a réalisé un déplacement alors que la "sélection" était le seul mode de manipulation permis constitue une erreur. Les deux autres cas, mènent à **F**, vers la figure 12.6..

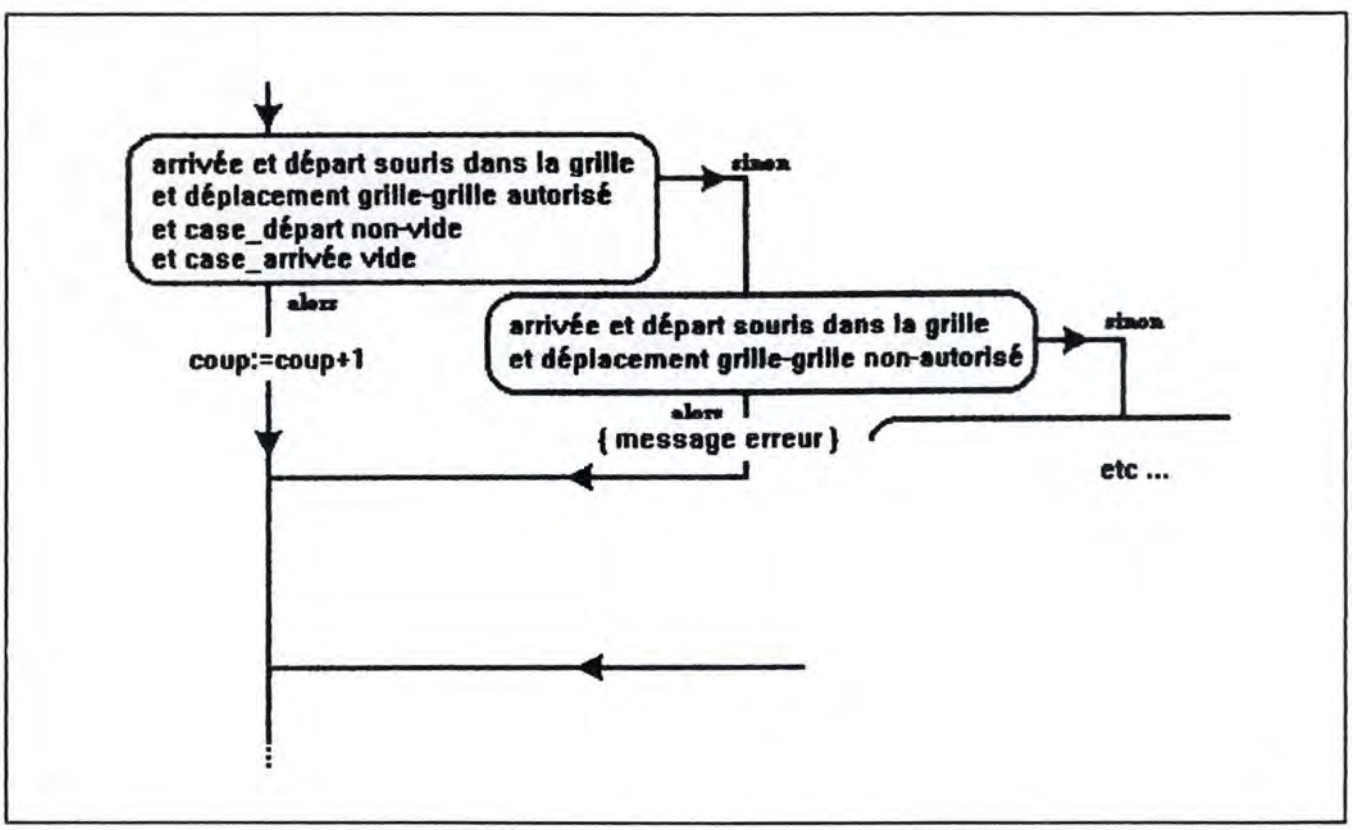


figure 12.6.

■ Dans la figure 13.6. Nous ne proposons pas une représentation complète pour cette partie de la procédure étant donné le nombre important de tests qu'elle comporte et la taille de chacun de ceux-ci. Les explications qui suivent suffiront à la compréhension de son fonctionnement.

■ Les deux tests suivants concluent à une action, et donc à **coup:=coup+1** :

▪ Si les coordonnées de départ sont dans le sabot, que la case de départ est non-vide, que les coordonnées d'arrivée sont dans la grille et que la case d'arrivée est vide, alors la manipulation est une action valide qui correspond à un coup joué. La variable coup est incrémentée d'une unité. De plus, si le mode de manipulation est la sélection cojuguée aux déplacements, il est nécessaire de mettre à jour la liste des jetons sélectionnés.

▪ Si les coordonnées de départ sont dans la zone d'attente, que les déplacements entre la zone d'attente et la grille sont autorisés, que la case de départ est non-vide, que les coordonnées d'arrivée sont dans la grille et que la case d'arrivée est vide, alors la manipulation est une action valide qui correspond à un coup joué. La variable coup est incrémentée d'une unité. De plus, si le mode de manipulation est la sélection cojuguée aux déplacements, il est nécessaire de mettre à jour la liste des jetons sélectionnés.

■ Le test suivant conclut à un statu quo, et donc à **coup:=coup** :

▪ Si les coordonnées de départ et d'arrivée sont dans la grille, que la case de départ est occupée, que la case d'arrivée est vide et que le déplacement au sein de la grille est autorisé, alors la manipulation est une action valide qui correspond à un simple changement de position d'un coup joué. La variable coup est inchangée.

■ Les deux tests suivants concluent à un retrait d'action, et donc à **coup:=coup-1** :

▪ Si les coordonnées de départ sont dans la grille et que les coordonnées de la case d'arrivée sont dans le sabot, que la case de départ est occupée, que la case d'arrivée est vide et que le déplacement de la grille vers le sabot est autorisé, alors la manipulation est une action valide qui correspond au retrait d'un coup joué. La variable coup est décrémentée d'une unité. De plus, si le mode de manipulation est la conjugaison du déplacement et de la sélection, il est nécessaire d'éliminer le jeton de la liste de sélection.

▪ Si les coordonnées de départ sont dans la grille et que les coordonnées de la case d'arrivée sont dans la zone d'attente, que la case de départ est occupée, que la case d'arrivée est vide et que le déplacement de la grille vers la zone d'attente est autorisé, alors la manipulation est une action valide qui correspond au retrait d'un coup joué. La variable coup est décrémentée d'une unité. De plus, si le mode de manipulation est la

conjugaison du déplacement et de la sélection, il est nécessaire d'éliminer le jeton de la liste de sélection.

■ Les situations suivantes sont deux actions de réorganisation de l'espace de jeu par le joueur. Elles engendreront, pour la plupart, l'activation d'une procédure dont la fonction est la réorganisation de la zone dont il est question.

▪ Si les coordonnées de départ sont dans la zone d'attente, que les déplacements entre la zone d'attente et le sabot sont autorisés, que la case de départ est non-vide, que les coordonnées d'arrivée sont dans le sabot et que la case d'arrivée est vide, alors la manipulation est une action valide mais ne correspondant pas à un coup joué. La procédure de gestion du sabot doit être appelée.

▪ Si les coordonnées de départ sont dans le sabot, que les déplacements entre la zone d'attente et le sabot sont autorisés, que la case de départ est non-vide, que les coordonnées d'arrivée sont dans la zone d'attente et que la case d'arrivée est vide, alors la manipulation est une action valide, mais ne correspondant pas à un coup joué. La procédure de gestion de la zone d'attente doit être appelée.

■ En dehors de ces situations, il s'agira d'une manipulation erronée et **coup sera inchangée**.

12.8. procédure Gestion_Liste_Sélection

12.8.1. situation

est appelée par : "Analyse _Evénements"

appelle : /

12.8.2. présentation

Cette procédure gère une liste contenant l'identifiant et la position courante dans la grille de tous les jetons sélectionnés depuis la dernière validation.

12.9. procédure Validation

12.9.1. situation

est appelée par : "Joue_Jeu"

appelle : "Vérification_Caractéristiques", "Vérification_Côtés", "Vérification_Positions"

12.9.2. présentation

Cette procédure renferme toutes les opérations qui assurent la validation de la (des) règle(s) d'association déterminée(s) par le concepteur. Dans l'architecture générale, elle intervient après la procédure "Gestion_Evénements" chaque fois qu'au sein de cette dernière le test gérant le moment de validation est vérifié (c'est-à-dire soit un timer dépassé, soit un nombre de coups déterminés atteint).

Le principe est de vérifier si la (les) règle(s) d'association est (sont) respectée(s). Cette vérification s'effectue au niveau du contenu (un jeton ou le vide) de chacune des cases de la grille. Nous avons décidé de réaliser cette vérification à partir de la case du coin supérieur gauche (indicée (1,1)), ligne par ligne, de gauche à droite pour atteindre en fin de vérification le coin inférieur droit (indicé (h,l)).

La gestion des indices est illustrée par la figure 12.7 :

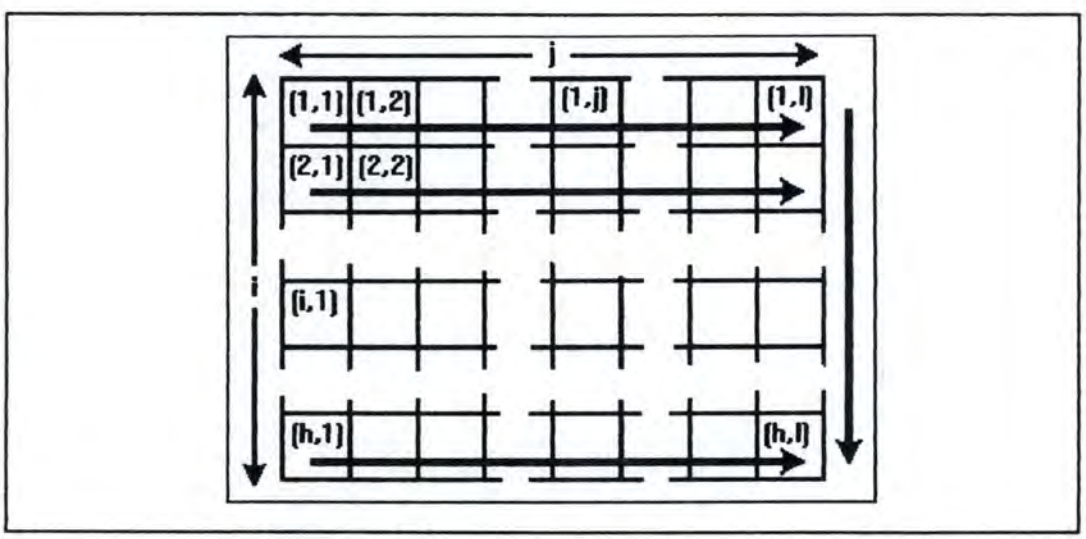


figure 12.7.

■ Après l'initialisation, on arrive en **A**.

■ On passe obligatoirement par au moins un des trois tests sur la valeur de l'indicateur. Si le premier test est vérifié on appelle la procédure gérant la vérification par caractéristiques. Sinon, on passe par le deuxième test. S'il est vérifié, on appelle la procédure gérant la vérification de l'association par côtés. Si ce deuxième test n'est pas vérifié, on passe au troisième test. Si celui-ci est positif, on appelle successivement les deux procédures précédemment citées. Sinon, il s'agit de l'association par positions.

12.10. procédure Validation_Association_Caractéristiques

12.10.1. situation

est appelée par : "Validation"

appelle : "Traitement_Zones_Libres", "Traitement_Zones_Fixes", "Traitement_Sélection"

12.10.2. présentation

Cette procédure vérifie la qualité de l'association par caractéristiques. Deux situations entraînent son exécution : une valeur pour "ind" = 1 ou = 3 dans la procédure "Validation". Elle assure la vérification de trois situations liées à ce mode d'association : association en zones libres, association en zones fixées et association de pièces sélectionnées.

12.10.3. spécifications

objets utilisés

/

préconditions

■ La condition de déclenchement de la validation est réalisée.

postconditions

■ Toutes les cases de la grille ont été vérifiées, ligne par ligne, de haut en bas, en partant du coin supérieur gauche de la grille. Le contenu de la grille est inchangé si les jetons qu'elle comprenait avant l'entrée dans le module étaient correctement associés par rapport au choix de l'association par caractéristiques défini par le concepteur du jeu. Dans le cas contraire, la grille est réorganisée pour répondre aux exigences de l'association.

12.10.4. déroulement interne

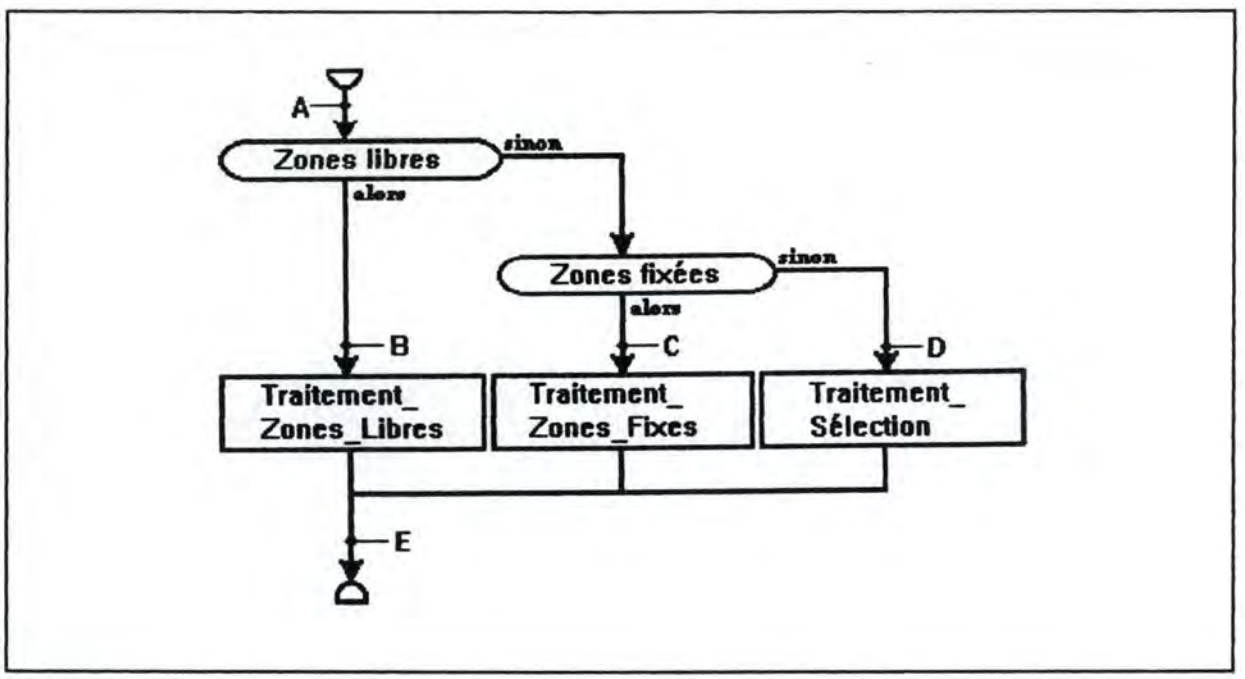


figure 12.9.

12.11. procédure Traitement_Zones_Libres

12.11.1. situation

est appelée par : "Validation_Association_Caractéristiques"

appelle : "Détermination_Zones", "Elimination", "Construction_Table", "Détermination_Caractéristique(s)",

12.11.2. présentation

Cette première procédure appelée par la procédure "Validation_Association_ Caractéristiques" gère la situation où l'association par caractéristiques a été définie en zones libres lors de la conception du jeu. Pour réaliser cette opération, on travaille de zones en zones dans la grille. Dans un premier temps, on recense pour toutes les cases non vide d'une zone, la (les) caractéristique(s) (définie(s) par le concepteur lors de la définition des jetons), de tous les jetons que le joueur a placés. Une fois cela réalisé, on regarde qu'elle(s) est (sont) la (les) caractéristique(s) que l'on retrouve(nt) le plus fréquemment pour un jeton. Enfin, on élimine de la zone tous les jetons ne correspondant pas à cette (ces) caractéristique(s). La nature de l'élimination dépend de la définition donnée pour la règle de validation.

L'exécution de cette procédure se fait en collaboration avec la procédure de détermination des zones de la grille, qui fournit les valeurs délimitant chaque zone, au fur et à mesure que l'on progresse dans la grille.

12.11.3. spécifications

objets utilisés

- **encore :** variable booléenne qui prend sa valeur dans la procédure de détermination des zones; si sa valeur est positive, il reste encore au moins une zone non encore vérifiée;
- **z :** variable compteur des éléments de la variable "listcar" (cfr "Détermination_ Caractéristique(s)"); elle est utilisée par la procédure "Détermination_Caractéristique(s)";

préconditions

- On se trouve dans la situation d'association par caractéristiques en zones libres.

postcondition

- On a vérifié la correction de l'association par caractéristiques en zones libres de tous les jetons placés dans la grille par le joueur depuis le début de la partie. On a réalisé cette opération au sein de chaque zone de la grille. Tous les jetons mal placés selon les conventions ont été éliminés de leur zone.

12.11.4. déroulement interne

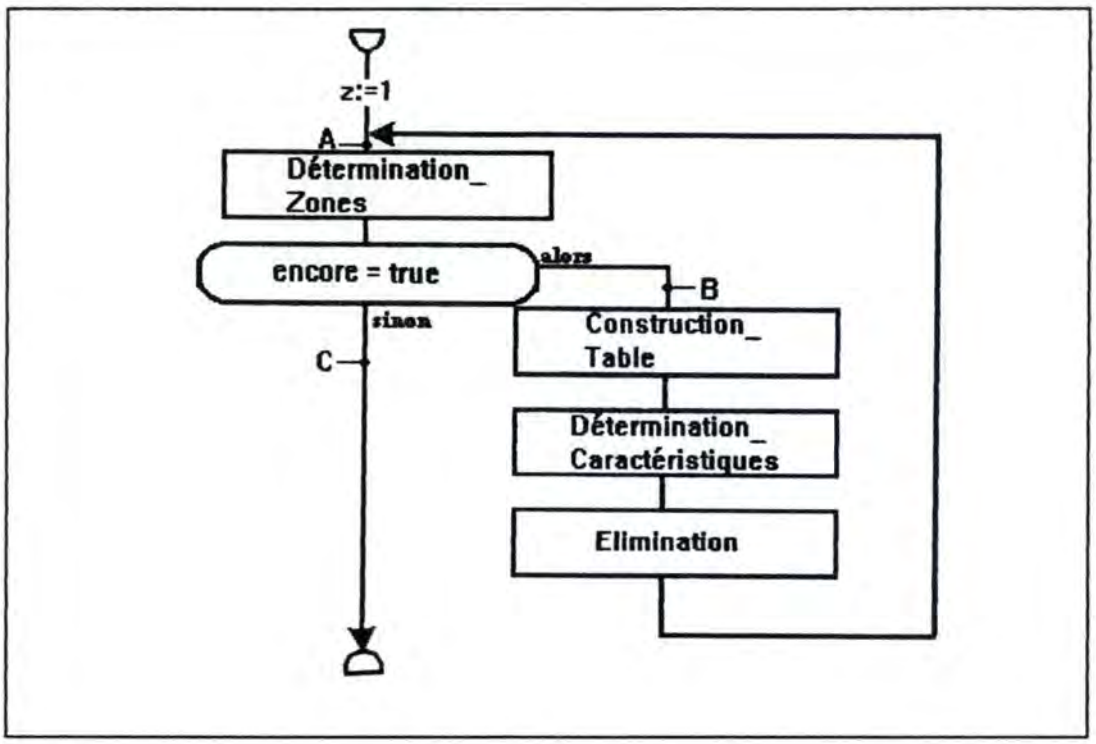


figure 12.10.

■ Le traitement de la grille s'effectue de zone en zone. Au sein d'une zone, on procède en trois étapes (correspondant chacune aux trois procédures qui suivent). On réalise dans un premier temps une table reprenant dans chacune de ses lignes le contenu d'une case. Ensuite, on détermine qu'elle(s) est (sont) la (les) caractéristique(s) qui apparai(ssen)t le plus fréquemment. Pour terminer, on élimine tous les jetons ne correspondant pas à celle(s)-ci.

■ Cette procédure débute, après passage en **A** par l'exécution de la procédure déterminant les zones de la grille. Son résultat est évidemment fonction de la définition donnée par le concepteur. Le test qui suit porte sur la valeur qui sera prise par le booléen "encore" à la sortie de la procédure "Détermination_Zones". Si ce test se révèle positif cela veut dire qu'il reste encore au moins une zone de la grille à traiter et on passe en **B**. Sinon toutes les zones on été vérifiées et on passe en **C**.

12.12. procédure Détermination_Zones

12.12.1. situation

est appelée par : "Traitement_Zones_Libres", "Traitement_Zones_Fixes", "Association_Côtés"

appelle : /

12.12.2. présentation

Cette procédure détermine les coordonnées de toutes les zones de la grille. Elle fournit les valeurs de celles-ci à d'autres procédures de même niveau. Par exemple à la procédure "Construction_Table". Ces valeurs ont été déterminées par le concepteur lors de la définition de la grille.

Elle fournit :

- **m :** variable entière qui est égale au nombre de colonnes de la grille jusqu'à la première colonne de la zone courante inclue; elle correspond à la première des deux valeurs constituant le couple des coordonnées désignant le coin supérieur gauche de la zone;
- **n :** variable entière qui est égale au nombre de lignes de la grille jusqu'à la première ligne de la zone courante inclue; elle correspond à la deuxième valeur constituant le couple des coordonnées désignant le coin supérieur gauche de la zone;
- **p :** variable entière qui est égale au nombre de colonnes de la zone courante;
- **q :** variable entière qui est égale au nombre de lignes de la zone courante;

De plus, elle détermine la valeur de la variable booléenne "encore" qui permet de tester si toutes les zones de la grille ont ou non été vérifiées.

12.13. procédure Eliminer_Jetons

12.13.1. situation

est appelée par : "Traitement_Zones_Libres", "Traitement_Zones_Fixes", "Traitement_Jetons_Sélectionnés", "Correction_Association_Côtés", "Association_Positions", "Elimination".

appelle : /

12.13.2. présentation

Cette procédure gère l'élimination des jetons mal associés. D'un jeu à l'autre, cette élimination peut consister en divers événements. Par exemple, dans certains jeux, les jetons seront éliminés de la grille et retournerons vers le sabot ou la zone d'attente. Dans d'autres jeux, l'élimination sera plus radicale et les jetons seront éliminés définitivement du jeu. L'événement correspondant à un jeu particulier aura été choisi, par le concepteur dans une liste, au cours de la définition de la règle de validation.

12.14. procédure Construction_Table

12.14.1. situation

est appelée par : "Traitement_Zones_Libres"

appelle : /

12.14.2. présentation

Pour une zone, nous allons lister le contenu de toutes les cases, en reprenant premièrement les coordonnées de la case, deuxièmement la (les) caractéristique(s) déterminée(s) par le concepteur pour le jeton qui est contenu dans cette case et troisièmement un compteur qui indique que c'est la xième fois que l'on rencontre cette (ces) caractéristique(s) depuis le début de la vérification de cette zone.

Si une case est vide, le champ consacré à la caractéristique reste vide et le compteur reste égal à zéro.

12.14.3. spécifications

objets utilisés

- **j :** variable entière jouant le rôle de compteur des colonnes de la zone courante de la grille;
- **i :** variable entière jouant le rôle de compteur des lignes de la zone courante de la grille;
- **m :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la grille jusqu'à la première colonne de la zone courante inclue;

■ **n** : variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la grille jusqu'à la première ligne de la zone courante inclue;
■ **p** : variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la zone courante;
■ **q** : variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la zone courante;
■ **t,u** : variables entières compteurs des éléments de la table;
■ **tab** : variable structurée reprenant les informations sur le contenu des jetons présents dans la zone; la taille de cette table est variable et dépend du nombre de cases de la zone;

tab.coord : champ prenant le couple des coordonnées d'une case;

tab.car : champ prenant la ou les caractéristique(s) déterminée(s) par le concepteur pour le jeton courant, cette variable est du type chaîne de caractère;

tab.cmpt : champ jouant le rôle d'une variable entière qui exprime le nombre de fois que l'on a déjà rencontré cette ou ces caractéristique(s) depuis le début du traitement de cette zone;

■ **case** : variable structurée pour laquelle nous avons retenu dans ce module les attributs couple de coordonnées et un état occupé ou inoccupé;
■ **jeton** : variable structurée pour laquelle nous avons retenu dans ce module les attributs couple de coordonnées et une (plusieurs) caractéristique(s) d'association;

préconditions

■ n et m ont acquis une valeur dans la routine de détermination des zones et assurent que l'on se situe, à l'entrée de la procédure, au niveau de la première case en haut à gauche de la zone courante.
■ p et q ont également une valeur qui permettent de définir exactement la hauteur et la largeur de la zone courante.

postconditions

■ On atteint le point **F** en ayant vérifié le contenu de chaque case de la zone courante.
■ La table est garnie et contient autant de lignes qu'il y avait de cases dans la grille (c'est en fait la valeur de u); chaque ligne de la table est garnie des attributs (tels que nous les avons définis) de la case qui y correspond; si la case était vide, le champ caractéristiques est garni d'un blanc et le champ compteur est égal à zéro.
■ m,n,p et q sont inchangées.
■ Le contenu de la zone est inchangé.

12.14.4. déroulement interne

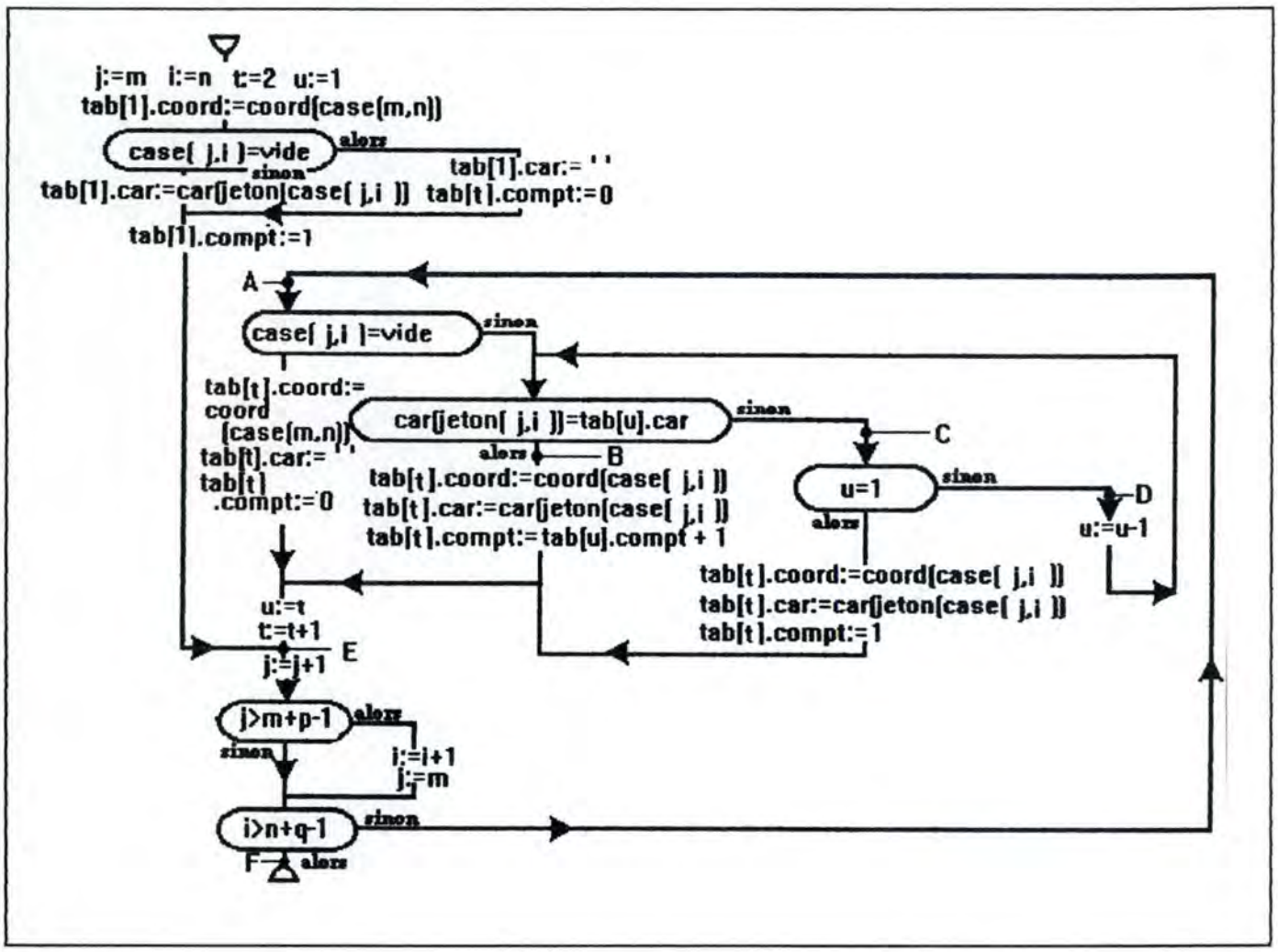


figure 12.11

■ On entre dans la procédure au point **E**. On passe alors à la case suivante de la zone courante.

■ On atteint le point **A** pour la première fois après avoir désigné la case suivante (pour autant que la zone contienne plus d'une case).

■ Si la case est vide, on complète le deuxième champ de la deuxième ligne de la table par un blanc et le troisième champ par un zéro.

■ Si la case n'est pas vide, on teste la (les) caractéristique(s) du jeton courant avec la (les) caractéristique(s) de la dernière ligne de la table, que l'on vient juste de garnir. Si elles sont égales, on arrive en **B**, on garnit les deux premiers champs de la ligne courante

de la table (indicée par "t") et on augmente la valeur compteur du champ compteur de la dernière ligne de la table (indicée par "u") de 1 et on place cette valeur dans le troisième champ de la ligne courante de la table.

■ Si la (les) caractéristique(s) n'étai(en)t pas la(les) même(s), on va rechercher dans la table, la dernière ligne la plus récente possédant la (les) caractéristique(s) identique(s) à celle(s) du jeton courant. D'où on décrémente "u" pour partir du dernier au premier élément de la table. Si on trouve une telle ligne, on passe en **B**. Sinon lorsque l'on atteint et vérifie sans fruit le premier élément de la table (quand u=1), on sait qu'il s'agit d'un jeton dont la (les) caractéristique(s) n'ont pas encore été rencontrée(s).

■ On a garni la ligne courante de la table, on prépare une nouvelle ligne avec "u" prenant la valeur de "t". "t" est ensuite augmentée d'une unité.

■ On revient en **E**. A ce point on se positionne sur la case suivante de la zone.

■ Tant que le test "i>n+q+1" n'est pas vérifié, on reste dans la boucle. Lorsqu'il est vérifié, on peut quitter celle-ci avec la certitude d'avoir vérifié le contenu de toutes les cases de la zone et d'avoir établit une table y correspondant.

12.15. procédure Détermination_Caractéristique(s)

12.15.1. situation

est appelée par : "Traitement_Zones_Libres"

appelle : /

12.15.2. présentation

En conclusion de la procédure précédente, nous avons à notre disposition une table. Dans celle-ci, nous allons déterminer qu'elle est (sont) la (les) caractéristique(s) pour un jeton qui est (sont) apparue(s) le plus souvent.

Rappelons que nous avons posé trois hypothèses pour le choix de la (des) caractéristique(s) dans une zone.

La première est que la (les) caractéristique(s) qui apparai(ssen)t en majorité correspond(ent) à la famille de jetons que le joueur désire associer dans cette zone de l'écran.

Cette hypothèse est cruciale si l'on désire une vérification automatique suivie d'une élimination automatique des jetons non correctement associés.

La deuxième hypothèse permet de départager des familles de jetons dont la (les) caractéristique(s) apparai(ssen)t simultanément un nombre de fois maximum et égal dans une même zone. Il faut se résoudre à en éliminer une des deux. Prenons, par exemple, le cas ou les zones de l'écran sont constituées de deux cases. Si cette hypothèse n'est pas appliquée, le fait de remplir la zone avec deux jetons quelconques possédant des caractéristiques différentes serait valable. En effet, celles-ci apparaîtraient un nombre de fois égal et maximum, égal à un. Arbitrairement, nous avons choisi de conserver la (les) caractéristique(s) apparue(s) le plus tôt dans la table.

La troisième hypothèse traite le cas où une même caractéristique apparaît un nombre de fois maximum dans plusieurs zones différentes. Nous avons choisi de conserver la caractéristique dans la zone vérifiée en premier lieu et de l'éliminer des autres.

12.15.3. spécifications

objets utilisés

- **i :** variable entière jouant le rôle de compteur;
- **u :** variable entière correspondant à la taille en nombre de lignes de la table, déterminée dans la procédure "Construction_Table";
- **tab :** variable structurée reprenant les informations sur le contenu des jetons présents dans la zone; la taille de cette table est variable et dépend du nombre de cases de la zone;
  - **tab.coord :** champ reprenant la coordonnée d'une case, cette variable est du type couple de coordonnées (x,y);
  - **tab.car :** champ reprenant la (les) caractéristique(s) déterminée(s) par le concepteur pour le jeton courant, cette variable est du type string;
  - **tab.compt :** champ reprenant une variable entière qui exprime le nombre de fois que l'on a déjà rencontré cette ou ces caractéristique(s) depuis le début du traitement de cette zone;
- **max :** variable entière, elle prendra la valeur maximale du troisième champ des lignes de la table;
- **car :** variable de type string, elle prendra la (les) caractéristique(s) de la ligne de la table dont le champ compteur a une valeur maximale;
- **listcar :** liste de toutes les caractéristiques apparues depuis le début de la validation de la grille; chaque élément de la liste correspond à la caractéristique apparue un nombre de fois maximum dans chacune des zones vérifiées depuis le début de cette nouvelle session de validation;
- **z :** variable entière jouant le rôle de compteur des éléments de "listcar";

préconditions

■ On dispose d'une table, dont le contenu a été déterminé par la procédure "Construction_Table"; la longueur de cette table est égale à la valeur finale de la variable "u" déterminée, elle aussi, par la procédure "Construction_Table"

postconditions

■ On atteint le point **C** en connaissant la (les) caractéristique(s) les plus fréquemment apparue(s) parmi les jetons de la zone courante contenue(s) dans la variable "car";
■ Le contenu de la zone de la grille est inchangé;

12.15.4. déroulement interne

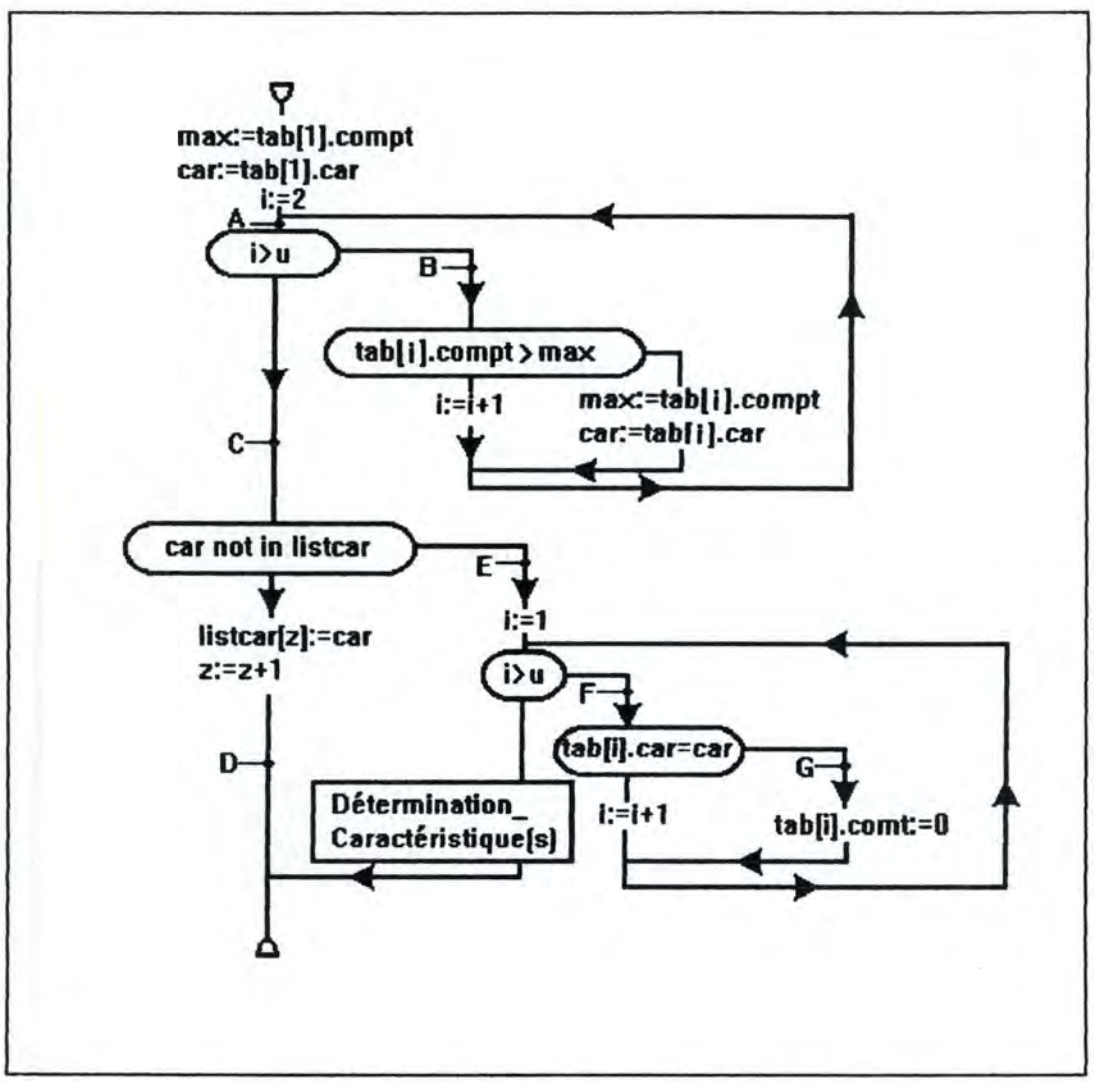


figure 12.12

■ Après l'initialisation, les préconditions sont respectées. On entre dans la procédure au point **A**.

■ Si le test n'est pas vérifié, on passe par **B** où l'on teste la valeur du champ compteur de la ligne courante avec la valeur de la variable max. Le max ne sera remplacé que si la valeur du champ est strictement supérieure à la variable max. Ceci pour respecter notre seconde hypothèse.

■ Que le test s'avère ou non positif, on passe ensuite à l'incrémentation du compteur i, puis on retourne au point **A**.

■ Si le test sur la valeur de i est vérifié on passe en **C**. Il faut alors vérifier que la (les) caractéristique(s) n'a (ont) pas déjà été choisie(s) pour une zone vérifiée précédemment (depuis le début de cette nouvelle session de validation). Comme premier test, on regarde si "car" appartient ou non à "listcar". Si elle n'appartient pas, on augmente "listcar" de la valeur de "car".

■ Si elle appartient, on passe par **E**. On doit éliminer cette caractéristique et choisir celle qui vient directement après elle. Pour réaliser cela, nous mettons à 0 tous les champs ".comp" des lignes de la table correspondant à des jetons dont la (les) caractéristique(s) est (sont) égale(s) à "car". On fait cela tant que "i" est inférieur à "u", c'est-à-dire pour toute la table.

■ Une fois que "i" dépasse "u", on sait que la table a été traitée, on quitte la boucle et on refait appel à la procédure elle-même afin de déterminer une nouvelle valeur pour "car", correspondant à la (aux) caractéristique(s) apparue(s) un nombre de fois juste inférieur à l' (aux) ancienne(s) caractéristique(s).

12.16. procédure Elimination

12.16.1. situation

est appelée par : "Traitement_Zones_Libres"

appelle : "Eliminer_Jeton"

12.16.2. présentation

En conclusion des deux procédures précédentes, nous avons à notre disposition la connaissance de la (des) caractéristique(s) sur base de laquelle (des quelles) le joueur a réalisé

(du moins c'est ce que nous supposons de façon automatique) l'association des jetons. Dans cette nouvelle procédure, nous allons réaliser l'élimination de tous les jetons non correctement associés, c'est-à-dire ne possédant pas la (les) caractéristique(s) choisie(s).

Cette procédure s'avère donc être assez simple, puisqu'il suffit de reprendre chaque case de la zone courante et de comparer la (les) caractéristique(s) du jeton contenu avec celle(s) de la variable "car".

12.16.3. spécifications

objets utilisés

- **j** : variable entière jouant le rôle de compteur des colonnes de la zone courante de la grille;
- **i** : variable entière jouant le rôle de compteur des lignes de la zone courante de la grille;
- **m** : variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la grille jusqu'à la première colonne de la zone courante inclue;
- **n** : variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la grille jusqu'à la première ligne de la zone courante inclue;
- **p** : variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la zone courante;
- **q** : variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la zone courante;
- **car** : variable du type chaîne de caractères, elle reprend la (les) caractéristique(s) de la ligne de la table dont le champ compteur a une valeur maximale et a été déterminée dans la procédure "Détermination_Caractéristique(s)";
- **case** : variable structurée pour laquelle nous avons retenu les attributs couple de coordonnées et un état occupé ou inoccupé;
- **jeton** : variable structurée pour laquelle nous avons retenu les attributs couple de coordonnées et une (plusieurs) caractéristique(s) d'association;

préconditions

- La variable "car" a acquis une valeur dans la procédure "Détermination_ Caractéristiques";
- m,n,p et q sont inchangées depuis la sortie de la procédure "Détermination_Zones";

postconditions

- On atteint le point **C** en ayant traité l'intégralité des jetons contenus dans les cases de la grille; ce traitement est l'élimination des jetons non correctement associés.

12.16.4. déroulement interne

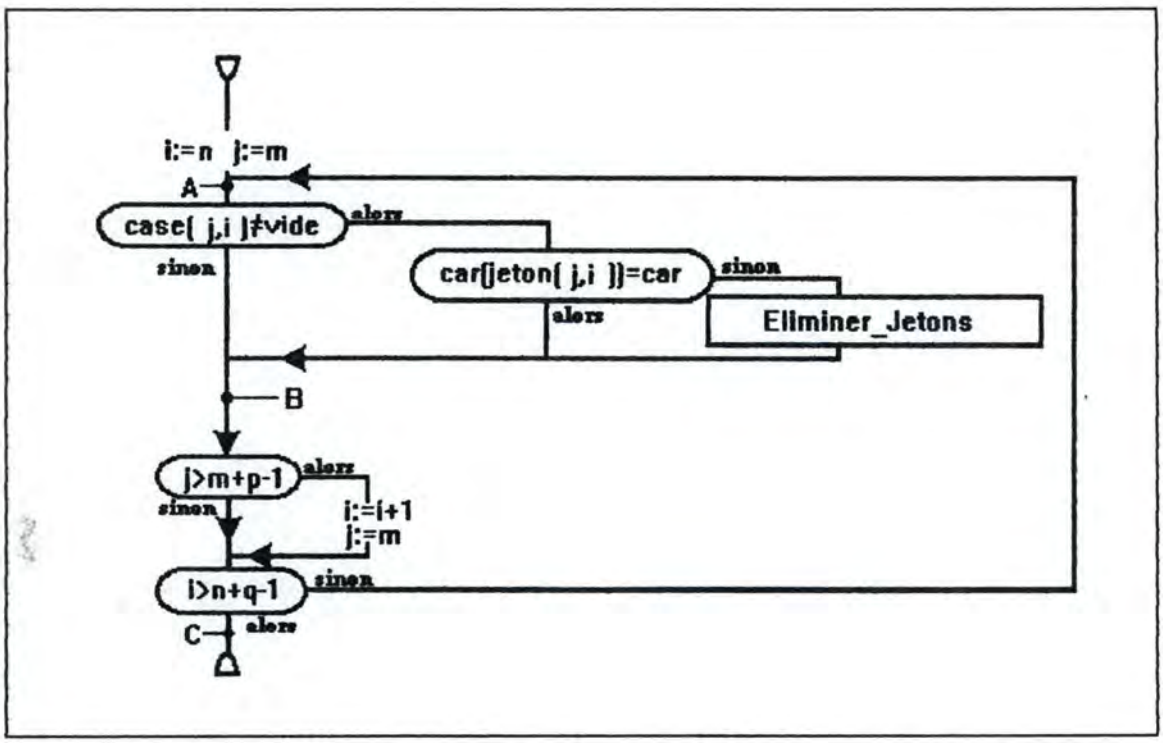


figure 12.13

■ On entre dans la procédure au point **A**. Si le test n'est pas vérifié, on passe en **B**, ce qui correspond à une case de coordonnées (m,n) vide. Si par contre la case n'est pas vide, on compare la (les) caractéristique(s) du jeton contenu avec la variable "car". Si le test est vérifié, le jeton est conservé, sinon, il est éliminé.

■ Après **B**, on gère le passage à la case suivante, qui est réalisé de la même façon que dans la section 12.13.. Si le test qui clôture cette série d'opérations s'avère positif, cela veut dire que l'on a vérifié chaque case de la zone courante. On retourne alors vers la procédure de détermination des zones de la grille afin de déterminer la zone suivante.

12.17. procédure Traitement_Zones_Fixes

12.17.1. situation

est appelée par : "Validation_Association_Caractéristiques"

appelle : "Détermination_Zones", "Elimination_Jetons", "Détermination_Caractéristique(s)_Z_Fixes"

12.17.2. présentation

Cette procédure gère la situation où l'association par caractéristiques à été définie en zones fixées. Pour réaliser cette opération, on travaille par zones dans la grille. On compare chacun des jetons contenu dans une zone à la (aux) caractéristique(s) définie(s) par le concepteur, pour cette zone.

La (les) caractéristique(s) de référence pourra(ont) être, par exemple, celle(s) d'un jeton posé sur la grille avant le début du jeu ou communiquée(s) dans la zone de message.

12.17.3. spécifications

objets utilisés

- **j :** variable entière jouant le rôle de compteur des cases d'une ligne de la zone courante de la grille et donc des colonnes de la grille;
- **i :** variable entière jouant le rôle de compteur des lignes de la zone courante de la grille;
- **m :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la grille jusqu'à la première colonne de la zone courante incluse;
- **n :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la grille jusqu'à la première ligne de la zone courante incluse;
- **p :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la zone courante;
- **q :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la zone courante;
- **case :** variable structurée pour laquelle nous avons retenu comme attributs un couple de coordonnées et un état occupé ou inoccupé;
- **jeton :** variable structurée pour laquelle nous avons retenu comme attributs un couple de coordonnées et une (plusieurs) caractéristique(s) d'association;

■ **car :** variable qui reprend pour la zone la (les) caractéristique(s) que le concepteur a déterminée(s); la valeur de cette variable est déterminée par la procédure "Détermination_Caractéristique(s)_Z_Fixes";
■ **encore :** variable booléenne qui prend sa valeur dans la procédure "Détermination_ Zones"; si sa valeur est true, cela signifie qu'il reste encore au moins une zone non encore vérifiée;

préconditions

■ Lors de la définition des zones de la grille, le concepteur a assigné une caractéristique par zone;

postconditions

■ On a vérifié le contenu de chacune des cases de la grille;
■ Toutes les zones de la grille ne contiennent que des jetons qui correspondent à la (aux) caractéristique(s) qui y avai(en)t été définie(s) pour ces zones.

12.17.4. déroulement interne

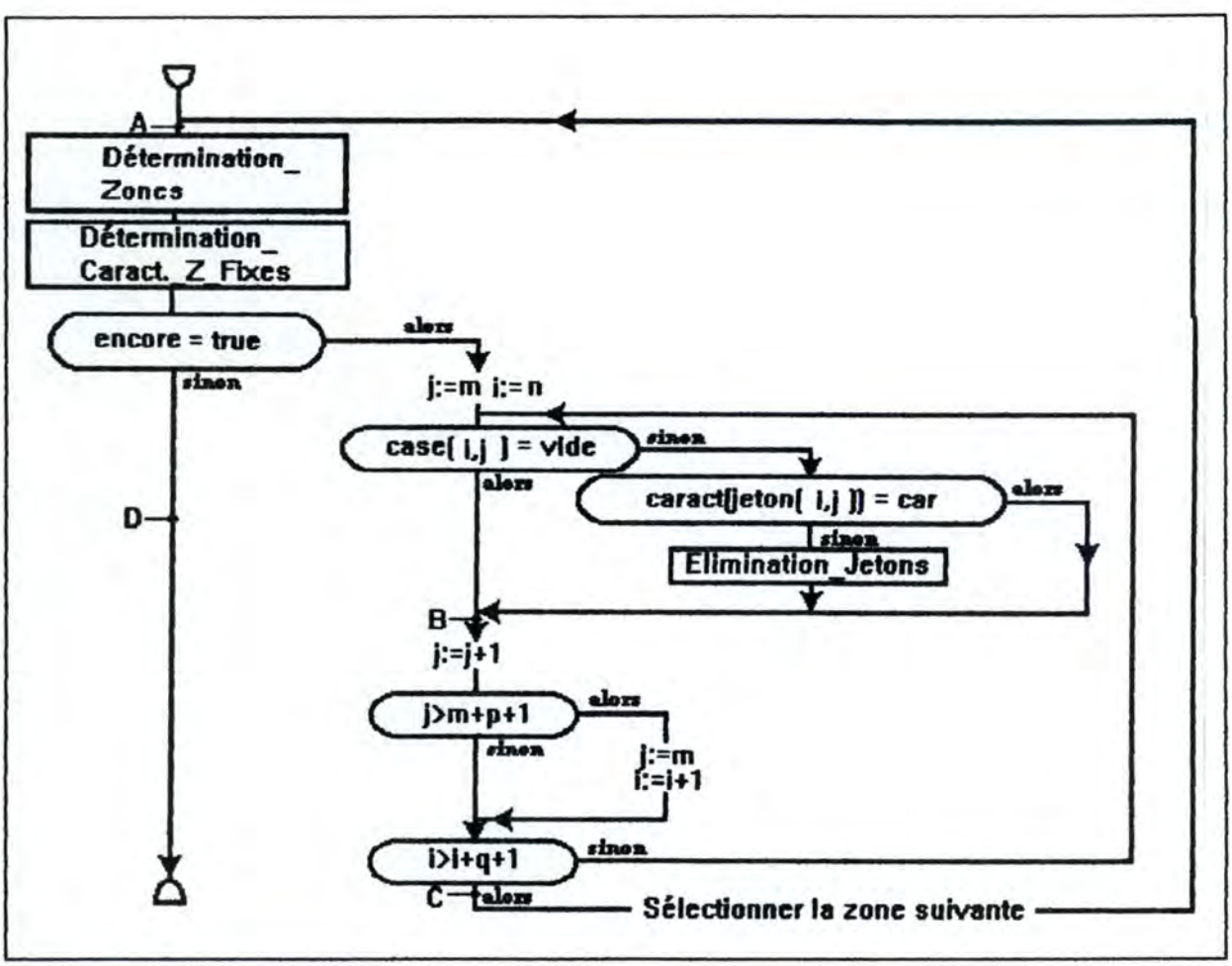


figure 12.14

■ On entre par **A** dans la procédure "Traitement_Zones_Fixes". On exécute une première fois les deux procédures qu'elle appelle. Elles vont fournir au corps de la boucle (tant que l'on n'a pas terminé de vérifier la dernière zone de la grille) les coordonnées d'une nouvelle zone non encore vérifiée et la (les) caractéristique(s) pour un jeton qui y a (ont) été rattachée(s) par le concepteur.

■ On entre dans le corps de la boucle s'il existe une zone non encore vérifiée. On teste dans un premier temps si la case est vide. Si elle ne l'est pas, on compare la (les) caractéristique(s) du jeton avec la (les) caractéristique(s) de référence. Si le test s'avère négatif, on élimine le jeton. On passe en **B**. De même, si la case était vide on passe directement en **B**.

■ A partir de ce point, on gère le passage à la case suivante de la même manière que ce que nous avons déjà vu pour ce problème. Si le dernier test est vérifié, cela signifie que la zone est terminée et on retourne vers les deux procédures, sinon on retourne vers le début de la boucle pour traiter la nouvelle case.

12.18. procédure Traitement_Jetons_Sélectionnés

12.18.1. situation

est appelée par : "Validation_Association_Caractéristiques"

appelle : "Eliminer_Jetons", "Désélectionner"

12.18.2. présentation

Cette troisième procédure appelée par la procédure "Validation_Association_Caractéristiques" gère la situation où l'association par caractéristiques a été définie, lors de la conception du jeu, sur les pièces sélectionnées par le joueur en cours de partie. Pour réaliser cette opération, on travaille uniquement sur les jetons sélectionnés dans la grille depuis la dernière validation. On compare chacun des jetons sélectionnés par rapport à une (plusieurs) caractéristique(s) qu'ils possèdent.

Cette procédure repose sur la procédure "Gestion_Liste_Sélection" qui est exécutée dans la procédure "Gestion_Evénements" et qui réunit, dans une liste, tous les jetons que le joueur sélectionne entre deux validations. La procédure présentée dans cette nouvelle section vérifie que tous les jetons sélectionnés possèdent la (les) même(s) caractéristique(s).

Pour ce faire, on considère la (les) caractéristique(s) du premier jeton sélectionné. Tous les autres jetons sélectionnés doivent également posséder celle(s)-ci. Si ce n'est pas le cas pour au moins un des jetons de la sélection, toute celle-ci est invalide et les jetons retrouvent leur aspect de l'avant sélection.

12.18.3. spécifications

objets utilisés

- **meme :** variable booléenne indiquant s'il elle possède la valeur "true" que tous les jetons possèdent la (les) même(s) caractéristique(s);
- **i :** variable entière, compteur des éléments de la liste;
- **liste :** variable structurée, dont le contenu est déterminé par la procédure "Gestion_Liste_Sélection"; ses éléments sont constitués chacun d'un couple de coordonnées de la position d'un jeton et de la (des) caractéristique(s) qu'il possède;
- **car :** variable qui reprend pour la liste des jetons constituée depuis la dernière validation la (les) caractéristique(s) que le concepteur a déterminé pour le premier jeton sélectionné;
- **t :** variable entière qui détermine la longueur de la liste, elle reçoit une valeur dans la procédure "Gestion_Liste_Sélection".

préconditions

- Une liste de jetons a été réalisée dans la procédure "Gestion_Liste_Sélection"; elle contient tous les jetons qui ont été sélectionnés depuis la dernière validation;
- t possède une valeur égale à la longueur de cette liste.

postconditions

- On a vérifié que les jetons sélectionnés étaient du même type. Si c'est le cas, ils subissent un traitement visant à signaler, au joueur, le fait qu'ils sont correctement associés. Si ce n'est pas le cas, ils ont retrouvé l'état qu'ils avaient avant la sélection;
- Les autres jetons sont inchangés

12.18.4. Déroulement interne

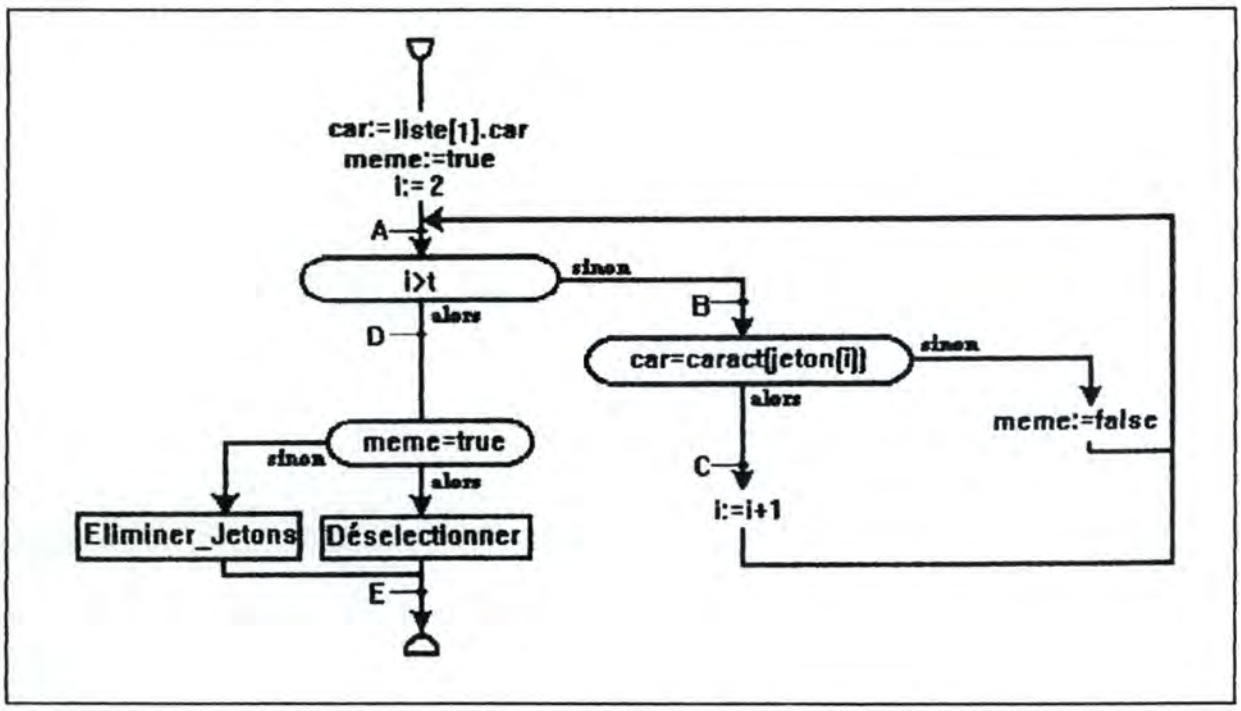


figure 12.15

■ On entre par **A** dans la procédure. On a réalisé les initialisations.

■ On entre dans le corps de la boucle tant que le compteur "i" ne dépasse pas la valeur représentant la longueur de la liste et que le booléen "meme" reste vrai. Si ce n'est pas le cas, on passe en **D**. Dans la boucle, après **B**, on teste la (les) caractéristique(s) du jeton de la ligne courante de la liste avec la valeur de la variable "car". Si le test s'avère positif, on augmente simplement "i", en passant par **C**. Si le test s'avère négatif, "meme" prend la valeur "false". De même, si la case était vide on passe directement en **B**.

■ Après avoir quitté la boucle et être passé en **D**, on teste la valeur de "meme". Si c'est "true", cela signifie que tous les jetons sont du même type. La procédure "Eliminer_ Jetons" se charge de l'événement qu'un tel résultat génère. Si par contre, la valeur est "false", il faut désélectionner les jetons. Nous avons choisi de réaliser cela par une procédure intitulée "Désélectionner". Dans les deux cas, une fois la procédure terminée, on atteint le point **E**, signalant la sortie.

12.19. procédure Gestion_Liste_Sélection

12.19.1. situation

est appelée par : "Analyse_Evénements"

appelle : /

12.19.2. présentation

Cette procédure place le couple des coordonnées et la (les) caractéristique(s) de chacun des jetons que le joueur sélectionne entre deux validations si la variable "etat" est égale à "select". Dans le cas contraire, si la variable "etat" est égale à "deselect", elle retire de la liste le jeton correspondant. La variable "etat" a été présentée page 109.

12.20. procédure Désélectionner

12.20.1. situation

est appelée par : "Traitement_Jetons_Sélectionnés"

appelle : /

12.20.2. présentation

Cette procédure remet les jetons dans l'état qu'ils avaient avant la sélection.

12.21. procédure Détermination_Caractéristiques_Z_Fixes

12.21.1. situation

est appelée par : "Traitement_Zones_Fixes"

appelle : /

12.21.2. présentation

Cette procédure rappelle pour chaque zone la (les) caractéristique(s) qui avai(en)t été définie(s) lors de la définition des zones.

12.22. procédure Association_Côtés

12.22.1. situation

est appelée par : "Validation"

appelle : "Détermination_Zones", "Correction_Association_Côtés"

12.22.2. présentation

Nous abordons la validation du type d'association par côtés. Cette validation consiste simplement à vérifier que tout jeton est correctement associé aux quatre jetons qui l'entourent. Si on prend deux jetons quelconques qui sont voisins, il suffit de vérifier que leurs côtés communs peuvent ou non être associés. S'ils sont simultanément associables ou non associables, la conclusion sera positive. Par contre, si le côté de l'un des deux jetons est associable alors que l'autre ne l'est pas, la conclusion sera négative et l'un des deux jetons devra être ôté de la grille. Nous avons décidé d'éliminer le jeton courant, c'est-à-dire le jeton sur lequel porte la vérification.

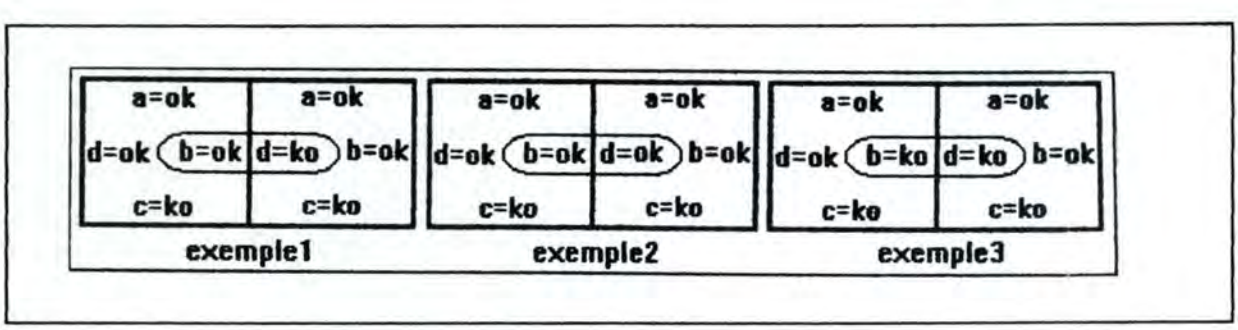


figure 12.16

Comme le présentent les trois exemples ci-dessus, seule la première association est non valide, les deux autres ne posent aucun problème.

La figure qui suit présente l'algorithme de cette procédure. Elle est quasiment identique à la procédure de la validation par caractéristiques en zones libres. Nous ne la verrons pas en détail car elle ne diffère de cette dernière que par la procédure qu'elle utilise dans sa boucle.

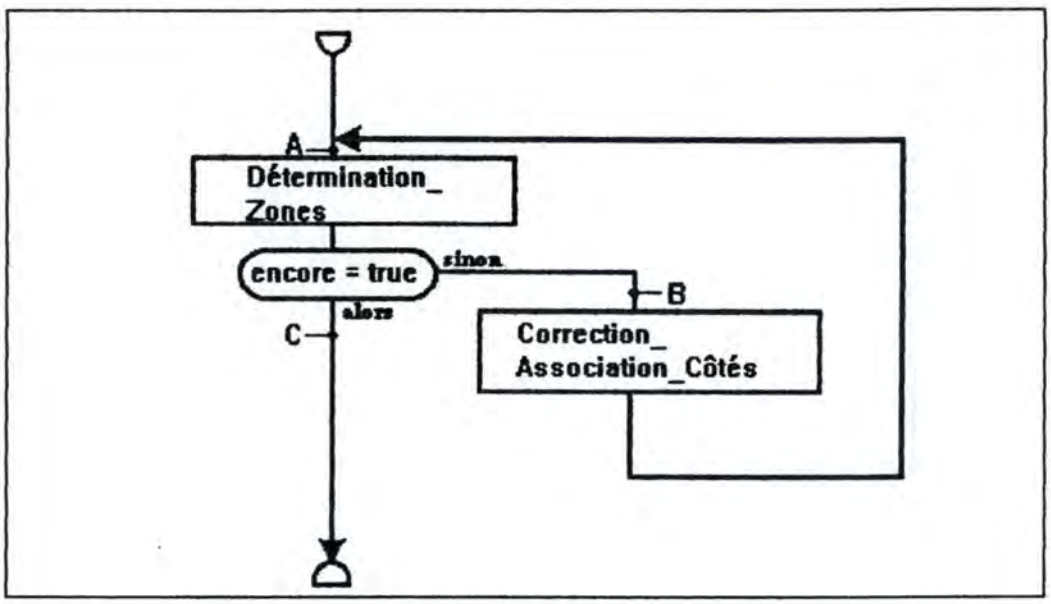


figure 12.17

Signalons simplement qu'elle vérifie les associations zone par zone dans la grille. On entre en **A**, on détermine la zone courante suivante. Tant qu'il reste des zones à traiter on entre dans la boucle où l'on exécute la procédure qui va être présentée ci-dessous. Quand toutes les zones ont été vérifiées, on quitte cette procédure.

12.23. procédure Correction_Association_Côtés

12.23.1. situation

est appelée par : "Association_Côtés"

appelle : "Eliminer_Jetons"

12.23.2. les situations de validation

■ On déclare un côté associable en définissant ce côté "ok". A l'inverse, un côté non associable prendra la valeur "ko". Pour que le jeton soit correctement placé, il faut que le test d'association appliqué à ses quatre côtés s'avèrent positif dans les quatre cas.

■ Un jeton possède un côté correctement associé si ce côté est conjugué avec le côté d'un autre jeton possédant pour ce côté une même valeur d'association. De même, si la case voisine est vide, l'association sera correcte.

■ Par contre, si les deux côtés associés possèdent des valeurs d'association différentes, l'association sera invalide. De même, si le côté est déclaré "ok" alors que le jeton est placé sur le bord d'une zone de la grille, l'association sera déclarée invalide.

12.23.3. présentation

On va vérifier pour chaque jeton de la zone courante de la grille que les recommandations du point 13.23.2. sont appliquées. Pour ce faire, on teste les quatre côtés du jeton courant. Chacun de ces tests attribue une valeur à une variable booléenne.

On obtient donc quatre variables booléennes qui doivent être toutes les quatre simultanément vraies si l'on veut conserver le jeton courant. Si ce n'est pas le cas, celui-ci est éliminé.

12.23.4. spécifications

objets utilisés

■ **j :** variable entière jouant le rôle de compteur des cases d'une ligne de la zone courante de la grille;
■ **i :** variable entière jouant le rôle de compteur des lignes de la zone courante de la grille;
■ **m :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la grille jusqu'à la première colonne de la zone courante inclue;
■ **n :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la grille jusqu'à la première ligne de la zone courante inclue;
■ **p :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de colonnes de la zone courante;
■ **q :** variable entière qui prend sa valeur dans la procédure de détermination des zones et qui est égale au nombre de lignes de la zone courante;
■ **case :** variable structurée qui possède un couple de coordonnées et un état occupé ou inoccupé;
■ **jeton :** variable structurée qui possède un couple de coordonnées et une définition pour l'association de ses quatre côtés;

jeton.a, jeton.b, jeton.c, jeton.d : ces quatre champs matérialisent les quatre côtés de tout jeton; ils désignent respectivement les côtés gauche, haut , droit et bas de tout jeton;

■ **cot1, cot2, cot3, cot4 :** variables booléennes qui permettent de déterminer en fin de module si les quatre côtés sont valides;

préconditions

■ On se trouve dans la situation de "l'association par côtés", au début de la nouvelle zone courante et m,n,p et q ont acquis une valeur dans le module "Détermination_Zones".

postconditions

■ On quitte la procédure en ayant vérifié que les jetons joués étaient correctement associés par leurs côtés; tout jeton ne vérifiant pas les conditions de cette association aura été éliminés; les autres seront inchangés.

12.23.5. déroulement interne

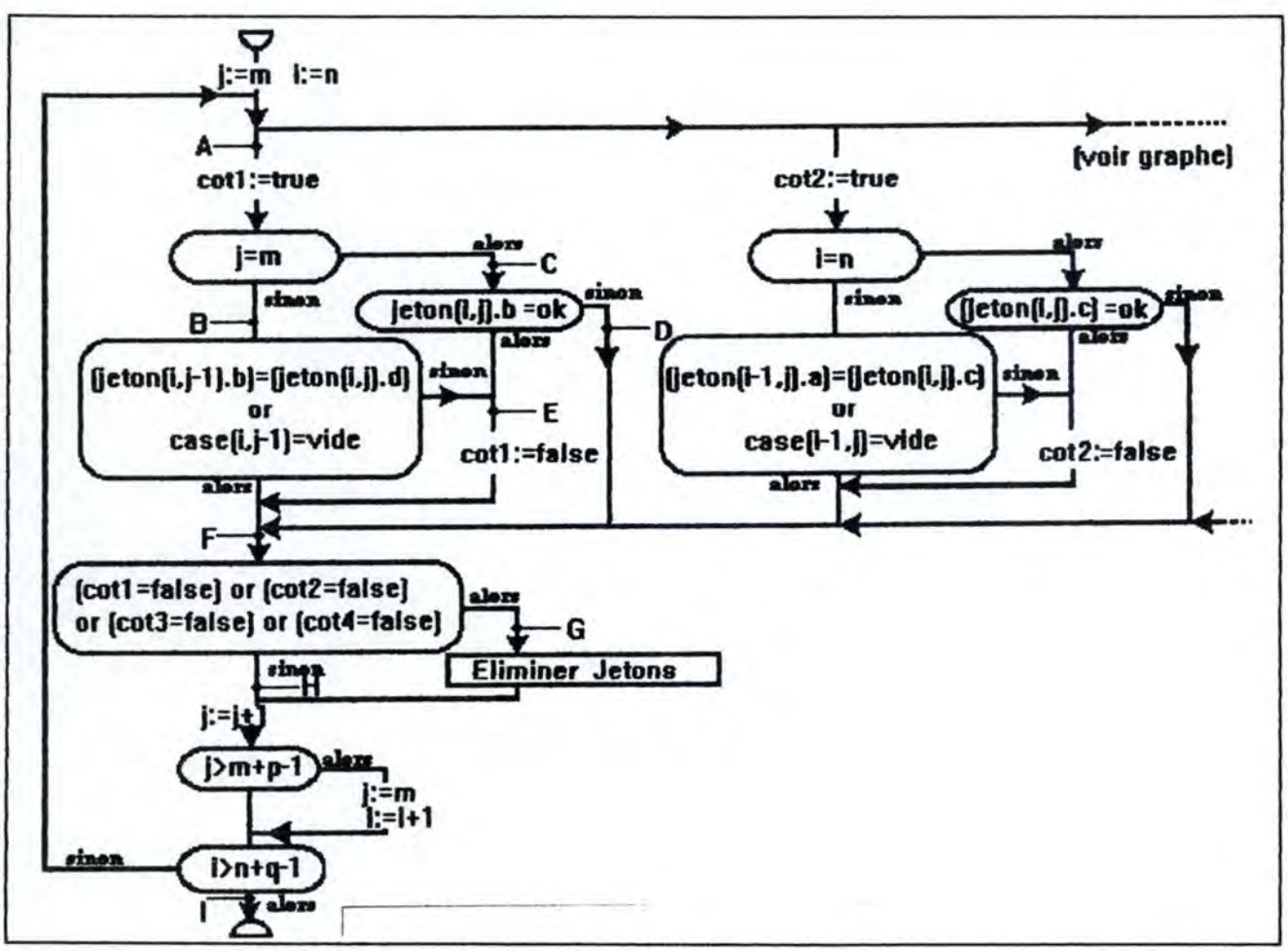


figure 12.18

■ On entre par **A** dans la procédure. On entame alors la vérification d'une nouvelle zone. Remarquons qu'entre les points **A** et **F**, les quatre tests se déroulent identiquement pour fournir une valeur aux quatre booléens. Nous n'étudierons donc que le premier des quatre tests.

■ Après avoir initialisé la variable booléenne "cot1", on teste si la case ne se trouve pas sur un des bords de la zone courante. Ici, il s'agit du bord vertical gauche. Si le jeton se trouve sur ce bord, on passe par **C** et il faut obligatoirement que le côté ne soit pas associable et donc défini "ko". Si ce n'est pas le cas, en passant par **E**, "cot1" prend la valeur "false", sinon, elle reste inchangée en passant par **D**.

■ Si la case ne se trouve pas sur un bord, on passe par **B**. Le test qui suit détermine si les côtés sont ou non associables simultanément (ok ou ko), ou si la case voisine est vide. Si ce test se révèle positif, "cot1" reste inchangée et on passe en **F**. Par contre, en cas de réponse négative, on passe par **E** et "cot1" prend la valeur "false".

■ Le test qui suit le passage en **F**, vérifie que les variables "cot1", "cot2", "cot3" et "cot4" sont simultanément vraies. Si ce n'est pas le cas, on doit conclure à l'élimination du jeton.

■ La boucle se termine avec le passage à la case suivante de la zone courante. Si le dernier test est vérifié, on quitte la boucle pour retourner vers le module de détermination des zones.

■ Voici l'autre partie de cet algorithme :

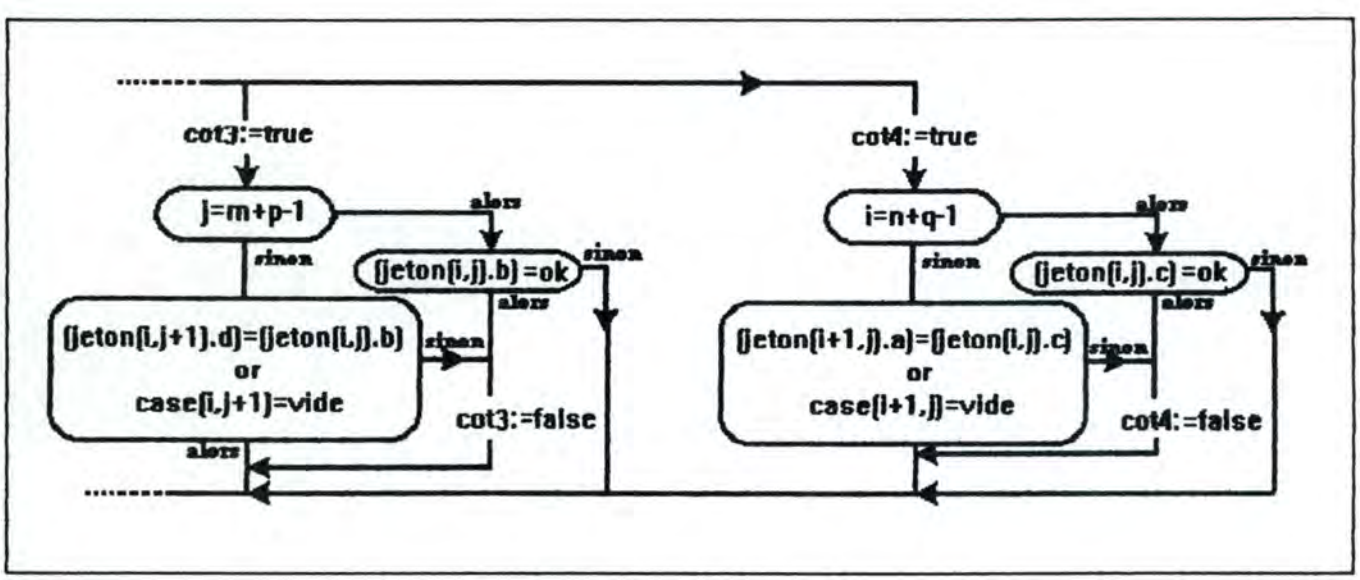


figure 13.19

12.24. procédure Association_Positions

12.24.1. situation

est appelée par : "Validation"

appelle : "Eliminer_Jetons"

12.24.2. présentation

Nous abordons la validation du dernier mode d'association. Cette validation consistera simplement à vérifier que tout jeton est correctement positionné dans la grille. Pour un jeton, il suffit de vérifier que l'un des couples de coordonnées d'arrivée qui lui ont été assignés lors de sa définition est identique au couple des coordonnées de la case où il se trouve. Si ce test s'avère négatif, le jeton doit être éliminé de la grille.

12.24.3. Spécifications

objets utilisés

- **i :** variable entière jouant le rôle de compteur des lignes de la grille;
- **j :** variable entière jouant le rôle de compteur des colonnes de la grille;
- **case :** variable structurée pour laquelle nous avons retenu comme attributs un couple de coordonnées et un état occupé ou inoccupé;
- **jeton :** variable structurée pour laquelle nous avons retenu un couple de coordonnées et la (les) position(s) finale(s) (exprimée(s) en couple(s) de coordonnées) que le jeton devra occuper;

  jeton.pos : position finale du jeton, définie par le joueur au moment de la définition du jeu (notons que dans l'algorithme qui suit, elle symbolise tous les couples d'arrivée définis pour un jeton);

préconditions

- On se trouve dans la situation de l'association par positions. Les valeurs de "h" et de "l" ont été déterminées à la conception du jeu, lors de la définition de la grille. "h" correspond à la hauteur de celle-ci, tandis que "l" correspond à sa largeur.

postconditions

- On quitte la procédure en ayant vérifié que les jetons joués étaient correctement positionnés. Tout jeton ne vérifiant pas les conditions de l'association aura été éliminé. Les autres seront inchangés.

12.24.4. déroulement interne

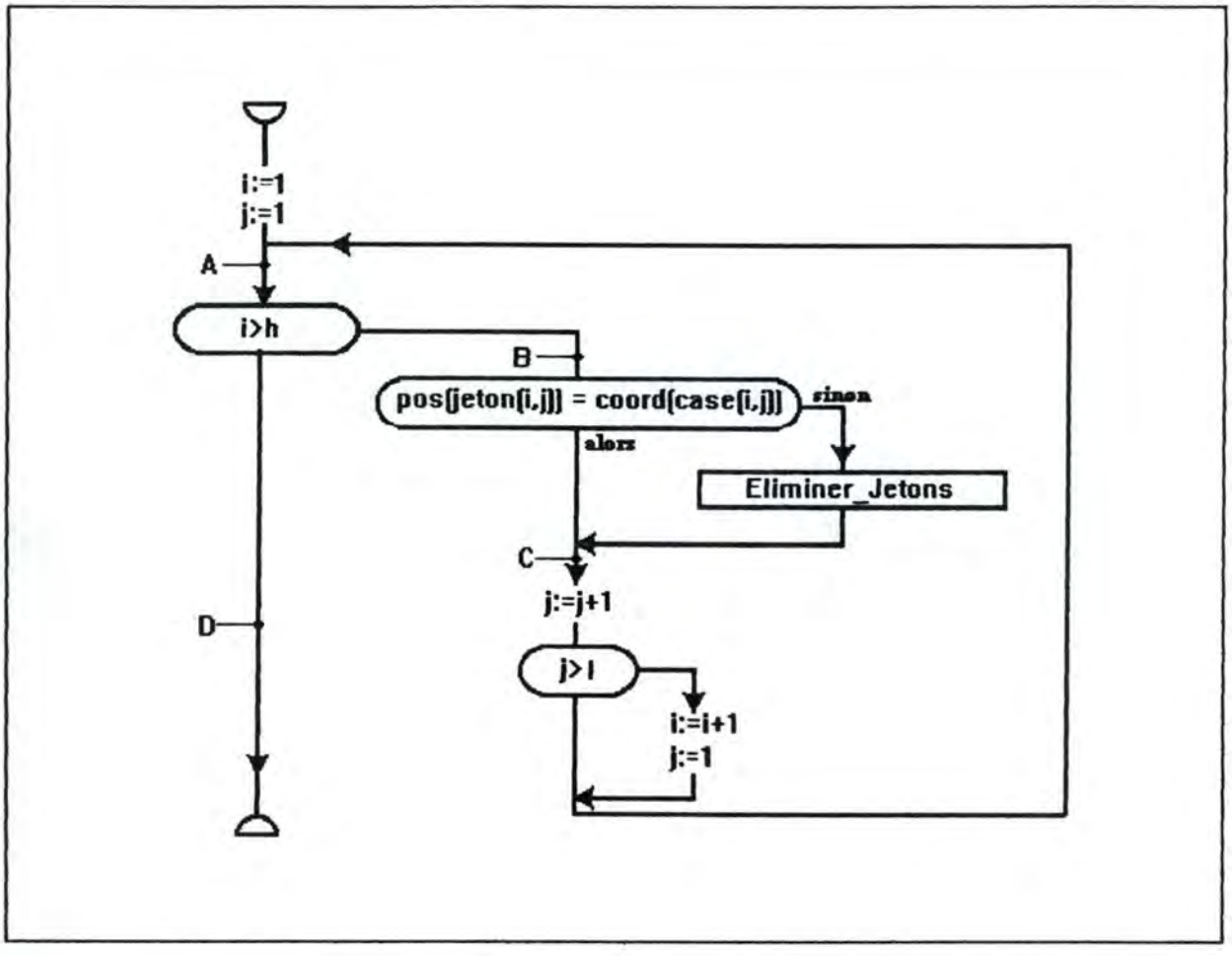


figure 12.20.

■ On entre par **A** dans le module. La grille ne pouvant pas être de taille nulle, le test ne sera pas vérifié et on passe au moins une fois par la boucle.

■ On entre dans la boucle. On compare la position d'arrivée du jeton (telle que définie lors de la conception du jeu) aux coordonnées de la case dans laquelle il se trouve. Si la comparaison conclut que le jeton n'est pas bien placé, il faut l'éliminer.

■ Ceci étant fait, on passe en **C**, où l'on réalise le passage à la case suivante dans la grille. Si "j" est supérieur à "l", on a donc atteint avec la case précédente la fin d'une ligne. On augmente "i" d'une unité pour passer à la ligne suivante et on remet "j" à un pour se positionner sur la première case de la nouvelle ligne.

■ Cette procédure se termine lorsque la grille à été complètement vérifiée, ce qui est connu lorsque "i" dépasse "h". On passe alors par **D** pour quitter le module.

12.25. procédure Gestion_Sabot

12.25.1. situation

est appelée par : "Analyse_Evénements"

appelle : /

12.25.2. présentation

Cette procédure réorganise le sabot en fonction des actions menées par le joueur.

12.26. procédure Gestion_Zone_Attente

12.26.1. situation

est appelée par : "Analyse_Evénements"

appelle : /

12.26.2. présentation

Cette procédure réorganise la zone d'attente en fonction des actions menées par le joueur.

partie 4 : Interface

13. Présentation
14.Le logiciel
15. Conception d'une interface pour l'animateur

13. Présentation

13.1. Contenu

Nous proposons, dans cette dernière partie, une présentation de l'interface conçue pour la partie implémentée du logiciel, la partie définition. La chapitre concernant la présentation de celle-ci a été conçu de façon à ce qu'il couvre les besoins d'un manuel de l'utilisateur.

Ce chapitre portera donc sur une présentation des objets utilisés, leur fonctionnement, leur organisation et leurs relations.

13.2. L'interface

13.2.1. L'interface Windows

Nous avons voulu, grâce à notre interface, retrouver la simplicité du formulaire. Elle a été conçue sous Windows et repose sur certaines recommandations reconnues pour la définition d'objets interactifs que ce type d'interface offre.

Notre logiciel comporte un menu principal et emploie un très grand nombre de boîtes de dialogue. Ces objets sont forts simples à comprendre, à employer et à assimiler. La gestion de l'interface a été conçue pour être réalisée au moyen de la souris.

13.2.2. Fonctionnalités des objets de dialogue

Les **menus** assurent un rappel constant des fonctions offertes par le logiciel. Les **boîtes de dialogue** reprennent systématiquement, par type abstrait, les éléments qu'il faut définir pour créer des jeux. Dans celles-ci, nous employons, en majorité, des **champs d'édition**, des **échelles** et des **cases à cocher** grâce auxquels le concepteur pourra introduire ses choix.

13.2.3. Terminologie

Bien que nous ayons opté pour une terminologie correspondant à ce que nous avons défini, une remarque importante s'impose. Au cours de l'étape de conception, l'accent a été mis sur les types de jetons et les occurrences de ces types que sont les jetons. Les notions de "type" et d'"occurrence" sont des notions dont le vocabulaire a été emprunté au domaine informatique. Nous avons jugé qu'ils seraient inadéquats pour l'interface proposée aux utilisateurs.

Plutôt que de parler de types de jetons et de jetons, nous proposons les termes "jetons" et "pions" aux animateurs. Lorsque tout animateur désire créer un type de jeton, il le réalise en sélectionnant l'item "jetons", lorsqu'il désire créer des occurrences des types de jetons, il le fait en créant des pions via l'activation de l'item "pions". Ces deux items du menu principal vont être plus largement expliqués dans le chapitre qui suit.

14.
Le logiciel

Ce chapitre présente ce que le logiciel offre actuellement aux concepteurs de jeux.

14.1. Jeux

Nous offrons la possibilité de réaliser la définition de jeux. Cette définition comprend:

- la définition d'un ensemble de jetons;
- la définition des cinq zones de l'écran;
- la définition des règles de jeu.

14.2. Jetons

Nous offrons la possibilité de réaliser la définition de jetons. Cette définition comprend:

- le choix d'un contenu;
- la définition des caractéristiques (couples de préfixes et de suffixes);
- la définition des côtés associables;
- la définition d'un nombre d'occurrences.

14.3 Pions

Nous offrons la possibilité de réaliser la définition de pions. Cette définition comprend pour un pion:

- le choix du jeton de référence;
- la présentation du contenu, des caractéristiques et des côtés associables du jeton duquel ce pion est issu (couples de préfixes et de suffixes);
- la définition de la position de départ et de la (des) position(s) d'arrivée;
- la définition de la représentation du pion.

14.4. Grille

Nous offrons la possibilité de réaliser la définition d'une grille. Cette définition comprend:

- la définition de la taille et de l'emplacement;
- la définition d'une découpe automatique en spécifiant le nombre de zones;
- pour la découpe en zones fixées, il est possible de définir la (les) caractéristiques(s) de chaque zone.

14.5. Sabot

Nous offrons la possibilité de réaliser la définition d'un sabot. Cette définition comprend:

- la définition de la taille et de l'emplacement;
- la définition de la présentation des pions : nombre de pions présents au début du jeu, taille des wagons, présentation à plat ou en piles (avec le nombre de piles), la définition d'un compteur des pions et son emplacement.

14.6. Zone d'attente et de résultat

Nous offrons la possibilité de réaliser la définition de ces deux zones. Cette définition comprend:

- la définition de la taille et de l'emplacement.

14.7. Zone de message

Nous offrons la possibilité de réaliser la définition d'un sabot. Cette définition comprend:

- la définition de la taille et de l'emplacement;
- la définition du contenu.

14.8. Règle de validation

Nous offrons la possibilité de réaliser la définition des règles de validation. Cette définition comprend:

- la définition du moment de validation tous les x coups joués;
- la définition d'un taux d'erreur;
- la définition des actions liées à la fin de la validation : une action si incorrecte ou une action si correcte;

14.9. Règles d'association

Nous offrons la possibilité de réaliser la définition des règles d'association. Cette définition comprend:

- le (les) mode(s) d'association;

14.10. Règles de déplacement

Nous offrons la possibilité de réaliser la définition des règles de déplacement. Cette définition comprend:

- *la (les) règle(s) de déplacement;*

14.11. Règles de fin de jeu

Nous offrons la possibilité de réaliser la définition des règles de fin de jeu. Cette définition comprend:

- la définition du temps limite de jeu en nombre de coups ou en temps;
- la définition de l'action liée au dépassement de la limite de temps de jeu;
- la définition de l'état final de chacune des zones de l'écran;

14.12. Règle de manipulation

Nous offrons la possibilité de réaliser la définition de la règle de manipulation. Cette définition comprend:

- la définition du mode de manipulation;

15. Conception d'une interface pour l'animateur

15.1. La barre de menu

Pour le menu principal, on trouve une barre de menu déroulant comptant dans l'ordre les items "Jeux", "Edition", "Jetons", "Fenêtre" et "A propos de" et ayant la présentation suivante.

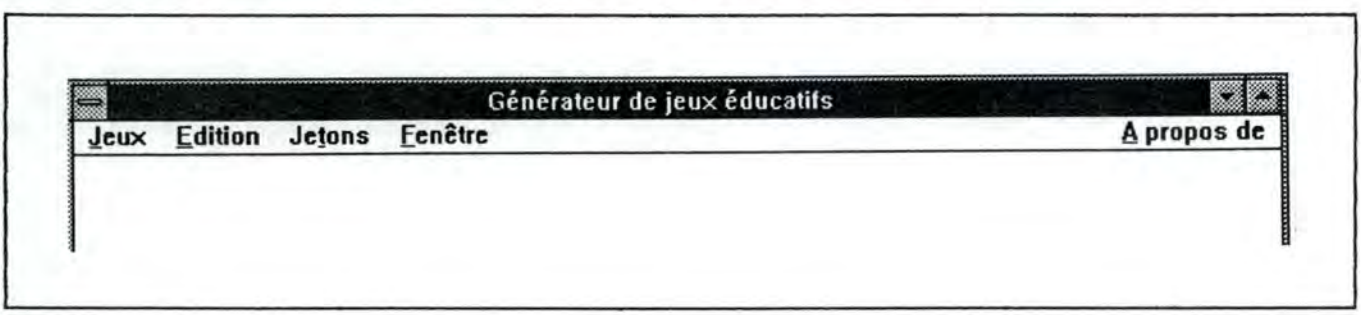


figure 15.1.

15.2. Les items de la barre de menu

L'item **"Jeux"** fournit un menu reprenant l'ensemble des fonctions de gestion des jeux offertes à l'animateur. Comme nous le verrons plus en détail, il s'agit entre autres de la création, de la mise à jour, de la sauvegarde, de fermeture, de la suppression et du lancement d'un jeu. Le menu attaché à cet item reprend également l'option "Quitter" qui permet de quitter le programme. Cet item est accessible à tout moment.

L'item **"Edition"** propose trois fonctions au concepteur : "Couper", "Copier", "Coller". Elles vont lui permettre de créer des jeux à partir des définitions de plusieurs autres jeux. Cet item n'est accessible qu'à partir du moment où l'animateur a sélectionné l'item "Nouveau ..." ou l'item "Ouvrir ..." dans le menu "Jeux". En dehors de ces situations, cet item est affiché en grisé

dans la barre du menu principal, pour signaler que les fonctions ne sont pas accessibles. Ces fonctions non pas été implémentées.

L'item **"Jetons"**, lorsqu'il est activé, génère un menu présentant les options offertes sur les définitions des jetons. En résumé, la création, la modification et la suppression. Cet item est accessible à tout moment.

L'item **"Fenêtre"** correspond à la possibilité de travailler avec plusieurs fenêtres reprenant chacune une définition d'un jeu (déjà défini ou en cours de création) afin de créer un nouveau jeu à partir des définitions de ceux-ci. Cet item offre l'accès aux fonctions de contrôle de ces fenêtres. Cet item apparaît dans la barre du menu principal en grisé ou en affichage normal pour les mêmes événements que l'item "Edition".

15.3. Item "Jeux"

15.3.1. Nouveau ...

Si cette option est choisie, l'animateur peut créer de toute pièce un jeu. Un écran reprenant l'ensemble des éléments à définir apparaît. Il donne accès à la définition des pions, du plan de jeu et des règles du jeu.

15.3.2. Ouvrir ...

Cette fonction permet à l'animateur de créer un nouveau jeu à partir de la définition d'un ou plusieurs jeux déjà existants ou, de modifier la définition d'un jeu. Cette fonction va donc utiliser les mêmes moyens de définition que "Nouveau ...". Il existe cependant deux différences avec celle-ci. Premièrement, l'activation de la fonction "Ouvrir ..." génère une liste déroulante des noms des jeux existants. Cette étape supplémentaire est nécessaire afin que l'animateur puisse choisir le(s) nom(s) du(des) jeu(x) à partir duquel (desquels) il désire en créer un autre ou plus simplement de choisir le jeu qu'il désire modifier.

Deuxièmement, si l'éducateur désire créer un nouveau jeu à partir d'un seul autre ou s'il désire modifier la définition d'un jeu, l'activation de l'item "Ouvrir ..." modifie l'affichage grisé des items "Edition" et "Ecran". En effet, l'animateur n'a pas besoin de gestion de ce type.

15.3.3. Fermer ...

Cette fonction permet à l'animateur de fermer un jeu sur lequel il est en train de travailler. Lors de l'accomplissement de cette tâche, il est nécessaire de demander à l'animateur si la

fermeture du jeu doit s'accompagner ou non d'une sauvegarde des éléments éventuellement modifiés ou créés.

15.3.4. Sauvegarder ...

Cette option permet à l'animateur de sauver un jeu qui vient d'être modifié. Notons que si par mégarde l'animateur décide de sauver un jeu qu'il vient de créer en activant cette fonction plutôt que la suivante, il obtiendra la présentation de la boîte de sauvegarde, attachée à l'option suivante. En effet, il lui faut absolument nommer tout jeu nouvellement créé. Sans cela, la définition sera perdue.

15.3.5. Sauvegarder sous ...

Cette option permet à l'animateur de nommer et de sauver un jeu qu'il vient de créer ou de modifier. L'activation de l'item y correspondant déclenche la présentation d'une boîte de dialogue dans laquelle il pourra nommer de façon univoque tout jeu nouvellement créé.

Remarquons que s'il sauve un jeu modifié au moyen de cette option, il va en fait créer une nouvelle définition sans remplacer la version qui existait avant la modification. De cette façon il peut disposer de deux définitions. Ceci peut être intéressant s'il désire créer plusieurs versions différentes d'un jeu pour différents enfants ou pour en ressortir la plus intéressante.

15.3.6. Sauvegarder tout ...

Cette option permet de réaliser une sauvegarde portant sur toutes les définitions de jeux ouvertes au moment de la sélection de l'option.

15.3.7. Supprimer ...

Cette option fournit à l'animateur la possibilité de supprimer tout jeu ne l'intéressant plus. En sélectionnant l'item référant cette fonction, une liste, comparable à celle proposée par l'activation de l'item "Ouvrir ...", est affichée. Au sein de cette liste, il peut choisir le(s) jeu(x) qu'il désire voir disparaître.

15.3.8. Jouer ...

Cette fonction permet à l'animateur de consulter la liste de tous les jeux définis et de choisir celui qu'il désire "lancer".

15.3.9. Valider ...

Cette fonction permet à l'animateur de réaliser la vérification syntaxique présentée au chapitre 10.

15.3.10. Résumer ...

L'animateur en activant cet item dispose de la liste de tous les jeux. La définition du jeu choisi est présentée dans le même écran que celui utilisé lors de la création et de la modification. La différence réside dans le fait que l'animateur ne sait modifier aucun élément de la définition du jeu.

15.3.11. Quitter

Cette option lorsqu'elle est activée permet la sortie du programme.

15.4. Item "Edition"

Les fonctions liées à cet item ne sont en fait pas cruciales pour le bon déroulement du programme. On peut même dire que l'on aurait pu s'en passer. Cependant cette option permet de gérer la création d'un jeu à partir de la définition d'autres jeux. Sans les fonctions d'édition, l'animateur serait systématiquement obligé de redéfinir de A à Z tout nouveau jeu. Il ne lui serait donc possible de créer un jeu ayant les mêmes jetons et règles qu'un autre jeu qu'il a déjà défini que via la redéfinition complète des pions et des règles. Ce travail serait long et inutile. Grâce aux fonctions d'édition l'animateur va simplement pouvoir sélectionner les éléments de définition d'un jeu qui l'intéresse, en faire une copie et la coller dans la définition du jeu en cours de création..

La sélection de cet item entraîne donc l'affichage d'un menu reprenant les items "Couper", "Copier" et "Coller". L'opération de création d'un jeu, selon cette technique, est réalisée en ouvrant simultanément des fenêtres correspondant chacune à un des jeux à partir desquels l'animateur désire construire ce nouveau jeu. La gestion de ces fenêtre est régie par les fonctions de l'item "Ecran".

15.5. Item "Jetons"

Pour rappel, cet item correspond à ce que nous avons appelé précédemment les types de jetons. Lorsqu'un animateur désire créer ce que nous avons appelé un type de jeton, il doit

sélectionner l'item de menu "Jetons" qui lui permet de créer ce que nous appellerons pour lui une définition générale d'un jeton. Plus tard, lors de la définition d'un jeu, celui-ci pourra créer des pions à partir de cette définition générale.

15.5.1. Nouveau ...

La sélection de cet item permet au concepteur du jeu de créer des nouveaux jetons et ce indépendamment de tout jeton déjà défini. Cette création, c'est-à-dire la détermination des caractéristiques associées à ce jeton, se fait conformément à ce qui a été défini dans le chapitre consacré à la définition du concept de type de jeton.

Le concepteur peut donc choisir un dessin, lui associer une série de caractéristiques (préfixes et suffixes), déterminer une position de départ, une (plusieurs) position(s) d'arrivée(s) et déterminer les côtés associables. Pour la détermination des caractéristiques, le concepteur dispose d'une aide lui assurant de choisir correctement celles-ci. Ceci est indispensable car elles doivent être compatibles avec les définitions d'autres jetons de façon à ce que les pions qui seront choisis à partir de ces définitions puissent être associables en fonction de leurs caractéristiques.

Pour illustrer cette facilité, supposons que deux fourchettes aient reçu comme définition, pour trois de leurs suffixes les termes suivants:

- FOURCHETTE, METAL, 4
- Fourch., METALLIQUE, quatre

Cette illustration montre qu'il sera impossible d'associer des occurrences de ces deux jetons car les trois suffixes qui les caractérisent sont complètement incompatibles : FOURCHETTE <> Fourch., METAL <> METALLIQUE et 4 <> quatre.

15.5.2. Ouvrir ...

Grâce à cette fonction, l'animateur va pouvoir modifier un jeton ou définir de nouveaux jetons à partir de la définition d'un autre jeton déjà existant. L'activation de cet item entraîne la présentation de la liste reprenant les intitulés des jetons définis. L'animateur doit alors choisir celui sur lequel il veut réaliser une modification ou à partir duquel il veut réaliser une nouvelle définition de jeton. La procédure de définition est alors identique à celle de "Nouveau ..." à cela près que les options sont déjà déterminées.

Quand il quitte cette option, s'il renomme le jeton, il dispose d'une nouvelle définition et l'ancienne est inchangée. Il dispose donc deux jetons. S'il ne renomme pas le jeton, il s'agit alors d'une simple modification et il le même jeton possède une nouvelle définition.

15.5.3. Supprimer ...

Cette option offerte au concepteur, comme son intitulé l'identique, lui permet de supprimer toutes les définitions qu'il désire.

15.6. Item "Fenêtre"

Cette option offerte à l'utilisateur est le complément de l'édition qui permet la définition ou de la modification de jeux à partir des définitions d'autres jeux. Les définitions des autres jeux étant présentées chacune dans une fenêtre, on comprend donc aisément le besoin de cet item assurant les fonctions de gestion de ces fenêtres. Le menu attaché à cette item est divisé en deux parties. La première partie reprend les quatre fonctions offertes tandis que la deuxième reprend la liste des identifiants des fenêtres ouvertes.

15.6.1. Damier

Place les fenêtres les unes à côté des autres sur l'écran. Elle offre ainsi à l'animateur la possibilité d'avoir un accès direct à n'importe laquelle des définition en cours.

15.6.2. Cascade

A l'inverse de l'option précédente où toutes les fenêtres sont présentées de façon réduite à l'utilisateur, les fenêtres sont ici superposées, présentant au sommet de la pile celle qui est actuellement courante.

15.6.3. Ranger icône

Lorsque les fenêtres sont minimisées, une icône vient s'inscrire dans le fond de l'écran Elle symbolise l'existence de cette fenêtre. Cette option permet de réorganiser automatiquement l'arrangement des icônes.

15.6.4. Fermer toutes

Cette option permet comme son intitulé l'identique de fermer toutes les fenêtres.

15.7. L'écran principal pour la création et le résumé des jeux

15.7.1. Présentation

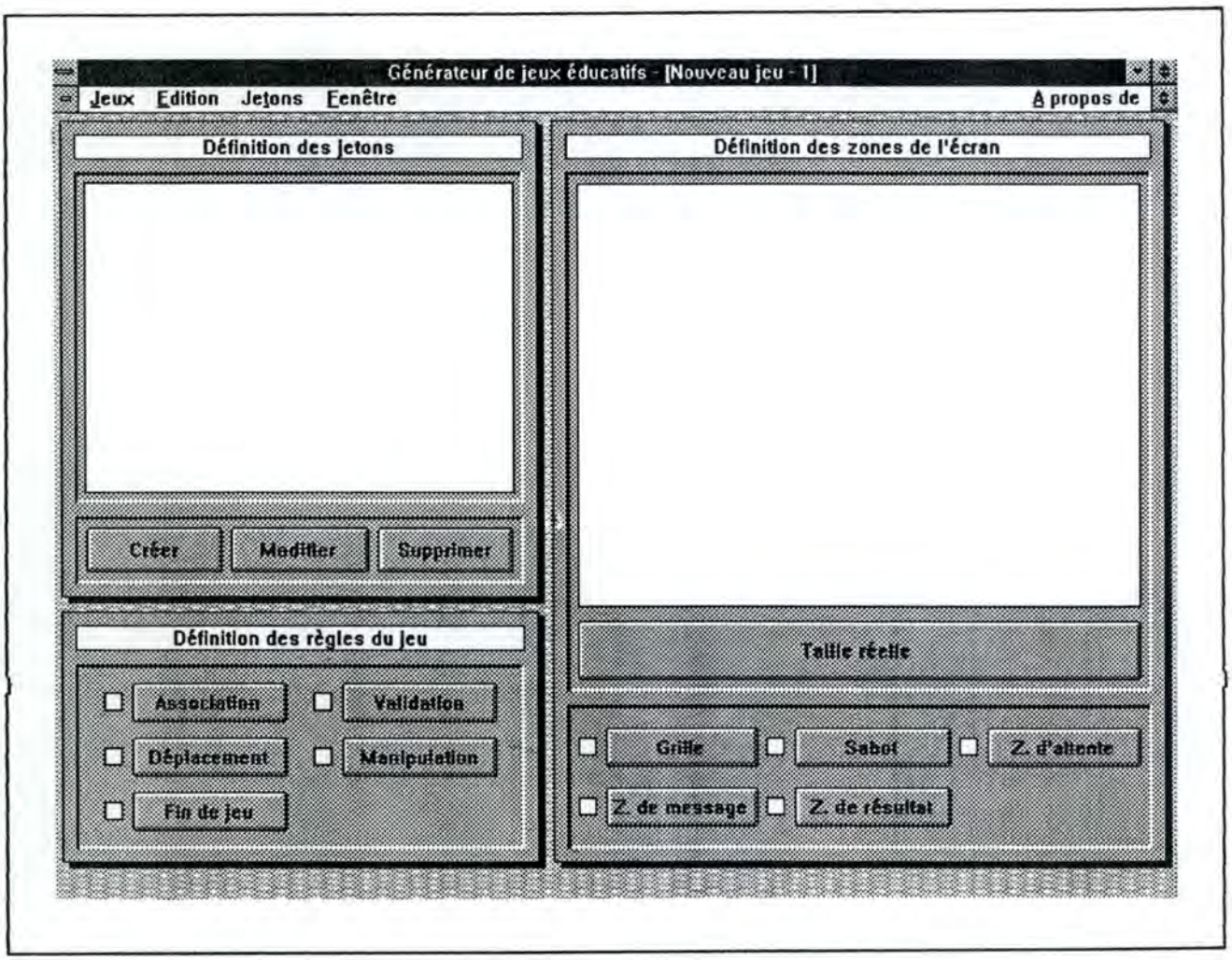


figure 15.2.

Cet écran offre l'accès aux fonctions de gestion des pions, des zones de l'écran de jeu et des règles.

15.7.2. La définition des PIONS

Une première zone de cet écran reprend les fonctions de définition des pions. Celle-ci se compose de deux parties. Dans la partie supérieure, on trouve la liste des pions déjà définis. Dans la seconde partie, on dispose de trois boutons de contrôle permettant la création, la modification et la suppression de pions.

Bouton Créer : Permet à l'animateur de créer des pions à partir des définitions des jetons. S'il enclenche le bouton "Créer", l'animateur voit s'afficher une boîte de dialogue au sein de laquelle il peut créer un nouveau pion. La création terminée, il retourne vers l'écran principal.

Bouton Modifier : Il faut d'abord sélectionner un pion dans la liste. Ensuite, il suffit d'activer le bouton "Modifier". La même boîte de dialogue qu'au point précédent est affichée. Elle est évidemment déjà garnie par les attributs du pions.

Bouton Supprimer : Il faut d'abord sélectionner un pion dans la liste. Ensuite, il lui suffit d'activer le bouton supprimer. Une boîte de dialogue demandant la confirmation est alors présentée au concepteur.

15.7.3. La définition des REGLES du jeu

La partie de l'écran dévolue à la définition des règles du jeu regroupe cinq boutons de contrôle accompagnés d'une case à cocher. Chacun de ces cinq boutons de contrôle permet d'activer une boîte de dialogue assurant la définition d'une des règles du jeu.

Lorsqu'il quitte la définition d'une règle, en fermant la boîte de dialogue, il retourne à l'écran principal, dans lequel la case à cocher attachée au bouton de contrôle de la règle qu'il vient de définir se garni automatiquement d'une croix. Ce cochage est réalisé de façon automatique. Grâce à celui-ci, le concepteur sait, à tout moment les règles qu'il a déjà définies.

15.7.4. La définition des zones de l'ECRAN du Jeu

Cette dernière partie de l'écran principal de création des jeux est constituée de deux zones. Sa partie supérieure est un feed-back du produit des définitions réalisables au moyen des boîtes de dialogue de la définition des zones de l'écran de jeu. Ce feed-back est donné par défaut à une échelle 1/2. Le concepteur peut demander un affichage plein écran par simple pression du bouton "Taille réelle".

La deuxième zone de cette partie de l'écran regroupe un ensemble de cinq boutons et de cinq cases à cocher. Les comportements de ces objets sont identiques à ceux de la définition des règles vues au point précédent.

défaut à une échelle 1/2. Le concepteur peut demander un affichage plein écran par simple pression du bouton "Taille réelle".

La deuxième zone de cette partie de l'écran regroupe un ensemble de cinq boutons et de cinq cases à cocher. Les comportements de ces objets sont identiques à ceux de la définition des règles vues au point précédent.

15.8. Quelques boîtes de dialogue

Dans cette section, nous présentons quatre boîtes de dialogue. Toutes les boîtes de l'interface emploient les mêmes objets de dialogue et sont organisés de la même façon. Nous avons jugé que l'explication de leur fonctionnement était suffisante pour comprendre l'ensemble de toutes les boîtes utilisées dans l'interface.

Ces boîtes emploient principalement les objets de dialogue suivants. Des boutons radios (petits carrés debout sur un coin) qui ne permettent le choix que d'une possibilité parmi plusieurs. Des cases à cocher (carrés dans lesquels s'inscrivent une croix lorsque l'on clique dessus) qui permettent de sélectionner autant d'éléments que l'on veut parmi plusieurs possibilités. Des combo-listes dans lesquelles on peut choisir des éléments et en introduire de nouveaux. Des listesde sélection dans lesquelles on ne peut que choisir des éléments. Des champs d'édition dans lesquels on peut introduire une chaîne de caractères. Des échelles grâce auxquelles on peut incrémenter ou décrémenter des valeurs.

15.8.1. Création des jetons

■ **Activée par :**

Items "Nouveau ..." et "Ouvrir ..." du menu "Jetons".

Pour définir une caractéristique, le concepteur doit déterminer un préfixe et un suffixe. Il peut soit introduire une chaîne de caractères dans la partie d'édition de la combo-liste pour déterminer une nouvelle caractéristique, soit choisir un élément dans la liste en déroulant celle-ci au moyen du bouton supportant une flèche, situé à la droite du champ d'édition. La chaîne de caractères choisie dans cette liste garnira le champ d'édition automatiquement.

A chaque élément de la liste des préfixes correspond une liste de suffixes.

Lorsqu'il a choisi un élément dans la liste des préfixes et qu'il veut ensuite déterminer un suffixe, il peut soit introduire une nouvelle valeur, soit dérouler la liste qui ne contient que les suffixes attachés au préfixe qu'il a choisi. S'il a introduit un nouveau préfixe, la liste des suffixes est évidemment vide. Signalons enfin que l'ajout d'élements dans les listes est gèré automatiquement.

- **Boutons de contrôle :**

Sauve : permet de sauver la définition en quittant la boîte de dialogue;

Annule : permet de quitter la boîte de dialogue, mais sans sauver la définition en cours ou modifiée.

15.8.2. Création des pions

- **Activée par :**

Boutons "Créer" et "Modifier" de la zone consacrée aux pions dans l'écran principal.

■ **Présentation :**

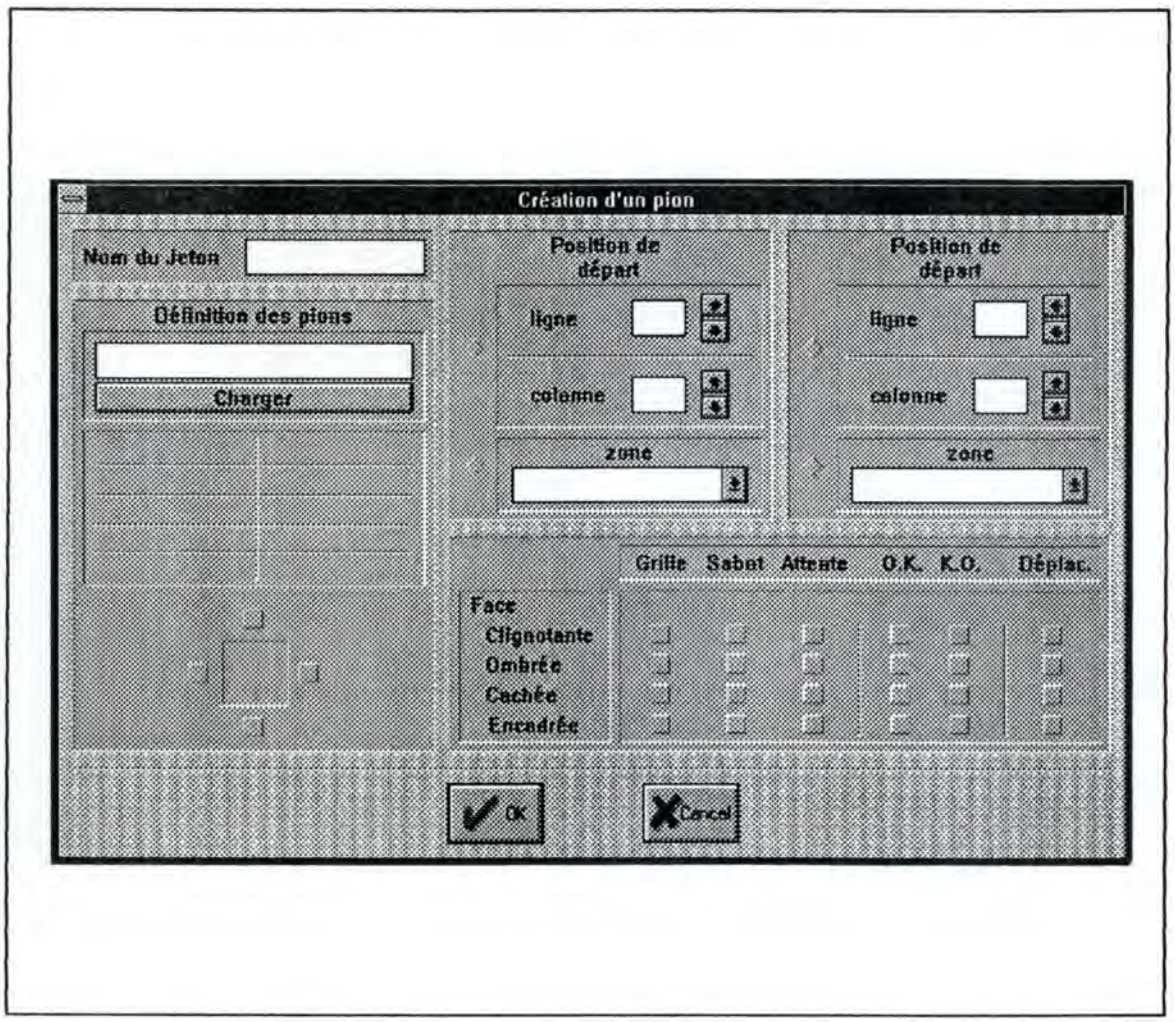


figure 15.4.

La zone du coin supérieur gauche affiche l'intitulé du jeton de référence. La zone juste au-dessous permet de choisir le jeton. Elle présente également les caractéristiques et les côtés associables définis pour le jeton.

Les deux zones du coin supérieur droit permettent de définir une position de départ et une position d'arrivée pour le pion. Ces positions peuvent être soit un couple de coordonnées, soit une zone de l'écran. La dernière zone permet de déterminer la représentation de la face du pion dans diverses situations.

■ **Fonctionnement :**

Le bouton "Charger définition jeton" permet, lorsqu'il est activé, de choisir le jeton de référence du pion. Le champ "Nom du jeton", la zone des caractéristiques, le contenu

et les côtés associables se garnissent alors automatiquement des valeurs définies pour le jetons.

■ **Boutons de contrôle :**

OK : permet de quitter la boîte en sauvant la définition;

Cancel : permet de quitter la boîte, mais sans sauver la définition.

15.8.3. Règles de fin de jeu

■ **Activée par :**

Bouton "Fin de Jeu" de la zone consacrée aux règles dans l'écran principal.

■ **Présentation :**

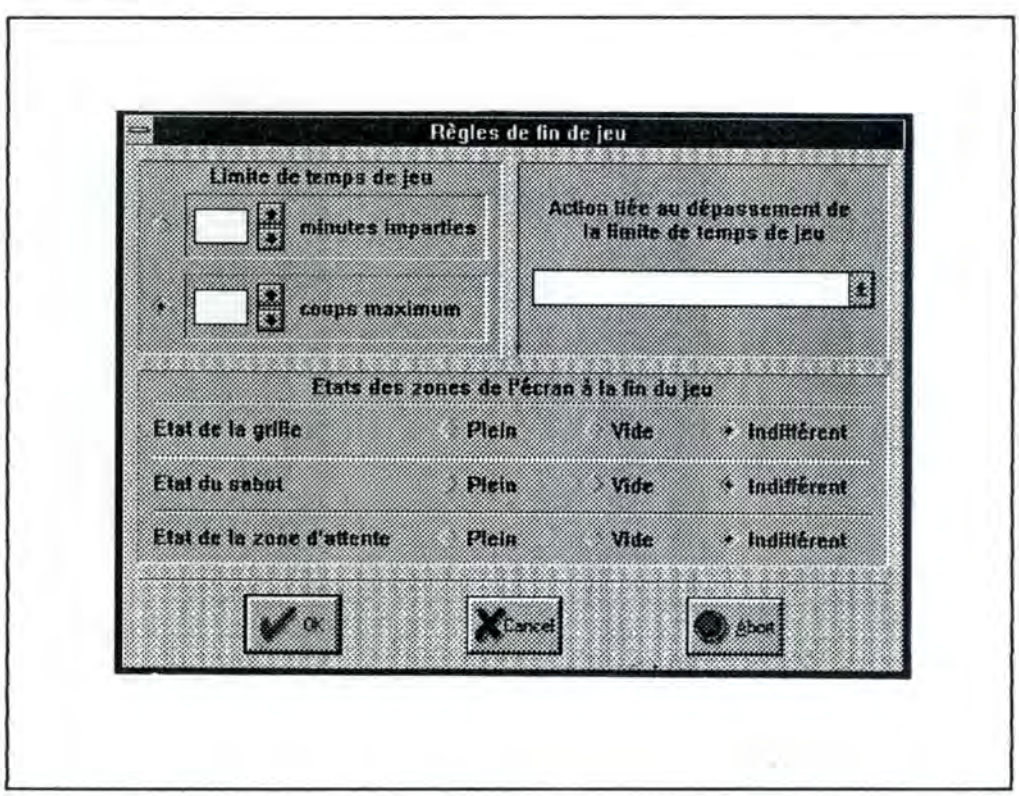


figure 15.5.

La zone du coin supérieur gauche permet de définir un temps ou un nombre de coups limites. La zone du coin supérieur droit offre la possibilité de choisir un événement accompagnant la fin du jeu.

La partie du bas donne accès à la définition des états que les zones de l'écran doivent posséder à la fin du jeu.

■ **Fonctionnement :**

Pour définir la limite du temps de jeu le concepteur dispose de deux champs d'édition accompagnés chacun d'une échelle. Le choix de l'un excluant l'autre, un bouton radio est placé devant chacun de ces deux champs.

L'action liée est simplement choisie en sélectionnant une action dans la liste (du type liste de sélection) qui est déroulable par un simple clic du bouton supportant une flèche vers le bas.

Les états étant mutuellement exclusifs. Cette contrainte est respectée grâce à l'emploi de boutons-radio.

■ **Boutons de contrôle :**

OK : permet de quitter la boîte en sauvant la définition;

Abort : permet de mettre à blanc toute la boîte de dialogue, de façon à pouvoir recommencer la définition;

Cancel : permet de quitter la boîte, mais sans sauver la définition.

15.8.4. Sabot

■ **Activée par :**

Bouton "Sabot" de la zone consacrée aux zones de l'écran de jeu dans l'écran principal.

■ **Présentation :**

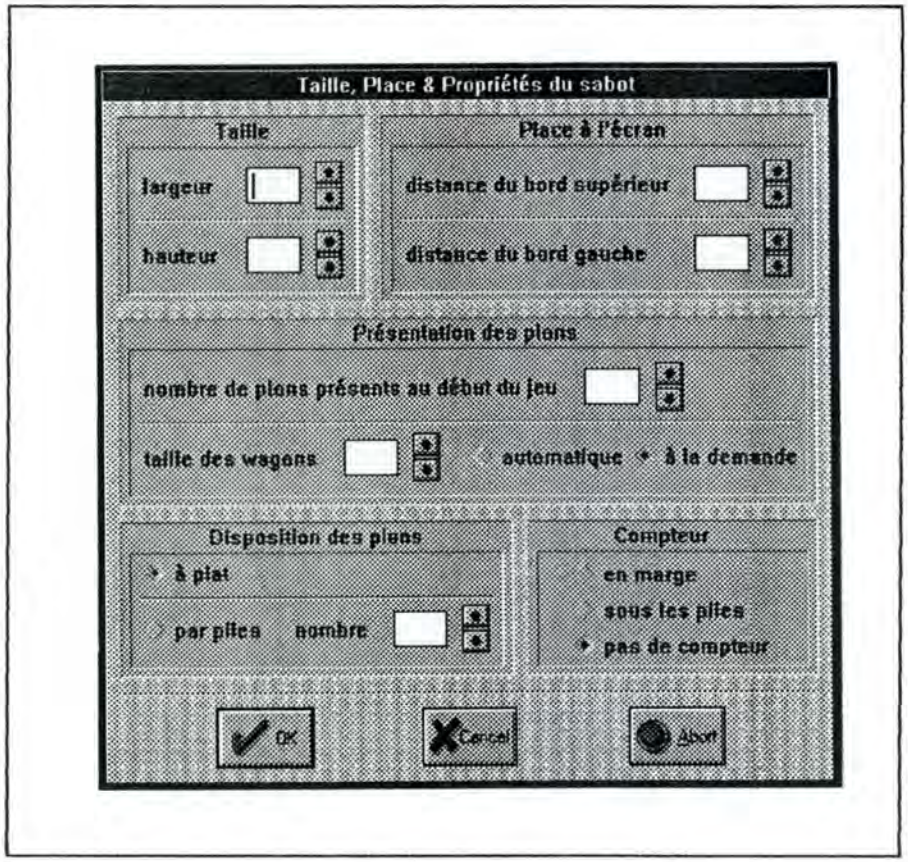


figure 15.6.

La zone du coin supérieur gauche permet la définition de la taille du sabot en nombre de cases, tandis que la zone droite permet de définir le point d'ancrage de celle-ci. La zone intermédiaire permet de définir le nombre de pions qui seront présents au début du jeu et la taille des wagons au cas où tous les pions ne seraient pas présents au départ du jeu.

La zone du coin inférieur gauche permet de décider le mode de disposition des pions tandis que la zone droite permet de choisir un compteur et sa représentation.

■ **Fonctionnement :**

Tous les champs d'édition de nombre sont accompagnés d'une échelle et toutes les cases à cocher sont du type bouton radio.

■ **Boutons de contrôle :**

OK : permet de quitter la boîte en sauvant la définition;

Abort : permet de remettre toute la boîte à blanc;

Cancel : permet de quitter la boîte, mais sans sauver la définition.

Conclusion

Rappel de l'objectif

Notre objectif était de concevoir et de réaliser un générateur de jeux. Ce générateur, nous le destinions à des éducateurs, ayant des personnes handicapées sous leur responsabilité, afin qu'ils puissent créer, sur mesure, des jeux possédant un intérêt éducatif.

Nous voulions que cet outil soit simple d'emploi, c'est-à-dire qu'il offre aux éducateurs un mode aisé de définition de jeux. Nous le voulions également efficace, de façon à ce qu'il permette de créer des jeux ayant un réel intérêt pour les personnes que ces éducateurs encadrent.

Ce que nous avons réalisé

L'analyse et la synthèse

Nous sommes partis de l'idée intuitive que les jeux auraient à se pratiquer sur une grille et que le joueur y manipulerait des jetons. Suite à l'analyse d'un ensemble de jeux correspondant à cette idée, nous avons pu aboutir à l'élaboration d'un paradigme unificateur. Les concepts constituant ce paradigme, organisés en un formulaire, nous ont alors fourni un moyen très simple pour la définition de jeux. Cette simplicité apparaît tout particulièrement au niveau de la définition des règles.

En effet, l'originalité de notre travail réside notamment dans le fait qu'elles peuvent être définies par simple remplissage de la zone qui leur est réservée dans le formulaire. Ce mode de définition ne demande aucun apprentissage particulier, à l'inverse d'un langage qui serait dédié à cette tâche.

Nous avons également tout lieu de croire que cet outil de conception est efficace car, d'une part, il permet de reconstruire les définitions des jeux à partir desquels nous avons réalisé l'analyse, et d'autre part, il permet de concevoir de nouvelles définitions qui donneront naissance à d'autres jeux.

La conception

L'analyse et la synthèse étant réalisées, nous avons entamé la partie purement informatique de notre travail, consistant à élaborer et implémenter un logiciel supportant le générateur.

Dans un premier temps, nous avons dégagé un ensemble de fonctions et de services. Une première partie de cet ensemble est destinée aux concepteurs (c'est-à-dire les animateurs) afin qu'ils puissent définir les éléments constitutifs de la définition des jeux. Nous voulions, évidemment, qu'au travers de l'ensemble des ces fonctions, les éducateurs puissent retrouver la simplicité d'emploi du formulaire.

Le deuxième groupe de fonctions a été mis au point afin que la gestion de la partie dynamique des jeux soit réalisée de façon automatique.

Au cours de cette étape de conception, nous avons également élaboré une interface qui organise l'ensemble de toutes les fonctions consacrées à la partie définition, celle qui sera mise à la disposition du concepteur pour qu'il puisse créer des jeux.

L'implémentation

En ce qui concerne la dernière partie de notre travail, nous nous sommes limités, faute de temps, à l'implémentation de la partie définition des jeux. La partie dynamique des jeux a reçu, tout comme la partie définition, une étude algorithmique, mais sa programmation n'a pas été réalisée.

Prolongement de notre travail

Dans un premier temps, il faudrait terminer la partie implémentation, c'est-à-dire réaliser la dynamique des jeux. De plus, il serait peut-être utile de compléter le logiciel au moyen des concepts et caractéristiques que nous avons mis de côté au moment d'aborder cette implémentation. Il s'agit du concept de pièce et des nombreuses caractéristiques que les objets peuvent posséder (pivotable, superposable, ...).

Enfin, il pourrait être intéressant d'étendre les possibilités de certains concepts. On peut envisager que les jetons puissent contnir des sons ou des séquences animées. Les listes d'actions de l'après validation ou de la fin du jeu pourraient être étendues. Pour l'interface, il serait utile d'élaborer une aide en ligne.

L'interface avec le joueur

Au cours de notre travail, nous nous sommes principalement concentrés sur le logiciel. Il présente l'intérêt de pouvoir générer des jeux au moyen d'un outil simple. Cependant, compte tenu de la population de joueurs à laquelle les jeux définissables s'adressent, le logiciel devrait être accompagné d'une interface adaptée aux difficultés rencontrées par les joueurs.

Bien qu'étant conscients de l'importance d'une telle interface, nous ne l'avons pas abordée car sa conception est un sujet indépendant du problème que nous avons traité.

Conclusion

L'outil que nous avons conçu présente deux particularités. D'une part, il offre la possibilité de créer des jeux, et d'autre part, cette création a été rendue très simple à réaliser. Ces deux atouts nous font croire que le générateur, une fois complètement implémenté pourra rendre de nombreux services à des éducateurs.

En effet, non seulement le générateur leur permettra de définir des jeux existants (pour autant qu'ils appartiennent à la famille de jeux sur lesquels nous nous sommes basés pour l'analyse) mais il leur offira aussi la possibilité de créer de nouveaux jeux. Ils pourront également faire constamment évoluer les définitions des jeux et ainsi les adapter aux progrès que les joueurs réaliseront. De même, au départ d'une idée de jeu, ils pourront envisager de nombreuses variantes, destinées chacune, à rencontrer les déficiences d'une personne handicapée particulière.

Bibliographie

Baudrenghien S., Demo R. (1992), "Elaboration d'une boîte à outils d'aide à la réalisation d'interfaces pour personnes handicapées", F.U.N.D.P., Namur

Bodart F. (1992), Cours interface homme machine, F.U.N.D.P., Namur

Bossuet G. (1982), "L'ordinateur à l'école", col. "L'éducateur", Presses Universitaires de France, Paris

Dubois E. (1992), Cours de Méthodologie de Développement de Logiciel, F.U.N.D.P., Namur

Lefevre L., Delchet R. (1977), "Les handicapés psychiques" in "L'éducation des enfants et des adolescents handicapés en milieu scolaire et para-scolaire", tome 2, E. S. F., Paris

Vanderdonckt J. (1992), "Corpus ergonomique minimal des applications de gestion", F.U.N.D.P., Namur

Vial J. (1981), "Jeu et éducation. Les ludothèques", col. "L'éducateur", Presses Universitaires de France, Paris