TP « Jeu de la vie »

N. Delestre

Règles [1]

En préambule, il faut préciser que le jeu de la vie n'est pas vraiment un jeu au sens ludique, puisqu'il ne nécessite aucun joueur ; il s'agit d'un automate cellulaire, un modèle où chaque état conduit mécaniquement à l'état suivant à partir de règles pré-établies.

Le jeu se déroule sur une grille à deux dimensions, théoriquement infinie (mais de longueur et de largeur finies et plus ou moins grandes dans la pratique), dont les cases, qu'on appelle des « cellules », par analogie avec les cellules vivantes, peuvent prendre deux états distincts : « vivantes » ou « mortes ».

À chaque étape, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines de la façon suivante :

— Une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît).
— Une cellule vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt.

Objectif général

Écrire un programme qui affiche les n premières générations d'une grille initialisée aléatoirement (avec un seuil de remplissage) et dont la taille est donnée par l'utilisateur. On suppose pour cela que l'on possède la fonction suivante :

— **fonction** obtenirNaturelAléatoire (borneMax : **NaturelNonNul) : Naturel**

Analyse

L'analyse descendante du problème *simulerJeuDeLaVie* est présentée par la figure 1.

Conception préliminaire

Voici les signatures des fonctions et procédures issues de cette analyse descendante.

— **procédure** initialiser (**E/S** g : Grille,**E** tauxDeRemplissage : 0..100)
— **procédure** calculerNouvelleGeneration (**E/S** g : Grille)
— **procédure** afficher (**E** g : Grille)
— **fonction** vaNaitreOuContinuerAVivre (g : Grille, x,y : **NaturelNonNul) : Booleen**

⌊**précondition(s)** x≤obtenirLargeur(g)
y≤obtenirHauteur(g)

1. Wikipédia

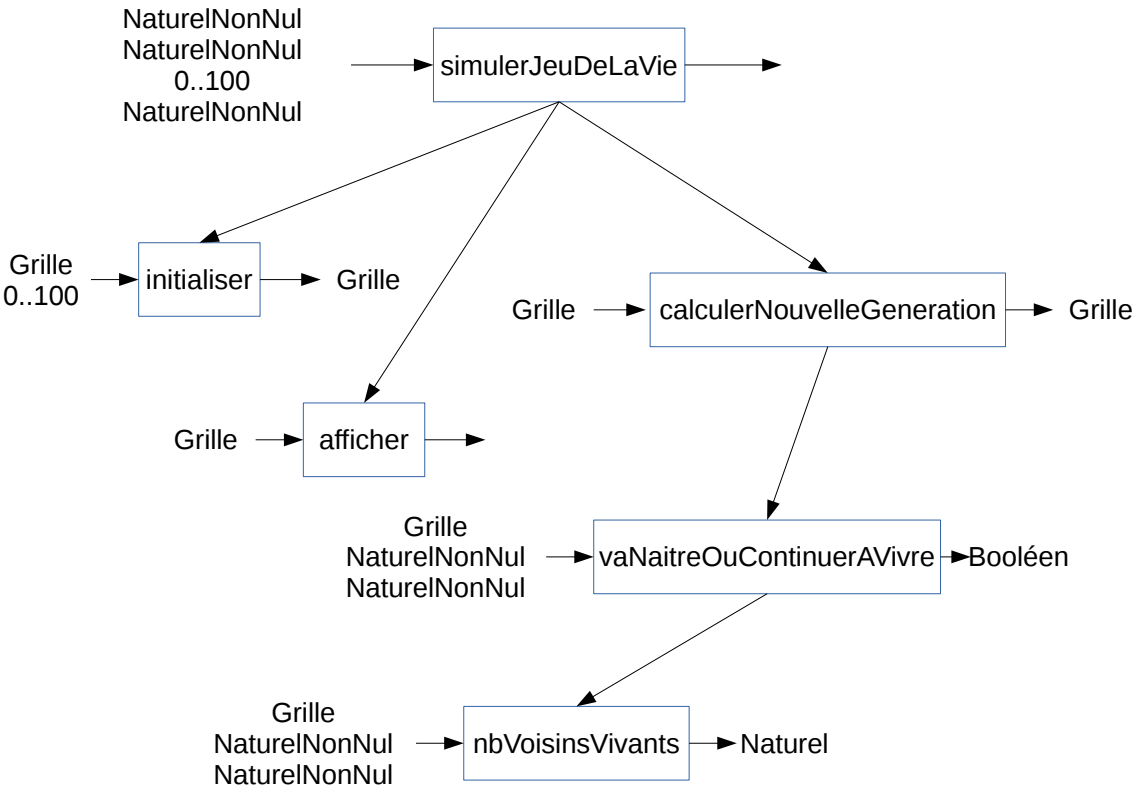

FIGURE 1 – Analyse descendante

— **fonction** nombreVoisinsVivants (g : Grille, x,y : **NaturelNonNul) : Naturel**

⌊**précondition(s)** x≤obtenirLargeur(g)
y≤obtenirHauteur(g)

Et voici celles permettant de manipuler le type `Grille` :

— **fonction** grille (largeur, hauteur : **NaturelNonNul) :** Grille
— **fonction** obtenirLargeur (g : Grille**) : NaturelNonNul**
— **fonction** obtenirHauteur (g : Grille**) : NaturelNonNul**
— **fonction** estUneCelluleVivante (g : Grille, x,y :**NaturelNonNul) : Booleen**

⌊**précondition(s)** x≤obtenirHauteur(g) et y≤obtenirHauteur(g)

— **procédure** tuerCellule **(E/S** g : Grille, **E** x,y :**NaturelNonNul)**

⌊**précondition(s)** x≤obtenirHauteur(g) et y≤obtenirHauteur(g) et estUneCelluleVivante(g,x,y)

— **procédure** faireNaitreCellule **(E/S** g : Grille, **E** x,y :**NaturelNonNul)**

⌊**précondition(s)** x≤obtenirHauteur(g) et y≤obtenirHauteur(g) et non estUneCelluleVivante(g,x,y)

Conception détaillée

Voici les algorithmes de certaines fonctions et procédures.

fonction nombreVoisinsVivants (g : Grille, x,y : **NaturelNonNul) : Naturel**

⌊**précondition(s)** x≤obtenirLargeur(g)
y≤obtenirHauteur(g)

Déclaration i,j : **NaturelNonNul**
minx,miny,maxx,maxy, resultat : **Naturel**

debut
resultat ← 0
minx ← max(1,x-1)
miny ← max(1,y-1)
maxx ← min(obtenirLargeur(g),x+1)
maxy ← min(obtenirHauteur(g),y+1)
pour i ←minx **à** maxx **faire**
pour j ←miny **à** maxy **faire**
si estUneCelluleVivante(g,i,j) et (i≠x ou j ≠ y) **alors**
resultat ← resultat+1
finsi
finpour
finpour
retourner resultat
fin

fonction vaNaitreOuContinuerAVivre (g : Grille, x,y : **NaturelNonNul) : Booleen**

⌊**précondition(s)** x≤obtenirLargeur(g)
y≤obtenirHauteur(g)

Déclaration nbVoisinsVivants : **Naturel**

debut

```
    nbVoisinsVivants ← nombreVoisinsVivants(g,x,y)
    si (non estUneCelluleVivante(g,x,y) et nbVoisinsVivants=3) ou (estUneCelluleVivante(g,x,y) et
    (nbVoisinsVivants=3 ou nbVoisinsVivants=2)) alors
        retourner VRAI
    sinon
        retourner FAUX
    finsi
fin
procédure calculerNouvelleGeneration (E/S g : Grille)
    Déclaration  temp : Grille
                 i,j : NaturelNonNul
debut
    temp ← grille(obtenirLargeur(g),obtenirHauteur(g))
    pour i ←1 à obtenirLargeur(g) faire
        pour j ←1 à obtenirHauteur(g) faire
            si vaNaitreOuContinuerAVivre(g,i,j) alors
                faireNaitreCellule(temp,i,j)
            finsi
        finpour
    finpour
    g ← temp
fin
```

Développement

Compléter le fichier src/JeuDeLaVie.c proposé dans l'archive disponible depuis Ubiquity (un paramètre à la fonction *simulerJeuDeLaVie* a été ajouté pour séparer la logique métier de l'interface).

Pour rappel, il ne faut pas supprimer les lignes qui commencent par `// @u:start` et `// @u:end`.

Pour aller plus loin

Pour en savoir plus sur les jeux de la vie, je vous invite (très fortement) à voir l'excellente vidéo de la chaîne youtube « Science étonnante » : `https://www.youtube.com/watch?v=S-W0NX97DB0`