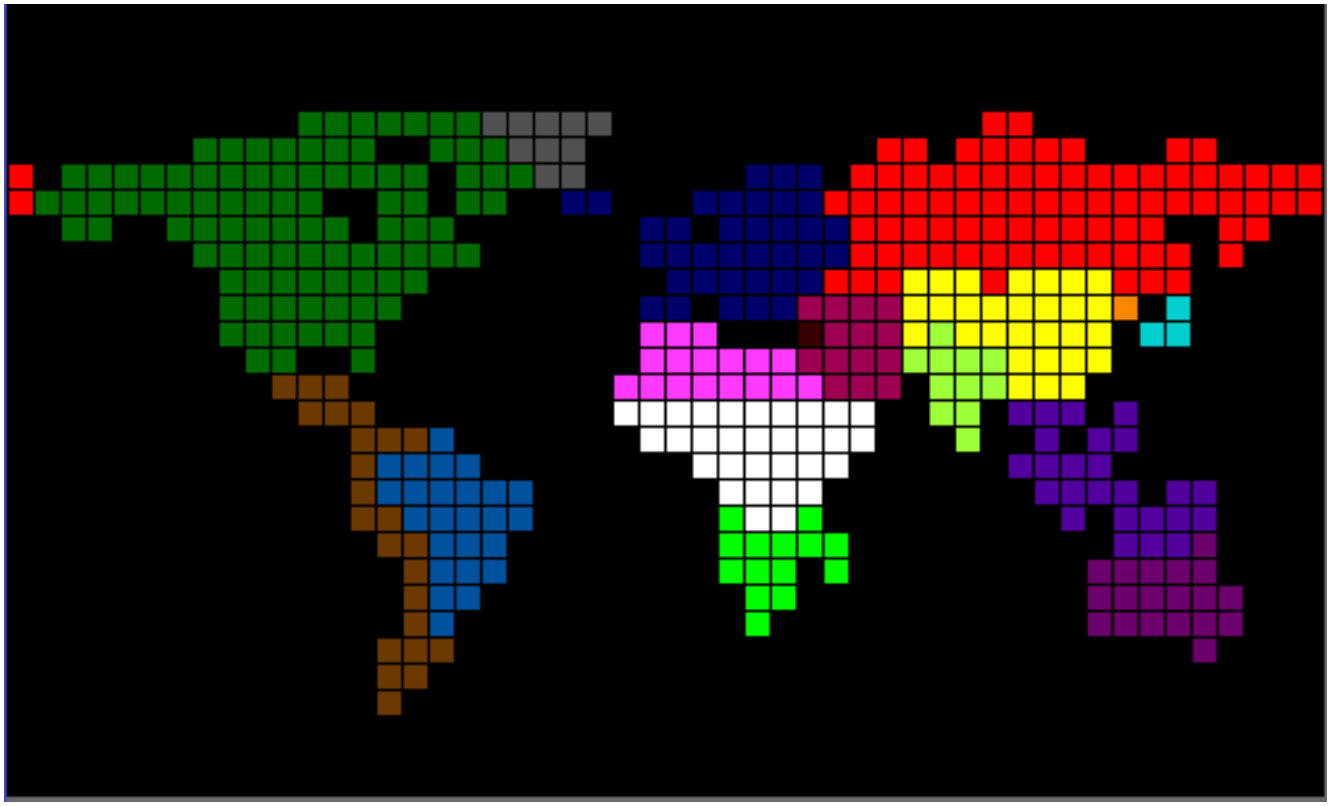
TPE

Travaux Personnels Encadrés



Comment pourrait on prédire l'Avenir d'une Civilisation ?

ROMAINPC |Nono13064 |theroxas898
Lycée ****** ******

Thème : L'Aléatoire, l'Insolite, le Prévisible
Matières : Mathématiques et Sciences de L'Ingénieur

Sommaire :

Introduction :

En voyant les thèmes proposés pour les TPE, nous n'avons au début pas trouvé d'inspiration face à ces thèmes un peu trop vastes… Or l'un d'entre eux nous a quand même attiré : L'aléatoire, l'insolite et le prévisible. Effectivement nous aimons tous les trois l'informatique et la programmation et ce sujet sous-entend ces notions très fortement car avec un ordinateur l'on peut concevoir des algorithmes capables de calculer ou de simuler à peu près tout, de même l'on peut y insérer de l'aléatoire (du « random ») ce qui permet de concevoir des programmes (et surtout des jeux en fait) qui pourrait donner des résultats différents et il devient alors intéressant d'en tirer des études statistiques pour voir les tendances des résultats : c'est le domaine de la probabilité.

C'est en cherchant un sujet sur ce thème que nous avons pensé au Jeu De La Vie. Ce « Jeu » n'est en fait pas trop un jeu mais un algorithme capable de prédire le comportement de petites cellules, il suffit de les placer comme on le souhaite au départ et le programme les fait évoluer sous nos yeux, ainsi l'on peut savoir facilement leur position à tel ou tel instant car l'ordinateur se charge instantanément (ou presque) de faire les calculs. Nous détaillerons l'utilité de ce programme après cette introduction.

Le sujet fut alors rapidement trouvé, nous nous sommes en effet poser cette question : peut-on comme pour la météo et de la même manière que le Jeu De La Vie prédire quelque chose qui nous concerne tous, prédire l'avenir de l'humanité ?

Nous avons donc choisi ce sujet pour tenter non pas de prédire exactement ce qui se passera à l'avenir mais démontrer qu'il serait possible de le faire. De plus ce sujet va nous permettre de tester nos talents en programmation et de produire quelque chose de concret: un programme qui permettrait d'après un modèle de fonctionnement fait de lois mathématiques de simuler des civilisations humaines.

Le Jeu De La Vie:

Le jeu de la vie fut imaginé en 1970 par John Horton Conway, mathématicien anglais. Le jeu de la vie est en fait un simulateur ou un « automate cellulaire ». Le principe est simple (l'on peut utiliser un ordinateur ou même le faire sur papier avec de la patience), il suffit de prendre un plan avec dessus un quadrillage, dans chaque case l'on peut placer ou non une cellule (on colorie la case ou on place une bille ou n'importe quoi).

Ensuite chaque case du tableau est testée :

- si elle est vide alors si elle est entourée de exactement 3 cellules vivantes il va y naître une cellule.
- si elle est pleine, la cellule ne peut y survivre que si elle est entouré de 2 ou 3 cellules.

Ainsi la position des cellules évolue au cours du temps.

Pour tester ce jeu nous vous conseillons ce site :

http://jm.davalan.org/divers/jeuvie/index.html

Image issue de Wikipédia

Ainsi le Jeu De La Vie à démontrer qu'il était possible d'illustrer simplement des cellules qui interagissent via des lois très précises. Et cette méthode est applicable à d'autres choses, comme par exemple au trafic routier (si il y a de la place devant, la voiture avance) ce qui permis d'observer l'effet accordéon lors des bouchons.

Et nous nous apprêtons à appliquer cette méthode pour régir le comportement de civilisations.

Comme vous avez pu le constater le « jeu » de la vie porte quand même bien son nom car c'est il est amusant de tester différentes configurations et l'on peut créer des canons, des vaisseaux, des dessins et bien d'autres choses.

Ainsi en hommage à ce Jeu de la vie et pour espérer aboutir sur un programme ou l'on prend plaisir à faire interagir des civilisations, nous allons nommer notre programme :

Le Jeu Des Civilisations

Ce que fera le programme et comment :

Tout d'abord parlons de la programmation :

Notre programme sera codé en Java, le Java est un langage de programmation appartenant à la société Oracle et créé en 1995 par Sun Microsystems.

Un langage de programmation est une suite d'instruction permettant de concevoir un programme, les personnes n'en n'ayant jamais vu se dise souvent « on comprends rien c'est l'ordinateur qui nous sort ça! », mais rassurez vous l'ordinateur n'y comprends rien non plus. En effet l'ordinateur comprends uniquement les chiffres 0 et 1 et rien d'autre, c'est le binaire. Or au niveau du processeur, le courant électrique passe(1) ou pas(0) dans des « portes logiques » , ces portes permettent de faire des calculs de bases (+ - * et /) ou de tester des conditions (si , ou ,et ,…)
Ainsi pour programmer un ordinateur il faut lui entrer des suites d'instructions, de conditions à tester, d'opération à faire, … Mais il serait très fastidieux de lui dire en binaire, d'où l'intérêt d'un langage de programmation ! Il est chargé de traduire ce que vous lui dites vers ce langage en binaire pour que l'ordinateur le comprenne, c'est la « compilation ».

Il existe de nombreux langages de programmation, certains comme le java, le C, le C++,... permette de faire des applications (jeux, logiciels,…) et d'autre se spécialise pour le langage internet (comme HTML) ou autre. Globalement ces langages présente les mêmes possibilités et la syntaxe (la façon d'écrire) des commandes et similaire.

Pour notre part nous avons choisi le Java car nous avions déjà quelques bases car nous avons tous les trois participé au concours algorea et avons appri les bases du langage sur le site France IOI, de même le Java est un langage très connu et c'est l'un des plus puissant sur le marché, il est concurrencé par le C++, langage similaire quoique un peu moins rapide mais qui contrairement au Java n'est pas privé.
Nous avons donc par conséquent approfondi nos connaissance en Java sur le site OpenClassroom, connaissant les bases de la programmation en Java, ce site nous permis d'apprendre quelques notions utiles, la programmation orientée objet et surtout de savoir créer des fenêtres, y dessiner dedans et placer des composants(boutons,…) et interagir avec.

Pour ce qui est du programme, nous le coderons sur le logiciel Éclipse, ce type de logiciel permet de lancer le programme facilement et il nous signale toutes erreurs. Dans ce document nous vous présenterons le code complet mais nous ne nous étendrons pas sur l'apprentissage du Java.

Objectifs du programme :

Voici ce que le programme devra faire, nous avons bien sûr imaginé ces formules, car comme nous le répétons, ce modèle de fonctionnement ne représente pas exactement la réalité mais nous avons tenté du moins de s'en approcher d'après nos connaissances.

Il devra :

- Créer un monde (30 x 50 cases) avec des zones inexploitables, d'autres avec des ressources alimentaires et économiques aléatoires. Si l'on sort d'un côté au ressort de l'autre, notre monde est donc un tore (un donut).

- Créer un certain nombre de civilisations et leur donner une case de départ.

Elle évolueront via des facteurs : Population, Production, Économie, Armée, Politique, Politique d'extension, Culture, Éducation, Science, Nourriture, Bonheur et Influence.

- Les civilisations s'étendront

- Les civilisations auront entre elles des relations amicales ou non, qui aboutirons à des fusions ou à des guerres.

- Le programme affichera la carte ainsi que 2 tableaux informant sur les facteurs des civilisations et un autre pour afficher leurs relations.

- Il sera possible de faire deux types de simulations :

  - avec des événements aléatoires
  - ou sans

- Éventuellement si nous avons le temps : rajouter des boutons pour mettre en pause la simulation, pour rajouter ou détruire une civilisation ou pour modifier certains de leurs paramètres. Nous envisageons aussi de pouvoir sauvegarder et charger des simulations. NB : ces modifications ne seront pas présente dans ce document, vous en verrez le résultat sur le programme même.

- Éventuellement aussi nous testerons notre modèle sur une reproduction du monde actuel, pour voir d'après nos lois comment le monde évoluera.

Comment avons-nous codé le programme :

NB : Vous trouverez au chapitre suivant le code source du programme, nous y avons ajouté des numéros (N) pour que vous vous y retrouviez. Notez que nous ne vous expliquerons pas toutes les subtilités du java mais comment « raisonne » le programme. Autre point : vous remarquerez qu'un programme s'exécute de haut en bas.

(0) Petite subtilité, le programme démarre en fait dans ce bloc à la fin du programme(p 26), ici il s'occupe juste d'appeller la classe « JeuDesCivilisations », donc de repartir au début du code.

(1) Pour commencer le programme crée des « variables », c'est la base de la programmation et le plus dur à comprendre (au début seulement), une variable n'est ni plus ni moins qu'un emplacement mémoire (situé dans la RAM, ou « mémoire vive » pour être précis) qui va stocker une valeur, l'on peut ainsi y garder des valeurs et les modifier pour les utiliser dans le programme.
Dans notre cas il y a deux variables, une pour savoir combien de civilisations il y aura et une autre pour le temps (en tour) d'exécution.
Nous initialisons aussi un « tableau », un regroupement de variables pour stocker toutes valeurs liées à la carte, il est en 3 dimensions, nous pouvons ainsi stocker 4 valeurs dans chaque case de la carte, chaque case est située à un emplacement ligne, colonne du tableau. Ainsi le monde fait 30 x 50 cases et chacune stocke 4 variables : le climat (qui défini l'utilisation ou non d'une case), les ressources alimentaires, les ressources économiques et matérielles et une valeur indiquant à quelle civilisation appartient la case. Les 3 premières valeurs sont entre 0 et 9.
Un autre tableau plus simple à 2 dimensions stocke les facteurs des civilisations, on identifie la civilisation par son numéro puis on va à la colonne (elle aussi numéroté) qui stocke le facteur voulu. (ex : 3 = population).
Enfin un tableau lui aussi à deux dimensions stocke les relations (numérotées entre 0(mauvaise entente) et 9(très bonne entente)) entre civilisations, vous remarquerez lors de l'exécution qu'il est triangulaire tout simplement car la moitié inférieur est inutile (car il est inutile de savoir les relations entre la civilisation 5 et la 1 si l'on connaît celles entre la 1 et 5). Le dernier tableau est identique mais il contient des nombres à virgules permettant plus de paliers possibles de relations.

(2) Par la suite le programme paramètre la fenêtre, la rend visible et lance la partie principale du programme : la méthode Go()

(3) Le programme commence par paramétrer la partie :
-il génère la carte en définissant les 3 premières valeurs que stockent chaque case

aléatoirement.
-il génère les civilisations en définissant certaines valeurs de bases, définit leur couleur et choisit aléatoirement leurs politiques.
-il choisit ensuite les cases de départ des civilisations.

(4) Le programme lance ensuite une boucle infinie(qui ne se termine jamais) qui constitue la partie la plus important, chaque boucle correspond à un tour, ou une unitée temps. Tout d'abord le programme met un temps d'attente (Thread.sleep(nb de ms);), puis il parcours la carte pour se renseigner de divers choses, notamment le nombre de cases possédées par chaque civilisation.

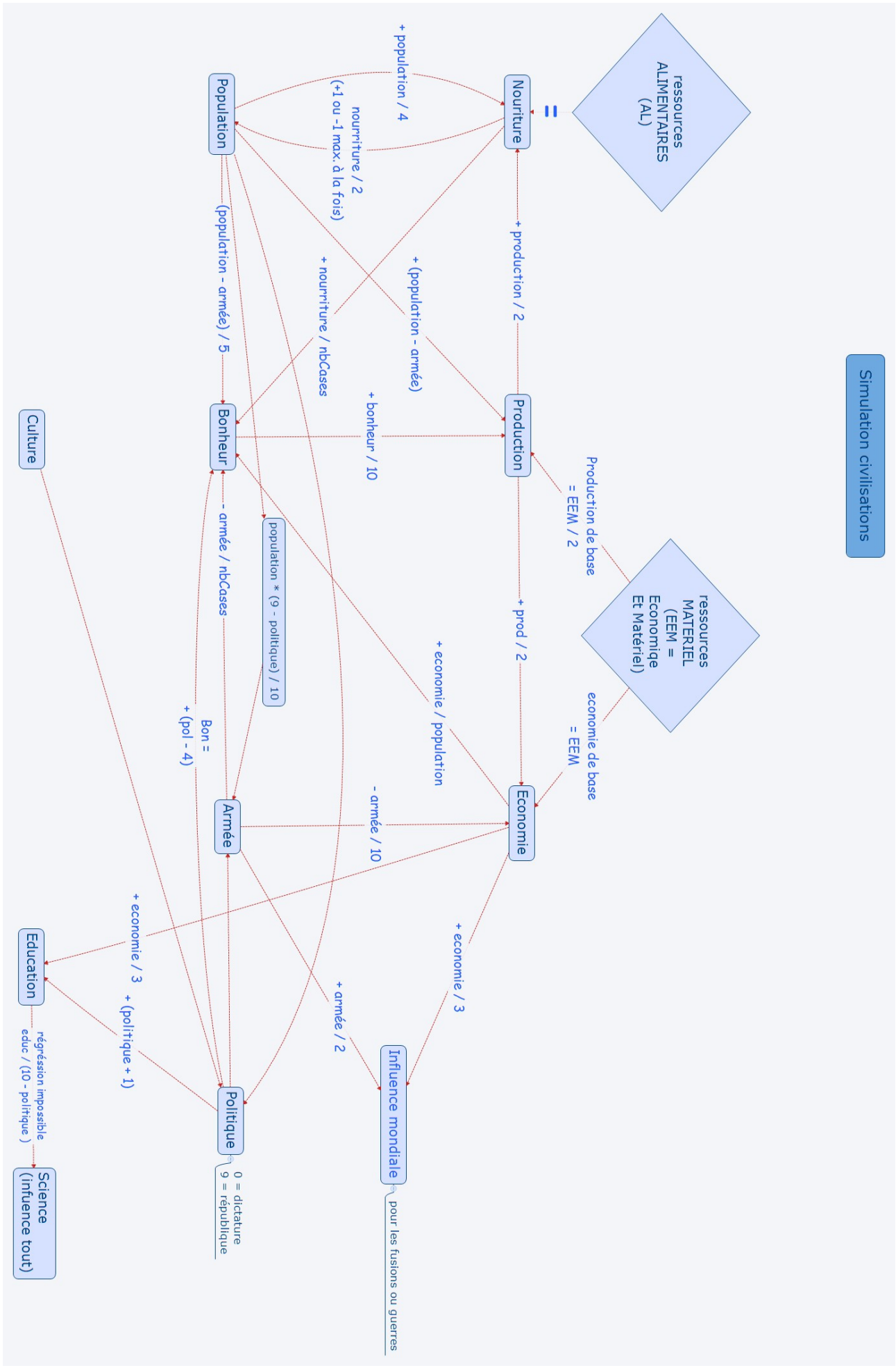
(5) Par la suite le programme va mettre à jour les facteurs que nous avons défini, vous pouvez voir page suivante un arbre indiquant quel facteur influe sur quoi. Il s'occupe de chaque civilisation une par une .
La plupart des facteurs sont définis par une formule, de ce fait leur valeur est donnée instantanément par le programme, en effet d'autres facteurs ne sont pas régis par une formule : la population ne peut augmenter ou baisser que de 1 par tour, la science ne peut jamais baisser et la culture ou la politique peuvent changer dans certaines conditions seulement (augmentation au fil du temps pour la culture et augmentation de la politique en cas de bonheur trop bas).
Ensuite le programme vérifie que la civilisation est toujours en vie. Sinon il retient qu'il ne faut plus l'étudier.

(6) La gestion des événements commence, pour commencer pour chaque civilisation on test si elle va s'étendre d'une case ou pas, ceci se déclenche si la population stagne ou si elle a attendu un certain nombre de tours (variable selon sa politique d'extension), le programme lui choisit alors une case propice pour s'installer.

(7) Ensuite le programme regarde les relations entre civilisations, si elles sont à 0 alors la guerre est déclenchée et le programme défini les fronts où auront lieu les batailles, chaque bataille confronte deux cases situées sur les frontières entre les civilisations concernées.
Si les relations sont à 9 alors les civilisations fusionnent, le programme ajuste les facteurs non définis par une formule (les autre seront actualisés au prochain tour instantanément), crée une nouvelle couleur, ajuste les relations avec le reste du monde et supprime une des deux civilisation du tableau après avoir donné son territoire.

Suite dans deux pages...

Arbre d'influence des facteurs entre eux



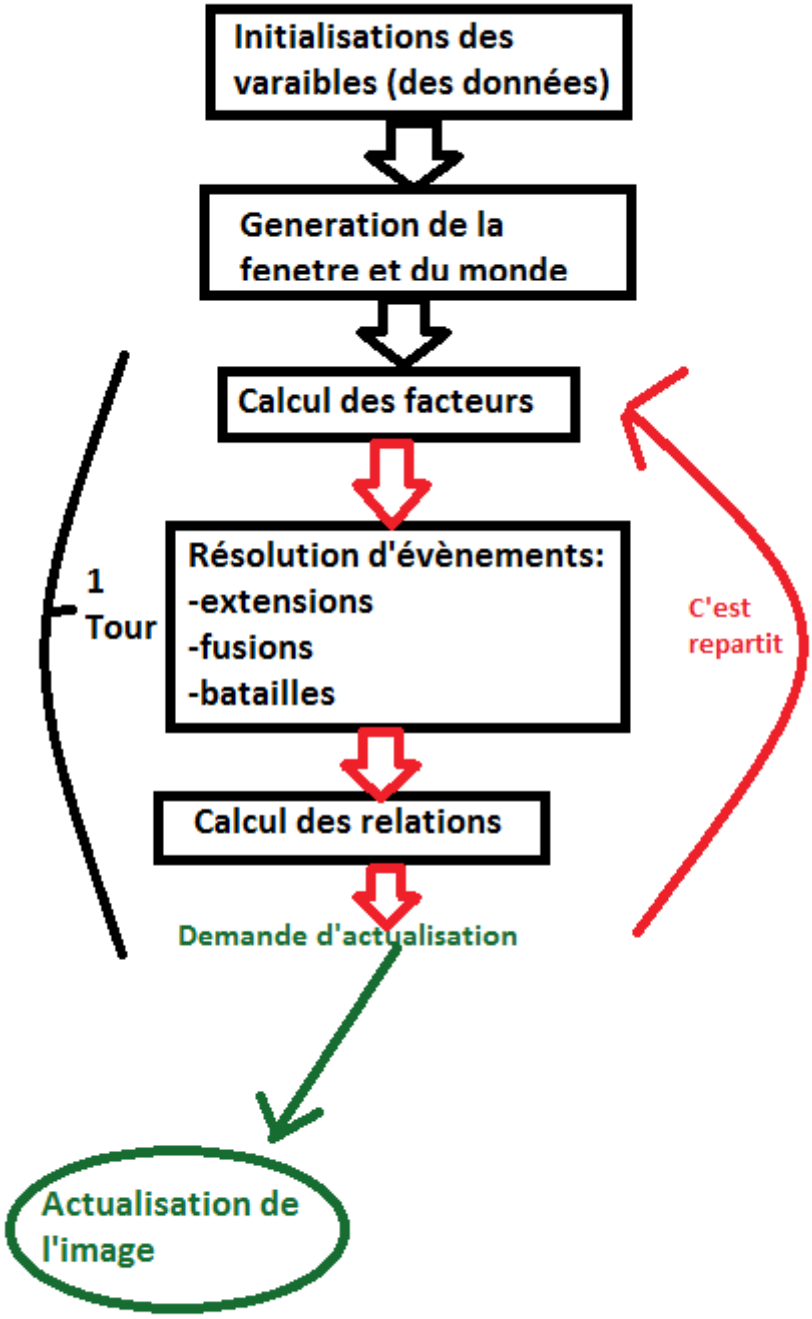
(8) Le programme va alors s 'occuper des guerres, nous avons avant le point (7) créé un tableau informant de la situation des batailles et de quelle case est contre quelle case, il peut aussi stocker le nombre de soldats sur le terrain. Le programme place donc les soldats sur place, pour ce faire il répartit équitablement (mais sans pour autant diviser pour éviter d'avoir par exemple 42,33 soldats) les soldats, donc l'armée, puis les points de sciences, et chaque case gagne un nombre de soldats égal au nombre de ressources économiques et matérielles sur la case.
Ensuite il reste à faire les batailles, chaque front est étudié un par un :
- les soldats science et ressources meurent les premiers (1 mort d'un coté 1 de l'autre)
- si une civilisation n'a plus ces soldats, alors les vrais soldats meurent à leur tour, ils sont directement soustrait à la population de la civilisation, au prochain tout la formule de l'armée reprendra un certain pourcentage de soldats dans la population, notez que ce pourcentage augmente en cas de guerre et varie selon la politique.
- une case est vaincue lorsqu'il n'y a plus aucun soldat, cette case appartient alors à la civilisation gagnante, cette bataille est résolue, le front supprimé.

(9) Ensuite le programme va faire évoluer les relations entre civilisations, l'on étudie chaque relation une par une(la 1-2, la1-3,..., la 3-5,...,la 9-10)
Les relations évoluent tout d'abord entre 1 et 8 (plus c'est bas moins ça va) et avec des nombres à virgules, ainsi cela baisse ou monte petit à petit (de 0,01), elle évolue en fonction des différences de politiques et de politiques d'extension.
Puis si leur relations sont à 1 alors le programme peut envisager de déclencher la guerre (donc relations à 0) en fonction des différences économiques et politiques ou si l'une d'entre elle a perdu une proportion (variable a) trop grande de soldats. Dans le cas ou la guerre est en cours, si tous ces critères ne sont plus validés la guerre s'arrête mais la civilisation la plus influente ajoute à son territoire les cases frontières.
Dans le même principe des relations à 8 peuvent passer à 9 si leur influence n'est pas trop grande et si elles ont la même politique.

(10) Le programme effectue ensuite des test pour vérifier si les civilisations entre en contact, vous remarquerez que si elle ne se connaisse pas leurs relations sont à – 1 et dès qu'elles se rencontrent cela passe à 5 (neutre).

(11) Pour finir le programme actualise la fenêtre (remarquez à la fin de la boucle la commande repaint() pour « repeindre » la fenêtre, de même le temps augmente de 1), un tour s'est écoulé avec des changements dans les facteurs des civilisations, dans leur disposition ou leur relations. Le résultat est instantanément visible et la boucle repart pour un nouveau tour...

Pour résumer :



Le programme code source :

Voici le code source du programme version simple(cf chapitre notes de versions).

Rappel : cherchez les nombres (N))

Aidez vous des commentaires (« // ») qui décrive le code

Nous conseillons de se référer aussi à la version PC qui n'a pas les problèmes de retour à la ligne qui rendent ce code un peu sale dans un logiciel de traitement de texte. Nous avons d'ailleur supprimé « l'indentation » ici et donc chaque ligne de code commence tout à gauche de la page.

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;

public class JeuDesCivilisations extends JFrame {
//VERSION SIMPLE
//PAR ROMAINPC Nono13064 theroxas898

(1)//Variables:
public int nombreCivilisations = 10;
public long temps = 0L;
//tableau de génération de la carte:
//0=climat    1=ressources AL    2=ressources EEM   3=appartenance de la case
public int[][][] monde = new int[30][50][4];
//tableau des civilisations:
//0->R 1->G   2->B   3->population 4->production 5->économie   6->influence
//7->bonheur  8->culture    9->armée      10->éducation 11->science
//12->nourriture    13->politique 14->politique territoriale 15->AL total  16->EEM total
//17->cases possédées      18->1=décès2=alliance      19-> pop provisoire
//20->est en guerre avec qq`(1=oui)            21->nb batailles à faire   NB:int[0][x]-
>case n'appartenant à personne
public int[][] civilisations = new int[nombreCivilisations + 1][22];
//tableau des relations:
public int[][] relations = new int[nombreCivilisations][nombreCivilisations - 1];
//tableau des relations provisoires:
public double[][] relationsProv = new double[nombreCivilisations][nombreCivilisations -
1];

(2)//création de la Fenetre:
public JeuDesCivilisations(){
//paramétrage:
this.setTitle("Jeu des Civilisations");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setResizable(false);
this.setSize(1350, 720);
this.setLocationRelativeTo(null);
this.setBackground(new Color(100, 100, 100));

//visibilité:
this.setVisible(true);

//définition du content pane:
Panneau container = new Panneau();
this.setContentPane(container);

//debut du programme:
Go();
}
```

```
//fonction principale:
public void Go(){

(3)//génération de la carte:
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
monde[ligne][colonne][0] = (int)(Math.random() * 7);
monde[ligne][colonne][1] = (int)(Math.random() * 10);
monde[ligne][colonne][2] = (int)(Math.random() * 10);
}
}


//génération des Civilisations:
for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
civilisations[0][0] = 75; civilisations[0][1] = 75; civilisations[0][2] = 75;
//couleurs:
switch(loop){
case 1:
civilisations[loop][0] = 0;civilisations[loop][1] = 0;civilisations[loop][2] = 255;
break;
case 2:
civilisations[loop][0] = 255;civilisations[loop][1] = 0;civilisations[loop][2] = 0;
break;
case 3:
civilisations[loop][0] = 0;civilisations[loop][1] = 255;civilisations[loop][2] = 0;
break;
case 4:
civilisations[loop][0] = 255;civilisations[loop][1] = 0;civilisations[loop][2] = 255;
break;
case 5:
civilisations[loop][0] = 255;civilisations[loop][1] = 255;civilisations[loop][2] = 0;
break;
case 6:
civilisations[loop][0] = 0;civilisations[loop][1] = 255;civilisations[loop][2] = 255;
break;
case 7:
civilisations[loop][0] = 100;civilisations[loop][1] = 0;civilisations[loop][2] = 0;
break;
case 8:
civilisations[loop][0] = 0;civilisations[loop][1] = 0;civilisations[loop][2] = 100;
break;
case 9:
civilisations[loop][0] = 0;civilisations[loop][1] = 100;civilisations[loop][2] = 0;
break;
case 10:
civilisations[loop][0] = 255;civilisations[loop][1] = 255;civilisations[loop][2] = 255;
break;
default:
civilisations[loop][0] = (int)(Math.random() * 256);
civilisations[loop][1] = (int)(Math.random() * 256);
civilisations[loop][2] = (int)(Math.random() * 256);
}


civilisations[loop][7] = 10;
civilisations[loop][3] = 1;
civilisations[loop][8] = - 1;
civilisations[loop][13] = (int)(Math.random() * 10);
civilisations[loop][10] = civilisations[loop][13] + 1;
civilisations[loop][14] = (int)(Math.random() * 10);
}
civilisations[0][18] = 1;
```

```
//gestion des relations:
for(int ligne = 0 ; ligne < nombreCivilisations ; ligne++){
for(int colonne = 0 ; colonne < nombreCivilisations - 1 ; colonne++){
if(colonne > ligne - 1){
relations[ligne][colonne] = - 1;
}else{
relations[ligne][colonne] = 42;
}
}
}

//placement des civilisations:
for(int loop2 = 1 ; loop2 <= nombreCivilisations ; loop2++){
int climatCase = 0;
int présence = 42;
int nourriture = 0;
int x = 0; int y = 0;
while(climatCase == 0 || présence != 0 || nourriture < 7){
x = (int)(Math.random() * (30 - 0));
y = (int)(Math.random() * (50 - 0));
climatCase = monde[x][y][0];
présence = monde[x][y][3];
nourriture = monde[x][y][1];
}
monde[x][y][3] = loop2;
}

//boucle principale:
(4)while(true){
//temps d'attente:
try {
Thread.sleep(10);
} catch (InterruptedException e) {
e.printStackTrace();
}

//addition des facteurs AL et EEM et cases des civilisations:
for(int loop = 0 ; loop <= nombreCivilisations ; loop++){
civilisations[loop][15] = 0; civilisations[loop][16] = 0; civilisations[loop][17] = 0;
}
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
civilisations[monde[ligne][colonne][3]][15] += monde[ligne][colonne][1];
civilisations[monde[ligne][colonne][3]][16] += monde[ligne][colonne][2];
civilisations[monde[ligne][colonne][3]][17]++;
}
}

//calculs de gestion des civilistions:
(5)for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
if(civilisations[loop][18] == 0){
//bonheur:
if(civilisations[loop][3] > 0 && civilisations[loop][17] > 0)
civilisations[loop][7] = (- civilisations[loop][9] + civilisations[loop][12] +
(civilisations[loop][3] - civilisations[loop][9]) / 5 + civilisations[loop][5] /
civilisations[loop][3]) / civilisations[loop][17];

//science:
int scienceProvisoire = civilisations[loop][10] / (10 - civilisations[loop][13]);
if(civilisations[loop][11] < scienceProvisoire)
civilisations[loop][11] = scienceProvisoire;

//nourriture:
civilisations[loop][12] = civilisations[loop][15] + civilisations[loop][4] / 2 +
civilisations[loop][11] / 3 - civilisations[loop][3] / 4;;
```

```
//population 1hab == 2AL
civilisations[loop][19] = civilisations[loop][3];
if((civilisations[loop][12] - (civilisations[loop][3] * 2)) >= 2){
civilisations[loop][3]++;
}
if((civilisations[loop][12] - (civilisations[loop][3] * 2)) < 0){
civilisations[loop][3]--;
}

//production:
civilisations[loop][4] = civilisations[loop][7] / 10 + (civilisations[loop][3] -
civilisations[loop][9]) + civilisations[loop][16] / 2;

//économie:
civilisations[loop][5] = civilisations[loop][16] + civilisations[loop][4] / 2 -
civilisations[loop][9] / 10;

//éducation:
civilisations[loop][10] = civilisations[loop][5] / 3 + (civilisations[loop][13] + 1);

//culture:
int cultureProvisoire = civilisations[loop][7] + civilisations[loop][10] / 5;
if(cultureProvisoire != 100){
if(temps % (100 - civilisations[loop][7] - civilisations[loop][10] / 5) == 0){
civilisations[loop][8]++;
}
if(civilisations[loop][8] < 0)
civilisations[loop][8] = 0;
if(civilisations[loop][8] > 9)
civilisations[loop][8] = 9;
}
//influence:
civilisations[loop][6] = civilisations[loop][5] / 3 + civilisations[loop][9] / 2;

//armée:
int politiqueProvisoire = civilisations[loop][13];
if(civilisations[loop][20] == 1)
politiqueProvisoire = politiqueProvisoire - 2;
if(politiqueProvisoire < 0)
politiqueProvisoire = 0;
civilisations[loop][9] = civilisations[loop][3] * (9 - politiqueProvisoire) / 10;

//politique (si changement)
if(civilisations[loop][7] < civilisations[loop][8] - 4){
civilisations[loop][13]++;
}
if(civilisations[loop][13] < 0)
civilisations[loop][13] = 0;
if(civilisations[loop][13] > 9)
civilisations[loop][13] = 9;



//décès d'une civilisation:
if(civilisations[loop][3] <= 0 || civilisations[loop][17] <= 0){
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][3] == loop){
monde[ligne][colonne][3] = 0;
}
}
}

civilisations[loop][18] = 1;
civilisations[loop][20] = 0;
}
}
}
```

```
//gestion des évènements:
(6)//expension:
for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
if(civilisations[loop][18] == 0){

if(civilisations[loop][3] == civilisations[loop][19] || temps % (10 - civilisations[loop]
[14]) == 0){
int totalZone = -1;
int x2 = -1; int y2 = -1;
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][3] == loop){
for(int loop2 = 0 ; loop2 <  4 ; loop2++){
int x = ligne;
int y = colonne;
if(loop2 == 0){
x--;
if(x == -1){
x = 29;
}
}
if(loop2 == 1){
x++;
if(x == 30){
x = 0;
}
}
if(loop2 == 2){
y--;
if(y == -1){
y = 49;
}
}
if(loop2 == 3){
y++;
if(y == 50){
y = 0;
}
}
if(monde[x][y][3] == 0 && monde[x][y][0] != 0){
int totalZone2 = monde[x][y][1] + monde[x][y][2];
if(totalZone2 > totalZone){
totalZone = totalZone2;
x2 = x;
y2 = y;
}
}
}
}
}
}
if(totalZone != -1){
monde[x2][y2][3] = loop;
}
}
}
}

//tableau des batailles:
//[x][y]  []--> 0-> il y a une bataille si ==1   1->coord x de l'ennemi   2->y   3->nb
soldats      4->nb soldats sc ou eem
int batailles[][][] =  new int[30][50][5];
```

```
(7)//resolutions de front, alliance:
for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
civilisations[loop][20] = 0;
civilisations[loop][21] = 0;
}
for(int première = 1 ; première <= nombreCivilisations ; première++){
for(int deuxième = première + 1 ; deuxième <= nombreCivilisations ; deuxième++){

if(civilisations[première][18] == 0 && civilisations[deuxième][18] == 0){

//définition des fronts:
if(relations[première - 1][deuxième - 2] == 0){

civilisations[première][20] = 1;
civilisations[deuxième][20] = 1;

//association des paires de cases pour faire une bataille
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][3] == première){
for(int loopFront = 0 ; loopFront < 4 ; loopFront++){
int x = ligne; int y = colonne;
if(loopFront == 0){
x = ligne - 1;
if(x == -1) x = 29;
}
if(loopFront == 1){
x = ligne + 1;
if(x == 30) x = 0;
}
if(loopFront == 2){
y = colonne + 1;
if(y == 50) y = 0;
}
if(loopFront == 3){
y = colonne - 1;
if(y == -1) y = 49;
}
if(monde[x][y][3] == deuxième && batailles[x][y][0] != 1 && batailles[ligne][colonne]
[0] != 1){
civilisations[première][21]++;civilisations[deuxième][21]++;
batailles[ligne][colonne][0] = 1; batailles[x][y][0] = 1;
batailles[ligne][colonne][1] = x; batailles[x][y][1] = ligne;
batailles[ligne][colonne][2] = y; batailles[x][y][2] = colonne;
loopFront = 42;
}
}
}
}
}
}

//aliance:
if(relations[première - 1][deuxième - 2] == 9){
//couleur:
int RMoy = ((civilisations[première][0] + civilisations[deuxième][0]) / 2);
int GMoy = ((civilisations[première][1] + civilisations[deuxième][1]) / 2);
int BMoy = ((civilisations[première][2] + civilisations[deuxième][2]) / 2);
                                                civilisations[première][0] = RMoy;
civilisations[première][1] = GMoy; civilisations[première][2] = BMoy;
//territoire:
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][3] == deuxième){
monde[ligne][colonne][3] = première;
}
}
}
```

```
//facteurs:
civilisations[première][3] = civilisations[première][3] + civilisations[deuxième][3];
civilisations[première][8] = ((civilisations[première][8] + civilisations[deuxième][8]) /
2);
civilisations[première][13] = ((civilisations[première][13] + civilisations[deuxième]
[13]) / 2);
civilisations[première][14] = ((civilisations[première][14] + civilisations[deuxième]
[14]) / 2);
//relations:
for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
boolean contactPremière = true; boolean contactDeuxième = true;
if(loop != première && loop != deuxième){
if(loop > première && loop < deuxième){
if(relations[première - 1][loop - 2] == - 1){relations[première - 1][loop - 2] = 5;
contactPremière = false;}
if(relations[loop - 1][deuxième - 2] == - 1){relations[loop - 1][deuxième - 2] = 5;
contactDeuxième = false;}
if(contactPremière || contactDeuxième){
relations[première - 1][loop - 2] = (relations[première - 1][loop - 2] + relations[loop -
1][deuxième - 2]) / 2;
}else{
relations[première - 1][loop - 2] = - 1; relations[loop - 1][deuxième - 2] = - 1;
}
}else{
if(loop > deuxième){
if(relations[première - 1][loop - 2] == - 1){relations[première - 1][loop - 2] = 5;
contactPremière = false;}
if(relations[deuxième - 1][loop - 2] == - 1){relations[deuxième - 1][loop - 2] = 5;
contactDeuxième = false;}
if(contactPremière || contactDeuxième){
relations[première - 1][loop - 2] = (relations[première - 1][loop - 2] +
relations[deuxième - 1][loop - 2]) / 2;
}else{
relations[première - 1][loop - 2] = - 1; relations[deuxième - 1][loop - 2] = - 1;
}
}else{
if(relations[loop - 1][première - 2] == - 1){relations[loop - 1][première - 2] = 5;
contactPremière = false;}
if(relations[loop - 1][deuxième - 2] == - 1){relations[loop - 1][deuxième - 2] = 5;
contactDeuxième = false;}
if(contactPremière || contactDeuxième){
relations[loop - 1][première - 2] = (relations[loop - 1][première - 2] + relations[loop -
1][deuxième - 2]) / 2;
}else{
relations[loop - 1][première - 2] = - 1; relations[loop - 1][deuxième - 2] = - 1;
}
}
}
}
}
//suppresion:
civilisations[deuxième][18] = 2;
}
}
}
}
```

```
(8)//placement soldats:
for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
if(civilisations[loop][20] == 1 && civilisations[loop][18] == 0 && civilisations[loop]
[21] != 0){
int resteSoldats = civilisations[loop][9] % civilisations[loop][21];
int resteSoldatsSc = civilisations[loop][11] % civilisations[loop][21];
int nbSoldats = (civilisations[loop][9] - resteSoldats) / civilisations[loop][21];
int nbSoldatsSc = (civilisations[loop][11] - resteSoldatsSc) / civilisations[loop][21];
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][3] == loop){
if(batailles[ligne][colonne][0] == 1){
batailles[ligne][colonne][3] = nbSoldats;
if(resteSoldats > 0){batailles[ligne][colonne][3]++; resteSoldats--; }
batailles[ligne][colonne][4] = nbSoldatsSc + monde[ligne][colonne][2];
if(resteSoldatsSc > 0){batailles[ligne][colonne][4]++; resteSoldatsSc--; }
}
}
}
}
}
}


//resolution des batailles:
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(batailles[ligne][colonne][0] == 1){
int nbSoldatsTotalCase = batailles[ligne][colonne][3] + batailles[ligne][colonne][4];
int nbSoldatsTotalEnnemi = batailles[batailles[ligne][colonne][1]][batailles[ligne]
[colonne][2]][3] + batailles[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]]
[4];
if(nbSoldatsTotalCase > nbSoldatsTotalEnnemi){
monde[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][3] = monde[ligne]
[colonne][3];
civilisations[monde[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][3]][3] -=
batailles[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][3];
int soldatsSc = batailles[ligne][colonne][4];
if(soldatsSc < nbSoldatsTotalEnnemi)
civilisations[monde[ligne][colonne][3]][3] -= nbSoldatsTotalEnnemi - soldatsSc;
}else{
if(nbSoldatsTotalCase < nbSoldatsTotalEnnemi){
monde[ligne][colonne][3] = monde[batailles[ligne][colonne][1]][batailles[ligne][colonne]
[2]][3];
civilisations[monde[ligne][colonne][3]][3] -= batailles[ligne][colonne][3];
int soldatsSc = batailles[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][4];
if(soldatsSc < nbSoldatsTotalCase)
civilisations[monde[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][3]][3] -=
nbSoldatsTotalCase - soldatsSc;
}else{
civilisations[monde[ligne][colonne][3]][3] -= batailles[ligne][colonne][3];
civilisations[monde[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][3]][3] -=
batailles[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][3];
}
}
batailles[ligne][colonne][0] = 0;
batailles[batailles[ligne][colonne][1]][batailles[ligne][colonne][2]][0] = 0;
}
}
}
```

```
(9)//gestion des relations:
for(int première = 1 ; première <= nombreCivilisations ; première++){
for(int deuxième = première + 1 ; deuxième <= nombreCivilisations ; deuxième++){

if(relations[première - 1][deuxième - 2] != -1 && civilisations[première][18] == 0 &&
civilisations[deuxième][18] == 0){

//initialisations des diférances
double difInfl = (double)(civilisations[première][6]) - (double)(civilisations[deuxième]
[6]);
if(difInfl < 0) difInfl = difInfl * - 1;
double difEco = (double)(civilisations[première][5]) - (double)(civilisations[deuxième]
[5]);
if(difEco < 0) difEco = difEco * - 1;
double difPol = (double)(civilisations[première][13]) - (double)(civilisations[deuxième]
[13]);
if(difPol < 0) difPol = difPol * - 1;

//relations entre 1 et 8
if(relationsProv[première - 1][deuxième - 2] >= 1 && relationsProv[première - 1][deuxième
- 2] <= 8){

//par politique
if(difPol != 0){
relationsProv[première - 1][deuxième - 2] -= difPol / 100;
}else{
relationsProv[première - 1][deuxième - 2] += 0.01;
}

//par extension
relationsProv[première - 1][deuxième - 2] -= (civilisations[première][14] +
civilisations[deuxième][14]) / 100;

//limites à 1 ou 8
if(relationsProv[première - 1][deuxième - 2] < 1)
relationsProv[première - 1][deuxième - 2] = 1;
if(relationsProv[première - 1][deuxième - 2] > 8)
relationsProv[première - 1][deuxième - 2] = 8;
}

int guerreProvisoire =     relations[première - 1][deuxième - 2];
//adaptation Provisoire -> tableau
relations[première - 1][deuxième - 2] = (int)(relationsProv[première - 1][deuxième - 2]);

//déclaration de guerre
if(relations[première - 1][deuxième - 2] <= 1){
//valeur de la proportion de population pour début et fin de guerre (a / 10)
double a = 4;
//pop[1] et [2] > a% pop max
if((civilisations[première][3] > (civilisations[première][12] * 1 / 2) * a / 10) &&
(civilisations[deuxième][3] > (civilisations[deuxième][12] * 1 / 2) * a / 10)){
if((difEco > 200) || (difPol > 3)){
relations[première - 1][deuxième - 2] = 0;
}
}else{
//dons des fronts
if(guerreProvisoire == 0){
int gagnante; int perdante;
if(civilisations[première][6] > civilisations[deuxième][6]){
gagnante = première; perdante = deuxième;
}else{
gagnante = deuxième; perdante = première;
}
if(civilisations[première][6] != civilisations[deuxième][6]){
int[][] marquage = new int[30][50];
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][3] == gagnante && marquage[ligne][colonne] == 0){
```

```
for(int loopFront = 0 ; loopFront < 4 ; loopFront++){
int x = ligne; int y = colonne;
if(loopFront == 0){
x = ligne - 1;
if(x == -1) x = 29;
}
if(loopFront == 1){
x = ligne + 1;
if(x == 30) x = 0;
}
if(loopFront == 2){
y = colonne + 1;
if(y == 50) y = 0;
}
if(loopFront == 3){
y = colonne - 1;
if(y == -1) y = 49;
}
if(monde[x][y][3] == perdante && marquage[x][y] == 0){
marquage[x][y] = 1; marquage[ligne][colonne] = 1;
monde[x][y][3] = gagnante;
loopFront = 42;
}
}
}
}
}
}
}
//arret de guerre
relations[première - 1][deuxième - 2] = 1;
}
}


//aliance
if(relations[première - 1][deuxième - 2] == 8){
if((difPol == 0) && (civilisations[première][6] < temps && civilisations[deuxième][6] <
temps)){
relations[première - 1][deuxième - 2] = 9;
}
}
}
}
}
```

```
(10)//génération et vérification des relations:
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][3] > 0){
for(int loop = 0 ; loop < 4 ; loop++){
int x = ligne; int y = colonne;
if(loop == 0){
x--;
if(x == -1){
x = 29;
}
}
if(loop == 1){
x++;
if(x == 30){
x = 0;
}
}
if(loop == 2){
y--;
if(y == -1){
y = 49;
}
}
if(loop == 3){
y++;
if(y == 50){
y = 0;
}
}
if(monde[ligne][colonne][3] != monde[x][y][3] && monde[x][y][3] > 0){
int première = monde[ligne][colonne][3]; int deuxième = monde[x][y][3];
if(première > deuxième){
int provisoire = première;
première = deuxième;
deuxième = provisoire;
}
if(relations[première - 1][deuxième - 2] == - 1){
relations[première - 1][deuxième - 2] = 5;
relationsProv[première - 1][deuxième - 2] = 5;
}
}
}
}
}
}

repaint();
temps = temps + 1L;
}
}
```

```
(11)//Actualisation de l'image:
public class Panneau extends JPanel{
public void paintComponent(Graphics g){
g.setColor(new Color(100, 100, 100));
//carte
for(int ligne = 0 ; ligne <= 29 ; ligne++){
for(int colonne = 0 ; colonne <= 49 ; colonne++){
if(monde[ligne][colonne][0] == 0){
g.setColor(Color.black);
}else{
g.setColor(new Color(civilisations[monde[ligne][colonne][3]][0],
civilisations[monde[ligne][colonne][3]][1], civilisations[monde[ligne][colonne][3]][2]));
}
g.fillRect(colonne * 10, ligne * 10, 10, 10);
g.setColor(Color.black);
g.drawRect(colonne * 10, ligne * 10, 10, 10);
}}
//tableau de statistiques:
Font font = new Font("Arial", Font.BOLD, 32);
g.setFont(font); g.setColor(Color.red);
g.drawString("Statistiques", 800, 30);
g.drawString("Relations", 40, 440);
g.setColor(Color.black);
Font font2 = new Font("Arial", Font.BOLD, 10);
g.setFont(font2);
g.drawString("Date :", 520, 10);
g.drawString(String.valueOf(temps), 550, 10);
g.drawString("ID", 520, 45);
g.drawString("Pop", 570, 45);
g.drawString("Prod", 620, 45);
g.drawString("Eco", 670, 45);
g.drawString("Inf", 720, 45);
g.drawString("Bonh", 770, 45);
g.drawString("Cult", 820, 45);
g.drawString("Arm", 870, 45);
g.drawString("Edu", 920, 45);
g.drawString("Sci", 970, 45);
g.drawString("Nour", 1020, 45);
g.drawString("Pol", 1070, 45);
g.drawString("Exten", 1115, 45);
g.drawString("AL", 1170, 45);
g.drawString("EEM", 1220, 45);
g.drawString("Cases", 1265, 45);
for(int ligne = 0 ; ligne < nombreCivilisations ; ligne++){
for(int colonne = 0 ; colonne <= 15 ; colonne++){
g.drawRect(colonne * 50 + 502, ligne * 20 + 50, 50, 20);
}
g.drawString(String.valueOf(ligne + 1), 505, ligne * 20 + 68);
}
for(int ligne = 0 ; ligne < nombreCivilisations ; ligne++){
for(int colonne = 0 ; colonne <= 14 ; colonne++){
g.drawString(String.valueOf(civilisations[ligne + 1][colonne + 3]), colonne * 50 + 555,
ligne * 20 + 68);
}
}
for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
g.setColor(new Color(civilisations[loop][0], civilisations[loop][1], civilisations[loop]
[2]));
g.fillRect(525, loop * 20 + 38, 10, 10);
if(civilisations[loop][18] == 1){
g.setColor(Color.red);
g.drawLine(502, loop * 20 + 42, 1300, loop * 20 + 42);
}
if(civilisations[loop][18] == 2){
g.setColor(Color.blue);
g.drawLine(502, loop * 20 + 42, 1300, loop * 20 + 42);
}
}
```

```
//tableau des relations:
g.setColor(Color.black);
for(int loop = 1 ; loop <= nombreCivilisations ; loop++){
if(loop != nombreCivilisations)
g.drawString(String.valueOf(loop), 10, 450 + 12 * loop);
if(loop != 1)
g.drawString(String.valueOf(loop), 10 + 12 * loop, 450);
}
for(int ligne = 1 ; ligne <= nombreCivilisations ; ligne++){
for(int colonne = 2 ; colonne <= nombreCivilisations ; colonne++){
if(relations[ligne - 1][colonne - 2] != 42){
g.drawRect(7 + colonne * 12, 439 + ligne * 12, 12, 12);
if(civilisations[ligne][18] == 0 && civilisations[colonne][18] == 0){
g.drawString(String.valueOf(relations[ligne - 1][colonne - 2]), 8 + colonne * 12, 450 +
ligne * 12);
}else{
g.drawString("x", 8 + colonne * 12, 450 + ligne * 12);
}
}
}
}
}
}
```

```
(0)//DEPART      main:
public static void main(String[] args) {
//appel de "JeuDesCivilisations" la classe principale:
new JeuDesCivilisations();
}
}
```

Et voilà !
Bon… la version PC est plus jolie et moins prise de tête.

Se procurer le programme et notes de versions :

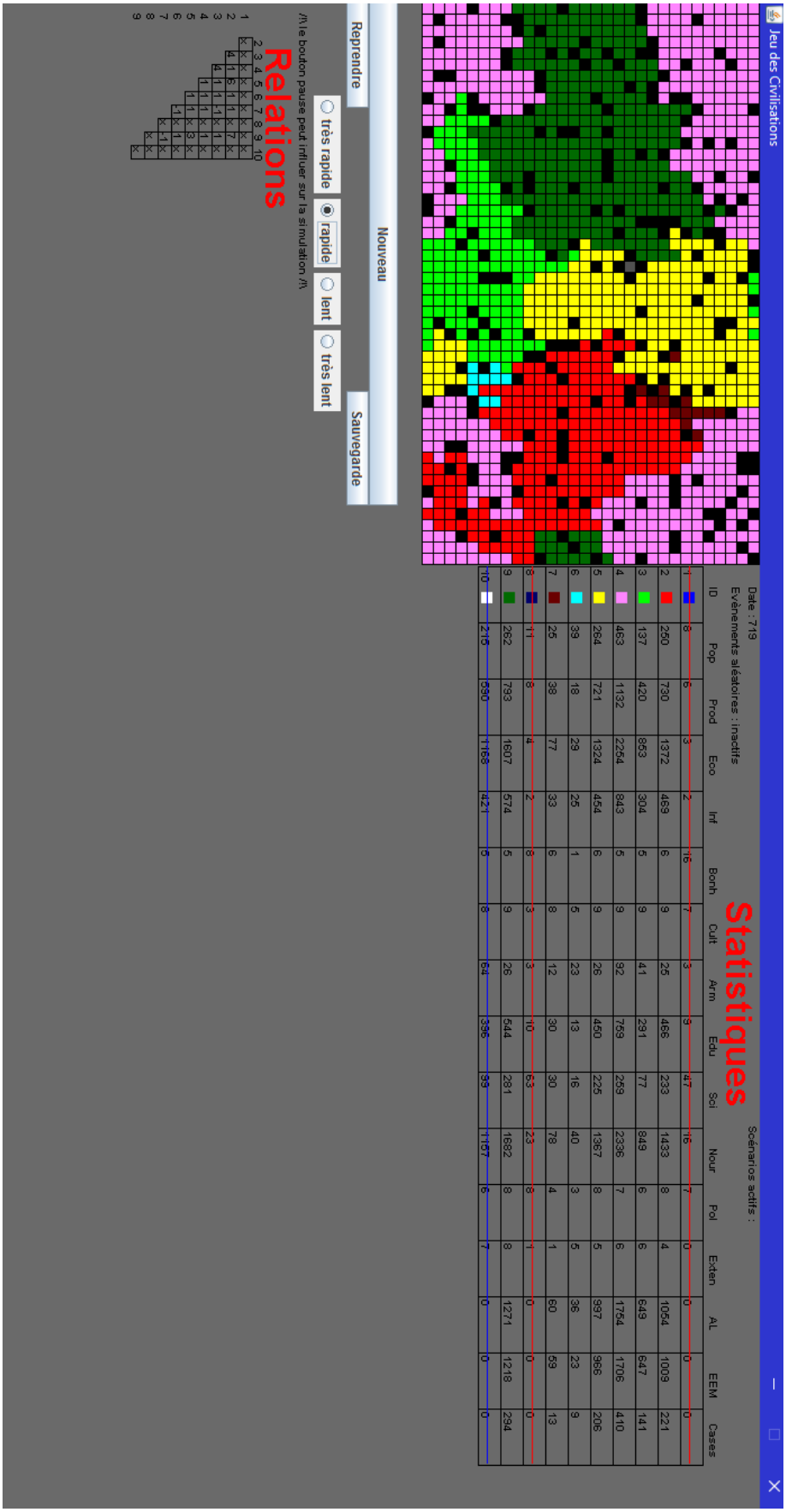
Vous pouvez maintenant télécharger le projet Eclipse avec ce lien :
http://www.mediafire.com/download/vxpzl0250ekfvq8/Jeu+Des+Civilisations+Version+Simple.zip
Décompressez le fichier (clic droit>extraire tout, ou utilisez winrar ou autre) Si vous n'avez pas Eclipse vous pouvez consulter le code en allant dans ce dossier puis dans « src » et ouvrez JeuDesCivilisations.java avec le bloc note windows.

Vous pouvez constater que ceci est la version simple, en effet nous avons eu le temps de faire une version finale beaucoup plus complète, elle ajoute :
- Un bouton pause
- D'autres pour régler la vitesse d'exécution
- Un autre pour sauvegarder la carte
- Un dernier pour créer une nouvelle partie
- Le programme fait une sauvegarde automatique à chaque nouvelle partie
- On peut paramétrer la partie et choisir :
    - le nombre de civilisations
    - si l'on active des événements aléatoires ou pas
    - on peut choisir des scénarios
- et on peut charger une carte sauvegardée ou que l'on a créé dans un bloc note !

Vous pouvez télécharger le projet Eclipse de cette version :
http://www.mediafire.com/download/s3a94ohf2i8hwoc/Jeu+Des+Civilisations.zip
=> Ou un .exe pour l'exécuter en double cliquant dessus (IMPORTANT : Si ça ne fonctionne pas, mettez à jour java via https://www.java.com/fr/ . Autre point : si Windows vous recommande de ne pas ouvrir l'application, faites le quand-même) :
http://www.mediafire.com/download/404lyq0py2qr9y0/Le+Jeu+Des+Civilisations.exe

Il est donc possible de créer (avec de la patience) des cartes toutes faites ainsi voici un dossier avec quelques cartes que nous avons créé et notamment celle du monde actuel :
http://www.mediafire.com/download/y145zv4bn1hv3pn/CARTES.zip

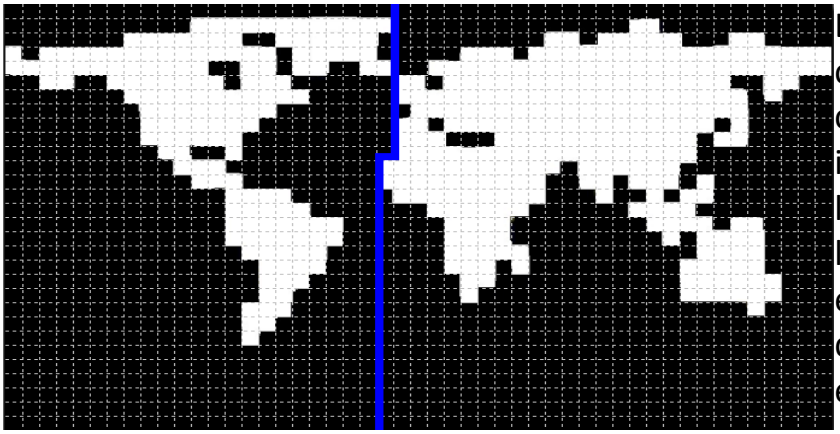
Voici un aperçu du programme :



Application au monde actuel :

Comme nous pouvons maintenant charger des cartes, nous avons décidé de créer notre propre carte, celle du monde. En effet les sauvegardes se font sous la forme d'un bloc-note (Windows) contenant toutes les valeurs de toutes les variables, le programme peut ainsi les lires, il suffit de les écrire dans le bon ordre. On obtiens une suite de nombres relativement peu claire, vous pouvez en avoir un aperçu en ouvrant une sauvegarde du programme avec le bloc note de Windows.

- Pour créer le monde il nous a fallu tout d'abord dessiner la carte :



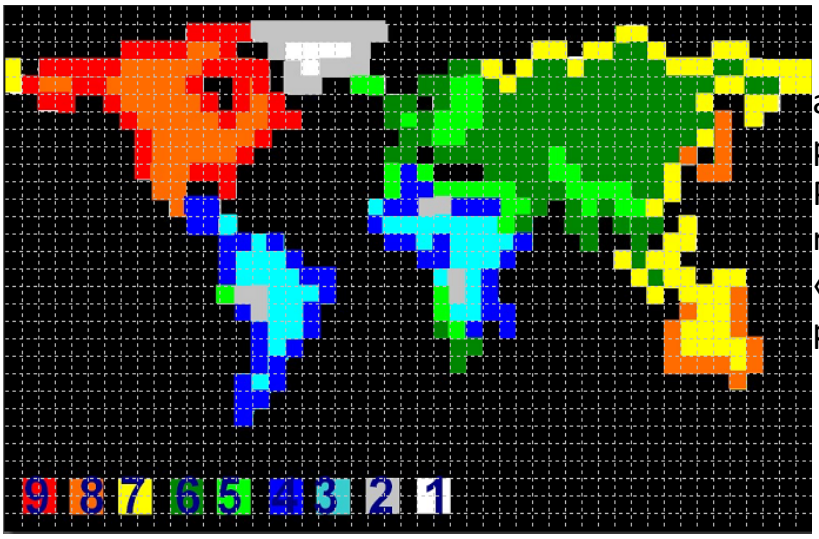
Les cases blanches auront un climat de 1 et les noires de 0, ces dernières seront donc inutilisable et affichée en noir lors de l'exécution,NB : la ligne bleue indique qu'il faut écarter les deux continents d'une case de plus (une petite erreur:-)).

Notez que nous n'avons pas mis l'Antarctique et n'avons pas collé l'Arctique au bord supérieur pour éviter qu'il y est liaison entre le nord et le sud, rappelons que nous sommes dans un tore, pas une sphère.

- Ensuite nous avons défini les ressources alimentaires :

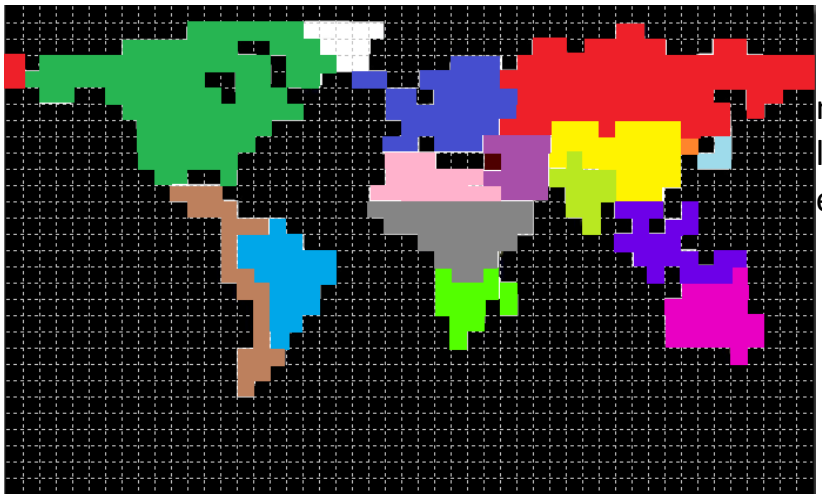
Nous avons fait des recherches sur la biodiversité et sur la « cultivabilité » du monde, sur les 9 points à attribuer, la biodiversité est sur 3 points, la possibilité de cultiver les terres sur 4 points et 2 points en plus si la case est sur une côte .

- Reste pour créer la carte de placer les ressources économiques et matérielles :



Pour ce faire nous avons attribué (sur 9 points) 1 point pour les côtes, 4 point pour le PIB des pays et 4 autres sur les ressources « précieuses »(charbon, or, pétrole, et uranium)

- Enfin nous avons placé des civilisations, en gardant celles actuelles mais en les regroupant pour ne pas avoir les 200 pays du monde.



Pour clore le bloc-note, il ne nous reste plus qu'à définir les relations entre civilisations et leur facteurs de base.

Une fois le bloc-note terminé, il suffit de le charger dans le programme (puisqu’il peut le faire) et l'on pourra enfin savoir l'avenir du monde.

Conclusion et avenir du programme :

Nous avons donc créé un simulateur capable de faire évoluer des civilisations et de calculer leurs relations, leur comportement,...

Le programme permet à lui tout seul de répondre à la problématique, en effet nous venons de prouver qu'il est possible d'appliquer le principe du jeu de la vie à des civilisations, nous avons ainsi un algorithme qui permet en plus de calculer tout ça peut aussi l'afficher en temps réel et donc afficher chaque étape de la simulation.

En revanche nous n'avons pas les connaissances historiques, géographiques, politiques et autre, nécessaires à faire une simulation fiable qui nous permettrait de savoir exactement comment sera le monde de demain.

Notre programme est un modèle de simulation fonctionnel mais pas réaliste et c'est seulement en trouvant un modèle qui collerait parfaitement au fonctionnement de l'univers que nous pourrions avoir LE simulateur le plus réaliste possible. Tout les scientifiques pourrait apporter leur contribution, aussi bien les historiens que les physiciens (s'ils résolvaient la théorie du tout notamment).

C'est pour cette raison que nous avons mis ce programme en téléchargement, car le monde pourra en profiter et reprendre le principe mais rajouter toujours plus de facteurs et les rendre de plus en plus réalistes pour avoir à l'avenir un Jeu Des Civilisations extrêmement réaliste.

Le programme à donc peut-être un avenir dans les plus gros ordinateurs mondiaux qui pourrait appliquer un modèle extrêmement fiable mais, sans aucun doute, pas encore à notre portée ni celle de l'humanité.

Le Jeu Des Civilisations est donc un simulateur dont le principe est résolu mais qui est pourtant incomplet, le monde ne sera réellement prévisible que si l'algorithme qui régit le programme colle au mieux à celui qui régit le monde et c'est seulement en y travaillant toujours plus qu'on finira par atteindre ce modèle mathématique et que l'avenir du monde sera aussi prévisible que la météo de demain...

Liens, sources et remerciements :

Avant tout revoici un site permettant de tester le Jeu de la Vie :
http://jm.davalan.org/divers/jeuvie/index.html

Remerciements aux sites où nous avons appris le Java :
- France ioi :
http://www.france-ioi.org/
- OpenClassroom :
https://openclassrooms.com/courses/apprenez-a-programmer-en-java

Remerciements à Oracle :
http://www.oracle.com/fr/java/overview/index.html
et Eclipse :
https://eclipse.org/

Divers sources de recherches pour la création de la carte du monde actuel :
wikipedia
protégeonslaterre.com
bretagnemontagne.wordpress.com
raymond.rodriguez1.free.fr