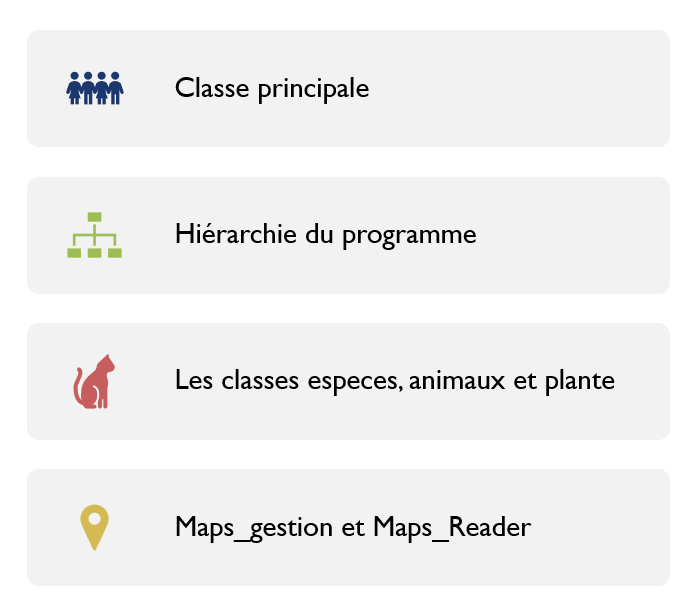
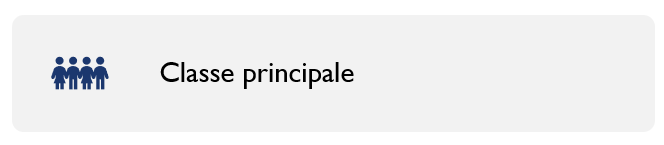
PROJET : Simulateur d’écosystème

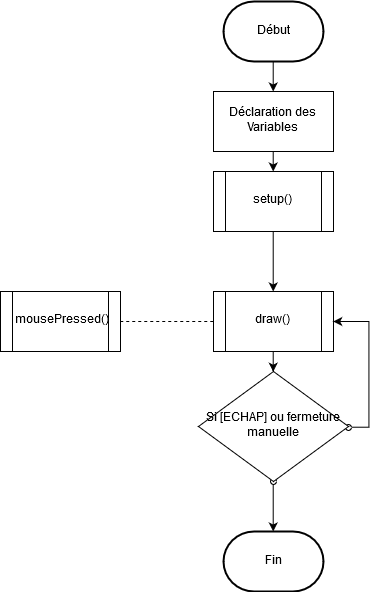
Par LEGRAND Alexia

Sommaire





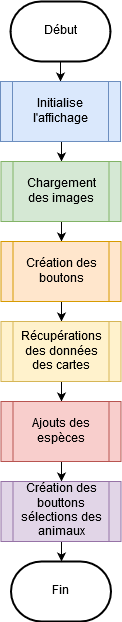
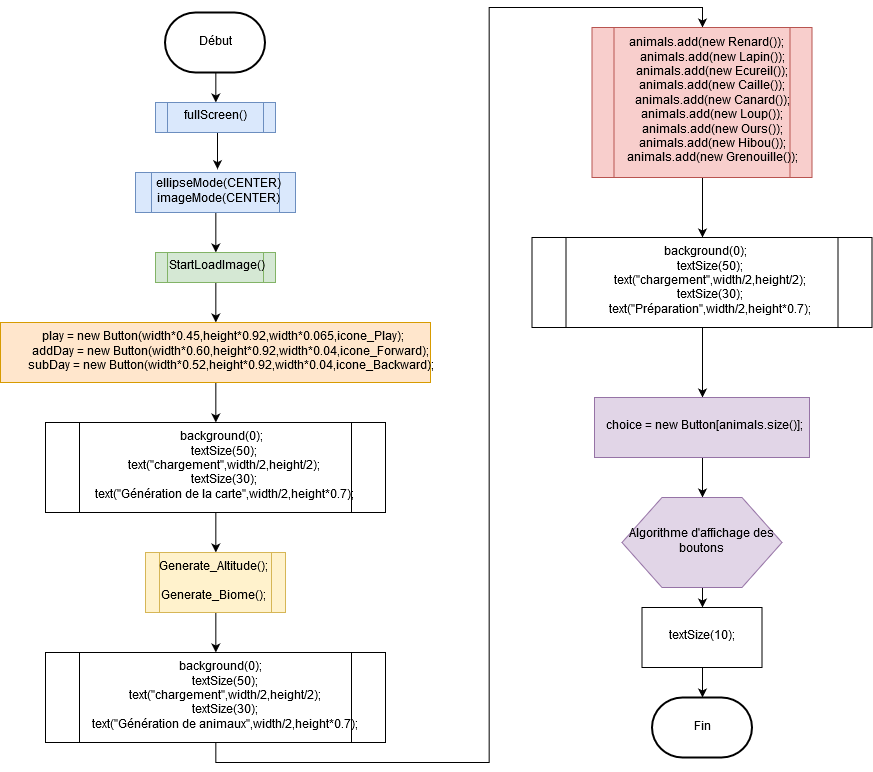
Contenu de la classe principale



Variables de la fonction pricipale

|  |  |  |
| --- | --- | --- |
| Type | Nom | Contenu |
| ArrayList<Animal> | animals | New ArrayList<Animal>() |
| Int | actualChoice | -1 |
| Int | day | 1 |
| Boolean | dayPass | false |
| Int | Xmouse | -1 |
| Int | Ymouse | -1 |
| Button | play | N/A |
| Button[] | choice | N/A |
| Button | addDay | N/A |
| Button | subDay | N/A |

Setup



Graphique du déroulement général de l’initialisation

Graphique du déroulement complet de l’initialisation

Code du Setup

void setup(){

fullScreen();

ellipseMode(CENTER);

imageMode(CENTER);

StartLoadImage();

//init boutton

play = new Button(width\*0.45,height\*0.92,width\*0.065,icone\_Play);

addDay = new Button(width\*0.60,height\*0.92,width\*0.04,icone\_Forward);

subDay = new Button(width\*0.52,height\*0.92,width\*0.04,icone\_Backward);

background(0);

textSize(50);

text("chargement",width/2,height/2);

textSize(30);

text("Génération de la carte",width/2,height\*0.7) ;

Generate\_Altitude();

Generate\_Biome();

background(0);

textSize(50);

text("chargement",width/2,height/2);

textSize(30);

text("Génération de animaux",width/2,height\*0.7);

animals.add(new Renard());

animals.add(new Lapin());

animals.add(new Ecureil());

animals.add(new Caille());

animals.add(new Canard());

animals.add(new Loup());

animals.add(new Ours());

animals.add(new Hibou());

animals.add(new Grenouille());

background(0);

textSize(50);

text("chargement",width/2,height/2);

textSize(30);

text("Préparation",width/2,height\*0.7);

choice = new Button[animals.size()];

final float XborneInf = 10;

final float XborneSup = width\*0.4;

final float YborneInf = 10;

final float YborneSup = height\*0.6;

int carre = int(sqrt(choice.length));

if(choice.length-carre\*carre > 0) carre++;

float deplacement;

if(YborneSup-YborneInf<XborneSup-XborneInf)

deplacement = (YborneSup-YborneInf)/carre;

else deplacement = (XborneSup-XborneInf)/carre;

for(int i=0;i<choice.length;i++)

choice[i] = new Button(XborneInf+deplacement/2+deplacement\*(i-(carre\*(i/carre)))+10,YborneInf+deplacement/2+deplacement\*int(i/carre)+10,deplacement-20,animals.get(i).GetIcone());

textSize(10);

}

background(0);

textSize(50);

text("chargement",width/2,height/2);

textSize(30);

text("Préparation",width/2,height\*0.7);

choice = new Button[animals.size()];

final float XborneInf = 10;

final float XborneSup = width\*0.4;

final float YborneInf = 10;

final float YborneSup = height\*0.6;

int carre = int(sqrt(choice.length));

if(choice.length-carre\*carre > 0) carre++;

float deplacement;

if(YborneSup-YborneInf<XborneSup-XborneInf)

deplacement = (YborneSup-YborneInf)/carre;

else deplacement = (XborneSup-XborneInf)/carre;

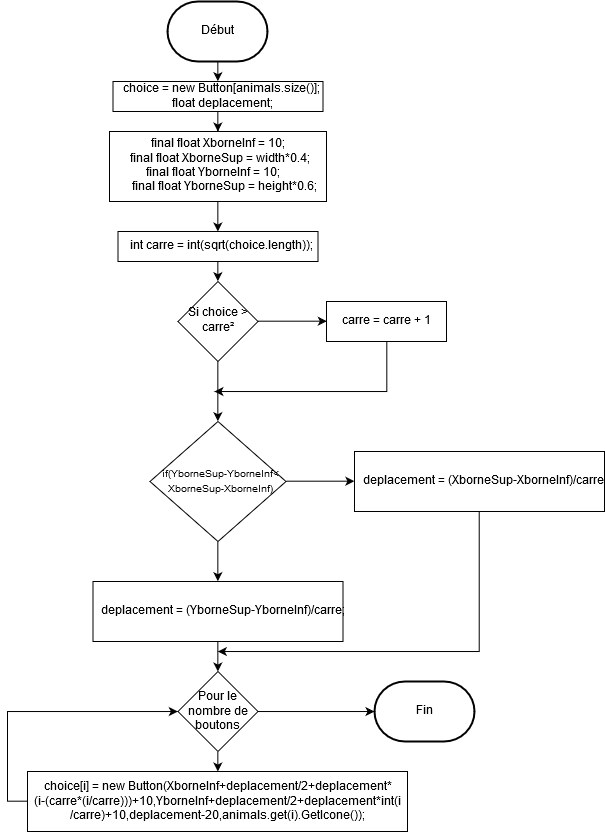
for(int i=0;i<choice.length;i++)

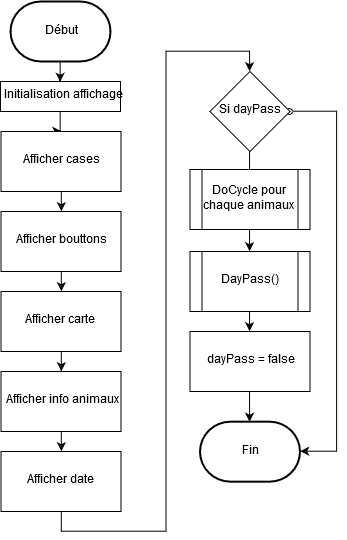
choice[i] = new Button(XborneInf+deplacement/2+deplacement\*(i-(carre\*(i/carre)))+10,YborneInf+deplacement/2+deplacement\*int(i/carre)+10,deplacement-20,animals.get(i).GetIcone());

textSize(10);

}

Algorithme d’affichage des bouttons



draw



Carre

Deplacement

Code du Draw

void draw() {

fill(0,0,0);

stroke(0,0,0);

rect(width\*0.41,10,width\*0.58,height\*0.98);

rect(10,10,width\*0.4,height\*0.6);

fill(255,255,255);

stroke(255,255,255);

//--------------------------------DESSINER CASES----------------------------------

noFill();

stroke(255);

//selection

rect(10,10,width\*0.4,height\*0.6,10);

//carte

rect(10,height\*0.62,width\*0.4,height\*0.37,10);

//info

rect(width\*0.41,10,width\*0.58,height\*0.98,10);

//--------------------------------DESSINER BOUTTONS----------------------------------

play.Display();

for(int i =0;i<choice.length;i++) choice[i].Display();

addDay.Display();

subDay.Display();

//-----------------------------DESSINER CARTE------------------------------------

for(int i =0; i < altitude.length;i++)

for(int j = 0; j< altitude[0].length;j++)

if(altitude[i][j] > 0) point(10+i,height\*0.62 + j);

//------------------------DESSINER ANIMAUX---------------------------------------

if(actualChoice != -1){

Animal show = animals.get(actualChoice);

textSize(30);

textAlign(CENTER);

text("Espèce : " + show.GetSpecies(), width\*0.41+(width\*0.29), height\*0.05);

//Infos générales

textAlign(LEFT);

textSize(20);

text("Nombre de population : " + NombreLisible(show.GetCount()), width\*0.41 + 10, height\*0.1);

text("Moyenne morts par jour : " + ((totalJours>0)?NombreLisible(show.GetTotalMorts()/totalJours):"0"), width\*0.41 + 10, height\*0.15);

text("Moyenne naissances par jour : " + ((totalJours>0)?NombreLisible(show.GetTotalNaissances()/totalJours):"0"), width\*0.41 + 10, height\*0.20);

text("Age moyen : " + show.GetAgeMoyen(), width\*0.41 + 10, height\*0.25);

text("Cause principale de mortalitée : " + show.GetCauseMort(), width\*0.41 + 10, height\*0.30);

//Infos de cycle

textAlign(RIGHT);

text("Infos du cycle",width\*0.41+(width\*0.57), height\*0.75);

textSize(15);

text("Nombre de morts : " + NombreLisible(show.GetMort()), width\*0.41+(width\*0.57), height\*0.80);

text("Nombre de naissances : " + NombreLisible(show.GetNaissance()), width\*0.41+(width\*0.57), height\*0.82);

//Infos sur carte

textAlign(LEFT);

textSize(20);

text("Infos Carte", width\*0.42, height\*0.75);

textSize(15);

text("Nombre de population : " + ((Xmouse>=0)?NombreLisible(show.posMap[Xmouse][Ymouse]):""), width\*0.42, height\*0.80);

text("Biome : " + ((Xmouse>=0)?GetBiomeName(biome[Xmouse][Ymouse]):""), width\*0.42, height\*0.82);

show.Display();

}

textSize(30);

textAlign(CENTER);

text(jour+"/" + mois+"/" + annee,width\*0.68,height\*0.925);

text(day,width\*0.56,height\*0.925);

if(dayPass){

for(int i =0; i<animals.size(); i++) animals.get(i).Zero();

for(int j=0;j<day;j++){

//println("Jour : " +j);

for(int i =0; i<animals.size(); i++){

Animal animal = animals.get(i);

//println("Do cycle : " + animal.GetSpecies());

animal.DoCycle();

}

DayPass();

}

dayPass = false;

}

}

if(dayPass){

for(int i =0; i<animals.size(); i++) animals.get(i).Zero();

for(int j=0;j<day;j++){

//println("Jour : " +j);

for(int i =0; i<animals.size(); i++){

Animal animal = animals.get(i);

//println("Do cycle : " + animal.GetSpecies());

animal.DoCycle();

}

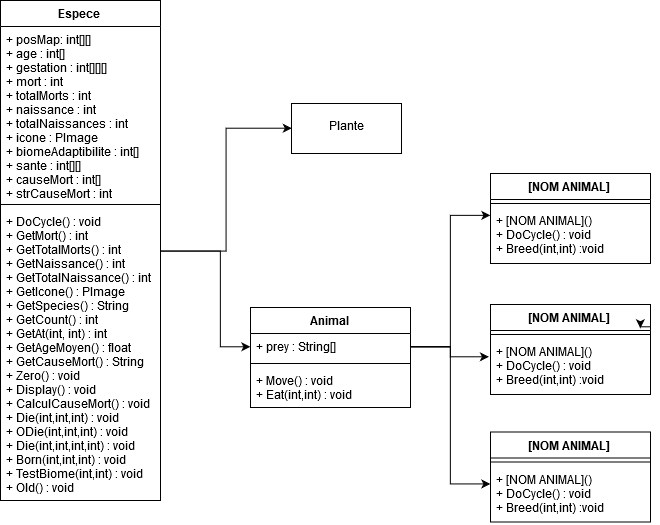
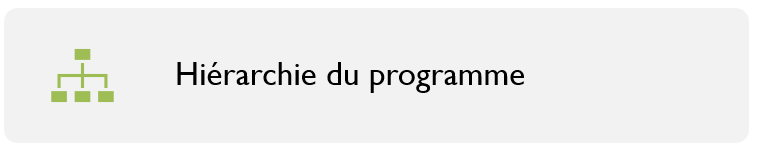
DayPass();

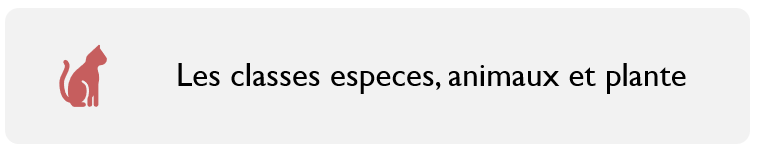
}

dayPass = false;

}

}





Classe espèce : OLD()

Une image contenant capture d’écran

Description générée automatiquement

Classe animal : move()

void Old(){ //Calcule l'age et tue les vieux membres de l'espèce

ArrayList<int[]> presence = new ArrayList<int[]>();

for(int i = 0; i<posMap.length;i++)

for(int y = 0; y <posMap[0].length;y++){

if(posMap[i][y] > 0) presence.add(new int[]{i,y});

}

if(jour == 31 && mois == 12)

for(int i =age[DUREE\_VIE-1];i>0;i--) age[i] = age[i-1];

for(int i=0;i<age[DUREE\_VIE-1];i++)

{

int ran = int(random(0,presence.size()));

while(posMap [presence.get(ran)[0]][presence.get(ran)[1]]== 0) ran = int(random(0,presence.size()));

ODie(presence.get(ran)[0],presence.get(ran)[1],DUREE\_VIE-1);

//println("Death at " + presence.get(ran)[0] + ", " + presence.get(ran)[1]);

}

for(int i=VIEILLESSE\_MIN;i<DUREE\_VIE-2;i++)

{

float rand = random(0,0.05\*i);

for(int j=0;j<age[i]\*rand/1000;j++){

int ran = int(random(0,presence.size()));

while(posMap [presence.get(ran)[0]][presence.get(ran)[1]]== 0) ran = int(random(0,presence.size()));

ODie(presence.get(ran)[0],presence.get(ran)[1],i);

//println("Death at " + presence.get(ran)[0] + ", " + presence.get(ran)[1]);

}

}

}

Une image contenant texte

Description générée automatiquement

Classe animal : eat()

void Move(){ //génére une migration aléatoire chaques mois

if(jour == 1){

for(int x = 0; x<posMap.length;x++)

for(int y = 0; y<posMap[0].length;y++){

if(posMap[x][y]>0){

for(int i =0;i<4;i++){

float ran = random(0,VITESSE\_PROPAGATION);

ran = ran/100\*posMap[x][y];

posMap[x][y]-= ran;

if(i == 1 && x>0 && altitude[x-1][y]>0) posMap[x-1][y]+= ran;

else if(i == 2 && x<posMap.length-1 && altitude[x+1][y]>0) posMap[x+1][y]+= ran;

else if(i == 3 && y>0 && altitude[x][y-1]>0) posMap[x][y-1]+= ran;

else if(i == 4 && y<posMap[0].length-1 && altitude[x][y+1]>0) posMap[x][y+1]+= ran;

else posMap[x][y]+= ran;

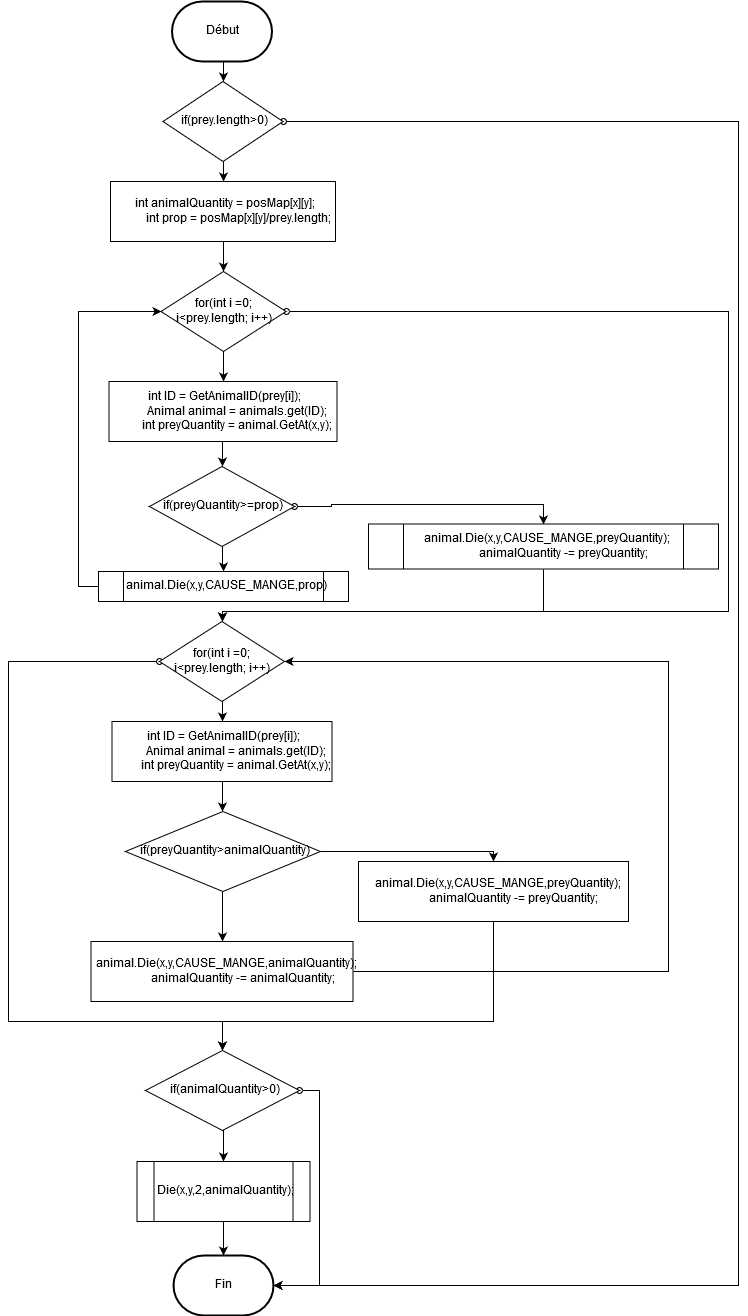
}

}

}

}

}



void Eat(int x, int y){ //Algorithme pour se nourrir (ne pas utiliser pour herbivores)

if(prey.length>0){

int animalQuantity = posMap[x][y];

int prop = posMap[x][y]/prey.length;

for(int i =0; i<prey.length; i++){

int ID = GetAnimalID(prey[i]);

Animal animal = animals.get(ID);

int preyQuantity = animal.GetAt(x,y);

if(preyQuantity>=prop){

animal.Die(x,y,CAUSE\_MANGE,prop);

}else{

animal.Die(x,y,CAUSE\_MANGE,preyQuantity);

animalQuantity -= preyQuantity;

}

}

for(int i =0; i<prey.length; i++){

int ID = GetAnimalID(prey[i]);

Animal animal = animals.get(ID);

int preyQuantity = animal.GetAt(x,y);

if(preyQuantity>animalQuantity){

animal.Die(x,y,CAUSE\_MANGE,animalQuantity);

animalQuantity -= animalQuantity;

}else{

animal.Die(x,y,CAUSE\_MANGE,preyQuantity);

animalQuantity -= preyQuantity;

}

}

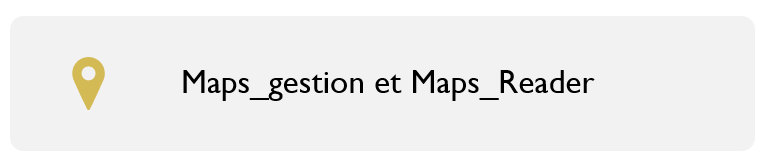
if(animalQuantity>0){

Die(x,y,2,animalQuantity);

}

}

}



Maps\_reader : TestColor()Une image contenant texte, carte de visite

Description générée automatiquement

boolean[][] TestColor(PImage image, color c){ //Retourne une carte renvoyant vrai en tout point où la couleur (c) est présente sur l'image (image)

boolean[][] map = new boolean[image.width][image.height];

image.loadPixels();

for(int i = 0; i< image.width\*image.height;i++)

{

int y\_map = i/image.width;

int x\_map = i-(y\_map\*image.width);

if(image.pixels[i] == c)

map[x\_map][y\_map] = true;

else map[x\_map][y\_map] = false;

}

return map;

}

Maps\_reader : map\_read()

Une image contenant texte

Description générée automatiquement

Maps\_reader : ResizeMap()

int[][] Map\_Read(String Imap, color[] tColor,int[] returnValue) { //Renvoie une carte de valeurs de la carte (Imap), conversant tColor --> returnValue

PImage map = GetMap(Imap);

int[][] finalMap = new int[map.width][map.height];

for(int i = 0; i<tColor.length;i++)

{

boolean[][] test = TestColor(map,tColor[i]);

for(int j = 0; j < test.length;j++)

for(int k = 0; k < test[0].length; k++)

{

if(test[j][k]) finalMap[j][k] = returnValue[i];

}

}

return finalMap;

}

Une image contenant capture d’écran

Description générée automatiquement

boolean[][] ResizeMap(boolean[][] original){ //Redimmensionne une carte (original)

boolean[][] nouvelle = new boolean[(int) (width\*0.4)][(int)(height\*0.37)];

for(int i = 0; i < nouvelle.length ; i++)

for(int j = 0; j < nouvelle[0].length ; j++)

nouvelle[i][j] = original[int(i\*(original.length/(float)nouvelle.length))][int(j\*(original[0].length/(float)nouvelle[0].length))];

return nouvelle;

}

Maps\_gestion : biome et altitude

Une image contenant capture d’écran

Description générée automatiquement

GetBiomeName(ID)

void Generate\_Altitude(){ //Génére la carte altitude

altitude = Map\_Read("ressources/carte profondeur.png",new color[]{color(0,0,79),color(0,25,168),color(0,64,207),color(0,112,255), color(71,184,255),color(135,224,255),color(184,240,255),color(255,255,255),color(0,153,51),color(255,204,0),color(255,125,0),color(153,102,51),color(135,97,87),color(184,169,169),color(224,209,209),color(240,232,232)},new int[]{-8000,-6000,-5000,-4000,-3000,-1000,-200,-100,100,200,500,1000,2000,3000,4000,5000});

int[][] newAltitude = new int [(int) (width\*0.4)][(int)(height\*0.37)];

for(int i = 0; i < newAltitude.length ; i++)

for(int j = 0; j < newAltitude[0].length ; j++)

newAltitude[i][j] = altitude[int(i\*(altitude.length/(float)newAltitude.length))][int(j\*(altitude[0].length/(float)newAltitude[0].length))];

altitude = newAltitude;

}

