

*TM n°15*

*Fabrication d'un logiciel  
didactique*

Logiciel capable de simuler le déplacement  
d'astres soumis à la gravitation

Adrien Briod et Mathieu Ackermann, 3M5

Sujet proposé par M. Lukas Schellenberg

## But

Nous nous sommes fixé comme objectif de réaliser une applet en Java<sup>1</sup> qui puisse illustrer les forces gravitationnelles exercées sur des masses en mouvement. Nous voulions ainsi permettre à l'utilisateur de notre programme de manipuler et de modifier la masse, la position et la vitesse initiale de plusieurs objets (planètes) et de voir directement l'effet de ses actes sur leur mouvement les uns par rapport aux autres dans un espace à deux dimensions (plan).

<sup>1</sup>: Java est un langage de programmation créé en 1991 par Sun Microsystems. Ayant été conçu à l'origine pour être intégré à des appareils électroménagers, il est rapidement devenu (en 1994) un langage de programmation pour internet, justement parce qu'il était compatible avec tous les modèles d'ordinateurs.

## Moyens

Nous ne connaissions absolument pas le langage Java avant de commencer et nous nous sommes basés sur deux livres principaux :

- « Maîtrisez Java 2 » d'Ivor Horton, aux éditions Wrox Press France, 2001
- « Java 2 », collection « Grand livre », par Mäurer, Baufeld, Müller, Wabnitz et Mühle, aux éditions Micro Application, 1999

Nous avons également consulté de nombreux sites web, dont notamment :

- Comment ça marche (<http://www.commentcamarche.net>) : Site très complet sur l'informatique en général, avec une section Java intéressante.
- Développez.com pour Java (<http://faqjava.developpez.com>) : Une foire aux questions parfois utile.
- Eteks (<http://www.eteks.com>) : Site très complet, proposant notamment un excellent tutorial ("Du C/C++ à Java").
- Self-Access pour Java (<http://www.self-access.com/java>) : Site intéressant pour débiter, mais les thèmes sont assez limités. Par contre les explications sont très détaillées.
- Touraivane (<http://www.esil.univ-mrs.fr/~tourai/Java/>) : Site très complet, avec une grande quantité de thèmes. Nous avons notamment beaucoup utilisé la partie n°25 intitulée "Gestion des évènements".
- <http://www.developpez.org/club/bkostrzewa/index.html> : Site complet, abordant un grand nombre de thèmes intéressants.

A noter que nous avons aussi de bonnes connaissances en HTML, ayant déjà un peu d'expérience dans la création de sites web. En ce qui concerne Java, en revanche, nous sommes partis de zéro et par de petits essais successifs, avons réussi à développer une applet de plus en plus complexe et complète.

Pour ce qui est des moyens utilisés pour les calculs, nous avons utilisé la méthode du pas à pas. C'est à dire qu'à chaque instant donné, le programme calcule la nouvelle trajectoire que doit prendre une masse. En effet, en fonction de sa position par rapport aux autres, une masse subit

une force d'attraction plus ou moins grande, ce qui change directement son accélération (par la deuxième loi de Newton), puis influe sur sa vitesse et finalement sur sa position.

## Principales difficultés

S'il y avait *une* difficulté à citer ce serait celle de la compatibilité : non seulement entre PC et Mac, mais même entre les différentes versions de Windows et de MacOS. En effet, entre le JDK (*Java Development Kit* : un kit de développement pour programmeur Java contenant divers outils dont un compilateur, qui permet de traduire le code en langage machine) version 1.0 et le 1.4 (avec toutes les versions intermédiaires) il y a un monde ! Une applet compilée en 1.4 aura toutes les chances de ne pas fonctionner avec un module Java 1.2, par exemple, pour autant que l'on ait utilisé de nouvelles fonctionnalités qui n'étaient pas encore disponibles dans l'ancienne version. Nous avons donc commencé par travailler avec la dernière version (1.4), mais presque aucun ordinateur (en dehors des nôtres) ne pouvait correctement interpréter ce que nous avons compilé. Nous avons ensuite tenté de compiler notre applet avec la version 1.2, mais tous les problèmes de compatibilité n'étaient pas résolus. Finalement nous avons réussi à avoir une applet compatible avec toutes les versions (ou presque !) de système d'exploitation grâce au JDK 1.1.

Nous n'avons pas rencontré d'autres difficultés aussi importantes, bien qu'il ait fallu aussi se mettre à « raisonner en Java », ce qui n'était pas évident au début (surtout pour comprendre le système des classes et des méthodes.)

## Finitions

Il n'est pas évident de mettre un point final à un programme et de dire qu'il est vraiment terminé. Nous aurions pu aller plus loin dans le développement de notre applet et lui ajouter de plus en plus de fonctionnalités ; cependant, il fallait bien s'arrêter quelque part. Plus concrètement, nous aurions pu développer un environnement à trois dimensions, améliorer les graphismes (textures, boutons réactifs, etc.) et éventuellement ajouter des fonctions d'enregistrement et d'impression de fichiers.

## Conclusion

Ce travail de maturité nous a permis de mieux apprécier les difficultés qui se présentent durant la conception d'un programme informatique. Nous avons pu nous faire une bonne idée de ce que représente la construction d'un logiciel, et cela en partant de zéro. Comme l'informatique est un domaine qui nous intéresse beaucoup, nous avons ainsi pu avoir un bon aperçu de ce en quoi consistait la programmation et nous sommes par conséquent très satisfaits d'avoir choisi ce travail de maturité.

## Code source

```
//GRAVITATION

//Bibliothèques à importer utiles dans l'applet
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

//Classe principale de l'applet
public class gravitation extends Applet implements ItemListener,
ActionListener, Runnable, MouseListener, MouseMotionListener, KeyListener,
TextListener, FocusListener, AdjustmentListener
{
    ////////////////////////////////////////////////////
    //DEFINITION DES VARIABLES//
    ////////////////////////////////////////////////////

    //Le Thread
    Thread firstThread;
    boolean stop = true;           //Variable pour l'arrêt du Thread

    //Variables graphiques générales
    Graphics gr;
    Image im;
    int hauteur, longueur;         //Dimensions de l'applet
    int hauteurdonnees = 230;      //Hauteur de la zone du menu
    double grillage = 50;          //Variable donnant l'incrémementation
    entre chaque ligne du grillage
    int mingrillagex, mingrillagey; //

    //Constantes
    double G = 500;                //Gravitation
    double dt = 0.01;              //Delta temps

    //Variables contenant les maximums autorisés pour l'applet
    int maxmasse = 5000;
    int minmasse = 1;
    int maxvitesse = 500;
    int maxpos = 10000;
    int nbmax = 50;

    //Champs de texte, listes, cases à cocher, boutons, barres de défilement
    et étiquettes du menu
    Label nomasse;
    Label insmasse;
    Label labelmasse;
    Label labelx;
    Label labely;
    Label labelvx;
    Label labelvy;
```

```

Label labelvitesse;
Label labelzoom;
Label labeltraj;
Label labelcouleur;
Label affichervecteurs;

Button boutonmarchearret;
Button boutonmenu;
Button nvllemasse;
Button delete;
Button deleteall;
Button reset;

TextField textemasse;
TextField textex;
TextField textey;
TextField textevx;
TextField textevy;

Checkbox affichagegrillage;
Checkbox affichageforces;
Checkbox affichagevitesses;
Checkbox affichagenoms;
Checkbox affichagelabels;
Checkbox casestationnaire;

java.awt.List listemasses;

Choice choixtraj;
Choice choixcouleur;

Scrollbar selectvitesse;
Scrollbar zoomsb;

//Variables utiles pour les options du menu
boolean menuvisible = true;           //Afficher le menu ?

double zoom = 1;                       //Taux de zoom
int vitesse = 20;                       //Vitesse de la simulation

boolean affichergrillage = false;      //Afficher le grillage ?
boolean afficherforces = true;         //Afficher les vecteurs force ?
boolean affichervitesses = true;      //Afficher les vecteurs vitesse ?
boolean affichernoms = false;          //Afficher les étiquettes des masses ?
boolean afficherlabels = true;        //Afficher les labels de position,
masse, vitesse

int traj = 0;                           //Type de trajectoire
int incrtraj = 20;                       //

int couleur = 0;                         //Couleur de fond (0: noir, 1: blanc)

//Variables diverses
int i = 0;

```

```

int j = 0;
int z;

int massetot = 0;           //Nombre de masses
int masseselectionnee = 0; //Numéro de la masse sélectionnée
int massesdefinies;       //nombre de masses définies au sein de la
page HTML (avec les <param>)
boolean selectionnee;      //Y a-t-il une masse sélectionnée ?
boolean masseaussiselect = false;

boolean startunefois = false; //Le bouton démarrer a-t-il été pressé au
moins une fois ?

boolean estsurmasse = false; //Variables enregistrant si le
pointeur est sur une masse...
int estsurmasseno = 0;       //et si oui, sur laquelle
boolean estsurvitesse = false; //de même pour un vecteur vitesse
int estsurvitesseno = 0;     //

boolean touchefrappee;      //Une touche a-t-elle été
dernièrement frappée ?
TextComponent textfieldselectionne; //Variable qui définit dans quel
textfield la dernière touche a été tapée
int code;                   //Variable contenant le code du
dernier caractère entré
char derchar;
int val;

//Définition des tableaux de caractéristiques des masses
boolean stationnairefixe[] = new boolean[nbmax];
boolean stationnaire[] = new boolean[nbmax];
int diam[] = new int[nbmax];
int masse[] = new int[nbmax];
int massefixe[] = new int[nbmax];
int nommasse[] = new int[nbmax];
double x[] = new double[nbmax];
int xfixe[] = new int[nbmax];
double y[] = new double[nbmax];
int yfixe[] = new int[nbmax];
int coordx[] = new int[nbmax];
int coordy[] = new int[nbmax];
double vx[] = new double[nbmax];
int vxfixe[] = new int[nbmax];
double vy[] = new double[nbmax];
int vyfixe[] = new int[nbmax];
int vitessex[] = new int[nbmax];
int vitessey[] = new int[nbmax];
double ax[] = new double[nbmax];
double ay[] = new double[nbmax];
double fx[] = new double[nbmax];
double fy[] = new double[nbmax];
int forcex[] = new int[nbmax];
int forcey[] = new int[nbmax];

```

```

//Variables de "transition" utilisées lorsqu'on appuie sur le bouton
effacer
int transmasse, transx, transy, transvx, transvy;
boolean transstat;

//Tableaux de variables stockant les distances entre les masses
double distance[][] = new double[nbmax][nbmax];
double distancecarre[][] = new double[nbmax][nbmax];
double totrayons[][] = new double[nbmax][nbmax];

//Vecteurs
Vector dernierspoints[] = new Vector[nbmax]; //Vecteur mémorisant les
points des trajectoires, on en crée 1 par masse
Vector collisionpoints = new Vector(); //Vecteur mémorisant les
points de collision
int indexderpoints[] = new int[nbmax];
int maxptspmasse; //Définit le nombre
maximal de points qu'un vecteur peut contenir pour 1 masse

//Période
int n = 0;

//Variables pour la gestion de la souris
int click = 0; //Comptes les clics lors de la création
d'une masse à la souris

int possourisx = 0;
int possourisy = 0;
int mousemovedx = 0; //Variables stockant les positions de la
souris (bougée, cliquée, tirée, pressée, etc...)
int mousemovedy = 0;
int mouseclickedx = 0;
int mouseclickedy = 0;
int mousedraggedx = 0;
int mousedraggedy = 0;
int mousepressedx = 0;
int mousepressedy = 0;
int distpressedsommetx = 0;
int distpressedsommety = 0;

boolean pressedmasse = false; //La souris est-elle enfoncée sur une
masse ?
boolean pressedvitesse = false; //La souris est-elle enfoncée sur un
vecteur ?

int position; //Position du pointeur

//Variables concernant le système de bulles d'aide
int debutssubstring, finsubstring;
boolean continuer, espace;
char dercharligne;
int bulleaide = 0;
String texte;

```

```

String trans;

////////////////////////////////////
//FIN DE LA DEFINITION DES VARIABLES//
////////////////////////////////////

//Classe permettant d'enregistrer la trace des masses en mouvement
public class trace
{
    double x,y;
    int p;
    trace(double coordx, double coordy)
    {
        this.x = coordx;
        this.y = coordy;
    }
    trace(double coordx, double coordy, int periode)
    {
        this.x = coordx;
        this.y = coordy;
        this.p = periode;
    }
}
trace ntrace, ntraceb;

public int chercherparametre(String nomparam)
{
    int val;

    if (getParameter(nomparam) == null)
    {
        val = 0;
    }
    else
    {
        val = Integer.parseInt(getParameter(nomparam));
    }

    return(val);
}

//Méthode appelée au lancement de l'applet (initialisation)
public void init()
{
    //Va chercher les dimensions de l'applet telle qu'elle est dans le
navigateur
    longueur = getSize().width;
    hauteur = getSize().height;

    //Afin que toute l'applet "écoute" les mouvements de la souris
    addMouseListener(this);
    addMouseMotionListener(this);
}

```

```
//Afin de pouvoir placer les éléments graphiques selon des coordonnées
(entrées plus bas grâce à .setBounds())
setLayout(null);

//Initialisation des Labels (étiquettes)

insmasse = new Label("Modifier la masse no");
insmasse.setBounds(115,hauteur-hauteurdonnees+65,120,20);
insmasse.setBackground(Color.lightGray);
insmasse.addMouseListener(this);
insmasse.setVisible(false);
add(insmasse);

nomasse = new Label();
nomasse.setVisible(false);
nomasse.setBounds(235,hauteur-hauteurdonnees+66,25,18);
nomasse.setBackground(Color.white);
nomasse.addMouseListener(this);
add(nomasse);

labelmasse = new Label("Masse :");
labelmasse.setBounds(115,hauteur-hauteurdonnees+93,95,20);
labelmasse.setBackground(Color.lightGray);
labelmasse.addMouseListener(this);
add(labelmasse);

labelx = new Label("Position en x :");
labelx.setBounds(115,hauteur-hauteurdonnees+115,95,20);
labelx.setBackground(Color.lightGray);
labelx.addMouseListener(this);
add(labelx);

labely= new Label("Position en y :");
labely.setBounds(115,hauteur-hauteurdonnees+137,95,20);
labely.setBackground(Color.lightGray);
labely.addMouseListener(this);
add(labely);

labelvx= new Label("Vitesse en x :");
labelvx.setBounds(115,hauteur-hauteurdonnees+181,95,20);
labelvx.setBackground(Color.lightGray);
labelvx.addMouseListener(this);
add(labelvx);

labelvy= new Label("Vitesse en y :");
labelvy.setBounds(115,hauteur-hauteurdonnees+203,95,20);
labelvy.setBackground(Color.lightGray);
labelvy.addMouseListener(this);
add(labelvy);

labelvitesse = new Label("Vitesse : "+vitesse+"x");
labelvitesse.setBounds(270,hauteur-hauteurdonnees+32,100,20);
labelvitesse.setBackground(Color.lightGray);
labelvitesse.addMouseListener(this);
```

```

add(labelvitesse);

labelzoom = new Label("Zoom : "+(int)(zoom*100)+"%");
labelzoom.setBounds(270,hauteur-hauteurdonnees+76,100,20);
labelzoom.setBackground(Color.lightGray);
labelzoom.addMouseListener(this);
add(labelzoom);

labelcouleur = new Label("Couleur de fond :");
labelcouleur.setBounds(270,hauteur-hauteurdonnees+164,100,20);
labelcouleur.setBackground(Color.lightGray);
labelcouleur.addMouseListener(this);
add(labelcouleur);

labeltraj = new Label("Type de trajectoire :");
labeltraj.setBounds(270,hauteur-hauteurdonnees+120,114,20);
labeltraj.setBackground(Color.lightGray);
labeltraj.addMouseListener(this);
add(labeltraj);

affichervecteurs = new Label("Afficher vecteurs :");
affichervecteurs.setBounds(435,hauteur-hauteurdonnees+54,100,20);
affichervecteurs.setBackground(Color.lightGray);
affichervecteurs.addMouseListener(this);
add(affichervecteurs);

//Initialisation des Boutons

boutonmarchearret = new Button("D\u00e9marrer");
boutonmarchearret.setBounds(longueur/2-80,hauteur-
hauteurdonnees,80,30);
boutonmarchearret.addActionListener(this);
boutonmarchearret.setEnabled(false);
boutonmarchearret.addMouseListener(this);
add(boutonmarchearret);

boutonmenu = new Button("Cacher le Menu");
boutonmenu.setBounds(longueur-100,hauteur-hauteurdonnees,100,30);
boutonmenu.addActionListener(this);
boutonmenu.setEnabled(true);
boutonmenu.addMouseListener(this);
add(boutonmenu);

nvllemasse = new Button("Nouvelle masse");
nvllemasse.setBounds(115,hauteur-hauteurdonnees+35,145,25);
nvllemasse.addActionListener(this);
nvllemasse.addMouseListener(this);
add(nvllemasse);

delete= new Button("Effacer");
delete.setBounds(5,hauteur-hauteurdonnees+172,50,25);
delete.addActionListener(this);
delete.setEnabled(false);
delete.addMouseListener(this);

```

```

add(delete);

deleteall= new Button("Effacer tout");
deleteall.setBounds(5,hauteur-hauteurdonnees+199,80,25);
deleteall.addActionListener(this);
deleteall.setEnabled(false);
deleteall.addMouseListener(this);
add(deleteall);

reset = new Button("R\u00e9initialiser");
reset.setBounds(longueur/2,hauteur-hauteurdonnees,80,30);
reset.addActionListener(this);
reset.setEnabled(false);
reset.addMouseListener(this);
add(reset);

//Initialisation des champs de texte

textemasse = new TextField();
//pour qu'il appelle la classe ActionEvent lorsque la touche "Entrée"
est frappée
textemasse.addActionListener(this);
//pour qu'il appelle la classe FocusEvent lorsque le textfield reçoit
le focus
textemasse.addFocusListener(this);
//pour qu'il appelle la classe KeyEvent lorsqu'un caractère y est
entré
textemasse.addKeyListener(this);
//pour qu'il appelle la classe TextEvent lorsqu'un texte y est entré
textemasse.addTextListener(this);
textemasse.addMouseListener(this);
textemasse.setEditable(false);
textemasse.setBounds(210,hauteur-hauteurdonnees+93,50,20);
add(textemasse);

textex = new TextField();
textex.addActionListener(this);
textex.addTextListener(this);
textex.addFocusListener(this);
textex.addKeyListener(this);
textex.addMouseListener(this);
textex.setEditable(false);
textex.setBounds(210,hauteur-hauteurdonnees+115,50,20);
add(textex);

textey = new TextField();
textey.addActionListener(this);
textey.addTextListener(this);
textey.addFocusListener(this);
textey.addKeyListener(this);
textey.addMouseListener(this);
textey.setEditable(false);
textey.setBounds(210,hauteur-hauteurdonnees+137,50,20);
add(textey);

```

```

textevx= new TextField();
textevx.addActionListener(this);
textevx.addTextListener(this);
textevx.addFocusListener(this);
textevx.addKeyListener(this);
textevx.addMouseListener(this);
textevx.setEditable(false);
textevx.setBounds(210,hauteur-hauteurdonnees+181,50,20);
add(textevx);

textevy= new TextField();
textevy.addActionListener(this);
textevy.addTextListener(this);
textevy.addFocusListener(this);
textevy.addKeyListener(this);
textevy.addMouseListener(this);
textevy.setEditable(false);
textevy.setBounds(210,hauteur-hauteurdonnees+203,50,20);
add(textevy);

// Initialisation des cases à cocher

casestationnaire = new Checkbox("Stationnaire", true);
casestationnaire.setState(false);
casestationnaire.setEnabled(false);
casestationnaire.setBounds(115,hauteur-hauteurdonnees+159,100,20);
casestationnaire.setBackground(Color.lightGray);
casestationnaire.addItemListener(this);
casestationnaire.addMouseListener(this);
add(casestationnaire);

affichagegrillage = new Checkbox("Afficher le grillage",
affichagegrillage);
affichagegrillage.setEnabled(true);
affichagegrillage.setBounds(435,hauteur-hauteurdonnees+32,120,20);
affichagegrillage.setBackground(Color.lightGray);
affichagegrillage.addItemListener(this);
affichagegrillage.addMouseListener(this);
add(affichagegrillage);

affichageforces = new Checkbox("force", afficherforces);
affichageforces.setEnabled(true);
affichageforces.setBounds(535,hauteur-hauteurdonnees+54,55,20);
affichageforces.setBackground(Color.lightGray);
affichageforces.addItemListener(this);
affichageforces.addMouseListener(this);
add(affichageforces);

affichagevitesses = new Checkbox("vitesse", affichervitesses);
affichagevitesses.setEnabled(true);
affichagevitesses.setBounds(590,hauteur-hauteurdonnees+54,65,20);
affichagevitesses.setBackground(Color.lightGray);
affichagevitesses.addItemListener(this);

```

```
affichagevitesses.addMouseListener(this);
add(affichagevitesses);

affichagegenoms = new Checkbox("Afficher les \u00e9tiquettes",
affichagegenoms);
affichagegenoms.setEnabled(true);
affichagegenoms.setBounds(435,hauteur-hauteurdonnees+76,140,20);
affichagegenoms.setBackground(Color.lightGray);
affichagegenoms.addItemListener(this);
affichagegenoms.addMouseListener(this);
add(affichagegenoms);

affichageelabels = new Checkbox("Afficher les labels de position,
masse, vitesse", true);
affichageelabels.setEnabled(true);
affichageelabels.setBounds(435,hauteur-hauteurdonnees+98,300,20);
affichageelabels.setBackground(Color.lightGray);
affichageelabels.addItemListener(this);
affichageelabels.addMouseListener(this);
add(affichageelabels);

// Initialisation de la liste

listemasses = new java.awt.List(11,false);
listemasses.setBounds(5,hauteur-hauteurdonnees+35,100,130);
listemasses.addItemListener(this);
listemasses.addMouseListener(this);
add(listemasses);

// Initialisation des menus déroulants

choixtraj = new Choice();
choixtraj.setBounds(270,hauteur-hauteurdonnees+142,156,20);
choixtraj.addItem("Aucun point");
choixtraj.addItem("Peu de points");
choixtraj.addItem("Moyennement de points");
choixtraj.addItem("Beaucoup de points");
choixtraj.addItem("Trait continu");
choixtraj.addItemListener(this);
choixtraj.addMouseListener(this);
add(choixtraj);

choixcouleur = new Choice();
choixcouleur.setBounds(270,hauteur-hauteurdonnees+186,80,20);
choixcouleur.addItem("Noir");
choixcouleur.addItem("Blanc");
choixcouleur.addItemListener(this);
choixcouleur.addMouseListener(this);
add(choixcouleur);

// Initialisation des "ascenseurs"

zoomsb = new Scrollbar();
```

```

zoomsb.setOrientation(Scrollbar.HORIZONTAL);
zoomsb.setMaximum(110);
zoomsb.setMinimum(10);
zoomsb.setBounds(270,hauteur-hauteurdonnees+98,100,20);
zoomsb.setValue(110);
zoomsb.addAdjustmentListener(this);
zoomsb.addMouseListener(this);
add(zoomsb);

selectvitesse = new Scrollbar();
selectvitesse.setBounds(270,hauteur-hauteurdonnees+54,100,20);
selectvitesse.setOrientation(Scrollbar.HORIZONTAL);
selectvitesse.setMaximum(210);
selectvitesse.setMinimum(1);
selectvitesse.setValue(vitesse);
selectvitesse.addAdjustmentListener(this);
selectvitesse.addMouseListener(this);
add(selectvitesse);

//Création du thread
firstThread = new Thread(this);

//Démarrage du thread
firstThread.start();

//Création d'une image remplissant toute l'applet
im = createImage(longueur,hauteur);

// On définit chacun des vecteurs contenus dans le tableau
dernierspoints[] 1 par 1
for(i = 0; i < dernierspoints.length; i++)
{
    dernierspoints[i] = new Vector();
}

//on définit le nom de chaque masse comme étant son numéro
for (i = 0; i < nbmax; i++)
{
    nommasse[i] = i;
}

massesdefinies = chercherparametre("massesdefinies");

if (massesdefinies == 0)
{
    // définit les caractéristiques de départ des masses par défaut
    que l'on insère.
    // ici, les masses 0 à 4 sont définies par des valeurs bien
    déterminées
    massefixe[0] = 3500; xfixe[0] = 0; yfixe[0] = 0;
stationnairefixe[0] = true; vxfixe[0] = 0; vyfixe[0] = 0;
    massefixe[1] = 300; xfixe[1] = 250; yfixe[1] = 0;
stationnairefixe[1] = false; vxfixe[1] = 0; vyfixe[1] = 85;

```

```

    massefixe[2] = 50; xfixe[2] = 105; yfixe[2] = 0;
stationnairefixe[2] = false; vxfixe[2] = 0; vyfixe[2] = 130;
    massefixe[3] = 400; xfixe[3] = -350; yfixe[3] = 0;
stationnairefixe[3] = false; vxfixe[3] = 0; vyfixe[3] = -70;
    massefixe[4] = 120; xfixe[4] = 840; yfixe[4] = 0;
stationnairefixe[4] = false; vxfixe[4] = 0; vyfixe[4] = 51;

    massesdefinies = 5;
}
else
{
    for (i=0; i<massesdefinies; i++)
    {
        massefixe[i] = chercherparametre("m["+i+"]");
        xfixe[i] = chercherparametre("x["+i+"]");
        yfixe[i] = chercherparametre("y["+i+"]");
        if (chercherparametre("s["+i+"]") == 0)
        {
            stationnairefixe[i] = false;
        }
        else
        {
            stationnairefixe[i] = true;
        }
        vxfixe[i] = chercherparametre("vx["+i+"]");
        vyfixe[i] = chercherparametre("vy["+i+"]");
    }
}

// les masses suivantes sont définies ici :
for (i = massesdefinies; i < nbmax; i++)
{
    double vitesseorbit;
    double distanceorbit;
    int negx = 0;
    int negy = 0;
    if (Math.random() < 0.5) negx = 1;
    if (Math.random() < 0.5) negy = 1;

    // leur masses est aléatoire
    massefixe[i] = 50 + (int)(400*Math.random());
    stationnairefixe[i] = false;

    // leur position est aléatoire
    xfixe[i] = 50 + (int)(4000*Math.random());
    if(negx == 1) xfixe[i] *= -1;

    yfixe[i] = 50 + (int)(3000*Math.random());
    if(negy == 1) yfixe[i] *= -1;

    distanceorbit =Math.sqrt((xfixe[i]*xfixe[i])+(yfixe[i]*yfixe[i]));
    vitesseorbit = 1.1*Math.sqrt((G*massefixe[0])/(distanceorbit));
}

```

```

        // leur vitesse est calculée de manière à ce qu'elle soit
perpendiculaire au vecteur position,
        // et inversement prop. à la distance les séparant du centre
vxfixe[i] = -(int)(vitesseorbit*(yfixe[i]/distanceorbit));
vyfixe[i] = (int)(vitesseorbit*(xfixe[i]/distanceorbit));
    }
}

//Méthode définissant l'action du thread
public void run()
{
    try
    {
        while(true)                //Boucle du thread
        {
            if(!stop)repaint();    //A chaque répétition de la
boucle, on ré-invoque la méthode paint
            firstThread.sleep(20); //Pause entre chaque action
        }
    }
    catch(InterruptedException e) {}
}

//Méthode transformant le système de coordonnées en x pour que 0 soit au
milieu de l'applet (renvoie la coordonnée transformée)
public double placex(double x)
{
    return(longueur/2+x);
}

//Méthode transformant le système de coordonnées en y pour que 0 soit au
milieu de l'applet (renvoie la coordonnée transformée)
public double placey(double y)
{
    return((hauteur-hauteurdonnees)/2-y);
}

//Méthode testant si une collision a lieu et retourne "true" ou "false"
public boolean testercollision(int i)
{
    boolean coll = false;
    for (int j=0; j<massetot; j++)
    {
        if (i!=j && j == nommasse[j]){
            diam[masseselectionnee]= (int)
Math.sqrt(4*masse[masseselectionnee]);
            totrayons[i][j] = diam[i]/2 + diam[j]/2;
            distancecarre[i][j]= (x[i]-x[j])*(x[i]-x[j])+(y[i]-
y[j])*(y[i]-y[j]);
            distance[i][j]=Math.sqrt(distancecarre[i][j]);

            if (totrayons[i][j]>distance[i][j])
            {
                coll = true;
            }
        }
    }
}

```

```

        }
    }
    }
    return(coll);
}

//Méthode renvoyant quelle masse entre en collision (en retournant son
numéro)
public int quellemassecollision(int i)
{
    int coll = 0;
    for (int j=0; j<massetot; j++)
    {
        if (i!=j && j == nommasse[j]){
            totrayons[i][j] = diam[i]/2 + diam[j]/2;
            distancecarre[i][j]= (x[i]-x[j])*(x[i]-x[j])+(y[i]-
y[j])*(y[i]-y[j]);
            distance[i][j]=Math.sqrt(distancecarre[i][j]);

            if (totrayons[i][j]>distance[i][j])
            {
                coll = j;
            }
        }
    }
    return(coll);
}

// Méthode testant si toutes les masses sont stationnaires (renvoie "true"
ou "false")
public boolean toutesmassesstats()
{
    boolean stat = true;
    for (i=0; i<massetot; i++)
    {
        if (i == nommasse[i])
        {
            if (!stationnaire[i])
            {
                stat = false;
            }
        }
    }
    return(stat);
}

// Méthode invoquée n fois à chaque repaint()
public void calculs()
{
    if (!stop)n++;

    for (int i=0; i<massetot; i++)
    {
        if (i == nommasse[i])

```

```

    {
        // ici se trouve la partie centrale du programme:
        // C'est là que les forces, accélérations, vitesses et
positions des masses
        // sont calculées grâce à la méthode du pas-à-pas.

        fx[i]=0;
        fy[i]=0;
        for (int j=0; j<massetot; j++)
        {
            if (i != j && j == nommasse[j])
            {
                distancecarre[i][j]= (x[i]-x[j])*(x[i]-x[j])+(y[i]-
y[j])*(y[i]-y[j]);
                distance[i][j]=Math.sqrt(distancecarre[i][j]);
                fx[i]= fx[i]+((x[j]-
x[i])/distance[i][j])*((G*masse[i]*masse[j])/distancecarre[i][j]);
                fy[i]= fy[i]+((y[j]-
y[i])/distance[i][j])*((G*masse[i]*masse[j])/distancecarre[i][j]);
            }
        }

        if (!stationnaire[i] && !stop)
        {
            ax[i]= fx[i]/masse[i];
            ay[i]= fy[i]/masse[i];

            vx[i] = vx[i] + ax[i]*dt;
            vy[i] = vy[i] + ay[i]*dt;

            x[i] = x[i] + vx[i]*dt;
            y[i] = y[i] + vy[i]*dt;
        }
    }

    // Enregistre dans un Vector() les points qui détermineront la
trajectoire de chaque masse
    if (!stationnaire[i] && n % 2 == 0 && !stop)
    {
        maxptspmasse = 60*(int)(1000000/(massetot*60));

        if(indexderpoints[i] < maxptspmasse)
        {
            dernierspoints[i].addElement(new
trace(x[nommasse[i]],y[nommasse[i]]));
        }
        else
        {
            dernierspoints[i].setElementAt(new
trace(x[nommasse[i]],y[nommasse[i]]),(indexderpoints[i] % maxptspmasse));
        }
        indexderpoints[i]++;
    }
}

```

```

//Gestion des collisions :
// Lors de la collision de 2 masses, la masse qui a le plus petit
numéro sera la nouvelle masse issue de la collision
// et l'autre masse ne sera plus prise en compte et prendra le nom de
la première (avec le tableau nommasse[]).
// Cela permet de faire le test if(i == nommasse[i]) pour savoir s'il
faut prendre en compte la masse[i] ou pas
// Les masses dont le nom ne correspondent pas à leur "numéro"
n'existent en fait plus vraiment.

for (int i=0; i<massetot; i++)
{
    // Une masse qui n'est plus prise en compte (et dont le nom ne
correspond plus à son numéro)
    // doit avoir le même nom que la masse avec laquelle elle est
rentrée en collision
    // (si cette dernière rentre elle aussi en collision, le nom de la
première doit "suivre")
    if(i != nommasse[i])
    {
        nommasse[i] = nommasse[nommasse[i]];
    }

    if(testercollision(i) && i == nommasse[i] && !stop)
    {
        int massecoll = quellemassecollision(i); //Variable
définissant avec quelle masse la masse i entre en collision

        nommasse[massecoll] = i;

        //Les propriétés de la nouvelle masse (vitesse, etc...)
s'obtiennent selon la règle des chocs mous
        //(c'est en fait une moyenne "pondérée", voir TN p.17)
        x[i] =
(masse[i]*x[i]+masse[massecoll]*x[massecoll])/(masse[i]+ masse[massecoll]);
        y[i] =
(masse[i]*y[i]+masse[massecoll]*y[massecoll])/(masse[i]+ masse[massecoll]);
        vx[i] =
(masse[i]*vx[i]+masse[massecoll]*vx[massecoll])/(masse[i]+ masse[massecoll]);
        vy[i] =
(masse[i]*vy[i]+masse[massecoll]*vy[massecoll])/(masse[i]+ masse[massecoll]);
        masse[i] += masse[massecoll];

        // Enregistre dans un Vector() les points de collision.
collisionpoints.addElement(new trace(x[i],y[i],n));

        if(stationnaire[i] || stationnaire[massecoll])
        {
            stationnaire[i] = true;
            vx[i] = 0;
            vy[i] = 0;
        }
    }
}

```

```

    }

    // définit stop comme étant "true" si toutes les masses sont
stationnaires
    if(toutemassesstats())
    {
        stop = true;
    }
}

// Méthode qui permet d'appeler un certain nombre nb de fois "calculs()"
// C'est utile pour régler la vitesse de calcul
public void fonctioncalculs(int nb)
{
    int nbdactualisation = 1;

    while (nbdactualisation % (nb+1) != 0)
    {
        nbdactualisation++;
        calculs();
    }
}

//Méthode qui détermine la couleur des différents éléments graphiques
public void choixcouleur(int i)
{
    //la masse n'est pas sélectionnée mais qu'elle est déjà rentrée en
collision avec la masse sélectionnée, masseaussiselect = true.
    masseaussiselect = false;

    for(j=0; j<massetot; j++)
    {
        if (j != i && nommasse[j] == i && j == masseselectionnee)
        {
            masseaussiselect = true;
        }
    }
    if(i == estsurmasseno && estsurmasse)
    {
        gr.setColor(new Color(250,150,23));
    }
    else if((i == masseselectionnee || masseaussiselect) && selectionnee)
    {
        gr.setColor(Color.orange);
    }
    else
    {
        gr.setColor(Color.red);
    }
}

//Méthode qui détermine la couleur du texte
public void choixcouleur_texte()
{

```

```

        if(couleur == 0)
        {
            gr.setColor(Color.lightGray);
        }
        else
        {
            gr.setColor(Color.black);
        }
    }

    //Méthode graphique qui affiche tous les éléments de l'applet et qui est
    actualisée à chaque repaint() (dans le thread)
    public void paint(Graphics rond)
    {
        gr = im.getGraphics();

        // appelle la fonction "fonctioncalculs()", qui appelle elle-même la
        fonction calculs() un certain nombre de fois
        fonctioncalculs(vitesse);

        // définit quelques grandeurs pour les dessins des masses et des
        vecteurs
        for (int i=0; i<massetot; i++)
        {
            diam[i]= (int) Math.sqrt(4*masse[i]);

            coordx[i] = (int)(placex((x[i]*zoom))-diam[i]*zoom/2);
            coordy[i] = (int)(placey((y[i]*zoom))-diam[i]*zoom/2);

            vitessex[i] = (int) vx[i];
            vitessey[i] = (int) vy[i];

            forcex[i] = (int) fx[i]/10;
            forcey[i] = (int) fy[i]/10;
        }

        // définit la couleur du fond d'écran
        if (couleur == 0)
        {
            gr.setColor(Color.black);
        }
        else
        {
            gr.setColor(Color.white);
        }

        gr.fillRect(0,0,longueur,hauteur);

        gr.setColor(Color.lightGray);

        //Construction du grillage

        if (affichergrillage)
        {

```

```

// L'espace entre les lignes du grillage est multiplié par le zoom
(diminue la valeur),
// ce qui est compensé par z (ce qui augmente la valeur). Cela
permet d'obtenir un grillage avec
// un espacement entre les lignes proportionné, tout en ayant des
chiffres ronds pour les valeurs.

    if (50*zoom > 30)
    {
        z = 1;
    }
    else if (50*zoom >= 20)
    {
        z = 2;
    }
    else if (50*zoom >= 10)
    {
        z = 5;
    }
    else
    {
        z = 10;
    }

    grillage = z*50*zoom;

// valeurs à partir desquelles il faut commencer à dessiner le
grillage
    mingrillagex = (int) Math.ceil(((hauteur-
hauteurdonnees)/2)/grillage);
    mingrillagey = (int) Math.ceil((longueur/2)/grillage);

// dessine le grillage proprement dit
    for (int i = -(int)(longueur/(50*zoom)); i <=
(int)(longueur/(50*zoom)); i++)
    {
        if(i==0)
        {
            choixcouleur_texte();
        }
        else
        {
            if (couleur == 0)
            {
                gr.setColor(new Color(65,65,65));
            }
            else
            {
                gr.setColor(Color.lightGray);
            }
        }

        gr.drawLine((int)(longueur/2 + i*grillage),0,(int)(longueur/2
+ i*grillage), hauteur);

```

```

        choixcouleur_texte();
        gr.drawString(""+i*z*50,(int)(longueur/2 + i*grillage),
(hauteur-hauteurdonnees)/2);
    }
    for (i = -(int)((hauteur-hauteurdonnees)/(50*zoom)) ; i <=
(int)((hauteur-hauteurdonnees)/(50*zoom)); i++)
    {
        if(i==0)
        {
            choixcouleur_texte();
        }
        else
        {
            if (couleur == 0)
            {
                gr.setColor(new Color(65,65,65));
            }
            else
            {
                gr.setColor(Color.lightGray);
            }
        }
        gr.drawLine(0, (int)((hauteur-hauteurdonnees)/2+i*grillage),
longueur, (int)((hauteur-hauteurdonnees)/2+i*grillage));
        choixcouleur_texte();
        gr.drawString(""+i*z*50, longueur/2, (int)((hauteur-
hauteurdonnees)/2+i*grillage));
    }
}

choixcouleur_texte();

// dessine la période
gr.drawString("p\u00e9riode : "+n,longueur-100,20);

//Gère l'affichage de la trajectoire
if (traj == 4)
{
    for (int i=0; i<massetot; i++)
    {
        if(indexderpoints[i] > maxptspmasse)
        {
            for (int j=0; j<dernierspoints[i].size()-1; j+=1)
            {
                choixcouleur_texte();

                if((j+1) != (indexderpoints[i] % maxptspmasse))
                {
                    ntrace = (trace)dernierspoints[i].elementAt(j);
                    ntraceb = (trace)dernierspoints[i].elementAt(j+1);
                }
                else
                {

```

```

        ntrace =
(trace)dernierspoints[i].elementAt(maxptspmasse-1);
        ntraceb = (trace)dernierspoints[i].elementAt(0);
    }
    gr.drawLine((int)(placex(ntrace.x*zoom))-
1),(int)(placey(ntrace.y*zoom))-1,(int)(placex(ntraceb.x*zoom)-
1),(int)(placey(ntraceb.y*zoom))-1);
    }
}
else
{
    for (int j=0; j<dernierspoints[i].size()-1; j+=1)
    {
        choixcouleur_texte();
        trace ntrace = (trace)dernierspoints[i].elementAt(j);
        trace ntraceb =
(trace)dernierspoints[i].elementAt(j+1);
        gr.drawLine((int)(placex(ntrace.x*zoom))-
1),(int)(placey(ntrace.y*zoom))-1,(int)(placex(ntraceb.x*zoom)-
1),(int)(placey(ntraceb.y*zoom))-1);
    }
}

//on repaint la trajectoire de la masse sélectionnée (qui est en
orange) afin qu'elle soit au premier plan
if (selectionnee)
{
    if(indexderpoints[masseselectionnee] > maxptspmasse)
    {
        for (int j=0; j<dernierspoints[masseselectionnee].size()-
1; j+=1)
        {
            gr.setColor(Color.orange);

            if((j+1) != (indexderpoints[masseselectionnee] %
maxptspmasse))
            {
                ntrace =
(trace)dernierspoints[masseselectionnee].elementAt(j);
                ntraceb =
(trace)dernierspoints[masseselectionnee].elementAt(j+1);
            }
            else
            {
                ntrace =
(trace)dernierspoints[masseselectionnee].elementAt(maxptspmasse-1);
                ntraceb =
(trace)dernierspoints[masseselectionnee].elementAt(0);
            }
            gr.drawLine((int)(placex(ntrace.x*zoom))-
1),(int)(placey(ntrace.y*zoom))-1,(int)(placex(ntraceb.x*zoom)-
1),(int)(placey(ntraceb.y*zoom))-1);
        }
    }
}

```

```

    }
    else
    {
        for (int j=0; j<dernierspoints[masseselectionnee].size()-
1; j+=1)
        {
            gr.setColor(Color.orange);
            trace ntrace =
(trace)dernierspoints[masseselectionnee].elementAt(j);
            trace ntraceb =
(trace)dernierspoints[masseselectionnee].elementAt(j+1);
            gr.drawLine((int)(placex(ntrace.x*zoom)-
1),(int)(placey(ntrace.y*zoom))-1,(int)(placex(ntraceb.x*zoom)-
1),(int)(placey(ntraceb.y*zoom))-1);
        }
    }
}
else if(traj == 1 || traj == 2 || traj == 3) //Détermine
l'espace entre les points de la trajectoire
{
    if (traj == 3)
    {
        incrtraj = 10;
    }
    else if(traj == 2)
    {
        incrtraj = 30;
    }
    else if(traj == 1)
    {
        incrtraj = 60;
    }

    for (int i=0; i<massetot; i++)
    {
        for (int j=0; j<dernierspoints[i].size(); j+=incrtraj)
        {
            choixcouleur_texte();
            trace ntrace = (trace)dernierspoints[i].elementAt(j);
            gr.fillOval((int)(placex(ntrace.x*zoom)-
1),(int)(placey(ntrace.y*zoom))-1,2,2);
        }
    }

    //Pour que la trajectoire de la masse selectionnee soit en orange
    //il faut refaire une boucle for, car la trajectoire doit être au
premier plan
    //(en particulier s'il y a des collisions, car deux trajectoires
identiques sont mémorisées)
    if (selectionnee)
    {
        for (int j=0; j<dernierspoints[masseselectionnee].size()-1;
j+=incrtraj)

```

```

        {
            gr.setColor(Color.orange);
            trace ntrace =
(trace)dernierspoints[masseselectionnee].elementAt(j);
            gr.fillOval((int)(placex(ntrace.x*zoom)-
1),(int)(placey(ntrace.y*zoom))-1,2,2);
        }
    }

//Affichage des croix aux points de collision
choixcouleur_texte();
if (traj != 0)
{
    for (int i=0; i<collisionpoints.size(); i++)
    {
        trace ntrace = (trace)collisionpoints.elementAt(i);
        if (n < (ntrace.p + maxptspmasse*2))
        {
            gr.drawLine((int)(placex(ntrace.x*zoom-
5)),(int)(placey(ntrace.y*zoom-
5)),(int)(placex(ntrace.x*zoom+5)),(int)(placey(ntrace.y*zoom+5)));

gr.drawLine((int)(placex(ntrace.x*zoom+5)),(int)(placey(ntrace.y*zoom-
5)),(int)(placex(ntrace.x*zoom-5)),(int)(placey(ntrace.y*zoom+5)));
        }
    }
}

// affiche les ronds représentant les masses
for (i=0; i<massetot; i++)
{
    if (i == nommasse[i])
    {
        choixcouleur(i);

        gr.fillOval(coordx[i],coordy[i],(int)(Math.ceil(diam[i]*zoom)),(int)(Math.
ceil(diam[i]*zoom)));
    }
}

// affiche les étiquettes (noms des masses) à côté de celles-ci
for (i=0; i<massetot; i++)
{
    String titremasse = ""+(nommasse[i]+1);
    int longueurtitre = 7*titremasse.length();
    //Permet que les numéros des masses qui sont rentrées en
collisions avec la masse i soient ajoutées à l'étiquette que porte la masse
    for (j=0; j<massetot; j++)
    {
        if(i == nommasse[j] && j != nommasse[j])
        {
            titremasse += ", "+(j+1);
        }
    }
}

```

```

        trans = ""+(j+1);
        longueurtitre += (6+7*trans.length());
    }
}

if (i == nommasse[i])
{
    if (affichernoms)
    {
        if (couleur == 0)
        {
            gr.setColor(Color.black);
        }
        else
        {
            gr.setColor(Color.white);
        }
        gr.fillRect(coordx[i],coordy[i]-10,longueurtitre,10);

        choixcouleur_texte();
        gr.drawString(""+titremasse,coordx[i],coordy[i]);
    }
}

//Affichage des vecteurs vitesse et force
for (int i=0; i<massetot; i++)
{
    if (i == nommasse[i])
    {
        if (i == estsurvitesseno && estsurvitesse)
        {
            gr.setColor(new Color(250,150,23));
        }
        else
        {
            if (couleur == 0)
            {
                gr.setColor(new Color(180,210,255));
            }
            else
            {
                gr.setColor(Color.blue);
            }
        }
    }

    if(affichervitesses | !startunefois)
    {
        gr.drawLine((int)(coordx[i]+diam[i]*zoom/2),(int)(coordy[i]+diam[i]*zoom/2),(int)(coordx[i]+diam[i]*zoom/2+vitesse[x][i]*zoom),(int)(coordy[i]+diam[i]*zoom/2-vitesse[y][i]*zoom));
    }
}

```

```

        if(!stationnaire[i] && afficherforces)
        {
            gr.setColor(Color.green);

gr.drawLine((int)(coordx[i]+diam[i]*zoom/2),(int)(coordy[i]+diam[i]*zoom/2),(i
nt)(coordx[i]+diam[i]*zoom/2+forcex[i]*zoom/5),(int)(coordy[i]+diam[i]*zoom/2-
forcey[i]*zoom/5));
        }
    }

    for (i=0; i<massetot; i++)
    {
        if (i == nommasse[i])
        {
            //Gère les petits triangles qui s'affichent dans les bords de
l'applet lorsqu'une masse sort du champ de vision
            if (x[i] >= (int)((longueur/2 + diam[i]*zoom/2)/zoom) | x[i]
<= (int)((-longueur/2 - diam[i]*zoom/2)/zoom) | y[i] >= (int)((hauteur-
hauteurdonnees)/2 + diam[i]*zoom/2)/zoom) | y[i] <= (int)((-(hauteur-
hauteurdonnees)/2 - diam[i]*zoom/2)/zoom))
            {
                int longueurtriangle = 15;
                int largeurtriangle = 8;

                int distancedehors;

                int signex = (int) (x[i]/Math.abs(x[i]));
                int xtraitgd = (int)placex(signex*longueur/2);
// = longueur/2+signex*longueur/2
                int ytraitgd =
(int)placey(signex*((longueur*y[i])/(2*x[i]))); // = ((hauteur-
hauteurdonnees)/2)-signex*((longueur*y[i])/(2*x[i]))+1

                int xtriangledg[] = {xtraitgd,(int)((xtraitgd-
signex*longueurtriangle)-
(longueurtriangle*y[i]/Math.sqrt(x[i]*x[i]+y[i]*y[i])),(int)((xtraitgd-
signex*longueurtriangle+(longueurtriangle*y[i]/Math.sqrt(x[i]*x[i]+y[i]*y[i]))
));

                int ytriangledg[] =
{ytraitgd,(int)((ytraitgd+signex*longueurtriangle*(y[i]/x[i]))-
(longueurtriangle*x[i]/Math.sqrt(x[i]*x[i]+y[i]*y[i])),(int)((ytraitgd+signex*
longueurtriangle*(y[i]/x[i]))+(longueurtriangle*x[i]/Math.sqrt(x[i]*x[i]+y[i]*y
[i]))));

                if(ytraitgd > 0 && ytraitgd < (hauteur-hauteurdonnees))
                {
                    choixcouleur(i);
                    gr.fillPolygon(xtriangledg,ytriangledg,3);
                    distancedehors =
(int)(Math.sqrt(x[i]*x[i]+y[i]*y[i])*(1-signex*((longueur)/(2*x[i]*zoom))));
                    trans = ""+distancedehors;
                    if (couleur == 0)
                    {

```

```

        gr.setColor(Color.black);
    }
    else
    {
        gr.setColor(Color.white);
    }
    gr.fillRect((int)(xtraitgd-signex*longueurtriangle-
((1+signex)/2)*(trans.length()*7)),(int)(ytraitgd+signex*(longueurtriangle*(y[
i]/x[i]))+10-10*ytraitgd/(hauteur-hauteurdonnees))-10,trans.length()*7,10);
    choixcouleur_texte();
    gr.drawString(""+distancedehors,(int)(xtraitgd-
signex*longueurtriangle-
((1+signex)/2)*(trans.length()*7)),(int)(ytraitgd+signex*(longueurtriangle*(y[
i]/x[i]))+10-10*ytraitgd/(hauteur-hauteurdonnees)));
    }

    int signey = (int) (y[i]/Math.abs(y[i]));
    int xtraithb = (int)placex(signey*((hauteur-
hauteurdonnees)*x[i])/(2*y[i]));
    int ytraithb = (int)placey(signey*(hauteur-
hauteurdonnees)/2);

    int xtrianglehb[] = {xtraithb,(int)(xtraithb-
signey*longueurtriangle*(x[i]/y[i])-
(largeurtriangle*y[i]/Math.sqrt(x[i]*x[i]+y[i]*y[i])),(int)(xtraithb-
signey*longueurtriangle*(x[i]/y[i])+(largeurtriangle*y[i]/Math.sqrt(x[i]*x[i]+
y[i]*y[i])))};
    int ytrianglehb[] =
{ytraithb,(int)(ytraithb+signey*longueurtriangle-
(largeurtriangle*x[i]/Math.sqrt(x[i]*x[i]+y[i]*y[i])),(int)(ytraithb+signey*1
ongueurtriangle+(largeurtriangle*x[i]/Math.sqrt(x[i]*x[i]+y[i]*y[i])))};

    if(xtraithb > 0 && xtraithb < longueur)
    {
        choixcouleur(i);
        gr.fillPolygon(xtrianglehb,ytrianglehb,3);
        distancedehors =
(int)(Math.sqrt(x[i]*x[i]+y[i]*y[i])*(1-signey*((hauteur-
hauteurdonnees)/(2*y[i]*zoom))));
        trans = ""+distancedehors;

        if (couleur == 0)
        {
            gr.setColor(Color.black);
        }
        else
        {
            gr.setColor(Color.white);
        }
        gr.fillRect((int)(xtraithb-
signey*longueurtriangle*(x[i]/y[i])-
xtraithb*trans.length()*7/longueur),(int)(ytraithb+signey*longueurtriangle+((1
+signey)/2)*10)-10,trans.length()*7,10);
        choixcouleur_texte();
    }

```

```

        gr.drawString(""+distancedehors,(int)(xtraithb-
signey*longueurtriangle*(x[i]/y[i])-
xtraithb*trans.length()*7/longueur),(int)(ytraithb+signey*longueurtriangle+((1
+signey)/2)*10));
    }
}
}

//dessine le rectangle gris du menu
gr.setColor(Color.lightGray);
gr.fillRect(0,hauteur-hauteurdonnees, longueur, hauteurdonnees);

// dessine le rectangle de 30 pixels en dessus du menu
gr.setColor(new Color(20,20,20));
if (couleur == 0)
{
    gr.setColor(Color.black);
}
else
{
    gr.setColor(Color.white);
}
gr.fillRect(0,hauteur-hauteurdonnees, longueur, 30);

// dessine le rectangle jaune des bulles d'aide
gr.setColor(new Color(255,255,170));
gr.fillRect(435,hauteur-hauteurdonnees+122,290,100);

//Affiche l'aide en fonction du numéro de "bulleaide"
gr.setColor(Color.black);
switch(bulleaide)
{
    case 0 : texte = "Bienvenue dans gravitation !"; break;
    case 1 : texte = "Pour démarrer une simulation, vous devez
avoir insérer 2 masses au moins; pour insérer une masse, pressez
le bouton \"Nouvelle masse\" ou cliquez quelque part dans l'espace"; break;
    case 2 : texte = "Vous pouvez démarrer la simulation en
pressant le bouton \"Démarrer\" ou vous pouvez insérer une nouvelle
masse soit en pressant le bouton \"Nouvelle masse\" soit en cliquant quelque
part dans l'espace"; break;
    case 3 : texte = "Vous ne pouvez pas insérer plus de masses
car le nombre maximum (" + nbmax + " masses) a été atteint; vous pouvez
démarrer la simulation en pressant le bouton \"Démarrer\"; vous
pouvez aussi modifier les paramètres des masses d'ajout"; break;
    case 4 : texte = "La simulation est en court; vous pouvez la
mettre en pause en pressant le bouton \"Arrêter\"; vous pouvez
réinitialiser le tout en cliquant sur le bouton \"Réinitialiser\" ou
vous pouvez régler les options d'affichage (zoom, trajectoire, ...)";
break;
    case 5 : texte = "La simulation est en pause; vous pouvez la
redémarrer en pressant le bouton \"Redémarrer\" ou

```

```

r\u00e9initialiser le tout en cliquant sur le bouton \"R\u00e9initialiser\";
break;
    case 6 : texte = "En cliquant, vous ins\u00e9rerez \u00e0
l'endroit en question une nouvelle masse dont vous pourrez choisir la masse et
la vitesse"; break;
    case 7 : texte = "Vous \u00eates en train d'ins\u00e9rer une
masse; cliquez dans l'espace lorsque vous d\u00e9sirez valider la grandeur de
la masse; vous ne pouvez pas effectuer d'autre action avant"; break;
    case 8 : texte = "Vous \u00eates en train d'ins\u00e9rer une
masse; cliquez dans l'espace lorsque vous d\u00e9sirez valider la vitesse
(repr\u00e9sent\u00e9e par le trait bleu); vous ne pouvez pas effectuer
d'autre action avant"; break;
    case 9 : texte = "Cliquez et laissez appuy\u00e9 si vous
d\u00e9sirez d\u00e9placer cette masse"; break;
    case 10 : texte = "Cliquez et laissez appuy\u00e9 si vous
d\u00e9sirez modifier cette vitesse"; break;
    case 11 : texte = "Vous \u00eates en train de d\u00e9placer une
masse; rel\u00e2chez la souris pour d\u00e9finir la position"; break;
    case 12 : texte = "Vous \u00eates en train de modifier une
vitesse; rel\u00e2chez la souris pour la d\u00e9finir"; break;
    case 13 : texte = "Vous ne pouvez pas ins\u00e9rer de masse ici,
car le curseur de la souris est trop proche d'une autre masse"; break;

    case 20 : texte = "Pressez ce bouton pour pour d\u00e9marrer la
simulation"; break;
    case 21 : texte = "Pressez ce bouton pour lancer la simulation";
break;
    case 22 : texte = "Pressez ce bouton pour mettre la simulation en
pause"; break;
    case 23 : texte = "Pressez ce bouton pour relancer la simulation";
break;
    case 24 : texte = "Pressez ce bouton pour masquer le menu; vous
pourrez ensuite le faire r\u00e9appara\u00eetre en cliquant sur le bouton \"Montrer
le menu\""; break;
    case 25 : texte = "Pressez ce bouton pour ins\u00e9rer une
nouvelle masse sans avoir \u00e0 choisir ses param\u00eatres (vous pourrez
cependant les modifier si vous le souhaitez); vous pouvez \u00e9galement
ins\u00e9rer une nouvelle masse en cliquant quelque part dans l'espace";
break;
    case 26 : texte = "Pressez ce bouton pour supprimer la masse
s\u00e9lectionn\u00e9e"; break;
    case 27 : texte = "Pressez ce bouton pour supprimer toutes les
masses de la liste"; break;
    case 28 : texte = "Pressez ce bouton pour remettre \u00e0
z\u00e9ro; cela repositionne les masses \u00e0 leur position d'origine, vous
pourrez \u00e9galement effectuer des modifications sur les param\u00eatres
d'origine des masses"; break;

    case 30 : texte = "Modifiez, gr\u00e2ce \u00e0 l'ascenseur, la
vitesse de calcul de la simulation"; break;
    case 31 : texte = "Modifiez, gr\u00e2ce \u00e0 l'ascenseur, le
zoom de 10 \u00e0 100"; break;

```

```

    case 32 : texte = "Choisissez, gr\u00e2ce au menu d\u00e9roulant,
le type de trajectoire que vous d\u00e9sirez; une trajectoire continue peut
provoquer des ralentissements"; break;
    case 33 : texte = "Choisissez, gr\u00e2ce au menu d\u00e9roulant,
la couleur de fond (noir ou blanc) que vous d\u00e9sirez"; break;
    case 34 : texte = "Cochez si vous d\u00e9sirez afficher le
grillage"; break;
    case 35 : texte = "Cochez si vous d\u00e9sirez afficher les
forces; ce sont les forces d'attraction dues \u00e0 la gravitation; elles sont
repr\u00e9sent\u00e9es par les traits verts"; break;
    case 36 : texte = "Cochez si vous d\u00e9sirez afficher les
num\u00e9ros des masses \u00e0 c\u00f4t\u00e9 de celles-ci"; break;
    case 37 : texte = "Cochez si vous voulez que les labels de
position, masse et vitesse apparaissent \u00e0 c\u00f4t\u00e9 de la souris
lors de l'ins\u00e9rtion d'une masse"; break;
    case 38 : texte = "D\u00e9cochez si vous ne d\u00e9sirez pas
afficher le grillage"; break;
    case 39 : texte = "D\u00e9cochez si vous ne d\u00e9sirez pas
afficher les forces; ce sont les forces d'attraction dues \u00e0 la
gravitation; elles sont repr\u00e9sent\u00e9es par les traits verts"; break;
    case 40 : texte = "D\u00e9cochez si vous ne d\u00e9sirez pas
afficher les num\u00e9ros des masses"; break;
    case 41 : texte = "D\u00e9cochez si vous ne voulez pas que les
labels de position, masse et vitesse apparaissent \u00e0 c\u00f4t\u00e9 de la
souris lors de l'ins\u00e9rtion d'une masse"; break;
    case 42 : texte = "Cochez si vous d\u00e9sirez afficher les
vecteurs vitesse; ils sont repr\u00e9sent\u00e9es par les traits bleus";
break;
    case 43 : texte = "D\u00e9cochez si vous ne d\u00e9sirez pas
afficher les vecteurs vitesse; ils sont repr\u00e9sent\u00e9es par les traits
bleus (il ne dispara\u00eetront qu'une fois la simulation lanc\u00e9e)"; break;
    case 44 : texte = "Choisissez gr\u00e2ce aux cases \u00e2 cocher
ci-contre, si vous d\u00e9sirez afficher les vecteurs force ou les vecteurs
vitesse"; break;

    case 50 : texte = "Indique la masse s\u00e9lectionn\u00e9e; vous
pouvez modifier ses param\u00e8tres \u00e0 l'aide des champs de texte ci-
dessous"; break;
    case 51 : texte = "Indique la masse s\u00e9lectionn\u00e9e; vous
ne pouvez pas modifier ses param\u00e8tres, car la simulation est en marche";
break;
    case 52 : texte = "Indique la valeur de la masse
s\u00e9lectionn\u00e9e; vous pouvez la modifier \u00e0 l'aide du champ de
texte"; break;
    case 53 : texte = "Indique la valeur de la masse
s\u00e9lectionn\u00e9e; vous ne pouvez pas la modifier, car la simulation est
en marche"; break;
    case 54 : texte = "Indique la valeur de la coordonn\u00e9e x de la
masse s\u00e9lectionn\u00e9e; vous pouvez la modifier \u00e0 l'aide du champ
de texte"; break;
    case 55 : texte = "Indique la valeur d'origine de la
coordonn\u00e9e x de la masse s\u00e9lectionn\u00e9e; vous ne pouvez pas la
modifier, car la simulation est en marche"; break;

```

```

        case 56 : texte = "Indique la valeur de la coordonn\u00e9e y de la
masse s\u00e9lectionn\u00e9e; vous pouvez la modifier \u00e0 l'aide du champ
de texte"; break;
        case 57 : texte = "Indique la valeur d'origine de la
coordonn\u00e9e y de la masse s\u00e9lectionn\u00e9e; vous ne pouvez pas la
modifier, car la simulation est en marche"; break;
        case 58 : texte = "Indique la valeur de la composante x de la
vitesse de la masse s\u00e9lectionn\u00e9e; vous pouvez la modifier \u00e0
l'aide du champ de texte"; break;
        case 59 : texte = "Indique la valeur d'origine de la composante x
de la vitesse de la masse s\u00e9lectionn\u00e9e; vous ne pouvez pas la
modifier, car la simulation est en marche"; break;
        case 60 : texte = "Indique la valeur de la composante x de la
vitesse de la masse s\u00e9lectionn\u00e9e; vous ne pouvez pas la modifier,
car la masse s\u00e9lectionn\u00e9e est d\u00e9finie comme \u00e9tant
stationnaire"; break;
        case 61 : texte = "Indique la valeur de la composante y de la
vitesse de la masse s\u00e9lectionn\u00e9e; vous pouvez la modifier \u00e0
l'aide du champ de texte"; break;
        case 62 : texte = "Indique la valeur d'origine de la composante y
de la vitesse de la masse s\u00e9lectionn\u00e9e; vous ne pouvez pas la
modifier, car la simulation est en marche"; break;
        case 63 : texte = "Indique la valeur de la composante y de la
vitesse de la masse s\u00e9lectionn\u00e9e; vous ne pouvez pas la modifier,
car la masse s\u00e9lectionn\u00e9e est d\u00e9finie comme \u00e9tant
stationnaire"; break;
        case 64 : texte = "Indique si la masse s\u00e9lectionn\u00e9e est
d\u00e9finie comme \u00e9tant stationnaire (coch\u00e9) ou non (pas
coch\u00e9)"; break;
        case 65 : texte = "Ceci est la liste de toutes les masses
ins\u00e9r\u00e9es; cliquez sur une ligne pour s\u00e9lectionner la masse
correspondante"; break;
        case 66 : texte = "Ceci est la liste des masses
ins\u00e9r\u00e9es; vous n'avez pas encore ins\u00e9r\u00e9 de masse; vous
pouvez ins\u00e9rer une nouvelle masse soit en pressant le bouton \"Nouvelle
masse\" soit en cliquant quelque part dans l'espace"; break;
    }

    //Appelle la fonction qui \u00e9crit le texte d\u00e9fini plus haut pour la
bulle d'aide sur plusieurs lignes
    ecrire(texte,50,435,hauteur-hauteurdonnees+135);

    // g\u00e8re l'affichage des labels de position, masse ou vitesse (qui
apparaissent lors de l'insertion de masses)
    if (click == 0 && possourisy < (hauteur-hauteurdonnees) &&
!startunefois && !estpresdunemasse(mousemovedx,mousemovedy)&&
!estsurunevitesse(mousemovedx,mousemovedy) && massetot < nbmax &&
!pressedmasse && !pressedvitesse)
    {
        afficher_souris_coords(possourisx,possourisy, mousemovedx,
mousemovedy);
    }
    if (click == 0 && possourisy < (hauteur-hauteurdonnees) &&
!startunefois && pressedmasse && !pressedvitesse)

```

```

        {
            afficher_souris_coords(possourisx,possourisy,
xfixe[masseselectionnee], yfixe[masseselectionnee]);
        }

        if (click == 1 && possourisy < (hauteur-hauteurdonnees) &&
!startunefois)
        {

afficher_souris_masse(possourisx,possourisy,massefixe[masseselectionnee]);
        }

        if (click == 2 && !startunefois && possourisy< (hauteur-
hauteurdonnees))
        {

afficher_souris_vitesse(possourisx,possourisy,vxfixe[masseselectionnee],vyfixe
[masseselectionnee]);
        }
        if (click == 0 && possourisy < (hauteur-hauteurdonnees) &&
!startunefois && pressedvitesse)
        {
            afficher_souris_vitesse(possourisx,possourisy,
vxfixe[masseselectionnee], vyfixe[masseselectionnee]);
        }

        rond.drawImage(im,0,0,this); //Dessine l'image de
l'applet

        actualiserbouttons();
    }

    //Méthode de mise à jour de tous les graphismes
    public void update(Graphics g)
    {
        paint(g);
    }

    //Méthode permettant d'écrire un texte sur plusieurs lignes (pour les
bulles d'aide)
    public void ecrire(String txt, int largeur, int posx, int posy)
    {
        //Initialisation des variables utilisées dans la boucle while ci-
dessous
        finsubstring = 0;
        i = 0;
        continuer = true;

        while (continuer)
        {
            //"debutsubstring" et "finsubstring" définissent la portion de
texte qui sera écrite pour chaque ligne (de no i)

```

```

        //debutsubstring prend la valeur de finsubstring (comme ça, si le
nombre de lignes est >1,
        //le début de la portion de texte est directement après la fin de
la ligne précédente)
        debutsubstring = finsubstring;
        finsubstring = debutsubstring + largeur;

        //Si la fin de la portion dépasse la longueur du texte, continuer
est false (ce qui arrête la boucle),
        //et finsubstring prend la valeur de la longueur du texte
        if (finsubstring >= txt.length())
        {
            finsubstring = txt.length();
            continuer = false;
        }

        //On définit le dernier caractère de la ligne
        dercharligne = txt.charAt(finsubstring-1);

        //Si ce caractère n'est pas un espace, on diminue la longueur de
la ligne jusqu'à avoir un espace.
        //Cela permet de terminer une ligne sans que cela soit au milieu
d'un mot.
        if (!Character.isWhitespace(dercharligne) && continuer)
        {
            j = finsubstring-1;

            espace = false;

            while(!espace)
            {
                j--;

                dercharligne = txt.charAt(j);

                if (Character.isWhitespace(dercharligne))
                {
                    espace = true;
                    finsubstring = j+1;
                }
            }
        }

        gr.drawString(txt.substring(debutsubstring, finsubstring), posx, posy+i*20);

        i++;
    }
}

```

```

//Méthode qui permet l'affichage à côté du pointeur des coordonnées de
celui-ci
public void afficher_souris_coords(int x, int y, int valx, int valy)

```

```

{
    String text_souris_coords = "x = " + valx + "; y = " + valy ;

    if(afficherlabels)
    {
        gr.setColor(new Color(225,225,225));
        gr.fillRect(x,y-20,text_souris_coords.length()*6-5,20);
        gr.setColor(Color.black);
        gr.drawString(text_souris_coords,x+5,y-5);
    }
}

//Méthode qui permet l'affichage à côté du pointeur de la masse en
création
public void afficher_souris_masse(int x, int y, int valmasse)
{
    // String text_souris_coords = "nlle masse " + massetot + " : x = " +
mousemovedx + "; y = " + mousemovedy ;
    String text_souris_masse = "masse = " + valmasse;

    if(afficherlabels)
    {
        gr.setColor(new Color(255,150,130));
        gr.fillRect(x,y-20,text_souris_masse.length()*7+5,20);
        gr.setColor(Color.black);
        gr.drawString(text_souris_masse,x+5,y-5);
    }
}

//Méthode qui permet l'affichage à côté du pointeur de la valeur du
vecteur vitesse de la masse en création
public void afficher_souris_vitesse(int x, int y, int valvx, int valvy)
{
    String text_souris_vitesse;

    int valvitesse = (int)(Math.sqrt(valvx*valvx + valvy*valvy));
    if (valvitesse != 0)
    {
        text_souris_vitesse = "vitesse = " + valvitesse;
    }
    else
    {
        text_souris_vitesse = "stationnaire";
    }

    if(afficherlabels)
    {
        gr.setColor(new Color(150,180,255));
        gr.fillRect(x,y-20,text_souris_vitesse.length()*6+5,20);
        gr.setColor(Color.black);
        gr.drawString(text_souris_vitesse,x+5,y-5);
    }
}

```

```

////////////////////////////////////
//GESTION DES TEXTFIELDS//
////////////////////////////////////

//Méthode comprenant un test renvoyant true ou false, suivant si le code
de caractère c correspond à un chiffre ou pas
//(en plus des touches : avant, arrière, en bas, en haut, effacer, delete,
home et end)
public boolean testcode(int c, char ch)
{
    if (Character.isDigit(ch) | c == 8 | (c >= 35 && c <= 40) | (c == 45
&& position == 0) | (c == 109 && position == 0) | c == 127)
    {
        return(true);
    }
    else
    {
        return(false);
    }
}

// Méthode appelée lorsqu'une touche est pressée
public void keyPressed(KeyEvent e)
{
    code = e.getKeyCode();
    derchar = e.getKeyChar();

    //La variable "touchefrappee" permet d'éviter le cas d'une boucle
infinie
    if(code != 10)
    {
        touchefrappee = true;
    }
    repaint();
}

//Méthode appelée lorsqu'une touche est relâchée
public void keyReleased(KeyEvent e)
{
    if(testcode(code,derchar))
    {
        position = textfieldselectionne.getCaretPosition(); //Va
chercher la position du caractère tapé dans le TextField
    }
    repaint();
}

//Autre écouteur des événements clavier (non utilisé mais doit être cité
dans l'applet)
public void keyTyped(KeyEvent e) {}

//Donne la valeur en argument à masse[masseselectionnee], après avoir
vérifié que cette

```

```

    //valeur est "dans les normes" (pas trop grande , et aussi si la masse
n'est pas en collision avec une autre)
    //ATTENTION : le test "testercollision" doit, pour fonctionner, avoir
accès à toutes ces variables :
    //x[masseselectionnee], y[masseselectionnee] et masse[masseselectionnee].
Donc, risque de boucle infinie
    //si une ce ces variables pas encore définie (=> lors de la création d'une
masse)
    public void donnervaleur_masse(int valeur)
    {
        massefixe[masseselectionnee] = valeur;
        if (massefixe[masseselectionnee] > maxmasse)
        {
            massefixe[masseselectionnee] = maxmasse;           //Si c'est trop
grand la masse = maxmasse
        }
        if (massefixe[masseselectionnee] < minmasse)
        {
            massefixe[masseselectionnee] = minmasse;           //Si c'est trop
petit la masse = minmasse
        }

        masse[masseselectionnee] = massefixe[masseselectionnee];

        while (testercollision(masseselectionnee))
        {
            massefixe[masseselectionnee]--;
            masse[masseselectionnee] = massefixe[masseselectionnee];
        }
    }

    //Méthode qui onne la valeur en argument à x[masseselectionnee], après
avoir vérifié que cette
    //valeur est "dans les normes" (pas trop grande ou trop petite, et aussi
si la masse n'est pas
    //en collision avec une autre)
    //ATTENTION : le test "testercollision" doit, comme c'est déjà expliqué
plus haut, avoir accès à toutes ces variables :
    //x[masseselectionnee], y[masseselectionnee] et masse[masseselectionnee].
    public void donnervaleur_posx(int valeur)
    {
        xfixe[masseselectionnee] = valeur;
        if (xfixe[masseselectionnee] > maxpos)
        {
            xfixe[masseselectionnee] = maxpos;           //Si c'est trop
grand le x = maxpos
        }
        if (xfixe[masseselectionnee] < -maxpos)
        {
            xfixe[masseselectionnee] = -maxpos;           //Si c'est trop
petit le x = -maxpos
        }

        x[masseselectionnee] = xfixe[masseselectionnee];
    }

```

```
while (testercollision(masseselectionnee))
{
    xfixe[masseselectionnee]++;
    x[masseselectionnee] = xfixe[masseselectionnee];
}

//Méthode semblable à celle ci-dessus mais pour la position en y
public void donnervaleur_posy(int valeur)
{
    yfixe[masseselectionnee] = valeur;
    if (yfixe[masseselectionnee] > maxpos)
    {
        yfixe[masseselectionnee] = maxpos;
    }
    if (yfixe[masseselectionnee] < -maxpos)
    {
        yfixe[masseselectionnee] = -maxpos;
    }

    y[masseselectionnee] = yfixe[masseselectionnee];

    while (testercollision(masseselectionnee))
    {
        yfixe[masseselectionnee]++;
        y[masseselectionnee] = yfixe[masseselectionnee];
    }
}

//Méthode semblable à celle ci-dessus mais pour la vitesse en x
public void donnervaleur_vitessex(int valeur)
{
    vxfixe[masseselectionnee] = valeur;
    if (vxfixe[masseselectionnee] > maxvitesse)
    {
        vxfixe[masseselectionnee] = maxvitesse;
    }
    if (vxfixe[masseselectionnee] < -maxvitesse)
    {
        vxfixe[masseselectionnee] = -maxvitesse;
    }

    vx[masseselectionnee] = vxfixe[masseselectionnee];
}

//Méthode semblable à celle ci-dessus mais pour la vitesse en y
public void donnervaleur_vitessey(int valeur)
{
    vyfixe[masseselectionnee] = valeur;
    if (vyfixe[masseselectionnee] > maxvitesse)
    {
        vyfixe[masseselectionnee] = maxvitesse;
    }
}
```

```

    if (vyfixe[masseselectionnee] < -maxvitesse)
    {
        vyfixe[masseselectionnee] = -maxvitesse;
    }

    vy[masseselectionnee] = vyfixe[masseselectionnee];
}

//Méthode appelée si le contenu d'un TextField change
public void textValueChanged(TextEvent e)
{
    if (touchefrappee && click == 0 && massetot>0 && !startunefois)
    {
        if (testcode(code,derchar))
        {
            if(textfieldselectionne.getText().length()!= 0)
            {
                val = Integer.parseInt(textfieldselectionne.getText());
            }
            else
            {
                val = 0;
            }

            if (textfieldselectionne == textmasse)
            {
                donnervaleur_masse(val);
                if(val > maxmasse | val < 0)
                {
                    val = massefixe[masseselectionnee];
                    textfieldselectionne.setText(" "+val);
                    textfieldselectionne.setCaretPosition(100);
                }
            }

            if (textfieldselectionne == textex)
            {
                donnervaleur_posx(val);
                if(val > maxpos | val < -maxpos)
                {
                    val = xfixe[masseselectionnee];
                    textfieldselectionne.setText(" "+val);
                    textfieldselectionne.setCaretPosition(100);
                }
            }

            if (textfieldselectionne == textey)
            {
                donnervaleur_posy(val);
                if(val > maxpos | val < -maxpos)
                {
                    val = yfixe[masseselectionnee];
                    textfieldselectionne.setText(" "+val);
                    textfieldselectionne.setCaretPosition(100);
                }
            }
        }
    }
}

```

```

    }
}

if (textfieldselectionne == textevx)
{
    donnervaleur_vitessex(val);
    if(val > maxvitesse | val < -maxvitesse)
    {
        val = vxfixe[masseselectionnee];
        textfieldselectionne.setText(""+val);
        textfieldselectionne.setCaretPosition(100);
    }
}

if (textfieldselectionne == textevy)
{
    donnervaleur_vitessey(val);
    if(val > maxvitesse | val < -maxvitesse)
    {
        val = vyfixe[masseselectionnee];
        textfieldselectionne.setText(""+val);
        textfieldselectionne.setCaretPosition(100);
    }
}

}
else
{
    textfieldselectionne.setText(""+val);
    textfieldselectionne.setCaretPosition(position);
}
}
touchefrappee = false;
repaint();
}

//Méthode appelée lorsqu'un TextField reçoit le "focus"
public void focusGained(FocusEvent e)
{
    if(massetot > 0)
    {
        textfieldselectionne = (TextComponent)e.getSource();

        //Sélectionne toute la ligne lorsque le textfield reçoit le focus
        if(!startunefois)
        {
            textfieldselectionne.setSelectionStart(0);
            textfieldselectionne.setSelectionEnd(100);
        }
        val = Integer.parseInt(textfieldselectionne.getText());
    }
    position = textfieldselectionne.getCaretPosition();
}
}

```

```

//Méthode appelée lorsqu'un TextField perd le "focus"
public void focusLost(FocusEvent e)
{
    if (massetot > 0)
    {
        if (e.getSource() == textemasse)
        {
            textemasse.setText(""+massefixe[masseselectionnee]);
        }
        if (e.getSource() == textex)
        {
            textex.setText(""+xfixe[masseselectionnee]);
        }
        if (e.getSource() == textey)
        {
            textey.setText(""+yfixe[masseselectionnee]);
        }
        if (e.getSource() == textevx)
        {
            textevx.setText(""+vxfixe[masseselectionnee]);
        }
        if (e.getSource() == textevy)
        {
            textevy.setText(""+vyfixe[masseselectionnee]);
        }
    }
}

////////////////////////////////////
//GESTION DE LA SOURIS//
////////////////////////////////////

//Méthode invoquée lorsque la souris est pressée, l'"attention" n'est
alors plus sur le textfield,
//et la méthode sortietextfield est alors invoquée, mais le false dit
qu'il ne faut pas donner le focus au champ suivant
public void mousePressed(MouseEvent e)
{
    if(e.getSource() == this)
    {
        mousepressedx = (int)((e.getX() - longueur/2)/zoom);
        mousepressedy = (int)(((hauteur-hauteurdonnees)/2 -
e.getY())/zoom);
        possourisx = e.getX();
        possourisy = e.getY();

        if (!startunefois && click == 0 &&
estsurunemasse(mousepressedx,mousepressedy) &&
!estsurunevitesse(mousepressedx,mousepressedy))
        {
            masseselectionnee =
estsurquellemasse(mousepressedx,mousepressedy);

```

```

        distpressedsommetx = mousepressedx - xfixe[masseselectionnee];
        distpressedsommety = mousepressedy - yfixe[masseselectionnee];

        pressedmasse = true;
        selectionnee = true;

        actualiserTF();

this.setCursor(Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR));

        bulleaide = 11;

        repaint();
    }

    if (!startunefois && click == 0 &&
estsurunevitesse(mousepressedx,mousepressedy))
    {
        masseselectionnee =
estsurquellevitesse(mousepressedx,mousepressedy);

        vxfixe[masseselectionnee] = mousepressedx -
xfixe[masseselectionnee];
        vyfixe[masseselectionnee] = mousepressedy -
yfixe[masseselectionnee];

        donnervaleur_vitesse(vxfixe[masseselectionnee]);
        donnervaleur_vitesse(vyfixe[masseselectionnee]);

        pressedvitesse = true;

        actualiserTF();

this.setCursor(Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR));

        bulleaide = 12;

        repaint();
    }
}

//Méthode invoquée lorsque la souris est cliquée
public void mouseClicked(MouseEvent e)
{
    if(e.getSource() == this)
    {
        mouseclickedx = (int)((e.getX() - longueur/2)/zoom);
        mouseclickedy = (int)((hauteur-hauteurdonnees)/2 -
e.getY())/zoom);
    }
}

```

```

    if (!startunefois && e.getY() < (hauteur-hauteurdonnees))
    {
        // pour insérer une masse, il faut effectuer 3 clics
        successifs:
        // 1. pour définir la position 2. pour définir la masse 3.
        pour définir la vitesse
        // La variable "click" indique dans laquelle de ces trois
        possibilité on se trouve
        if (click == 0)
        {
            if (!estpresdunemasse(mouseclickedx, mouseclickedy) &&
            !estsurunevitesse(mouseclickedx, mouseclickedy))
            {
                if (massetot < nbmax)
                {
                    massetot++;
                    massesélectionnee = massetot-1;

                    stationnairefixe[massesélectionnee] = false;

                    x[massesélectionnee] = mouseclickedx;
                    y[massesélectionnee] = mouseclickedy;
                    masse[massesélectionnee] = 1;
                    donnervaleur_posx(mouseclickedx);
                    donnervaleur_posy(mouseclickedy);
                    donnervaleur_masse(1);
                    donnervaleur_vitesse_x(0);
                    donnervaleur_vitesse_y(0);

                    click = 1;

                    listemasses.removeAll();
                    for ( i = 0; i < massetot; i++)
                    {
                        listemasses.add("masse "+ (i+1), i);
                    }
                    sélectionnee = true;

                    bulleaide = 7;
                }
                else
                {
                    sélectionnee = false;
                }
            }
        }
        else if (click == 1)
        {
            vxfixe[massesélectionnee] = (mouseclickedx-
            xfixe[massesélectionnee]);
            vyfixe[massesélectionnee] = (mouseclickedy-
            yfixe[massesélectionnee]);

```

```

        if(Math.abs(vxfixe[masseselectionnee])<5 &&
Math.abs(vyfixe[masseselectionnee])<5)
        {
            stationnairefixe[masseselectionnee]=true;
            vxfixe[masseselectionnee]= 0;
            vyfixe[masseselectionnee]= 0;
        }
        else
        {
            stationnairefixe[masseselectionnee]=false;
        }

        donnervaleur_vitessex(vxfixe[masseselectionnee]);
        donnervaleur_vitessey(vyfixe[masseselectionnee]);

        click = 2;

        bulleaide = 8;
    }

    else if (click == 2)
    {
        vxfixe[masseselectionnee] = mouseclickedx -
xfixe[masseselectionnee];
        vyfixe[masseselectionnee] = mouseclickedy -
yfixe[masseselectionnee];

        if(Math.abs(vxfixe[masseselectionnee])<5 &&
Math.abs(vyfixe[masseselectionnee])<5)
        {
            stationnairefixe[masseselectionnee]=true;
            vxfixe[masseselectionnee]= 0;
            vyfixe[masseselectionnee]= 0;
        }
        else
        {
            stationnairefixe[masseselectionnee]=false;
        }

        donnervaleur_vitessex(vxfixe[masseselectionnee]);
        donnervaleur_vitessey(vyfixe[masseselectionnee]);
        stationnaire[masseselectionnee] =
stationnairefixe[masseselectionnee];

        estsurvitesseno = masseselectionnee;
        estsurvitesse = true;

this.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        repaint();

        click = 0;

```

```

        bulleaide = 10;
    }
}
if (startunefois && e.getY() < (hauteur-hauteurdonnees))
{
    selectionnee = false;
}
actualiserTF();
}
repaint();
}

//Méthode invoquée lorsque la souris est relâchée
public void mouseReleased(MouseEvent e)
{
    if(e.getSource() == this)
    {
        if(estsurunemasse(possourisx,possourisy) |
estsurunevitesse(possourisx,possourisy))
        {
this.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

            if (estsurunemasse(mousedraggedx,mousedraggedy))
            {
                bulleaide = 9;
            }
            if (estsurunevitesse(mousedraggedx,mousedraggedy))
            {
                bulleaide = 10;
            }
        }
        pressedmasse = false;
        pressedvitesse = false;
    }
    repaint();
}

//Méthode qui renvoie "true" si les positions en arguments se trouvent
près d'une masse
public boolean estpresdunemasse(int posx, int posy)
{
    boolean estpresmasse = false;

    for (i = 0; i < massetot; i++)
    {
        if(Math.sqrt((xfixe[i]-posx)*(xfixe[i]-posx)+(yfixe[i]-
posy)*(yfixe[i]-posy))<diam[i]/2+8*zoom)
        {
            estpresmasse = true;
        }
    }
    return estpresmasse;
}

```

```

//Méthode qui renvoie "true" si les positions en arguments se trouvent sur
une masse
public boolean estsurunemasse(int posx, int posy)
{
    boolean estsurmasse = false;

    for (i = 0; i < massetot; i++)
    {
        if(Math.sqrt((xfixe[i]-posx)*(xfixe[i]-posx)+(yfixe[i]-
posy)*(yfixe[i]-posy))<diam[i]/2)
            {
                estsurmasse = true;
            }
    }
    return estsurmasse;
}

//Méthode qui renvoie le numéro de la masse sur laquelle se trouvent les
coordonnées en arguments
public int estsurquellemasse(int posx, int posy)
{
    int estsurlamasse = 0;

    for (i = 0; i < massetot; i++)
    {
        if(Math.sqrt((xfixe[i]-posx)*(xfixe[i]-posx)+(yfixe[i]-
posy)*(yfixe[i]-posy))<diam[i]/2)
            {
                estsurlamasse = i;
            }
    }
    return estsurlamasse;
}

//Méthode qui renvoie "true" si les positions en arguments se trouvent sur
un vecteur vitesse
public boolean estsurunevitesse(int posx, int posy)
{
    boolean estsurvit = false;

    for (i = 0; i < massetot; i++)
    {
        if((posx > xfixe[i] + vxfixe[i] - 5/zoom && posx < xfixe[i] +
vxfixe[i] + 5/zoom) && (posy > yfixe[i] + vyfixe[i] - 5/zoom && posy <
yfixe[i] + vyfixe[i] + 5/zoom))
            {
                estsurvit = true;
            }
    }
    return estsurvit;
}

```

```

//Méthode qui renvoie le numéro du vecteur vitesse sur lequel se trouvent
les positions en arguments
public int estsurquellevitesse(int posx, int posy)
{
    int estsurlavitesse = 0;

    for (i = 0; i < massetot; i++)
    {
        if((posx > xfixe[i] + vxfixe[i] - 5/zoom && posx < xfixe[i] +
vxfixe[i] + 5/zoom) && (posy > yfixe[i] + vyfixe[i] - 5/zoom && posy <
yfixe[i] + vyfixe[i] + 5/zoom))
        {
            estsurlavitesse = i;
        }
    }
    return estsurlavitesse;
}

//Méthode invoquée lorsque la souris est déplacée
public void mouseMoved(MouseEvent e)
{
    mousemovedx = (int)((e.getX() - longueur/2)/zoom);
    mousemovedy = (int)(((hauteur-hauteurdonnees)/2 - e.getY())/zoom);
    possourisx = e.getX();
    possourisy = e.getY();

    if (click == 0 && e.getY() < (hauteur-hauteurdonnees) && !startunefois
&&
(estsurunemasse(mousemovedx,mousemovedy)|estsurunevitesse(mousemovedx,mousemov
edy)))
    {
        this.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

        if (estsurunemasse(mousemovedx,mousemovedy)){
            estsurmasse = true;
            estsurvitesse = false;
            estsurmasseno = estsurquellemasse(mousemovedx,mousemovedy);
            bulleaide = 9;
            repaint();
        }
        else
        {
            estsurmasse = false;
        }

        if (estsurunevitesse(mousemovedx,mousemovedy)){
            estsurvitesse = true;
            estsurmasse = false;
            estsurvitesseno =
estsurquellevitesse(mousemovedx,mousemovedy);
            bulleaide = 10;
            repaint();
        }
        else

```

```

        {
            estsurvitesse = false;
        }
    }
else if (!startunefois)
{
    this.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
    estsurmasse = false;
    estsurvitesse = false;

    if (e.getY() >= (hauteur-hauteurdonnees))
    {
        if (massetot <=1)
        {
            bulleaide = 1;
        }
        else if (massetot == nbmax)
        {
            bulleaide = 3;
        }
        else
        {
            bulleaide = 2;
        }
    }
    else
    {
        if (estpresdunemasse(mousemovedx, mousemovedy))
        {
            bulleaide = 13;
        }
        else
        {
            if(massetot < nbmax)
            {
                bulleaide = 6;
            }
            else
            {
                bulleaide = 3;
            }
        }
    }
    repaint();
}

if (click == 1 && !startunefois && e.getY() < (hauteur-hauteurdonnees))
{
    // en fait, la masse est égale à son rayon au carré :
    massefixe[masseselectionnee] = ((xfixe[masseselectionnee]-
mousemovedx)*(xfixe[masseselectionnee]-mousemovedx)+(yfixe[masseselectionnee]-
mousemovedy)*(yfixe[masseselectionnee]-mousemovedy));

    donnervaleur_masse(massefixe[masseselectionnee]);
}

```

```

        actualiserTF();

        bulleaide = 7;

        repaint();
    }
    else
    {
        if(click == 1)
        {
            bulleaide = 7;
        }
    }

    if (click == 2 && !startunefois && e.getY() < (hauteur-hauteurdonnees))
    {
        vxfixe[masseselectionnee] = (mousemovedx-
xfixe[masseselectionnee]);
        vyfixe[masseselectionnee] = (mousemovedy-
yfixe[masseselectionnee]);

        if(Math.abs(vxfixe[masseselectionnee]) < 5 &&
Math.abs(vyfixe[masseselectionnee]) < 5)
        {
            stationnairefixe[masseselectionnee]=true;
            vxfixe[masseselectionnee]= 0;
            vyfixe[masseselectionnee]= 0;
        }
        else
        {
            stationnairefixe[masseselectionnee]=false;
        }

        donnervaleur_vitesse(vxfixe[masseselectionnee]);
        donnervaleur_vitesse(vyfixe[masseselectionnee]);

        actualiserTF();

        bulleaide = 8;

        repaint();
    }
    else
    {
        if(click == 2)
        {
            bulleaide = 8;
        }
    }
}

//Méthode invoquée lorsque la souris est déplacée avec le bouton gauche
enfoncé

```

```

public void mouseDragged(MouseEvent e)
{
    mousedraggedx = (int)((e.getX() - longueur/2)/zoom);
    mousedraggedy = (int)((hauteur-hauteurdonnees)/2 - e.getY())/zoom);
    possourisx = e.getX();
    possourisy = e.getY();

    if (!startunefois && click == 0 && pressedmasse)
    {
        xfixe[masseselectionnee] = (mousedraggedx - distpressedsommetx);
        yfixe[masseselectionnee] = (mousedraggedy - distpressedsommety);
        x[masseselectionnee] = xfixe[masseselectionnee];
        y[masseselectionnee] = yfixe[masseselectionnee];

        donnervaleur_posy(yfixe[masseselectionnee]);
        donnervaleur_posx(xfixe[masseselectionnee]);

        actualiserTF();

        bulleaide = 11;

        repaint();
    }

    if (!startunefois && click == 0 && pressedvitesse)
    {
        vxfixe[masseselectionnee] = mousedraggedx -
xfixe[masseselectionnee];
        vyfixe[masseselectionnee] = mousedraggedy -
yfixe[masseselectionnee];

        if(Math.abs(vxfixe[masseselectionnee])<5 &&
Math.abs(vyfixe[masseselectionnee])<5)
        {
            stationnairefixe[masseselectionnee]=true;
            vxfixe[masseselectionnee]= 0;
            vyfixe[masseselectionnee]= 0;
        }
        else
        {
            stationnairefixe[masseselectionnee]=false;
        }

        donnervaleur_vitesse(x(vxfixe[masseselectionnee]));
        donnervaleur_vitesse(y(vyfixe[masseselectionnee]));
        stationnaire[masseselectionnee] =
stationnairefixe[masseselectionnee];

        actualiserTF();

        bulleaide = 12;

        repaint();
    }
}

```

```
}

//Méthode invoquée lorsque la souris entre dans le cadre de l'applet (ou
de tout autre objet)
//C'est surtout utilisé pour gérer l'affichage des bulles d'aide...
public void mouseEntered(MouseEvent e)
{
    if(click == 0)
    {
        if(e.getSource() == this)
        {
            if (!pressedmasse && !pressedvitesse)
            {
                if(!startunefois)
                {
                    if (massetot <=1)
                    {
                        bulleaide = 1;
                    }
                    else if (massetot == nbmax)
                    {
                        bulleaide = 3;
                    }
                    else
                    {
                        bulleaide = 2;
                    }
                }
            }
            else
            {
                if(stop)
                {
                    bulleaide = 5;
                }
                else
                {
                    bulleaide = 4;
                }
            }
        }
    }
}

if(e.getSource() == boutonmarchearret)
{
    if(stop)
    {
        if(!startunefois)
        {
            bulleaide = 21;
        }
        else
        {
            bulleaide = 23;
        }
    }
}
```

```
    }
    else
    {
        bulleaide = 22;
    }
}

if(e.getSource() == boutonmenu)
{
    bulleaide = 24;
}

if(e.getSource() == nvllemasse)
{
    bulleaide = 25;
}

if(e.getSource() == delete)
{
    bulleaide = 26;
}

if(e.getSource() == deleteall)
{
    bulleaide = 27;
}

if(e.getSource() == reset)
{
    bulleaide = 28;
}

if(e.getSource() == labelvitesse | e.getSource() == selectvitesse)
{
    bulleaide = 30;
}

if(e.getSource() == labelzoom | e.getSource() == zoomsb)
{
    bulleaide = 31;
}

if(e.getSource() == labeltraj | e.getSource() == choixtraj)
{
    bulleaide = 32;
}

if(e.getSource() == labelcouleur | e.getSource() == choixcouleur)
{
    bulleaide = 33;
}

if(e.getSource() == affichagegrillage)
{
```

```
    if(!affichergrillage)
    {
        bulleaide = 34;
    }
    else
    {
        bulleaide = 38;
    }
}

if(e.getSource() == affichageforces)
{
    if(!afficherforces)
    {
        bulleaide = 35;
    }
    else
    {
        bulleaide = 39;
    }
}

if (e.getSource() == affichagevitesses)
{
    if(!affichervitesses)
    {
        bulleaide = 42;
    }
    else
    {
        bulleaide = 43;
    }
}

if (e.getSource() == affichervecteurs)
{
    bulleaide = 44;
}

if(e.getSource() == affichagenoms)
{
    if(!affichernoms)
    {
        bulleaide = 36;
    }
    else
    {
        bulleaide = 40;
    }
}
if(e.getSource() == affichagelabels)
{
    if(!afficherlabels)
    {
```

```
        bulleaide = 37;
    }
    else
    {
        bulleaide = 41;
    }
}

if(e.getSource() == insmasse | e.getSource() == nomasse)
{
    if(massetot > 0)
    {
        if(!startunefois)
        {
            bulleaide = 50;
        }
        else
        {
            bulleaide = 51;
        }
    }
}

if(e.getSource() == labelmasse | e.getSource() == textemasse)
{
    if(massetot>0)
    {
        if(!startunefois)
        {
            bulleaide = 52;
        }
        else
        {
            bulleaide = 53;
        }
    }
}

if(e.getSource() == labelx | e.getSource() == textex)
{
    if(massetot>0)
    {
        if(!startunefois)
        {
            bulleaide = 54;
        }
        else
        {
            bulleaide = 55;
        }
    }
}
```

```
if(e.getSource() == labely | e.getSource() == textey)
{
    if(massetot>0)
    {
        if(!startunefois)
        {
            bulleaide = 56;
        }
        else
        {
            bulleaide = 57;
        }
    }
}

if(e.getSource() == labelvx | e.getSource() == textevx)
{
    if(massetot>0)
    {
        if(!startunefois)
        {
            if(!stationnaire[masseselectionnee])
            {
                bulleaide = 58;
            }
            else
            {
                bulleaide = 50;
            }
        }
        else
        {
            bulleaide = 59;
        }
    }
}

if(e.getSource() == labelvy | e.getSource() == textevy)
{
    if(massetot>0)
    {
        if(!startunefois)
        {
            if(!stationnaire[masseselectionnee])
            {
                bulleaide = 61;
            }
            else
            {
                bulleaide = 63;
            }
        }
        else
        {
```

```

        bulleaide = 62;
    }
}

if(e.getSource() == casestationnaire)
{
    if(massetot>0)
    {
        bulleaide = 64;
    }
}

if(e.getSource() == listemasses)
{
    if(massetot>0)
    {
        bulleaide = 65;
    }
    else
    {
        bulleaide = 66;
    }
}
}

repaint();
}

//Méthode invoquée lorsque la souris sort du cadre de l'applet (ou de tout
autre objet)
public void mouseExited(MouseEvent e)
{
    repaint();
}

//Méthode qui actualise les TextFields (pour savoir s'ils doivent être
"editable" ou pas, quelle valeur ils doivent comporter, etc...)
public void actualiserTF()
{
    if(massetot>0)
    {
        textemasse.setText(" "+massefixe[masseselectionnee]);
        textex.setText(" "+xfixe[masseselectionnee]);
        textey.setText(" "+yfixe[masseselectionnee]);
        textevx.setText(" "+vxfixe[masseselectionnee]);
        textevy.setText(" "+vyfixe[masseselectionnee]);

        if(menuvisible)
        {
            nomasse.setVisible(true);
            nomasse.setText(" "+(masseselectionnee+1));
            insmasse.setVisible(true);
        }
    }
}

```

```

        casestationnaire.setState(stationnairefixe[masseselectionnee]);
        listemasses.select(masseselectionnee);
    }
    else
    {
        textemasse.setText("");
        textex.setText("");
        textey.setText("");
        casestationnaire.setState(false);
        textevx.setText("");
        textevy.setText("");
        nomasse.setVisible(false);
        insmasse.setVisible(false);
    }

    if (casestationnaire.getState())
    {
        textevx.setEditable(false);
        textevy.setEditable(false);
    }
    else
    {
        textevx.setEditable(true);
        textevy.setEditable(true);
    }

    if (!startunefois && massetot>0 && click == 0)
    {
        insmasse.setText("Modifier la masse no");
        textemasse.setEditable(true);
        textex.setEditable(true);
        textey.setEditable(true);
        textevx.setEditable(!stationnairefixe[masseselectionnee]);
        textevy.setEditable(!stationnairefixe[masseselectionnee]);
        casestationnaire.setEnabled(true);
    }
    else
    {
        insmasse.setText("                Masse no");
        textemasse.setEditable(false);
        textex.setEditable(false);
        textey.setEditable(false);
        textevx.setEditable(false);
        textevy.setEditable(false);
        casestationnaire.setEnabled(false);
    }
}

//Méthode qui gère l'état des boutons (pour savoir s'ils doivent être
"enabled" ou pas, etc...)
public void actualiserboutons()
{
    if (startunefois)

```

```

{
    nvllemasse.setEnabled(false);
    delete.setEnabled(false);
    deleteall.setEnabled(false);
    reset.setEnabled(true);
}
else
{
    if (massetot != nbmax && click == 0)
    {
        nvllemasse.setEnabled(true);
    }
    else
    {
        nvllemasse.setEnabled(false);
    }
    reset.setEnabled(false);

    if (massetot >= 2 && click == 0)
    {
        boutonmarchearret.setEnabled(true);
    }
    else
    {
        boutonmarchearret.setEnabled(false);
    }

    if (massetot >=1 && click == 0)
    {
        if (selectionnee)
        {
            delete.setEnabled(true);
        }
        else
        {
            delete.setEnabled(false);
        }
        deleteall.setEnabled(true);
    }
    else
    {
        delete.setEnabled(false);
        deleteall.setEnabled(false);
    }
}
if (stop)
{
    boutonmarchearret.setLabel("D\u00e9marrer");
}
else
{
    boutonmarchearret.setLabel("Arr\u00eater");
}
}

```

```
}

//Méthode qui gère les cases à cocher et les listes
public void itemStateChanged(ItemEvent e)
{

    if (e.getSource() == listemasses)
    {
        selectionnee = true;
        if(click == 0)
        {
            masseselectionnee = listemasses.getSelectedIndex();
        }
    }

    if (e.getSource() == choixraj)
    {
        traj = choixraj.getSelectedIndex();
    }

    if (e.getSource() == choixcouleur)
    {
        couleur = choixcouleur.getSelectedIndex();
    }

    if (e.getSource() == affichagegrillage)
    {
        affichergrillage = affichagegrillage.getState();

        if(!affichergrillage)
        {
            bulleaide = 34;
        }
        else
        {
            bulleaide = 38;
        }
    }

    if (e.getSource() == affichageforces)
    {
        afficherforces = affichageforces.getState();

        if(!afficherforces)
        {
            bulleaide = 35;
        }
        else
        {
            bulleaide = 39;
        }
    }

    if (e.getSource() == affichagevitesses)
```

```
{
    affichervitesses = affichagevitesses.getState();

    if(!affichervitesses)
    {
        bulleaide = 42;
    }
    else
    {
        bulleaide = 43;
    }
}

if (e.getSource() == affichagenoms)
{
    affichernoms = affichagenoms.getState();

    if(!affichernoms)
    {
        bulleaide = 36;
    }
    else
    {
        bulleaide = 40;
    }
}

if (e.getSource() == affichagelabels)
{
    afficherlabels = affichagelabels.getState();

    if(!afficherlabels)
    {
        bulleaide = 37;
    }
    else
    {
        bulleaide = 41;
    }
}

if (e.getSource() == casestationnaire)
{
    stationnairefixe[masseselectionnee] = casestationnaire.getState();
    stationnaire[masseselectionnee] =
stationnairefixe[masseselectionnee];
    vxfixe[masseselectionnee] = 0;
    vyfixe[masseselectionnee] = 0;
    vx[masseselectionnee] = vxfixe[masseselectionnee];
    vy[masseselectionnee] = vyfixe[masseselectionnee];
}

actualiserTF();
actualiserbouttons();
```

```

        repaint();
    }

    //Méthode qui gère les barres de défilement (les "ascenseurs")
    public void adjustmentValueChanged(AdjustmentEvent e)
    {
        if (e.getSource() == selectvitesse)
        {
            vitesse = e.getValue();
            labelvitesse.setText("Vitesse : "+vitesse+"x");
        }

        if (e.getSource() == zoomsb)
        {
            zoom = e.getValue();
            labelzoom.setText("Zoom : "+(int)zoom+"%");
            zoom = zoom/100;
        }
        repaint();
    }

    //Méthode qui gère les boutons (cette méthode est également invoquée
    lorsque la touche "enter" est frappée dans un textfield)
    public void actionPerformed(ActionEvent e)
    {
        if (massetot > 0)
        {
            // Si l'événement vient d'un champ de texte, le focus est donné
            au textfield suivant
            if (e.getSource() == textemasse && !startunefois)
            {
                textex.requestFocus();
            }
            if (e.getSource() == textex && !startunefois)
            {
                textey.requestFocus();
            }
            if (e.getSource() == textey && !startunefois)
            {
                if(!stationnaire[masseselectionnee])
                {
                    textevx.requestFocus();
                }
                else
                {
                    textemasse.requestFocus();
                }
            }
            if (e.getSource() == textevx && !startunefois)
            {
                textevy.requestFocus();
            }
            if (e.getSource() == textevy && !startunefois)
            {

```

```

        textemasse.requestFocus();
    }
}

if (e.getSource() == boutonmarchearret)
{
    if (stop)
    {
        stop = false;

        bulleaide = 22;
    }
    else
    {
        stop = true;

        bulleaide = 23;
    }
    startunefois = true;
}

//Lorsque le boutonmenu est pressé on cache tous les éléments du menu
et on pose hauteurdonnees = 30
if(e.getSource() == boutonmenu)
{
    if(hauteurdonnees>30)
    {
        hauteurdonnees = 30;

        menuvisible = false;

        boutonmenu.setLabel("Montrer le Menu");

        insmasse.setVisible(false);
        nomasse.setVisible(false);
        labelmasse.setVisible(false);
        labelx.setVisible(false);
        labely.setVisible(false);
        labelvx.setVisible(false);
        labelvy.setVisible(false);
        labelvitesse.setVisible(false);
        labelzoom.setVisible(false);
        labelcouleur.setVisible(false);
        nvllemasse.setVisible(false);
        delete.setVisible(false);
        deleteall.setVisible(false);
        textemasse.setVisible(false);
        textex.setVisible(false);
        textey.setVisible(false);
        textevx.setVisible(false);
        textevy.setVisible(false);
        casestationnaire.setVisible(false);
        affichagegrillage.setVisible(false);
        affichagenoms.setVisible(false);
    }
}

```

```

affichageforces.setVisible(false);
affichagevitesses.setVisible(false);
affichervecteurs.setVisible(false);
affichagelabels.setVisible(false);
listemasses.setVisible(false);
labeltraj.setVisible(false);
choixtraj.setVisible(false);
choixcouleur.setVisible(false);
zoomsb.setVisible(false);
selectvitesse.setVisible(false);

reset.setBounds(longueur/2, hauteur-hauteurdonnees, 80, 30);
boutonmarchearret.setBounds(longueur/2-80, hauteur-
hauteurdonnees, 80, 30);
    boutonmenu.setBounds(longueur-100, hauteur-
hauteurdonnees, 100, 30);
    }
    else //On rétablit tout lorsque l'utilisateur presse à
nouveau le boutonmenu
    {
        hauteurdonnees = 230;

        menuvisible = true;

        boutonmenu.setLabel("Cacher le Menu");

        insmasse.setVisible(true);
        nomasse.setVisible(true);
        labelmasse.setVisible(true);
        labelx.setVisible(true);
        labely.setVisible(true);
        labelvx.setVisible(true);
        labelvy.setVisible(true);
        labelvitesse.setVisible(true);
        labelzoom.setVisible(true);
        labelcouleur.setVisible(true);
        nvllemasse.setVisible(true);
        delete.setVisible(true);
        deleteall.setVisible(true);
        textemasse.setVisible(true);
        textex.setVisible(true);
        textey.setVisible(true);
        textevx.setVisible(true);
        textevy.setVisible(true);
        casestationnaire.setVisible(true);
        affichagegrillage.setVisible(true);
        affichagenoms.setVisible(true);
        affichageforces.setVisible(true);
        affichagevitesses.setVisible(true);
        affichervecteurs.setVisible(true);
        affichagelabels.setVisible(true);
        listemasses.setVisible(true);
        labeltraj.setVisible(true);
        choixtraj.setVisible(true);

```

```

        choixcouleur.setVisible(true);
        zoomsb.setVisible(true);
        selectvitesse.setVisible(true);

        reset.setBounds(longueur/2,hauteur-hauteurdonnees,80,30);
        boutonmarchearret.setBounds(longueur/2-80,hauteur-
hauteurdonnees,80,30);
        boutonmenu.setBounds(longueur-100,hauteur-
hauteurdonnees,100,30);
    }
}

// lorsque le bouton "Nouvelle masse" est pressé, on augmente
massetot, et on "initialise" les valeurs de chaque paramètre (masse, x, y,
...)
if (e.getSource() == nvllemasse)
{
    if (massetot < nbmax)
    {
        massetot++;
        masseselectionnee = massetot-1;

        x[masseselectionnee] = xfixe[masseselectionnee];
        y[masseselectionnee] = yfixe[masseselectionnee];
        masse[masseselectionnee] = massefixe[masseselectionnee];
        donnervaleur_posy(yfixe[masseselectionnee]);
        donnervaleur_posx(xfixe[masseselectionnee]);
        donnervaleur_masse(massefixe[masseselectionnee]);
        donnervaleur_vitessex(vxfixe[masseselectionnee]);
        donnervaleur_vitessey(vyfixe[masseselectionnee]);

        stationnaire[masseselectionnee] =
stationnairefixe[masseselectionnee];
    }
    listemasses.removeAll();

    for ( i = 0; i < massetot; i++)
    {
        listemasses.add("masse "+ (i+1),i);
    }

    selectionnee = true;
}

// lorsque le bouton "Effacer tout" est pressé, on dit que massetot =
0
if (e.getSource() == deleteall)
{
    massetot = 0;
    masseselectionnee = 0;

    listemasses.removeAll();
}

```

```

// voici ce qu'il se passe lorsque le bouton "effacer" est pressé
if (e.getSource() == delete)
{
    if (!startunefois && selectionnee && massetot > 0)
    {
        //Lorsqu'on supprime une masse, les propriétés des masses sont
"décalées",
        //c'est pourquoi on utilise des variables de "transition".
        transmasse = massefixe[masseselectionnee];
        transstat = stationnairefixe[masseselectionnee];
        transx = xfixe[masseselectionnee];
        transy = yfixe[masseselectionnee];
        transvx = vxfixe[masseselectionnee];
        transvy = vyfixe[masseselectionnee];

        for (i = masseselectionnee; i < massetot; i++)
        {
            if (i == massetot-1){
                massefixe[i] = transmasse;
                stationnairefixe[i]= transstat;
                xfixe[i] = transx;
                yfixe[i] = transy;
                vxfixe[i] = transvx;
                vyfixe[i] = transvy;
            }
            else
            {
                massefixe[i] = massefixe[i+1];
                stationnairefixe[i]=stationnaire[i+1];
                xfixe[i] = xfixe[i+1];
                yfixe[i] = yfixe[i+1];
                vxfixe[i] = vxfixe[i+1];
                vyfixe[i] = vyfixe[i+1];
            }
            masse[i] = massefixe[i];
            stationnaire[i] = stationnairefixe[i];
            x[i]=xfixe[i];
            y[i]=yfixe[i];
            vx[i]=vxfixe[i];
            vy[i]=vyfixe[i];
        }

        massetot--;

        if(masseselectionnee>=massetot) masseselectionnee = massetot-
1;

        listemasses.removeAll();

        for ( i = 0; i < massetot; i++)
        {
            listemasses.add("masse "+ (i+1),i);
        }
    }
}

```

```

    }

    // lorsque le bouton "Réinitialiser" est pressé, on redonne à chaque
    paramètre des masses sa valeur d'origine (valeur fixe)
    if (e.getSource() == reset)
    {
        if (startunefois)
        {
            for(i = 0; i < dernierspoints.length; i++)
            {
                dernierspoints[i].removeAllElements();
            }

            for(i = 0; i < nbmax; i++)
            {
                indexderpoints[i] = 0;
            }

            collisionpoints.removeAllElements();

            for (i = 0; i < massetot; i++)
            {
                masse[i] = massefixe[i];
                stationnaire[i] = stationnairefixe[i];
                x[i]=xfixe[i];
                y[i]=yfixe[i];
                vx[i]=vxfixe[i];
                vy[i]=vyfixe[i];
            }

            stop = true;
            startunefois = false;
            selectionnee = true;

            for (i = 0; i < nbmax; i++)
            {
                nommasse[i] = i;
            }

            //Remise à zéro de la période
            n = 0;

            boutonmarchearret.setLabel("D\u00e9marrer");
        }
    }

    actualiserTF();
    actualiserbouttons();
    repaint();
}
}

```

Le 28 novembre 2003

Mathieu Ackermann et Adrien Briod

*Tot. Lignes de code : 3'442*