PROJET JAVA Lours_Dorkenoo_Poinas

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Balls Class Reference

This class represents our balls with point table.

Public Member Functions

- Balls (Point[ ] balls)

  on construit une copie de notre tableau de balles pour en créer une nouvelle instance, puis on copie dans le tableau de points initiaux.
- Point[ ] getBalls ()

  on retourne notre tableau de balles
- Point[ ] getInitBalls ()

  on retourne notre tableau de balles initiales
- void translate (int dx, int dy)

  translate les balles du montant indiqué sur chaque coordonnée
- void reInit ()

  remet les balles comme à l'origine, d'où l'utilité de mes deux tableaux de balles
- String toString ()

  représentation de notre classe sous forme de string

4.1.1 Detailed Description

This class represents our balls with point table.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Balls()

```
Balls.Balls (
            Point[] balls )  [inline]
```

on construit une copie de notre tableau de balles pour en créer une nouvelle instance, puis on copie dans le tableau de points initiaux.

Parameters

balls	notre tableau de coordonnées de balles

4.1.3 Member Function Documentation

4.1.3.1 getBalls()

```
Point [] Balls.getBalls ( ) [inline]
```

on retourne notre tableau de balles

4.1.3.2 getInitBalls()

```
Point [] Balls.getInitBalls ( ) [inline]
```

on retourne notre tableau de balles initiales

4.1.3.3 reInit()

```
void Balls.reInit ( ) [inline]
```

remet les balles comme à l'origine, d'où l'utilité de mes deux tableaux de balles

4.1.3.4 toString()

```
String Balls.toString ( ) [inline]
```

représentation de notre classe sous forme de string

Returns

notre chaine clean

4.1.3.5 translate()

```
void Balls.translate (
             int dx,
             int dy ) [inline]
```

translate les balles du montant indiqué sur chaque coordonnée

Parameters

dx	x coord
dy	y coord

The documentation for this class was generated from the following file:

- src/Balls.java

4.2 BallsSimulator Class Reference

This class represents a balls simulator that implements the Simulable interface.

Inheritance diagram for BallsSimulator:

4.3 Boids Class Reference

Boids.java.

Inheritance diagram for Boids:

Public Member Functions

- Boids (int x, int y, int vx, int vy, int orientation, int taille_x, int taille_y)

  Constructs a new Boids object with the given initial position, velocity, orientation, and window size.
- int getOrientation ()

  Returns the orientation of the boid.
- void reset ()

  Resets the boid's position, velocity, and orientation to their initial values.
- void setOrientation (int orientation)

  Sets the orientation of the boid.
- int[ ] getPosition ()

  Returns the current position of the boid.
- int[ ] getVitesse ()

  Returns the current velocity of the boid.
- void update ()

  Updates the orientation and position of the boid.
- int distance (Boids other)

  Calculates the distance between the current boid and another boid.
- void separate (Boids[ ] list_boids, int distance_separation)

  Applies the separation rule to the boid based on its neighbors.
- void align (Boids[ ] boids, int distance_alignement)

  Applies the alignment rule to the boid based on its neighbors.
- void cohere (Boids[ ] boids, int distance_essaim)

  Applies the cohesion rule to the boid based on its neighbors.

Protected Member Functions

- void update_orientation ()

  Updates the orientation of the boid based on its velocity.
- void update_position ()

  Updates the position of the boid based on its velocity and window size.
- void update_vitesse (int[ ] vitesse2)

  Updates the velocity of the boid based on a given force.

Protected Attributes

- int[ ] position
- int[ ] init_position
- int[ ] init_vitesse
- int[ ] vitesse
- int orientation
- int init_orientation
- int taille_fen_X
- int taille_fen_Y

4.3.1 Detailed Description

Boids.java.

Summary: This class represents a boid object, which is a simulated bird-like creature that exhibits flocking behavior. It contains methods to update the boid's position and orientation, and to apply flocking rules such as separation, alignment, and cohesion. The class also includes methods to calculate the distance between boids and to reset the boid's state.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Boids()

```
Boids.Boids (
              int x,
              int y,
              int vx,
              int vy,
              int orientation,
              int taille_x,
              int taille_y )  [inline]
```

Constructs a new Boids object with the given initial position, velocity, orientation, and window size.

Parameters

x	The initial x-coordinate of the boid's position
y	The initial y-coordinate of the boid's position
vx	The initial x-component of the boid's velocity
vy	The initial y-component of the boid's velocity
orientation	The initial orientation of the boid
taille_x	The width of the window

4.3.3 Member Function Documentation

4.3.3.1 align()

```
void Boids.align (
             Boids[] boids,
             int distance_alignement )  [inline]
```

Applies the alignment rule to the boid based on its neighbors.

Parameters

boids	An array of boids representing the neighbors of the current boid
distance_alignement	The distance threshold for alignment

4.3.3.2 cohere()

```
void Boids.cohere (
             Boids[] boids,
             int distance_essaim )  [inline]
```

Applies the cohesion rule to the boid based on its neighbors.

Parameters

boids	An array of boids representing the neighbors of the current boid
distance_essaim	The distance threshold for cohesion

4.3.3.3 distance()

```
int Boids.distance (
             Boids other )  [inline]
```

Calculates the distance between the current boid and another boid.

Parameters

other	The other boid to calculate the distance to

Returns

The distance between the current boid and the other boid

4.3.3.4 getOrientation()

```
int Boids.getOrientation ( )  [inline]
```

Returns the orientation of the boid.

Returns

The orientation of the boid

4.3.3.5 getPosition()

```
int [] Boids.getPosition ( )  [inline]
```

Returns the current position of the boid.

Returns

The position of the boid as an array of x and y coordinates

4.3.3.6 getVitesse()

```
int [] Boids.getVitesse ( )  [inline]
```

Returns the current velocity of the boid.

Returns

The velocity of the boid as an array of x and y components

4.3.3.7 reset()

```
void Boids.reset ( )  [inline]
```

Resets the boid's position, velocity, and orientation to their initial values.

4.3.3.8 separate()

```
void Boids.separate (
            Boids[] list_boids,
            int distance_separation )  [inline]
```

Applies the separation rule to the boid based on its neighbors.

Parameters

list_boids	An array of boids representing the neighbors of the current boid
distance_separation	The distance threshold for separation

4.3.3.9 setOrientation()

```
void Boids.setOrientation (
             int orientation )  [inline]
```

Sets the orientation of the boid.

Parameters

orientation	The new orientation of the boid

4.3.3.10 update()

```
void Boids.update ( )  [inline]
```

Updates the orientation and position of the boid.

4.3.3.11 update_orientation()

```
void Boids.update_orientation ( )  [inline], [protected]
```

Updates the orientation of the boid based on its velocity.

4.3.3.12 update_position()

```
void Boids.update_position ( )  [inline], [protected]
```

Updates the position of the boid based on its velocity and window size.

4.3.3.13 update_vitesse()

```
void Boids.update_vitesse (
             int[] vitesse2 )  [inline], [protected]
```

Updates the velocity of the boid based on a given force.

Parameters

vitesse2	The force to be applied to the velocity of the boid

4.3.4 Member Data Documentation

4.3.4.1 init_orientation

```
int Boids.init_orientation  [protected]
```

4.3.4.2 init_position

```
int [] Boids.init_position  [protected]
```

4.3.4.3 init_vitesse

```
int [] Boids.init_vitesse  [protected]
```

4.3.4.4 orientation

```
int Boids.orientation  [protected]
```

4.3.4.5 position

```
int [] Boids.position  [protected]
```

4.3.4.6 taille_fen_X

```
int Boids.taille_fen_X  [protected]
```

4.3.4.7 taille_fen_Y

```
int Boids.taille_fen_Y  [protected]
```

4.3.4.8 vitesse

```
int [ ] Boids.vitesse  [protected]
```

The documentation for this class was generated from the following file:

- src/Boids.java

4.4 BoidsEvent Class Reference

BoidsEvent class represents an event that updates the behavior of a group of boids.

Inheritance diagram for BoidsEvent:

Collaboration diagram for BoidsEvent:

Public Member Functions

- BoidsEvent (int date, SpecialBoids[ ] Boids, Color colorClass, GUISimulator gui, EventManager Boids↩ Manager, SpecialBoids[ ] allBoids)

  Constructs a BoidsEvent object with the specified parameters.
- void execute ()

  Executes the event by updating the behavior of the boids.

4.4.1 Detailed Description

BoidsEvent class represents an event that updates the behavior of a group of boids.

It contains methods to separate, align, and cohere the boids, as well as update their positions. The class also handles the scheduling of new events based on the type of boids.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 BoidsEvent()

```
BoidsEvent.BoidsEvent (
            int date,
            SpecialBoids[ ] Boids,
            Color colorClass,
            GUISimulator gui,
            EventManager BoidsManager,
            SpecialBoids[ ] allBoids )  [inline]
```

Constructs a BoidsEvent object with the specified parameters.

Parameters

date	the date of the event
Boids	the boids to be processed
colorClass	the color class of the boids
gui	the GUISimulator object for visualization
BoidsManager	the EventManager object for scheduling events
allBoids	all the boids in the simulation

4.4.3 Member Function Documentation

4.4.3.1 execute()

```
void BoidsEvent.execute ( )  [inline]
```

Executes the event by updating the behavior of the boids.

If the boids are of type "poisson", a new BoidsEvent is scheduled every 1 unit of time. If the boids are of type "requin", a new BoidsEvent is scheduled every 2 units of time.

Reimplemented from Event.

The documentation for this class was generated from the following file:

- src/BoidsEvent.java

4.5 BoidsSimulator Class Reference

BoidsSimulator.java.

Inheritance diagram for BoidsSimulator:

Collaboration diagram for BoidsSimulator:

Public Member Functions

- BoidsSimulator (SpecialBoids[ ] list_Boids, GUISimulator gui, EventManager BoidsEvent)

  Constructor for BoidsSimulator class.
- void restart ()

  Restarts the simulation by resetting the position and orientation of each boid and updating the graphical elements.
- void next ()

  Advances the simulation to the next iteration by updating the boids' positions and redrawing them on the GUI.

4.5.1 Detailed Description

BoidsSimulator.java.

This file contains the implementation of a Boids simulator. It uses the GUISimulator library to display graphical elements representing boids. The simulator allows for restarting the simulation and advancing to the next iteration.

The BoidsSimulator class defines methods for drawing boids, restarting the simulation, and advancing to the next iteration. It also contains a constructor that initializes the simulator with a list of boids, a graphical user interface, and an event manager.

The simulator uses the list of boids to draw each boid on the GUI. The restart() method resets the position and orientation of each boid, and updates the graphical elements accordingly. The next() method advances the simulation to the next iteration by updating the boids' positions and redrawing them on the GUI.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 BoidsSimulator()

```
BoidsSimulator.BoidsSimulator (
            SpecialBoids[] list_Boids,
            GUISimulator gui,
            EventManager BoidsEvent )  [inline]
```

Constructor for BoidsSimulator class.

Parameters

list_Boids	an array of SpecialBoids representing the boids in the simulation
gui	the GUISimulator object used for displaying graphical elements
BoidsEvent	the EventManager object used for managing events in the simulation

4.5.3 Member Function Documentation

4.5.3.1 next()

```
void BoidsSimulator.next ( )  [inline]
```

Advances the simulation to the next iteration by updating the boids' positions and redrawing them on the GUI.

4.5.3.2 restart()

```
void BoidsSimulator.restart ( )  [inline]
```

Restarts the simulation by resetting the position and orientation of each boid and updating the graphical elements.

The documentation for this class was generated from the following file:

- src/BoidsSimulator.java

4.6 Cell Class Reference

This is a Java program that defines a Cell class.

Inheritance diagram for Cell:

Collaboration diagram for Cell:

Public Member Functions

- Cell (int size_x, int size_y)

  Constructs a Cell object with the given size of the grid.
- int getSize_y ()
- int getSize_x ()
- void InitConfigFirst ()

  Initialize our first config to make a copy which will be useful while reseting.
- void Init_cells ()

  Initialize isAlive and alive_before.
- Point getCellule (int i)

  Returns the Point object at the specified index of the cells array.
- int getlength ()

  Returns the total number of cells.
- int[ ] getIsAlive ()

  Returns the array containing the status of each cell (alive or dead).
- int[ ] getAlive_before ()

  This method returns the array of alive cells before the current state.
- void setBoolean_coord (int bool, int coord_x, int coord_y)

  Sets the boolean value at the specified coordinates in the Cell.
- void setBoolean (int bool, int i)

  Sets the boolean value at the specified index.
- void setnewEtapeConway ()

  This code updates the state of each cell in a Conway's Game of Life simulation, based on the rules of the game.
- String toString ()

  Cell.java.

Protected Member Functions

- boolean isNeighbor (Point cellule1, Point cellule2)

  This code defines a method called "isNeighbor" that checks if two given points are neighbors.

Protected Attributes

- Point[ ] cells
- int[ ] isAlive
- int[ ] alive_before
- int[ ] first_config
- int size_x
- int size_y

4.6.1 Detailed Description

This is a Java program that defines a Cell class.

The Cell class represents a cell in a grid. It contains methods for initializing the grid, setting the state of cells, calculating the number of neighbors for each cell, and updating the state of cells based on the rules of Conway's Game of Life.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Cell()

```
Cell.Cell (
             int size_x,
             int size_y )  [inline]
```

Constructs a Cell object with the given size of the grid.

Initializes the cells array, isAlive array, and alive_before array. Throws an IllegalArgumentException if the lengths of the arrays are not the same. Each cell is assigned a Point object and initialized with default state values.

Parameters

size← _x	the number of cells in the x-direction
size← _y	the number of cells in the y-direction

4.6.3 Member Function Documentation

4.6.3.1 getAlive_before()

```
int [ ] Cell.getAlive_before ( )  [inline]
```

This method returns the array of alive cells before the current state.

4.6.3.2 getCellule()

```
Point Cell.getCellule (
            int i )  [inline]
```

Returns the Point object at the specified index of the cells array.

Parameters

i	The index of the Point object to be returned.

Returns

The Point object at the specified index.

4.6.3.3 getIsAlive()

```
int [] Cell.getIsAlive ( )  [inline]
```

Returns the array containing the status of each cell (alive or dead).

Returns

The array containing the status of each cell.

4.6.3.4 getlength()

```
int Cell.getlength ( )  [inline]
```

Returns the total number of cells.

Returns

The total number of cells.

4.6.3.5 getSize_x()

```
int Cell.getSize_x ( )  [inline]
```

Returns

size_x

4.6.3.6 getSize_y()

```
int Cell.getSize_y ( )  [inline]
```

Returns

size_y

4.6.3.7 Init_cells()

```
void Cell.Init_cells ( )  [inline]
```

Initialize isAlive and alive_before.

Reimplemented in Schelling.

4.6.3.8 InitConfigFirst()

```
void Cell.InitConfigFirst ( )  [inline]
```

Initialize our first config to make a copy which will be useful while reseting.

4.6.3.9 isNeighbor()

```
boolean Cell.isNeighbor (
             Point cellule1,
             Point cellule2 )  [inline], [protected]
```

This code defines a method called "isNeighbor" that checks if two given points are neighbors.

It calculates the absolute difference in x and y coordinates between the two points and checks if either the x or y difference is equal to 1. If so, it also checks if the other difference is less than or equal to 1. The method returns true if the points are neighbors, and false otherwise.

Parameters

cellule1	first cell
cellule2	second cell

Returns

a boolean which indicate if cellule1/2 are neighbors.

4.6.3.10 setBoolean()

```
void Cell.setBoolean (
            int bool ,
            int i )  [inline]
```

Sets the boolean value at the specified index.

Parameters

bool	the boolean value to set
i	the index to set the value at

4.6.3.11 setBoolean_coord()

```
void Cell.setBoolean_coord (
            int bool ,
            int coord_x,
            int coord_y )  [inline]
```

Sets the boolean value at the specified coordinates in the Cell.

Parameters

bool	the boolean value to set
coord←↩ _x	the x-coordinate of the cell
coord←↩ _y	the y-coordinate of the cell

4.6.3.12 setnewEtapeConway()

```
void Cell.setnewEtapeConway ( )  [inline]
```

This code updates the state of each cell in a Conway's Game of Life simulation, based on the rules of the game.

4.6.3.13 toString()

```
String Cell.toString ( )  [inline]
```

Cell.java.

This code represents a Java class that defines a toString() method to generate a string representation of an array of cells. The method iterates through the cells array and constructs a string containing the x and y coordinates along with the alive status of each cell. The final string is returned as the result.

4.6.4 Member Data Documentation

4.6.4.1 alive_before

```
int [ ] Cell.alive_before  [protected]
```

4.6.4.2 cells

```
Point [ ] Cell.cells  [protected]
```

4.6.4.3 first_config

```
int [ ] Cell.first_config  [protected]
```

4.6.4.4 isAlive

```
int [ ] Cell.isAlive  [protected]
```

4.6.4.5 size_x

```
int Cell.size_x  [protected]
```

4.6.4.6 size_y

```
int Cell.size_y  [protected]
```

The documentation for this class was generated from the following file:

- src/Cell.java

4.7 CellSimulator Class Reference

This class implements the Simulable interface and simulates the behavior of a group of cells using Conway's Game of Life rules.

Inheritance diagram for CellSimulator:

Collaboration diagram for CellSimulator:

Public Member Functions

- CellSimulator (Cell cells, GUISimulator gui)

  Constructs a CellSimulator object with the given Cell and GUISimulator objects.
- Cell getCells ()

  Returns the Cell object used in the simulation.
- void next ()

  Advances the simulation by one step, updating the state of the cells and redrawing them on the GUI.
- void restart ()

  Restarts the simulation by resetting the state of the cells and redrawing them on the GUI.
- void setGraphicCell ()

  Draws the cells on the GUI using rectangles, with live cells represented by blue rectangles and dead cells represented by white rectangles.

4.7.1 Detailed Description

This class implements the Simulable interface and simulates the behavior of a group of cells using Conway's Game of Life rules.

The class takes in a Cell object and a GUISimulator object as parameters and uses them to display the state of the cells.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 CellSimulator()

```
CellSimulator.CellSimulator (
              Cell cells,
              GUISimulator gui )  [inline]
```

Constructs a CellSimulator object with the given Cell and GUISimulator objects.

Parameters

cells	the Cell object representing the group of cells
gui	the GUISimulator object used to display the state of the cells

4.7.3 Member Function Documentation

4.7.3.1 getCells()

```
Cell CellSimulator.getCells ( )  [inline]
```

Returns the Cell object used in the simulation.

Returns

the Cell object used in the simulation

4.7.3.2 next()

```
void CellSimulator.next ( )  [inline]
```

Advances the simulation by one step, updating the state of the cells and redrawing them on the GUI.

4.7.3.3 restart()

```
void CellSimulator.restart ( )  [inline]
```

Restarts the simulation by resetting the state of the cells and redrawing them on the GUI.

4.7.3.4 setGraphicCell()

```
void CellSimulator.setGraphicCell ( )  [inline]
```

Draws the cells on the GUI using rectangles, with live cells represented by blue rectangles and dead cells represented by white rectangles.

The documentation for this class was generated from the following file:

- src/CellSimulator.java

4.8 Event Class Reference

This is an abstract class representing an event.

Inheritance diagram for Event:

Public Member Functions

- Event (long date)

  Constructs an Event object with the given date.
- long getDate ()

  Gets the date of the event.
- abstract void execute ()

  Executes the event.

4.8.1 Detailed Description

This is an abstract class representing an event.

It contains a date field and provides methods to get the date and execute the event.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Event()

```
Event.Event (
            long date )  [inline]
```

Constructs an Event object with the given date.

Parameters

date	the date of the event

4.8.3 Member Function Documentation

4.8.3.1 execute()

```
abstract void Event.execute ( )  [abstract]
```

Executes the event.

Reimplemented in MessageEvent, and BoidsEvent.

4.8.3.2 getDate()

```
long Event.getDate ( )  [inline]
```

Gets the date of the event.

Returns

the date of the event

The documentation for this class was generated from the following file:

- src/Event.java

4.9 EventManager Class Reference

This class manages a priority queue of events and executes them in chronological order.

Public Member Functions

- EventManager ()
- void addEvent (Event event)

  Adds an event to the priority queue.
- void next ()

  Executes the next event in the queue that is scheduled to occur.
- boolean isFinished ()

  Checks if the event queue is empty.
- void restart ()

  Restarts the event manager by resetting the current date and clearing the event queue.

4.9.1 Detailed Description

This class manages a priority queue of events and executes them in chronological order.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 EventManager()

```
EventManager.EventManager ( )  [inline]
```

4.9.3 Member Function Documentation

4.9.3.1 addEvent()

```
void EventManager.addEvent (
             Event event )  [inline]
```

Adds an event to the priority queue.

Parameters

event	Event to be added

4.9.3.2 isFinished()

```
boolean EventManager.isFinished ( )  [inline]
```

Checks if the event queue is empty.

Returns

True if the queue is empty, false otherwise

4.9.3.3 next()

```
void EventManager.next ( )  [inline]
```

Executes the next event in the queue that is scheduled to occur.

4.9.3.4 restart()

```
void EventManager.restart ( )  [inline]
```

Restarts the event manager by resetting the current date and clearing the event queue.

The documentation for this class was generated from the following file:

- src/EventManager.java

4.10 Immigration Class Reference

This code defines a class called "Immigration" that represents a simulation of a cellular automaton with multiple states.

Inheritance diagram for Immigration:

Collaboration diagram for Immigration:

Public Member Functions

- Immigration (int size_x, int size_y, int nb_etats)

  Constructs an Immigration object with the specified size of the grid and number of states.
- int getNb_etats ()

  Returns the number of states in the simulation.
- void setnewEtapeImmigration ()

  Updates the state of the cells in the simulation based on the Immigration rules.
- void setBoolean_coord_Immi (int bool, int coord_x, int coord_y)

  Sets the state of a specific cell at the given coordinates in the Immigration simulation.

Additional Inherited Members

4.10.1 Detailed Description

This code defines a class called "Immigration" that represents a simulation of a cellular automaton with multiple states.

It extends the "Cell" class and provides methods to set the number of states, count alive neighbors, update the state of the cells, and set the state of a specific cell.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Immigration()

```
Immigration.Immigration (
              int size_x,
              int size_y,
              int nb_etats )  [inline]
```

Constructs an Immigration object with the specified size of the grid and number of states.

Parameters

size_x	the number of cells in the x-direction
size_y	the number of cells in the y-direction
nb_etats	the number of states in the simulation

4.10.3 Member Function Documentation

4.10.3.1 getNb_etats()

```
int Immigration.getNb_etats ( )  [inline]
```

Returns the number of states in the simulation.

Returns

the number of states

4.10.3.2 setBoolean_coord_Immi()

```
void Immigration.setBoolean_coord_Immi (
             int bool ,
             int coord_x,
             int coord_y )  [inline]
```

Sets the state of a specific cell at the given coordinates in the Immigration simulation.

Parameters

bool	the state to set for the cell
coord↩_x	the x-coordinate of the cell
coord↩_y	the y-coordinate of the cell

Exceptions

IllegalArgumentException	if the specified state is greater than or equal to the number of states

4.10.3.3 setnewEtapeImmigration()

```
void Immigration.setnewEtapeImmigration ( )  [inline]
```

Updates the state of the cells in the simulation based on the Immigration rules.

The documentation for this class was generated from the following file:

- src/Immigration.java

4.11 ImmiSimulator Class Reference

This is a Java program that simulates an immigration cellular automaton.

Inheritance diagram for ImmiSimulator:

Collaboration diagram for ImmiSimulator:

Public Member Functions

- ImmiSimulator (Immigration cells, GUISimulator gui)

  Constructs a new ImmiSimulator object with the given Immigration cells and GUISimulator gui.
- void setGraphicCell ()

  Sets the graphic representation of the cells.
- void next ()

  Advances the simulation to the next step.
- void restart ()

  Restarts the simulation.

Static Public Attributes

- static final int TAILLE_CELLULE = 30
- static final int MARGES = TAILLE_CELLULE / 2

4.11.1 Detailed Description

This is a Java program that simulates an immigration cellular automaton.

The program uses a GUI to display the cells and their states. It includes methods to set the graphic representation of the cells, advance to the next step of the simulation, and restart the simulation.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 ImmiSimulator()

```
ImmiSimulator.ImmiSimulator (
            Immigration cells,
            GUISimulator gui )  [inline]
```

Constructs a new ImmiSimulator object with the given Immigration cells and GUISimulator gui.

Initializes the graphic representation of the cells.

Parameters

cells	the Immigration object representing the cells
gui	the GUISimulator object for displaying the cells

4.11.3 Member Function Documentation

4.11.3.1 next()

```
void ImmiSimulator.next ( )  [inline]
```

Advances the simulation to the next step.

Resets the GUI, updates the cells' states, and sets the new graphic representation.

4.11.3.2 restart()

```
void ImmiSimulator.restart ( )  [inline]
```

Restarts the simulation.

Resets the GUI, initializes the cells, and sets the new graphic representation.

4.11.3.3 setGraphicCell()

```
void ImmiSimulator.setGraphicCell ( )  [inline]
```

Sets the graphic representation of the cells.

Iterates through the cells and adds rectangles to the GUI based on their states.

4.11.4 Member Data Documentation

4.11.4.1 MARGES

```
final int ImmiSimulator.MARGES = TAILLE_CELLULE / 2  [static]
```

4.11.4.2 TAILLE_CELLULE

```
final int ImmiSimulator.TAILLE_CELLULE = 30  [static]
```

The documentation for this class was generated from the following file:

- src/ImmiSimulator.java

4.12 Main Class Reference

Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`, then press Enter.

Static Public Member Functions

- static void main (String[ ] args)

4.12.1 Detailed Description

Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`, then press Enter.

You can now see whitespace characters in your code.

4.12.2 Member Function Documentation

4.12.2.1 main()

```
static void Main.main (
            String[ ] args )  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/Main.java

4.13 MessageEvent Class Reference

Represents a message event, derived from the base Event class.

Inheritance diagram for MessageEvent:

Collaboration diagram for MessageEvent:

Public Member Functions

- MessageEvent (int date, String message)

  Constructs a MessageEvent object with a specified date and message.
- void execute ()

  Executes the message event by printing the date and message.

4.13.1 Detailed Description

Represents a message event, derived from the base Event class.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 MessageEvent()

```
MessageEvent.MessageEvent (
            int date,
            String message )  [inline]
```

Constructs a MessageEvent object with a specified date and message.

Parameters

date	the date of the event
message	the message associated with the event

4.13.3 Member Function Documentation

4.13.3.1 execute()

```
void MessageEvent.execute ( )  [inline]
```

Executes the message event by printing the date and message.

Reimplemented from Event.

The documentation for this class was generated from the following file:

- src/MessageEvent.java

4.14 Schelling Class Reference

Schelling.java This class extends the Cell class and implements the Schelling model of segregation.

Inheritance diagram for Schelling:

Collaboration diagram for Schelling:

Public Member Functions

- Schelling (int size_x, int size_y, int nb_etats, int ndrDeVoisinDiffPourChanger)

    Initializes a Schelling object with the given parameters.
- HashMap< Point, Boolean > getDict ()

    Returns the dictionary of points.
- int getNb_etats ()
- void initDict (HashMap< Point, Boolean > dictPointToLibre)

    Initializes the dictionary with all points set to free.
- void setnewEtapeSchelling ()

    Updates the states of the cells based on the Schelling model.
- void setFree (Point Cellule)

    Marks a cell as free in the Schelling model.
- void SetNewDestination (Point Cellule)

    Sets a new destination for a cell in the Schelling model.
- void Init_cells ()

    Initializes the cells and dictionary based on the first configuration in the Schelling model.
- void setBoolean_coord_Sche (int bool, int coord_x, int coord_y)

    Sets the state of a cell at a given coordinate in the Schelling model.

Additional Inherited Members

4.14.1 Detailed Description

Schelling.java This class extends the Cell class and implements the Schelling model of segregation.

It initializes a dictionary mapping each point to a boolean value indicating whether it is free or not. It also provides methods to count the number of neighboring cells with a different state, set a new destination for a cell, and update the states of the cells.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Schelling()

```
Schelling.Schelling (
             int size_x,
             int size_y,
             int nb_etats,
             int ndrDeVoisinDiffPourChanger )  [inline]
```

Initializes a Schelling object with the given parameters.

Parameters

size_x	the size of the grid in the x direction
size_y	the size of the grid in the y direction
nb_etats	the number of states
ndrDeVoisinDiffPourChanger	the number of different neighbors required to trigger a change

4.14.3 Member Function Documentation

4.14.3.1 getDict()

```
HashMap<Point, Boolean> Schelling.getDict ( )  [inline]
```

Returns the dictionary of points.

Returns

the dictionary of points

4.14.3.2 getNb_etats()

```
int Schelling.getNb_etats ( )  [inline]
```

4.14.3.3 Init_cells()

```
void Schelling.Init_cells ( )  [inline]
```

Initializes the cells and dictionary based on the first configuration in the Schelling model.

Reimplemented from Cell.

4.14.3.4 initDict()

```
void Schelling.initDict (
             HashMap< Point, Boolean > dictPointToLibre )  [inline]
```

Initializes the dictionary with all points set to free.

Parameters

dictPointToLibre	The dictionary to initialize.

4.14.3.5 setBoolean_coord_Sche()

```
void Schelling.setBoolean_coord_Sche (
             int bool ,
             int coord_x,
             int coord_y )  [inline]
```

Sets the state of a cell at a given coordinate in the Schelling model.

Parameters

bool	The boolean value to set.
coord← _x	The x-coordinate of the cell.
coord← _y	The y-coordinate of the cell.

4.14.3.6 setFree()

```
void Schelling.setFree (
            Point Cellule )  [inline]
```

Marks a cell as free in the Schelling model.

Parameters

Cellule	The cell to mark as free.

4.14.3.7 SetNewDestination()

```
void Schelling.SetNewDestination (
            Point Cellule )  [inline]
```

Sets a new destination for a cell in the Schelling model.

Parameters

Cellule	The current cell.

4.14.3.8 setnewEtapeSchelling()

```
void Schelling.setnewEtapeSchelling ( )  [inline]
```

Updates the states of the cells based on the Schelling model.

The documentation for this class was generated from the following file:

- src/Schelling.java

4.15 SchellingSimulator Class Reference

SchellingSimulator.java.

Inheritance diagram for SchellingSimulator:

Collaboration diagram for SchellingSimulator:

Public Member Functions

- SchellingSimulator (Schelling cells, GUISimulator gui)

  Creates a new SchellingSimulator instance.
- void setGraphicCell ()

  Sets the graphical representation of the cells on the GUI.
- void next ()

  Proceeds to the next step of the simulation.
- void restart ()

  Restarts the simulation by resetting the cells and their states.

Static Public Attributes

- static final int TAILLE_CELLULE = 50
- static final int MARGES = TAILLE_CELLULE / 2

4.15.1 Detailed Description

SchellingSimulator.java.

This class implements a simulator for the Schelling model. It uses a GUI to display the cells and their states. The cells are represented as a grid, and each cell can be in one of several states. The simulator allows for the progression of time, with cells changing their states based on certain rules. It also provides a restart functionality to reset the simulation to its initial state.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 SchellingSimulator()

```
SchellingSimulator.SchellingSimulator (
            Schelling cells,
            GUISimulator gui )  [inline]
```

Creates a new SchellingSimulator instance.

Parameters

cells	The Schelling object representing the cells and their states
gui	The GUISimulator object used to display the cells

4.15.3 Member Function Documentation

4.15.3.1 next()

```
void SchellingSimulator.next ( ) [inline]
```

Proceeds to the next step of the simulation.

4.15.3.2 restart()

```
void SchellingSimulator.restart ( ) [inline]
```

Restarts the simulation by resetting the cells and their states.

4.15.3.3 setGraphicCell()

```
void SchellingSimulator.setGraphicCell ( ) [inline]
```

Sets the graphical representation of the cells on the GUI.

4.15.4 Member Data Documentation

4.15.4.1 MARGES

```
final int SchellingSimulator.MARGES = TAILLE_CELLULE / 2 [static]
```

4.15.4.2 TAILLE_CELLULE

```
final int SchellingSimulator.TAILLE_CELLULE = 50 [static]
```

The documentation for this class was generated from the following file:

- src/SchellingSimulator.java

4.16 SpecialBoids Class Reference

SpecialBoids class extends the Boids class and represents a special type of boid.

Inheritance diagram for SpecialBoids:

Collaboration diagram for SpecialBoids:

Public Member Functions

- SpecialBoids (int x, int y, int vx, int vy, int orientation, int taille_x, int taille_y, Color color, String name)

    Constructor for SpecialBoids class.
- Color getColor ()

    Returns the color of the special boid.
- String getName ()

    Returns the name of the special boid.
- void separate (SpecialBoids[ ] list_boids, int distance_separation)

    Separates the special boids from other boids based on the distance separation parameter.
- void align (SpecialBoids[ ] boids, int distance_alignement)

    Aligns the special boids with other boids based on the distance alignment parameter.
- void cohere (SpecialBoids[ ] boids, int distance_essaim)

    Cohers the special boids towards the center of mass of nearby boids based on the distance cohesion parameter.

Additional Inherited Members

4.16.1 Detailed Description

SpecialBoids class extends the Boids class and represents a special type of boid.

It includes additional properties such as name and color. The class provides methods for separating boids, aligning boids, and coherring boids based on their name and distance parameters.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 SpecialBoids()

```
SpecialBoids.SpecialBoids (
             int x,
             int y,
             int vx,
             int vy,
             int orientation,
             int taille_x,
             int taille_y,
             Color color,
             String name )  [inline]
```

Constructor for SpecialBoids class.

Initializes the special boid with the given position, velocity, orientation, size, color, and name.

Parameters

x	the x-coordinate of the boid's position
y	the y-coordinate of the boid's position
vx	the x-component of the boid's velocity
vy	the y-component of the boid's velocity
orientation	the orientation of the boid
taille_x	the x-size of the boid
taille_y	the y-size of the boid

4.16.3 Member Function Documentation

4.16.3.1 align()

```
void SpecialBoids.align (
             SpecialBoids[ ] boids,
             int distance_alignement )  [inline]
```

Aligns the special boids with other boids based on the distance alignment parameter.

If the special boid is a "poisson", it adjusts its velocity to match the average velocity of nearby "poisson" boids. If the special boid is a "requin", it does not perform any alignment.

Parameters

boids	an array of boids
distance_alignement	the distance threshold for alignment

4.16.3.2 cohere()

```
void SpecialBoids.cohere (
             SpecialBoids[ ] boids,
             int distance_essaim )  [inline]
```

Cohers the special boids towards the center of mass of nearby boids based on the distance cohesion parameter.

If the special boid is a "poisson", it adjusts its velocity towards the center of mass of nearby "poisson" boids. If the special boid is a "requin", it adjusts its velocity towards the center of mass of nearby "poisson" boids.

Parameters

boids	an array of boids
distance_essaim	the distance threshold for cohesion

4.16.3.3 getColor()

```
Color SpecialBoids.getColor ( )  [inline]
```

Returns the color of the special boid.

Returns

The color of the special boid.

4.16.3.4 getName()

```
String SpecialBoids.getName ( )  [inline]
```

Returns the name of the special boid.

Returns

The name of the special boid.

4.16.3.5 separate()

```
void SpecialBoids.separate (
            SpecialBoids[ ] list_boids,
            int distance_separation )  [inline]
```

Separates the special boids from other boids based on the distance separation parameter.

If the special boid is a "poisson", it adjusts its velocity based on the distance to other "poisson" boids. If the special boid is a "requin", it does not perform any separation.

Parameters

list_boids	an array of boids
distance_separation	the distance threshold for separation

The documentation for this class was generated from the following file:

- src/SpecialBoids.java

4.17 TestBalls Class Reference

TestBalls.java.

Static Public Member Functions

- static void main (String[ ] args)

4.17.1 Detailed Description

TestBalls.java.

This code demonstrates the usage of the Balls class to manipulate a collection of Point objects representing the positions of balls. It initializes a collection of balls with specific positions, translates the positions by a given amount, reinitializes the positions, and prints the positions before and after each operation.

4.17.2 Member Function Documentation

4.17.2.1 main()

```
static void TestBalls.main (
             String[ ] args )  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/TestBalls.java

4.18 TestBallsSimulator Class Reference

TestBallsSimulator.java.

Static Public Member Functions

- static void main (String[ ] args)

4.18.1 Detailed Description

TestBallsSimulator.java.

This program demonstrates the simulation of balls using a graphical user interface. It creates a GUISimulator object with a black background and initializes an array of Point objects representing the balls' positions. The BallsSimulator object is then created with the array of Point objects and the GUISimulator object as parameters. Finally, the GUISimulator's simulable is set to the BallsSimulator object.

4.18.2 Member Function Documentation

4.18.2.1 main()

```
static void TestBallsSimulator.main (
             String[ ] args )  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/TestBallsSimulator.java

4.19 TestBoidsSimulator Class Reference

This code is written in Java and is stored in the file TestBoidsSimulator.java.

Static Public Member Functions

- static void main (String[ ] args)

4.19.1 Detailed Description

This code is written in Java and is stored in the file TestBoidsSimulator.java.

It creates a simulation of boids, which are virtual creatures that exhibit collective behavior. The simulation is displayed using a graphical user interface (GUI) provided by the GUISimulator library. The code defines the behavior and properties of the boids, such as their positions, velocities, orientations, and colors. It also creates an event manager to handle the simulation events and manages the interaction between the boids. The boids are represented by instances of the SpecialBoids class, which is a subclass of the Boids class. The simulation is started by creating a BoidsSimulator object and setting it as the simulable for the GUI.

4.19.2 Member Function Documentation

4.19.2.1 main()

```
static void TestBoidsSimulator.main (
             String[ ] args )  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/TestBoidsSimulator.java

4.20 TestCell Class Reference

This code demonstrates the implementation of the Game of Conway using a Cell object.

Static Public Member Functions

- static void main (String[ ] args)

4.20.1 Detailed Description

This code demonstrates the implementation of the Game of Conway using a Cell object.

The code initializes a Cell object and performs various operations on it, such as setting boolean coordinates, initializing the configuration, and generating new steps in the Conway game. It also prints the final configuration of the Cell object after each operation.

4.20.2 Member Function Documentation

4.20.2.1 main()

```
static void TestCell.main (
            String[ ] args )  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/TestCell.java

4.21 TestCellSimulator Class Reference

This code initializes a graphical user interface (GUI) simulator and creates a cell object.

Static Public Member Functions

- static void main (String[ ] args)

4.21.1 Detailed Description

This code initializes a graphical user interface (GUI) simulator and creates a cell object.

The cell object is then modified by setting boolean coordinates. The initial configuration of the cell is set using the InitConfigFirst() method. Finally, a CellSimulator object is created and added to the GUI window.

4.21.2 Member Function Documentation

4.21.2.1 main()

```
static void TestCellSimulator.main (
            String[ ] args )  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/TestCellSimulator.java

4.22 TestEventManager Class Reference

TestEventManager.java.

Static Public Member Functions

- static void main (String[ ] args) throws InterruptedException

4.22.1 Detailed Description

TestEventManager.java.

This program demonstrates the usage of the EventManager class to manage message events. It creates an EventManager object and adds multiple MessageEvent objects with different messages and intervals. The program then iterates through the events, printing the messages at the specified intervals. The program waits for 1 second between each event.

4.22.2 Member Function Documentation

4.22.2.1 main()

```
static void TestEventManager.main (
             String[ ] args ) throws InterruptedException  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/TestEventManager.java

4.23 TestImmiSimulator Class Reference

This code is a Java program that demonstrates the simulation of an immigration process.

Static Public Member Functions

- static void main (String[ ] args)

Static Public Attributes

- static final double DIVISON_ECHELLE = 1.8
- static final int NB_ETAT = 4

4.23.1 Detailed Description

This code is a Java program that demonstrates the simulation of an immigration process.

It uses the GUISimulator library to create a graphical user interface and displays the simulation on a blue background. The simulation is based on the Immigration class, which represents a grid of cells with different states. The initial state of the cells is randomly assigned using the Random class. The ImmiSimulator class is responsible for running the simulation and updating the GUI accordingly. The main method initializes the necessary variables, creates an instance of GUISimulator, and adds the ImmiSimulator to the GUI.

4.23.2 Member Function Documentation

4.23.2.1 main()

```
static void TestImmiSimulator.main (
             String[] args )  [inline], [static]
```

4.23.3 Member Data Documentation

4.23.3.1 DIVISON_ECHELLE

```
final double TestImmiSimulator.DIVISON_ECHELLE = 1.8  [static]
```

4.23.3.2 NB_ETAT

```
final int TestImmiSimulator.NB_ETAT = 4  [static]
```

The documentation for this class was generated from the following file:

- src/TestImmiSimulator.java

4.24 TestInvader Class Reference

test invader, the first given gui Implementation that print a little monster of the GUI application

Static Public Member Functions

- static void main (String[ ] args)

4.24.1 Detailed Description

test invader, the first given gui Implementation that print a little monster of the GUI application

4.24.2 Member Function Documentation

4.24.2.1 main()

```
static void TestInvader.main (
              String[] args )  [inline], [static]
```

The documentation for this class was generated from the following file:

- src/TestInvader.java

4.25 TestShellingSimulator Class Reference

Summary: This code is a simulation of the Schelling model, which is a social simulation model used to study segregation in a population.

Static Public Member Functions

- static void main (String[ ] args)

Static Public Attributes

- static final double DIVISON_ECHELLE = 1.8
- static final int NB_ETAT = 4
- static final int NB_VOISIN_CHANGEMENT = 3

4.25.1 Detailed Description

Summary: This code is a simulation of the Schelling model, which is a social simulation model used to study segregation in a population.

The code initializes a grid of cells with random states, and then runs the simulation to observe the dynamics of segregation.

4.25.2 Member Function Documentation

4.25.2.1 main()

```
static void TestShellingSimulator.main (
              String[] args )  [inline], [static]
```

4.25.3 Member Data Documentation

4.25.3.1 DIVISON_ECHELLE

```
final double TestShellingSimulator.DIVISON_ECHELLE = 1.8  [static]
```

4.25.3.2 NB_ETAT

```
final int TestShellingSimulator.NB_ETAT = 4  [static]
```

4.25.3.3 NB_VOISIN_CHANGEMENT

```
final int TestShellingSimulator.NB_VOISIN_CHANGEMENT = 3  [static]
```

The documentation for this class was generated from the following file:

- src/TestShellingSimulator.java

4.26 TriangleElement Class Reference

Represents a triangle shape that can be painted on a graphical area.

Inheritance diagram for TriangleElement:

Collaboration diagram for TriangleElement:

Public Member Functions

- TriangleElement (int[ ] xPoints, int[ ] yPoints, Color color, int orientation)

  Constructs a TriangleElement object with the given points, color, and orientation.
- void setOrientation (int orientation)

  Sets the orientation of the triangle.
- void paint (Graphics2D graphics2D)

  Paints the triangle on the given Graphics2D object.

4.26.1 Detailed Description

Represents a triangle shape that can be painted on a graphical area.

The position, color, and orientation of the triangle can be set, and it can be painted with the specified attributes.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 TriangleElement()

```
TriangleElement.TriangleElement (
             int[ ] xPoints,
             int[ ] yPoints,
             Color color,
             int orientation )  [inline]
```

Constructs a TriangleElement object with the given points, color, and orientation.

Parameters

xPoints	the x-coordinates of the triangle's vertices
yPoints	the y-coordinates of the triangle's vertices
color	the color of the triangle
orientation	the orientation of the triangle in degrees

4.26.3 Member Function Documentation

4.26.3.1 paint()

```
void TriangleElement.paint (
            Graphics2D graphics2D )  [inline]
```

Paints the triangle on the given Graphics2D object.

Parameters

graphics2D	the Graphics2D object to paint on

4.26.3.2 setOrientation()

```
void TriangleElement.setOrientation (
            int orientation )  [inline]
```

Sets the orientation of the triangle.

Parameters

orientation	the new orientation of the triangle in degrees

The documentation for this class was generated from the following file:

- src/TriangleElement.java

Chapter 5

File Documentation

5.1 src/Balls.java File Reference

Classes

- class Balls

  This class represents our balls with point table.

5.2 src/BallsSimulator.java File Reference

Classes

- class BallsSimulator

  This class represents a balls simulator that implements the Simulable interface.

5.3 src/Boids.java File Reference

Classes

- class Boids

  Boids.java.

5.4 src/BoidsEvent.java File Reference

Classes

- class BoidsEvent

  BoidsEvent class represents an event that updates the behavior of a group of boids.

5.5 src/BoidsSimulator.java File Reference

Classes

- class BoidsSimulator

  BoidsSimulator.java.

5.6 src/Cell.java File Reference

Classes

- class Cell

  This is a Java program that defines a Cell class.

5.7 src/CellSimulator.java File Reference

Classes

- class CellSimulator

  This class implements the Simulable interface and simulates the behavior of a group of cells using Conway's Game of Life rules.

5.8 src/Event.java File Reference

Classes

- class Event

  This is an abstract class representing an event.

5.9 src/EventManager.java File Reference

Classes

- class EventManager

  This class manages a priority queue of events and executes them in chronological order.

5.10 src/Immigration.java File Reference

Classes

- class Immigration

  This code defines a class called "Immigration" that represents a simulation of a cellular automaton with multiple states.

5.11 src/ImmiSimulator.java File Reference

Classes

- class ImmiSimulator

  This is a Java program that simulates an immigration cellular automaton.

5.12 src/Main.java File Reference

Classes

- class Main

  Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`, then press Enter.

5.13 src/MessageEvent.java File Reference

Classes

- class MessageEvent

  Represents a message event, derived from the base Event class.

5.14 src/Schelling.java File Reference

Classes

- class Schelling

  Schelling.java This class extends the Cell class and implements the Schelling model of segregation.

5.15 src/SchellingSimulator.java File Reference

Classes

- class SchellingSimulator

  SchellingSimulator.java.

5.16 src/SpecialBoids.java File Reference

Classes

- class SpecialBoids

  SpecialBoids class extends the Boids class and represents a special type of boid.

5.17 src/TestBalls.java File Reference

Classes

- class TestBalls

  TestBalls.java.

5.18 src/TestBallsSimulator.java File Reference

Classes

- class TestBallsSimulator

  TestBallsSimulator.java.

5.19 src/TestBoidsSimulator.java File Reference

Classes

- class TestBoidsSimulator

  This code is written in Java and is stored in the file TestBoidsSimulator.java.

5.20 src/TestCell.java File Reference

Classes

- class TestCell

  This code demonstrates the implementation of the Game of Conway using a Cell object.

5.21 src/TestCellSimulator.java File Reference

Classes

- class TestCellSimulator

  This code initializes a graphical user interface (GUI) simulator and creates a cell object.

5.22 src/TestEventManager.java File Reference

Classes

- class TestEventManager

  TestEventManager.java.

5.23 src/TestImmiSimulator.java File Reference

Classes

- class TestImmiSimulator

  This code is a Java program that demonstrates the simulation of an immigration process.

5.24 src/TestInvader.java File Reference

Classes

- class TestInvader

  test invader, the first given gui Implementation that print a little monster of the GUI application
- class **Invader**

  Un mini-invader...

5.25 src/TestShellingSimulator.java File Reference

Classes

- class TestShellingSimulator

  Summary: This code is a simulation of the Schelling model, which is a social simulation model used to study segregation in a population.

5.26 src/TriangleElement.java File Reference

Classes

- class TriangleElement

  Represents a triangle shape that can be painted on a graphical area.