

A wide-angle photograph of a waterfall in a dense forest. The waterfall flows over several layers of dark, mossy rocks, creating a series of small pools. The water is white and turbulent as it falls. The surrounding forest is lush and green, with various trees and ferns visible. A large tree trunk is prominent on the left side.

# Flux de Contrôle

**Mais avant, méthodes et décomposition**

## Methodes: Definition et invocation

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}  
  
private void tournerADroite() {  
    turnLeft();  
    turnLeft();  
    turnLeft();  
}
```

## Methodes: Definition et invocation

Definition  
de méthodes

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}
```

```
private void tournerADroite() {  
    turnLeft();  
    turnLeft();  
    turnLeft();  
}
```

## Methodes: Definition et invocation

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}  
  
private void tournerADroite() {  
    turnLeft();  
    turnLeft();  
    turnLeft();  
}
```

Definition  
de méthodes

Invocation  
de méthodes

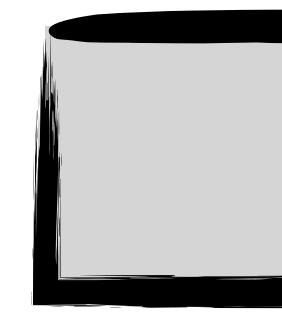
## Nom de méthodes

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}  
  
private void tournerADroite() {  
    turnLeft();  
    turnLeft();  
    turnLeft();  
}
```

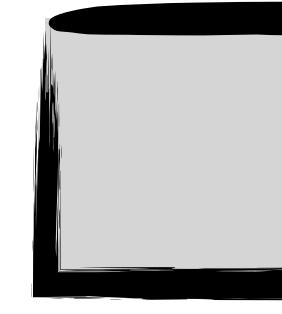
## Nom de méthodes

```
private void retourner Au Depart() {  
    ...  
}  
  
private void retournerAuDepart() {  
    ...  
}  
  
private void retourner A L'Interieur() {  
    ...  
}  
  
private void retournerAL'Interieur() {  
    ...  
}  
  
private void retournerALInterieur() {  
    ...  
}
```

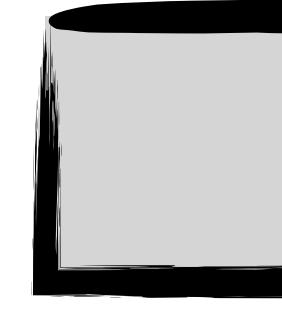
## Nom de méthodes



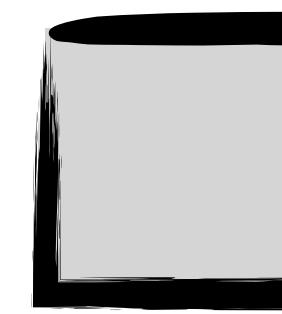
```
private void retourner Au Depart() {  
    ...  
}
```



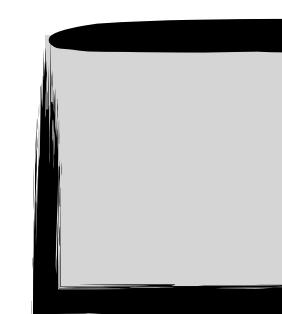
```
private void retournerAuDepart() {  
    ...  
}
```



```
private void retourner A L'Interieur() {  
    ...  
}
```



```
private void retournerAL'Interieur() {  
    ...  
}
```

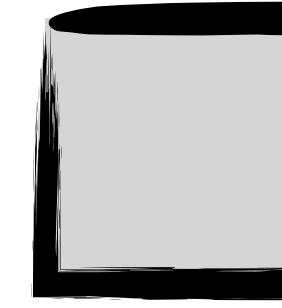


```
private void retournerALInterieur() {  
    ...  
}
```

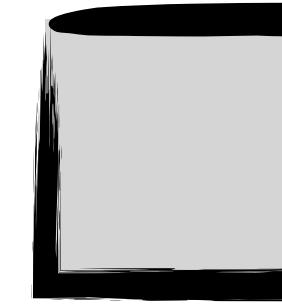
## Nom de méthodes



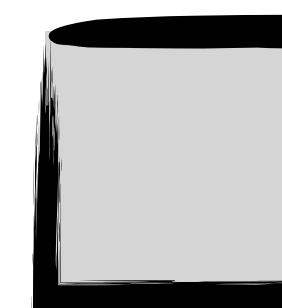
```
private void retourner Au Depart() {  
    ...  
}
```



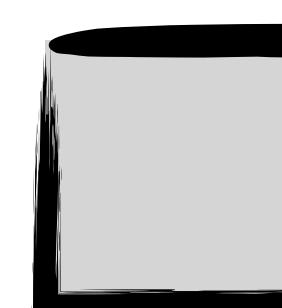
```
private void retournerAuDepart() {  
    ...  
}
```



```
private void retourner A L'Interieur() {  
    ...  
}
```

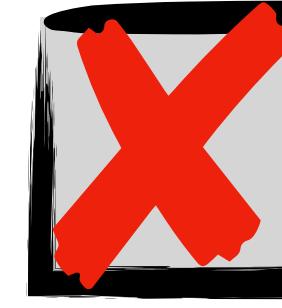


```
private void retournerAL'Interieur() {  
    ...  
}
```

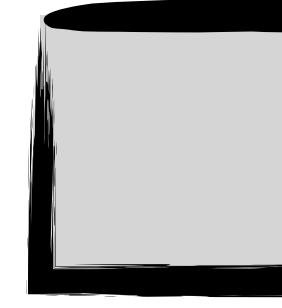


```
private void retournerALInterieur() {  
    ...  
}
```

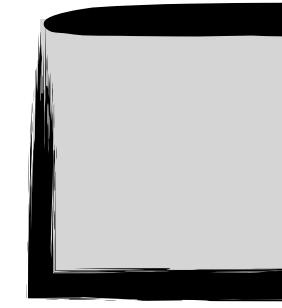
## Nom de méthodes



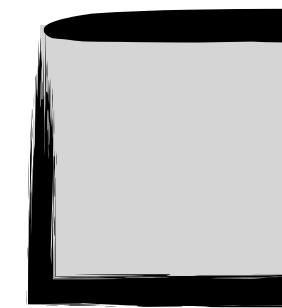
```
private void retourner Au Depart() {  
    ...  
}
```



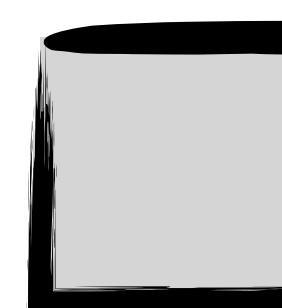
```
private void retournerAuDepart() {  
    ...  
}
```



```
private void retourner A L'Interieur() {  
    ...  
}
```



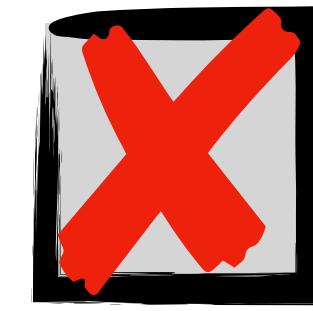
```
private void retournerAL'Interieur() {  
    ...  
}
```



```
private void retournerAInterior() {  
    ...  
}
```

Pas d'espace !

## Nom de méthodes

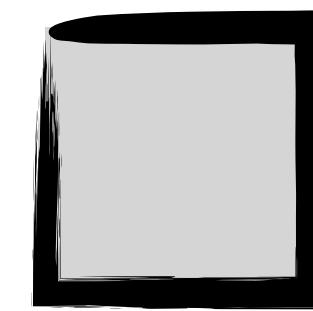


```
private void retourner Au Depart() {  
    ...  
}
```

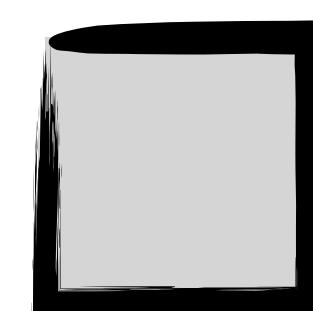


```
private void retournerAuDepart() {  
    ...  
}
```

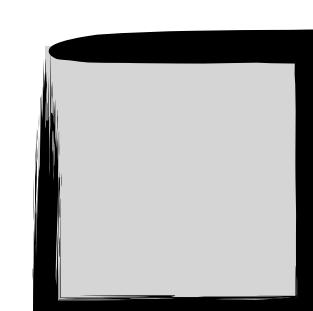
Pas d'espace !



```
private void retourner A L'interieur() {  
    ...  
}
```



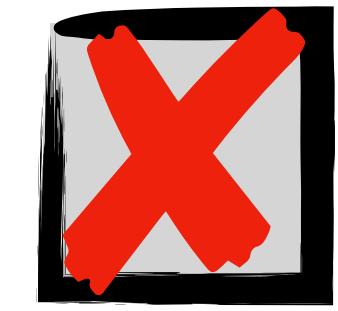
```
private void retournerAL'Interieur() {  
    ...  
}
```



```
private void retournerAInterieur() {  
    ...  
}
```

## Nom de méthodes

**private void retourner Au Depart()** {  
    ...  
}

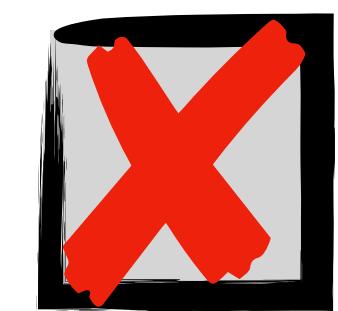


**private void retournerAuDepart()** {  
    ...  
}

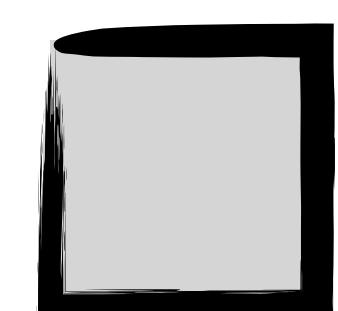


Pas d'espace !

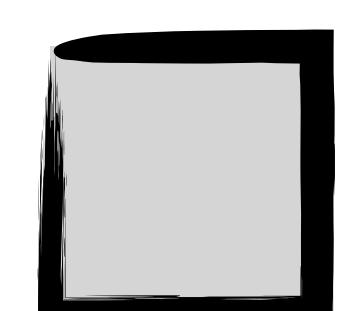
**private void retourner A L'interieur()** {  
    ...  
}



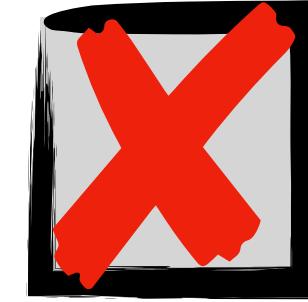
**private void retournerAL'Interieur()** {  
    ...  
}



**private void retournerALInterieur()** {  
    ...  
}



## Nom de méthodes

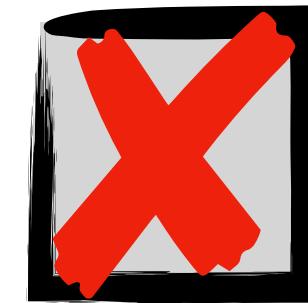


```
private void retourner Au Depart() {  
    ...  
}
```

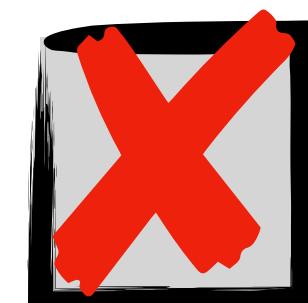


```
private void retournerAuDepart() {  
    ...  
}
```

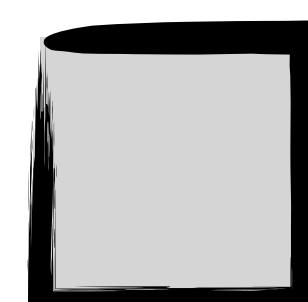
Pas d'espace !



```
private void retourner A L'interieur() {  
    ...  
}
```

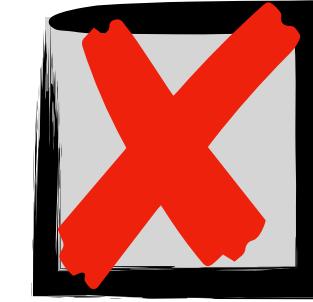


```
private void retournerAL'Interieur() {  
    ...  
}
```



```
private void retournerALInterieur() {  
    ...  
}
```

## Nom de méthodes

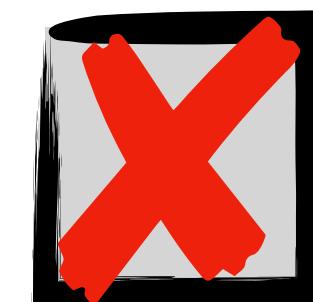


```
private void retourner Au Depart() {  
    ...  
}
```

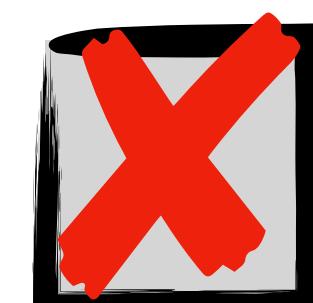


```
private void retournerAuDepart() {  
    ...  
}
```

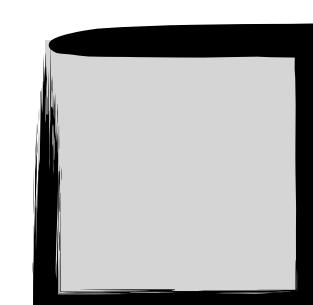
Pas d'espace !



```
private void retourner A L'Interieur() {  
    ...  
}
```



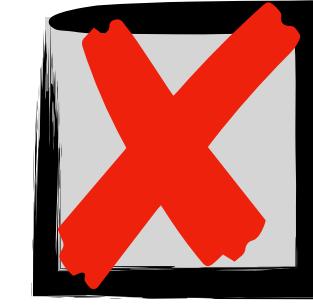
```
private void retournerAL'Interieur() {  
    ...  
}
```



```
private void retournerALInterieur() {  
    ...  
}
```

Pas apostrophe !

## Nom de méthodes



```
private void retourner Au Depart() {  
    ...  
}
```



```
private void retournerAuDepart() {  
    ...  
}
```

Pas d'espace !



```
private void retourner A L'Interieur() {  
    ...  
}
```



```
private void retournerAL'Interieur() {  
    ...  
}
```



```
private void retournerALInterieur() {  
    ...  
}
```

Pas apostrophe !

## Methodes: Indentation

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}
```

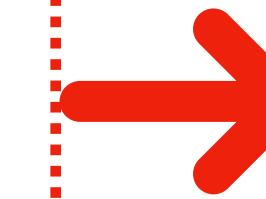
## Methods: Indentation

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}
```

## Methods: Indentation

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}
```

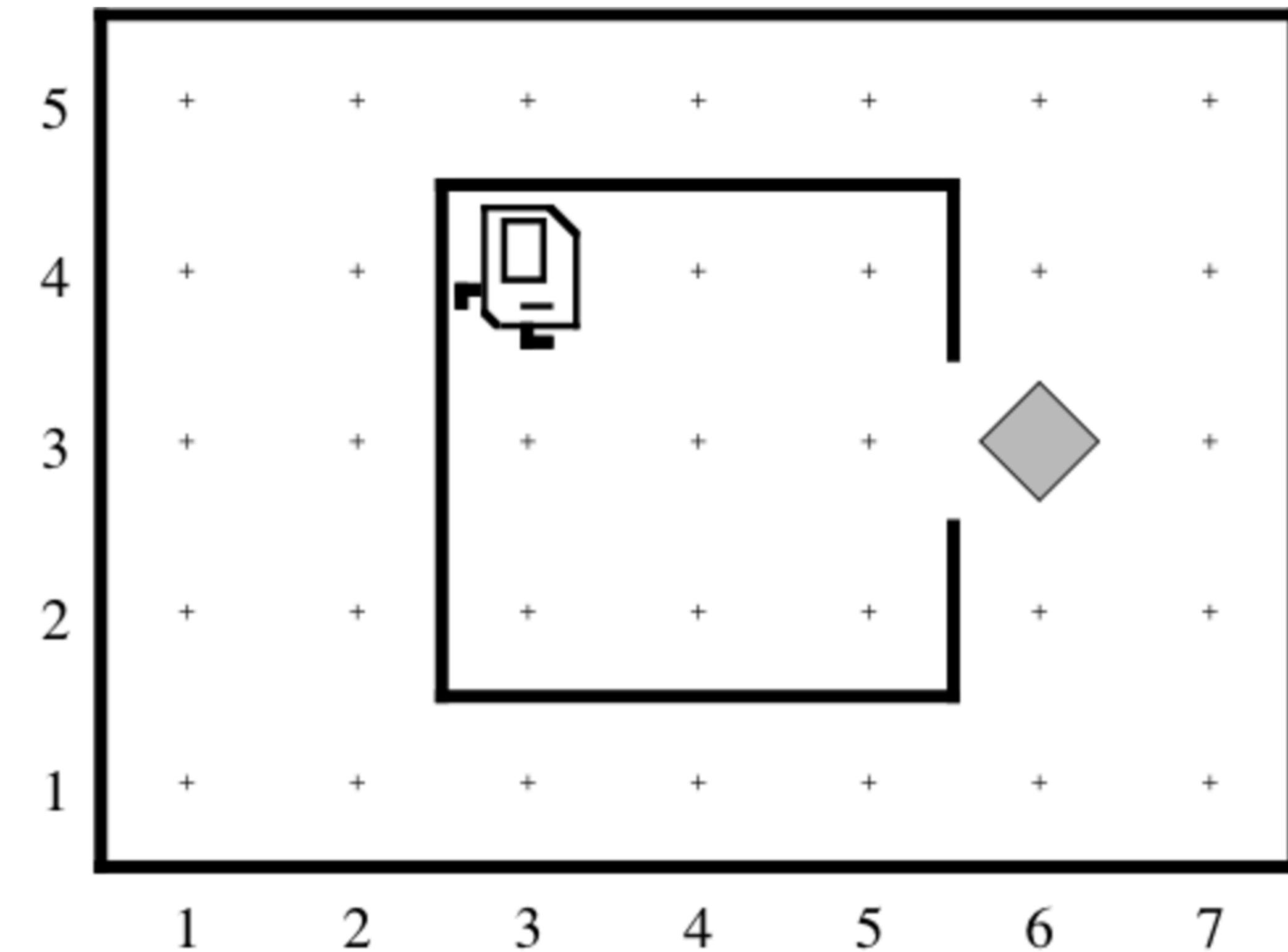
## Methodes: Indentation



Indentation

```
private void retournerAuDepart() {  
    faireDemiTour();  
    move();  
    move();  
    move();  
    tournerADroite();  
    move();  
    tournerADroite();  
}
```

# Ramasser Le Journal



```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move(); move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();  
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();        
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();  
    }
}

private void tournerADroite() {
    turnLeft();
    turnLeft();
    turnLeft();
}

private void faireDemiTour() {
    turnLeft();
    turnLeft();
}
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

```
import stanford.karel.*;

public class RamasserJournal extends Karel {

    public void run(){
        allerDehors();
        prendreJournal();
        retournerAuDepart();
    }

    private void allerDehors() {
        move();
        move();
        tournerADroite();
        move();
        turnLeft();
        move();
    }

    private void prendreJournal() {
        pickBeeper();
    }

    private void retournerAuDepart() {
        faireDemiTour();
        move();
        move();
        move();
        tournerADroite();
        move();
        tournerADroite();
    }

    private void tournerADroite() {
        turnLeft();
        turnLeft();
        turnLeft();
    }

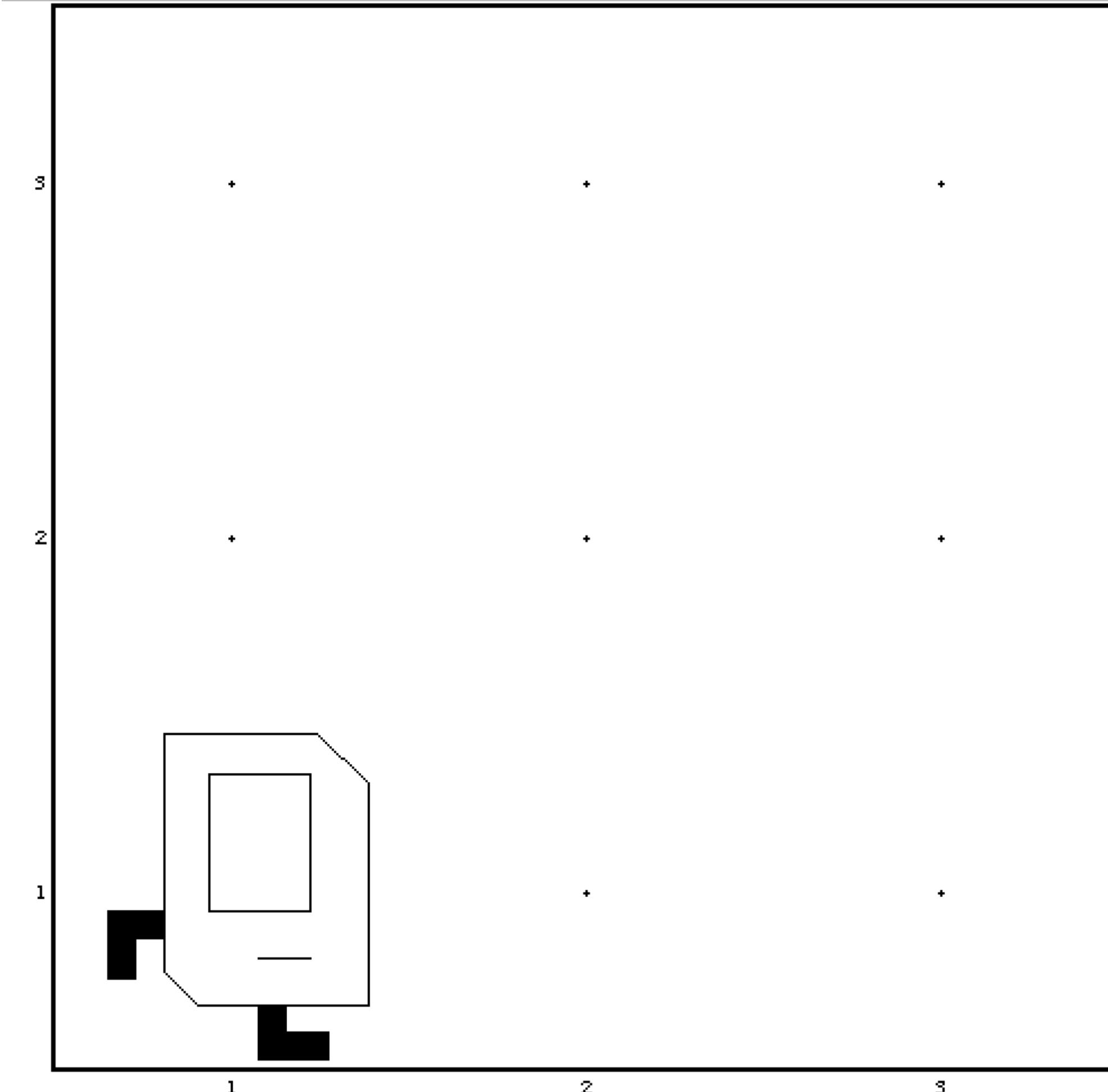
    private void faireDemiTour() {
        turnLeft();
        turnLeft();
    }
}
```

**Boucle "Pour"**

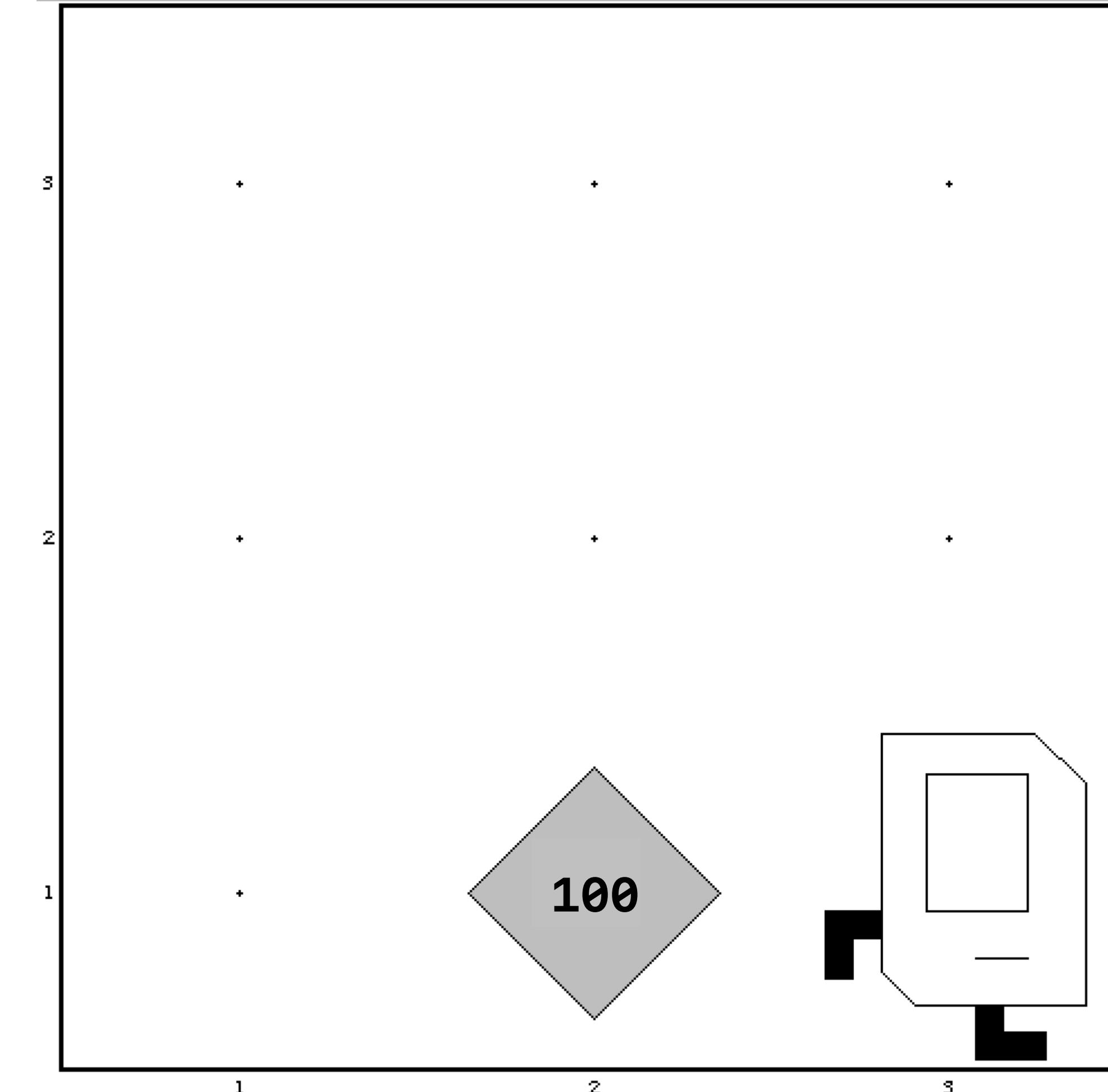
**"for" Loop**

# Placer 100 Beepers?

Avant



Après



# Placer 100 Beepers?

Sans Boucle

Avec boucle "pour"

## Placer 100 Beepers?

Sans Boucle

```
public void run() {  
    move();  
    putBeeper();  
    // putBeeper(); 90 fois de plus  
    move();  
}
```

Avec boucle "pour"

# Placer 100 Beepers?

Sans Boucle

```
public void run() {  
    move();  
    putBeeper();  
    // putBeeper(); 90 fois de plus  
    move();  
}
```

Avec boucle "pour"

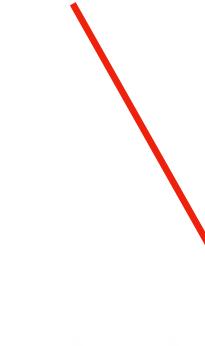
```
public void run() {  
    move();  
    for(int i=0; i<100; i++) {  
        putBeeper();  
    }  
    move();  
}
```

## Boucle « pour »

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

## Boucle « pour »

Initialization du  
compteur i



```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

## Boucle « pour »

Initialization du  
compteur i

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

int i=0

## Boucle « pour »

Initialization du  
compteur i

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

Type de données:  
Nombre entier  
(integer en anglais)

int i=0

## Boucle « pour »

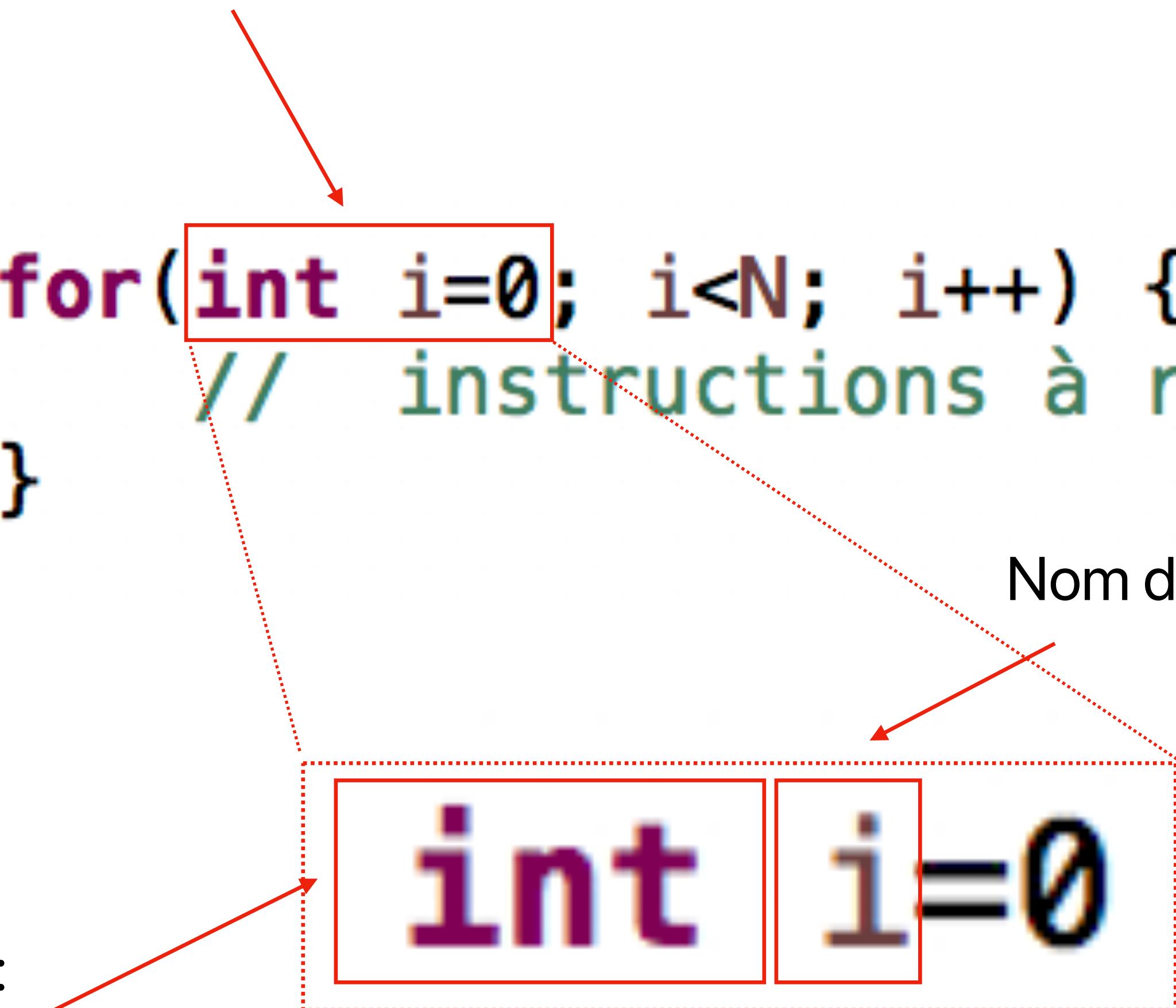
Initialization du  
compteur i

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

Type de données:  
Nombre entier  
(integer en anglais)

int i=0

Nom du compteur



## Boucle « pour »

Initialization du  
compteur i

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

Type de données:  
Nombre entier  
(integer en anglais)

int i=0

Nom du compteur

Valeur initiale du  
compteur

## Boucle « pour »

Initialization du  
compteur i

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

## Boucle « pour »

Condition de continuation  
de la boucle

Initialization du  
compteur i

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

## Boucle « pour »

Condition de continuation  
de la boucle

Initialization du  
compteur i

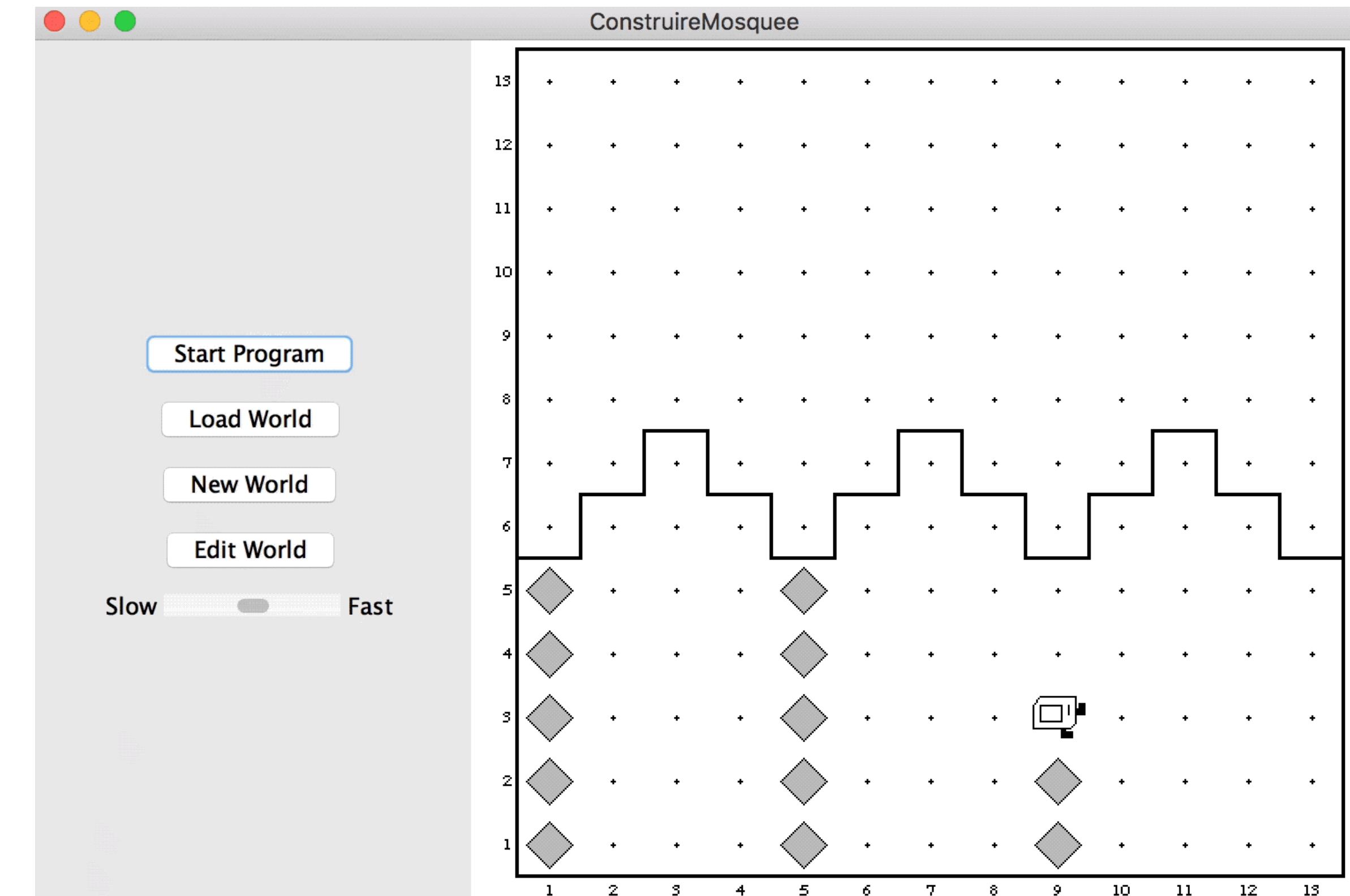
Incrementation du  
compteur i

```
for(int i=0; i<N; i++) {  
    // instructions à répéter N fois  
}
```

**Comment construire la grande mosquée de Conakry?**

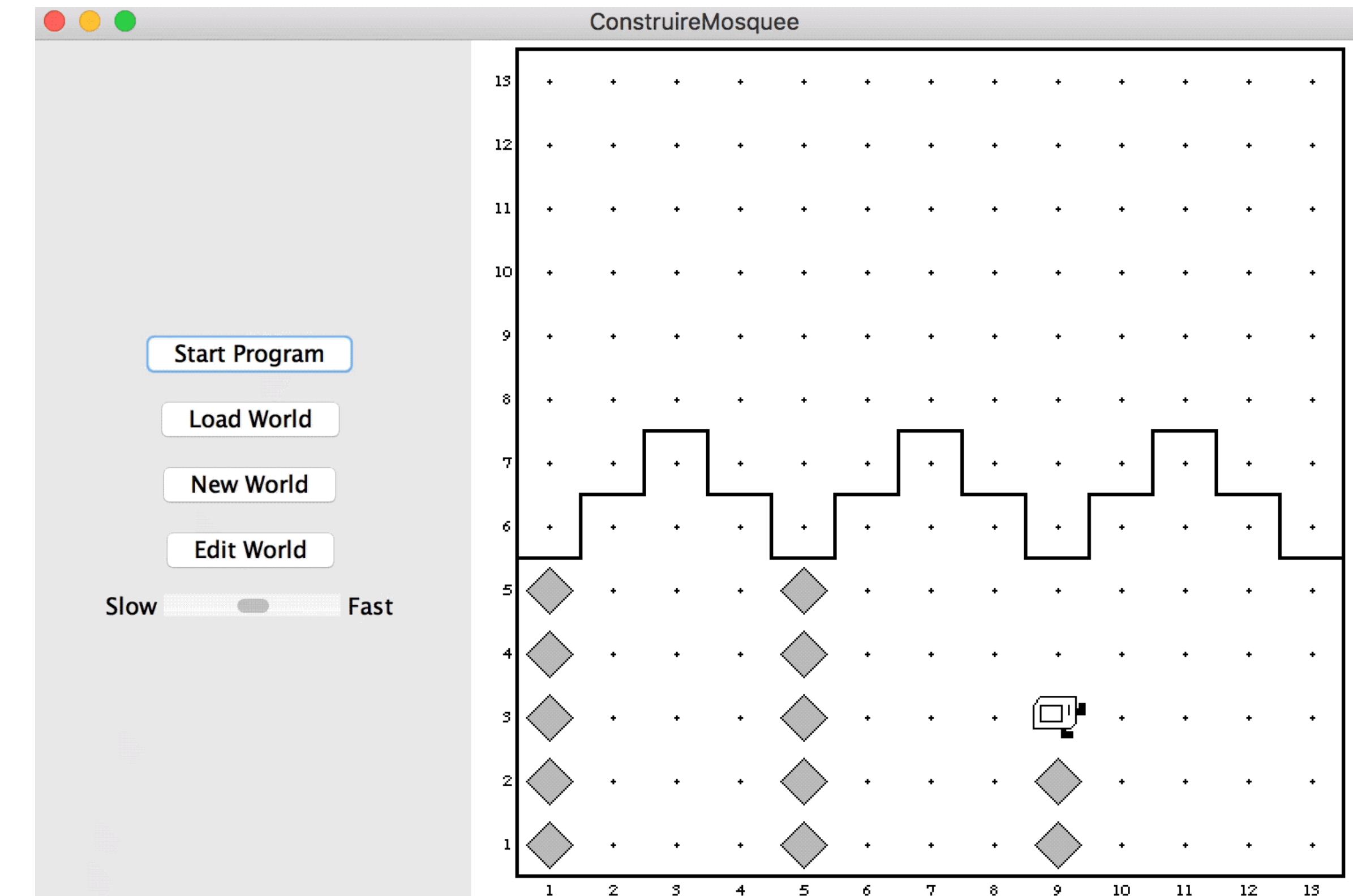
# Construire la grande mosquée de Conakry: « Méthode de Tokpa »

```
private void methodeDeTokpa() {  
    for(int i=0; i<3; i++) {  
        construireUnPilier();  
        allerAuPilierProchain();  
    }  
    construireUnPilier();  
}  
  
private void construireUnPilier() {  
    monterEnConstruisant();  
    descendre();  
}
```



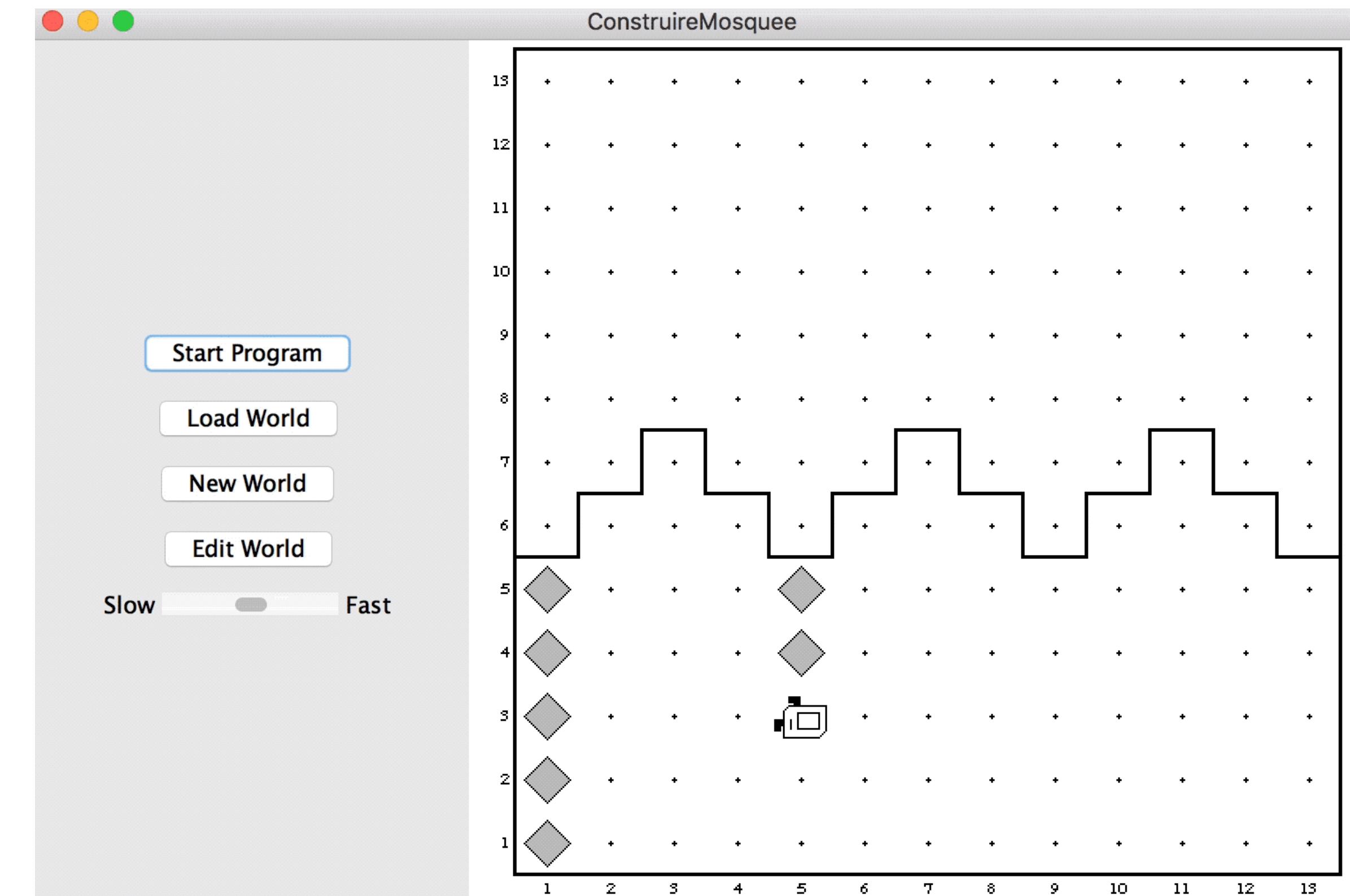
# Construire la grande mosquée de Conakry: « Méthode de Tokpa »

```
private void methodeDeTokpa() {  
    for(int i=0; i<3; i++) {  
        construireUnPilier();  
        allerAuPilierProchain();  
    }  
    construireUnPilier();  
}  
  
private void construireUnPilier() {  
    monterEnConstruisant();  
    descendre();  
}
```



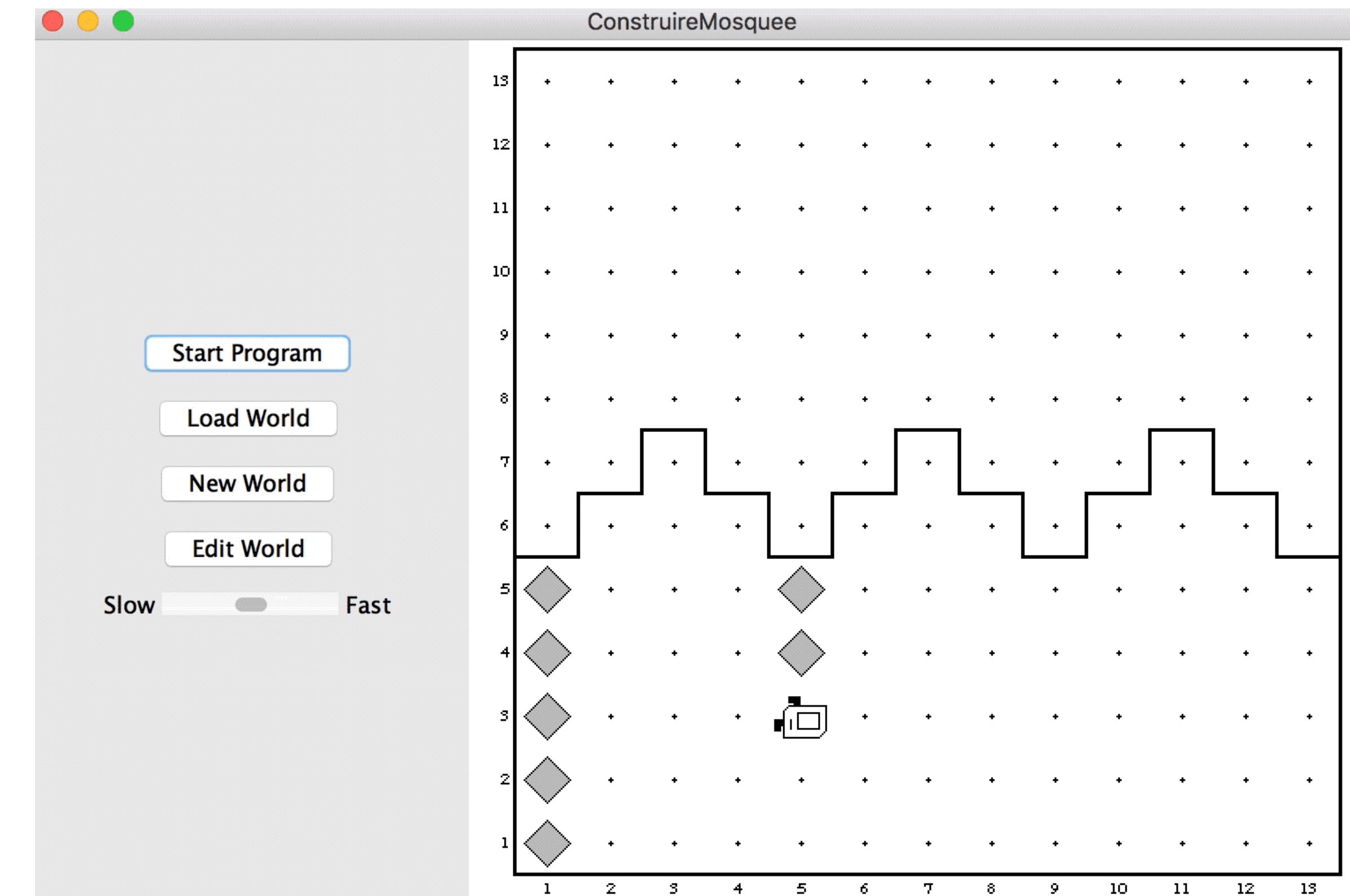
# Construire la grande mosquée de Conakry: « Méthode de Chanceux »

```
private void methodeDeChanceux() {  
    construireDeuxPiliers();  
    allerAuPilierProchain();  
    construireDeuxPiliers();  
}  
  
private void construireDeuxPiliers() {  
    monterEnConstruisant();  
    tournerADroite();  
    allerAuPilierProchain();  
    descendreEnConstruisant();  
}
```



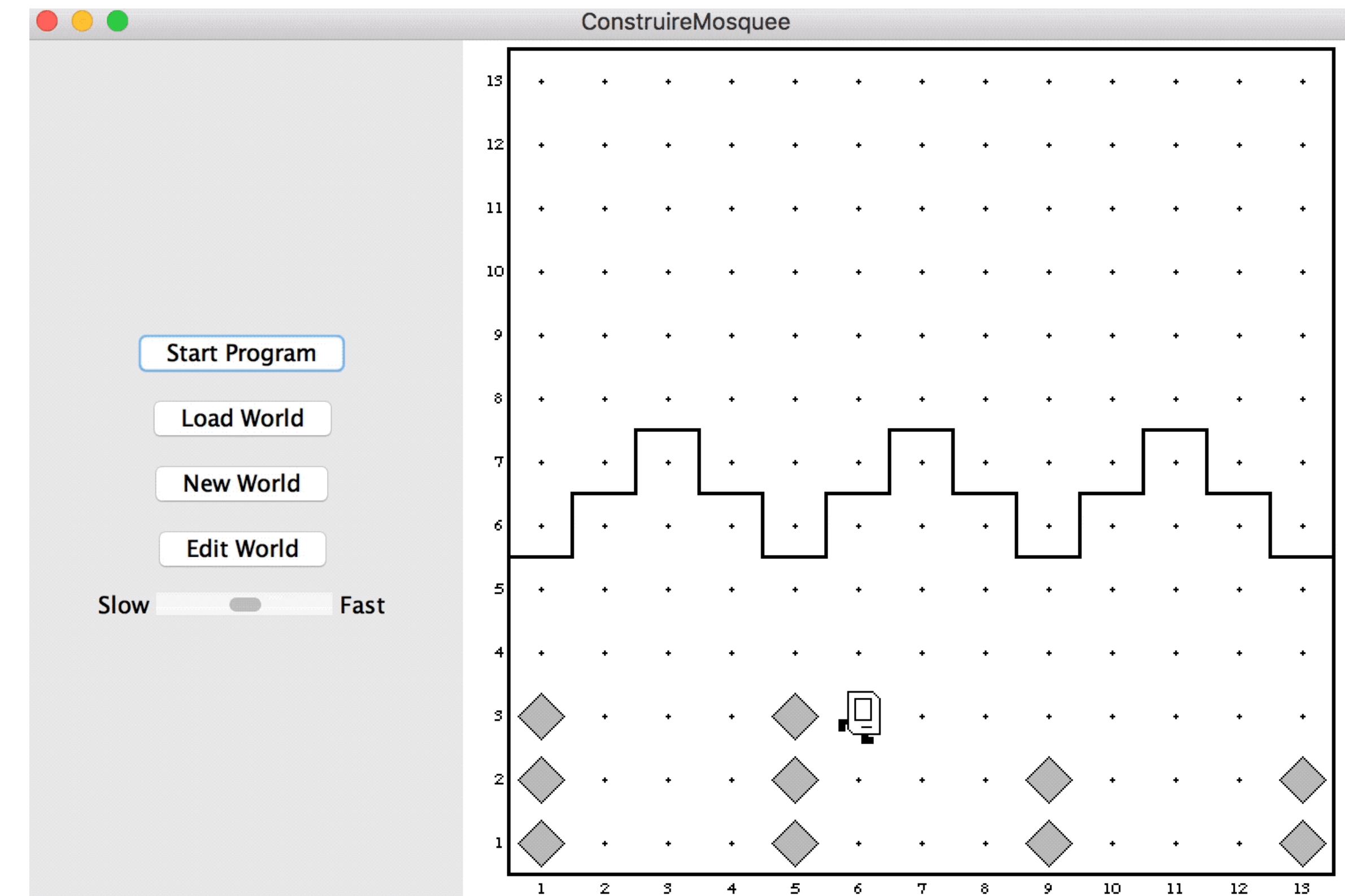
# Construire la grande mosquée de Conakry: « Méthode de Chanceux »

```
private void methodeDeChanceux() {  
    construireDeuxPiliers();  
    allerAuPilierProchain();  
    construireDeuxPiliers();  
}  
  
private void construireDeuxPiliers() {  
    monterEnConstruisant();  
    tournerADroite();  
    allerAuPilierProchain();  
    descendreEnConstruisant();  
}
```



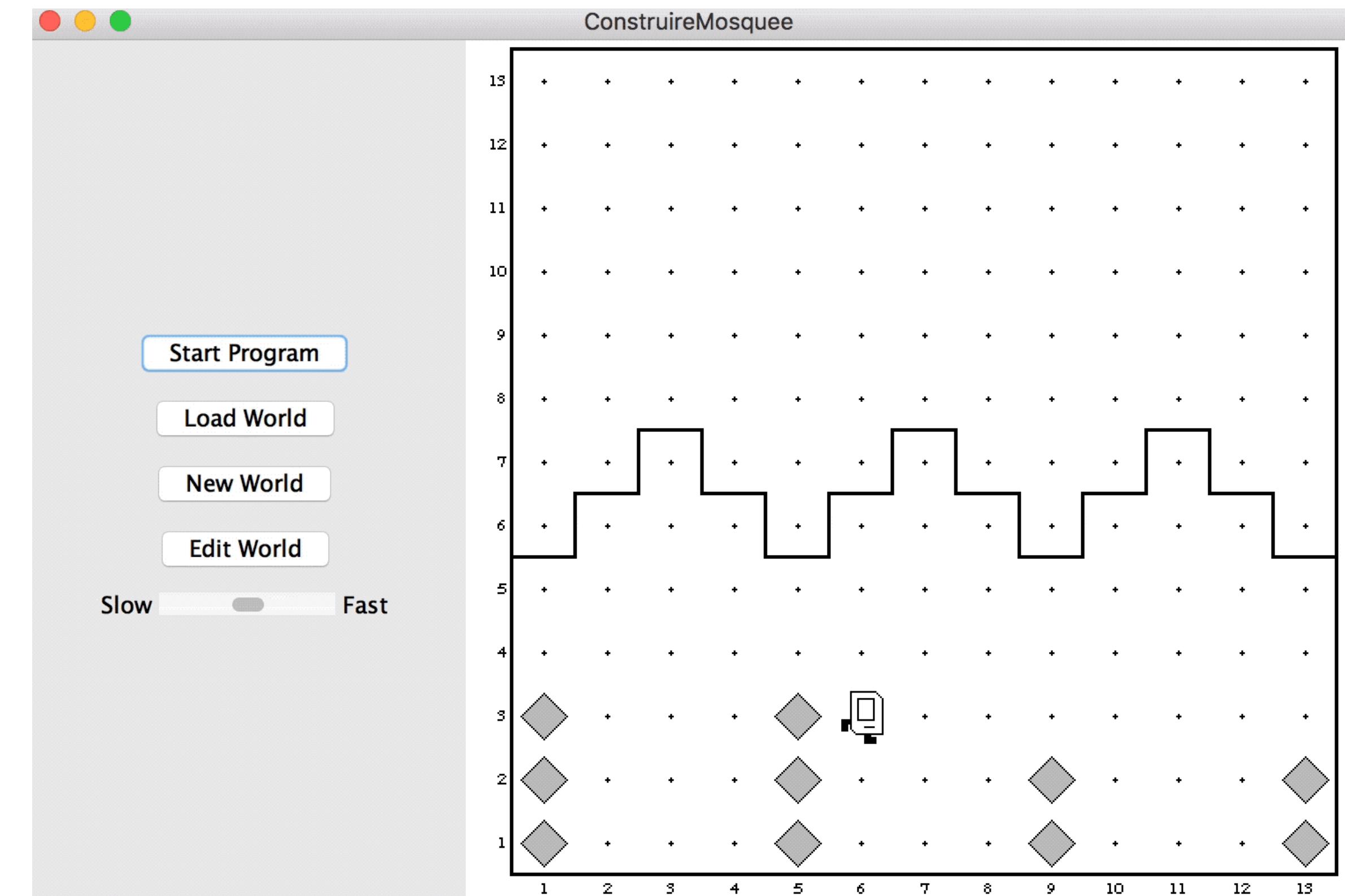
# Construire la grande mosquée de Conakry: « Méthode de Daphne »

```
private void methodeDeDaphne() {  
    for(int i=0; i<4; i++) {  
        construireUneLigne();  
        allerALaLigneSuivante();  
    }  
    construireUneLigne();  
  
    // redescendre  
    turnLeft();  
    descendre();  
}
```



# Construire la grande mosquée de Conakry: « Méthode de Daphne »

```
private void methodeDeDaphne() {  
    for(int i=0; i<4; i++) {  
        construireUneLigne();  
        allerALaLigneSuivante();  
    }  
    construireUneLigne();  
  
    // redescendre  
    turnLeft();  
    descendre();  
}
```

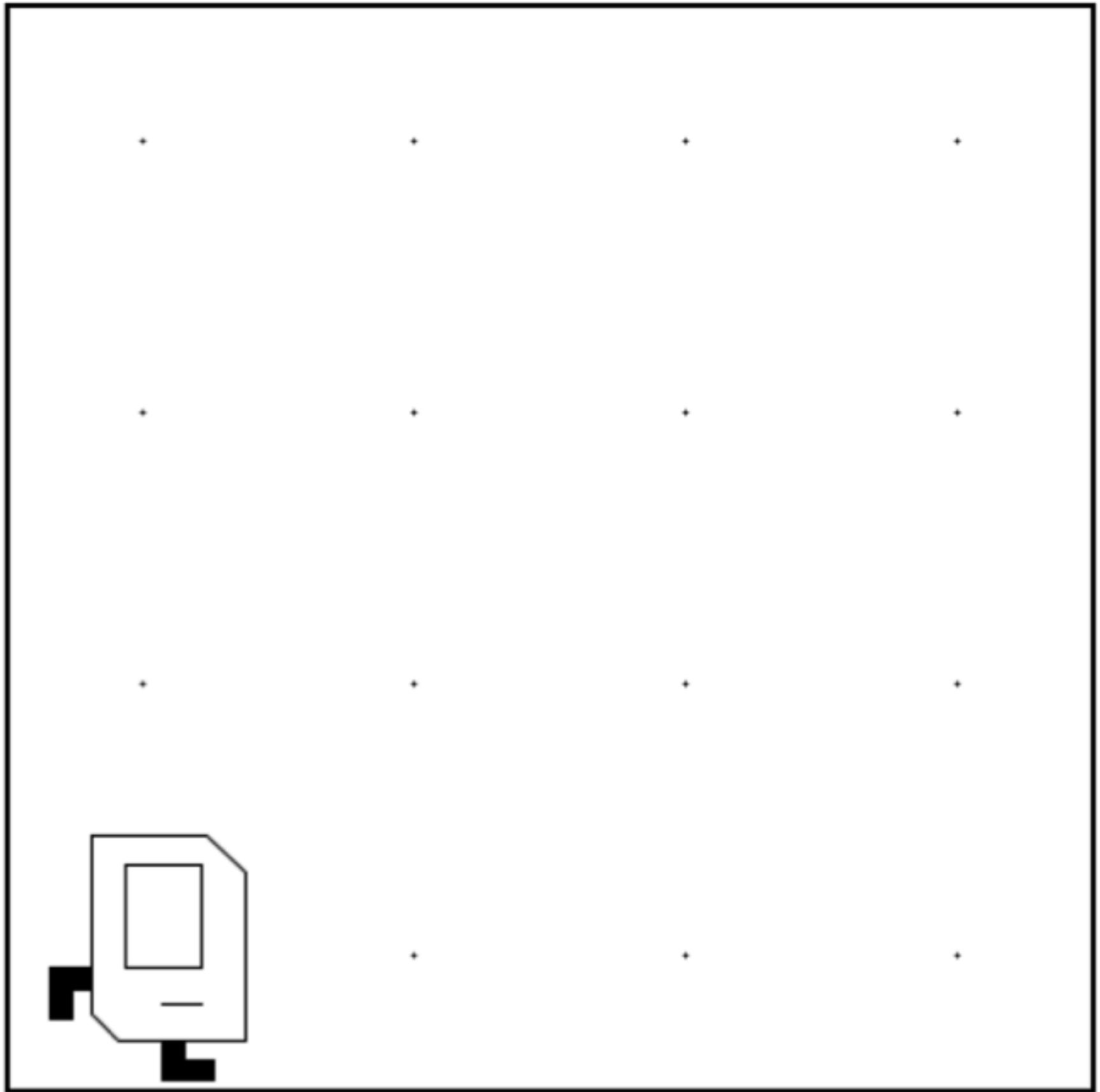


**Boucle "tant que"**

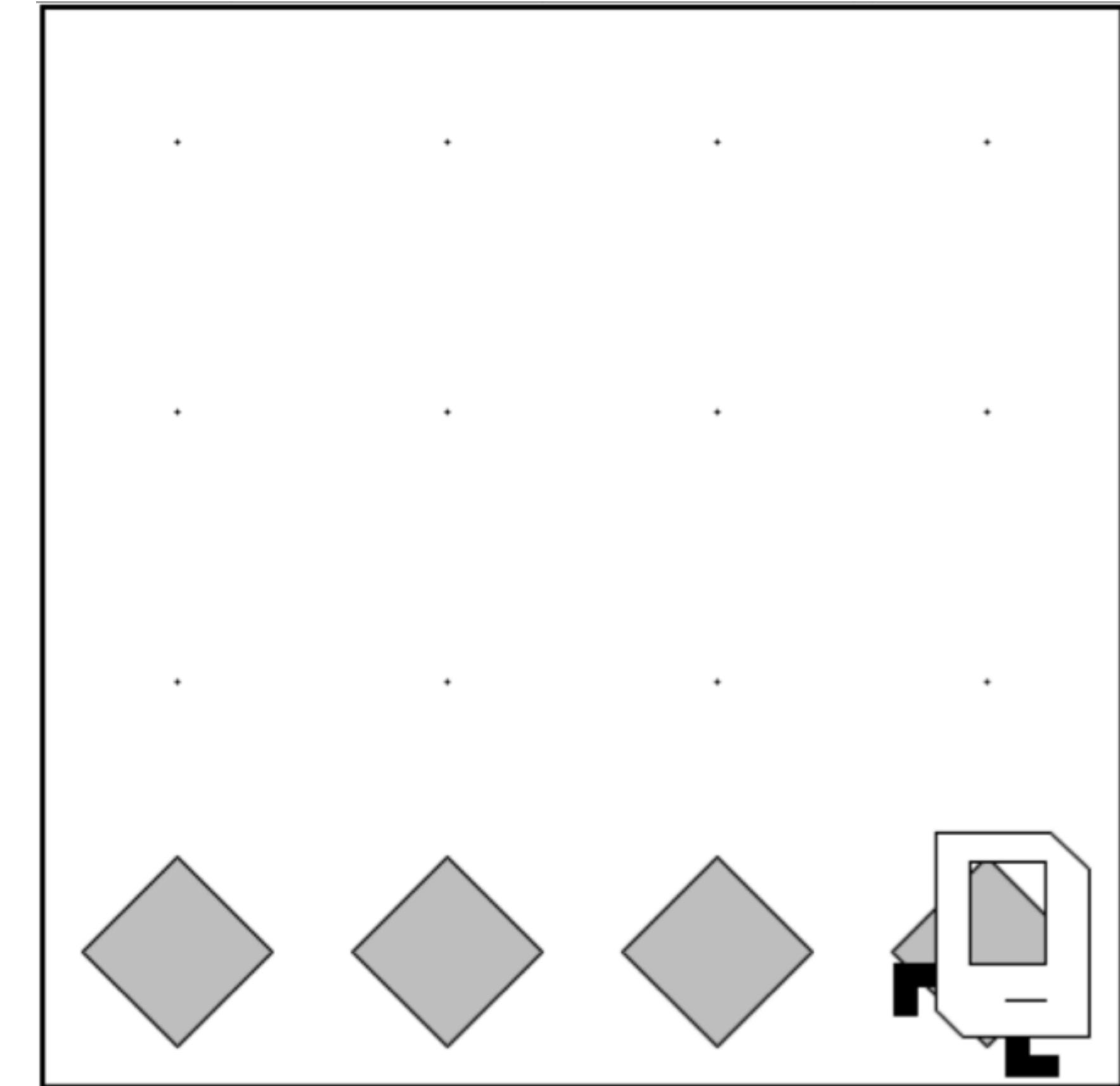
**"while" Loop**

Placer une ligne de beepers:  
Monde de taille (4 lignes, 4 colonnes)

Avant

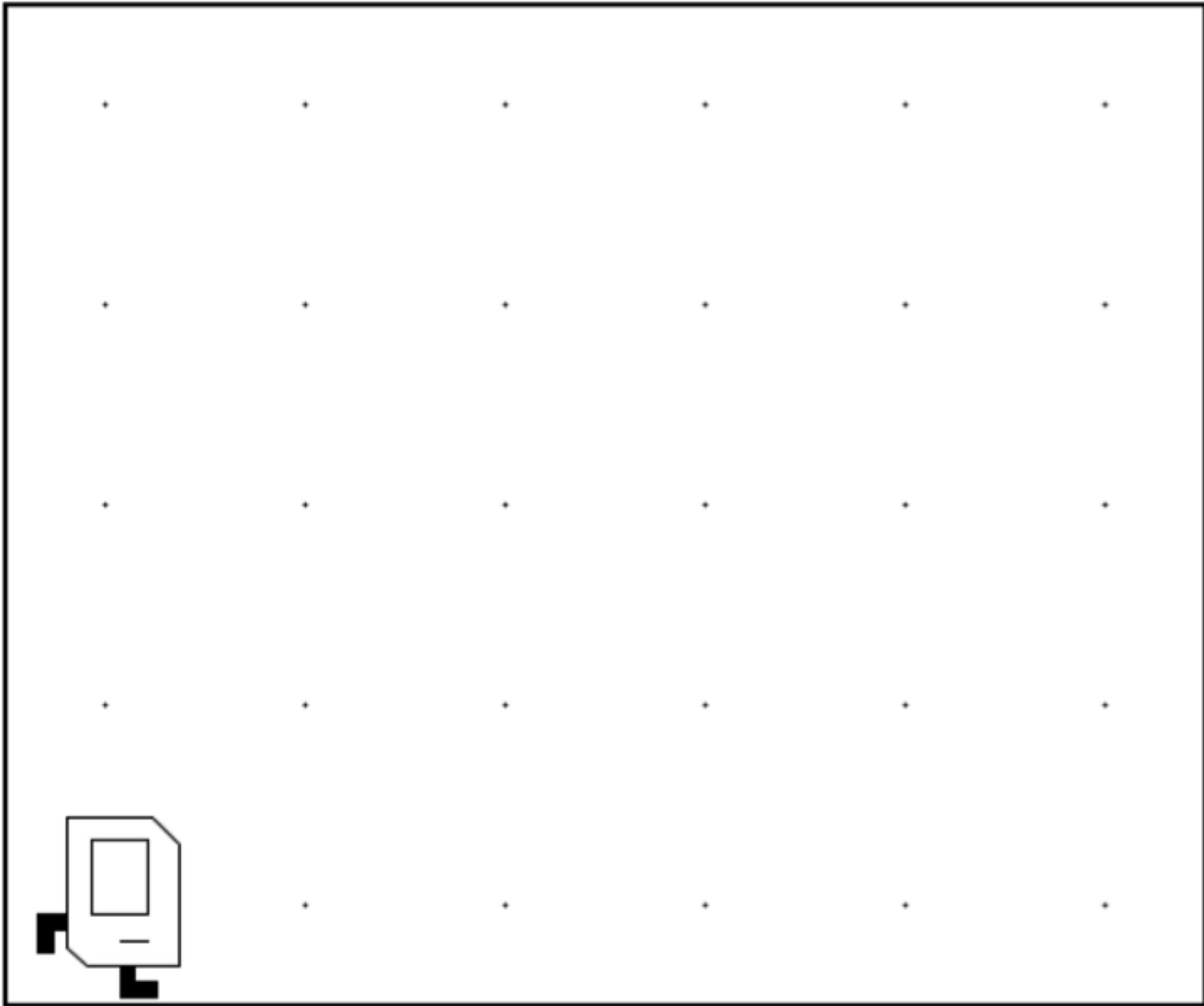


Après

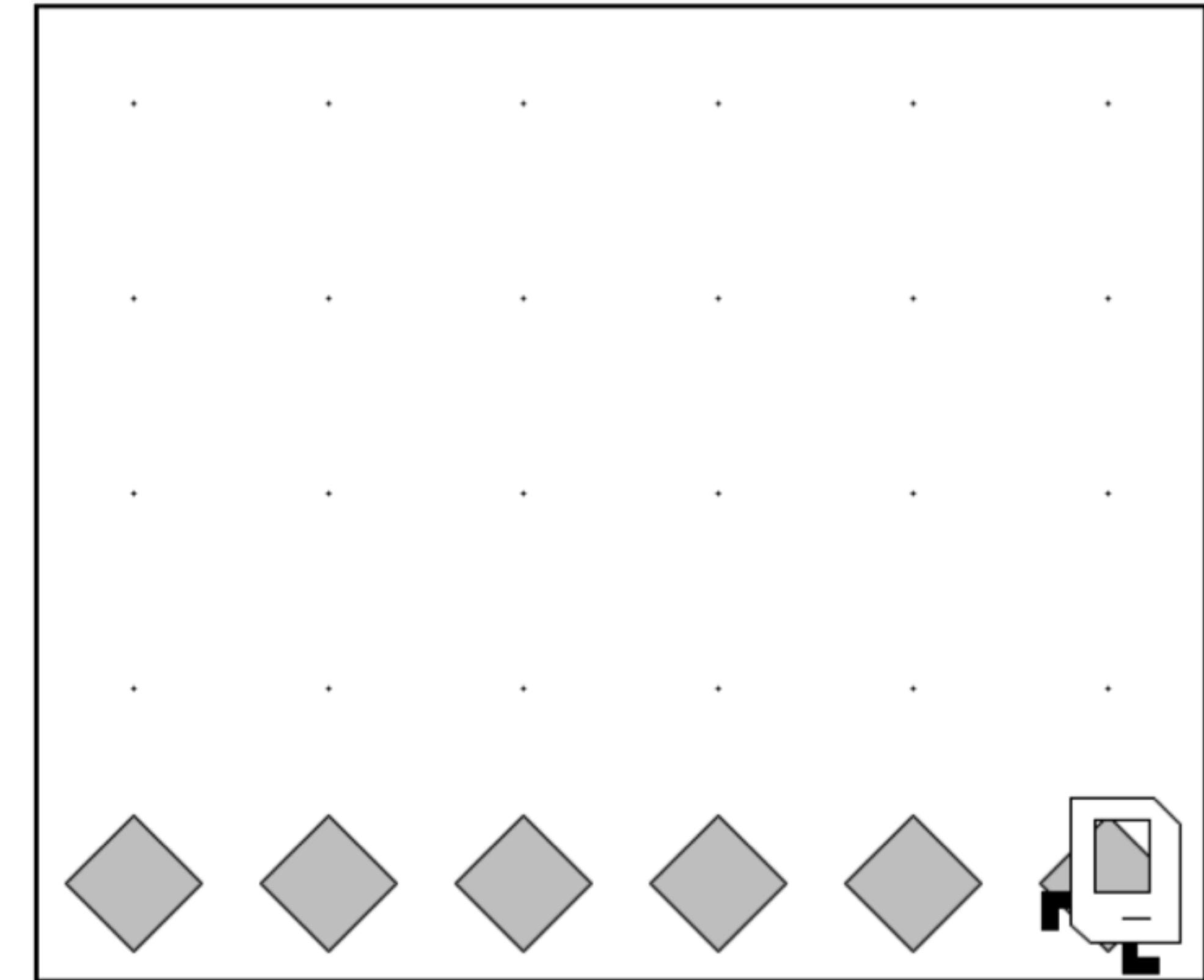


Placer une ligne de beepers:  
Monde de taille (5 lignes, 6 colonnes)

Avant

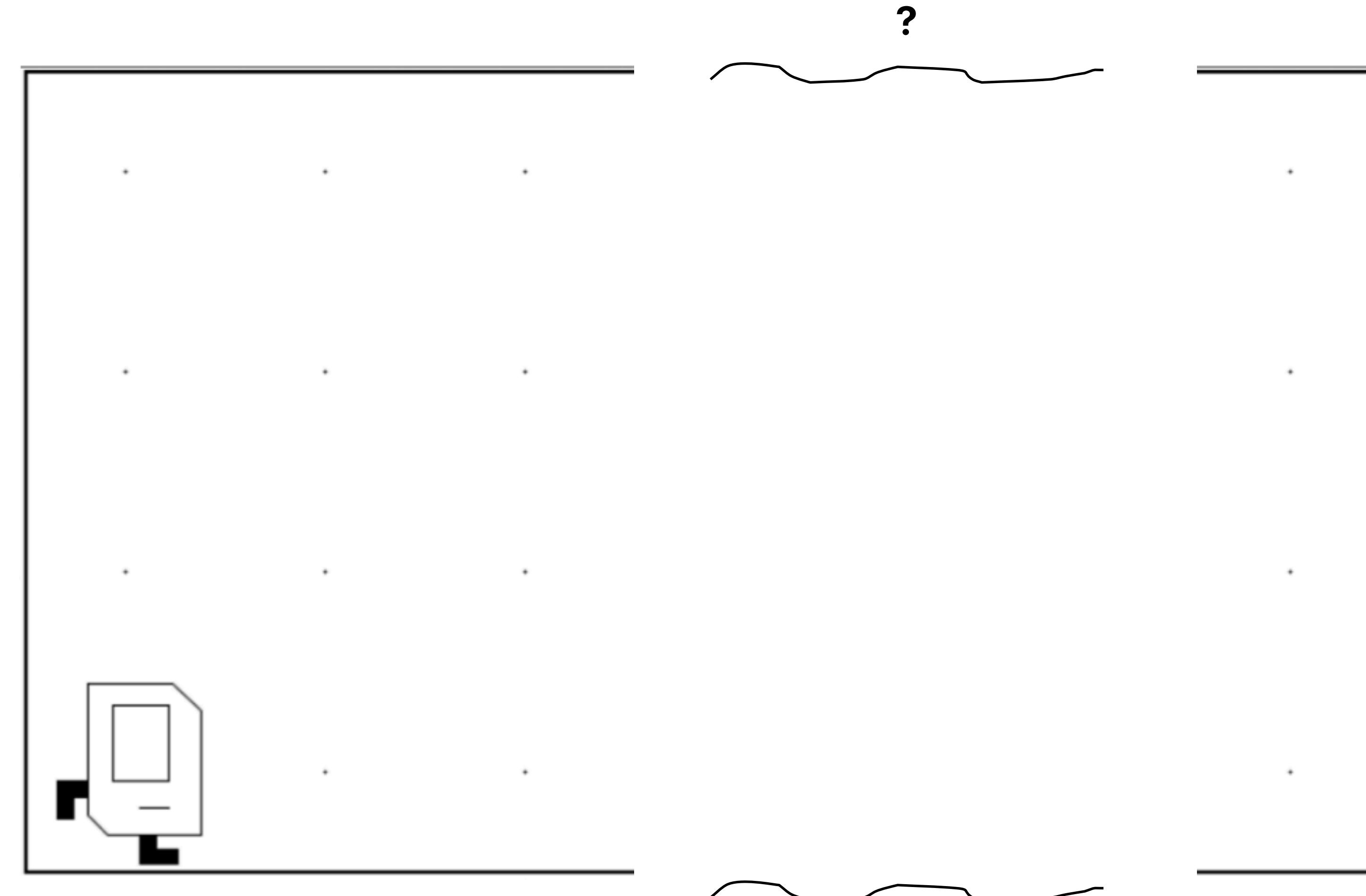


Après



Placer une ligne de beepers:

Comment le faire dans un monde de n'importe quel taille?



## Boucle "tant que"

```
while(condition) {  
    . . .  
}
```

## Boucle "tant que"

Une expression booléenne:  
Est soit vrai (true) ou faux (false)

Exemple: **frontIsClear()**  
(est ce que l'avant est dégagé ?)

```
while(condition) {
```

```
}
```

## Boucle "tant que"

Une expression booléenne:  
Est soit vrai (true) ou faux (false)

Exemple: **frontIsClear()**  
(est ce que l'avant est dégagé ?)

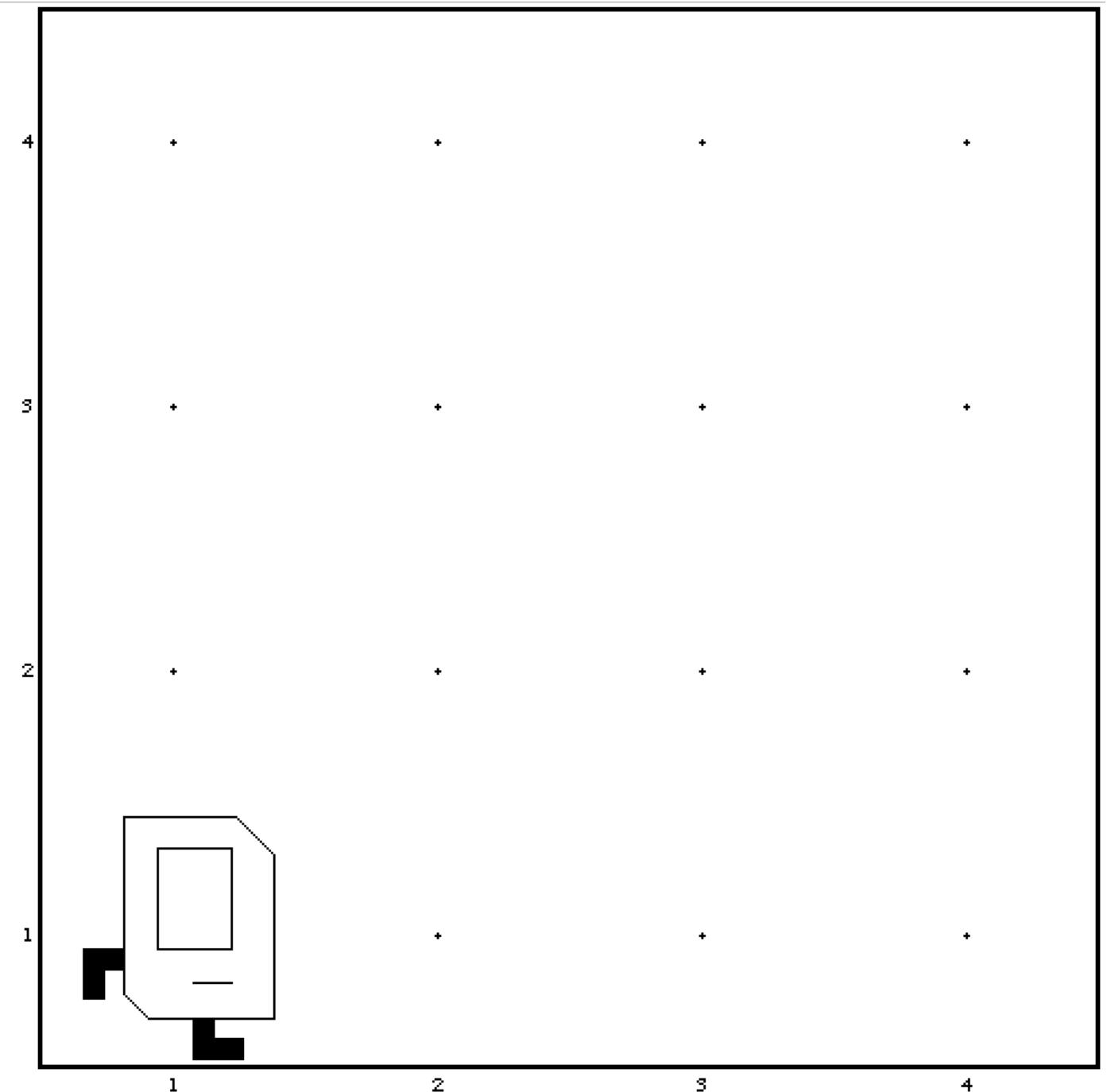
```
while(condition) {  
    // instructions à répéter  
    // tant que condition est vrai  
}
```

## Boucle "tant que"

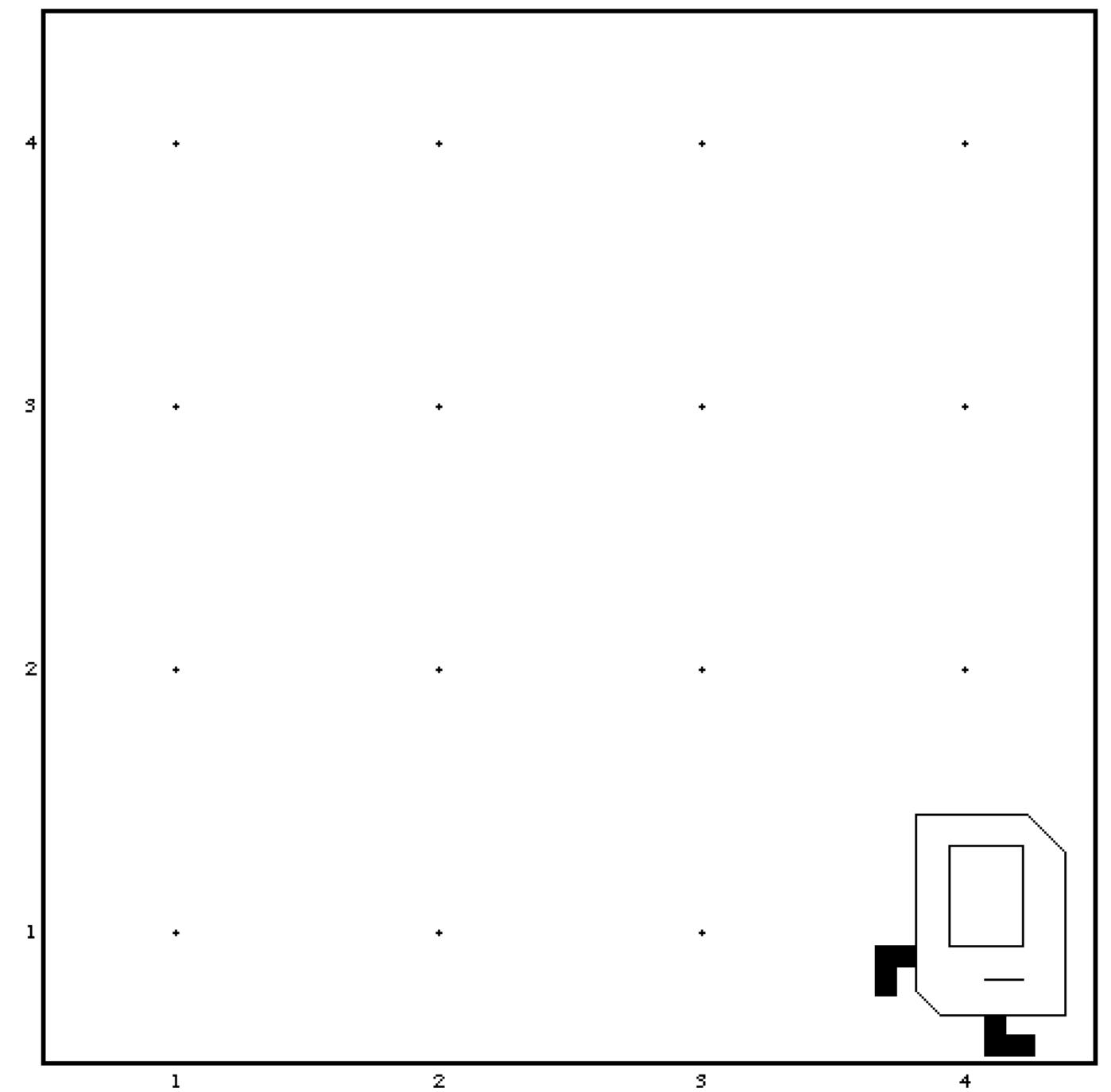
Exemple: Aller au mur

```
private void allerAuMur() {  
  
    while(frontIsClear()) {  
        move();  
    }  
}
```

Avant



Après

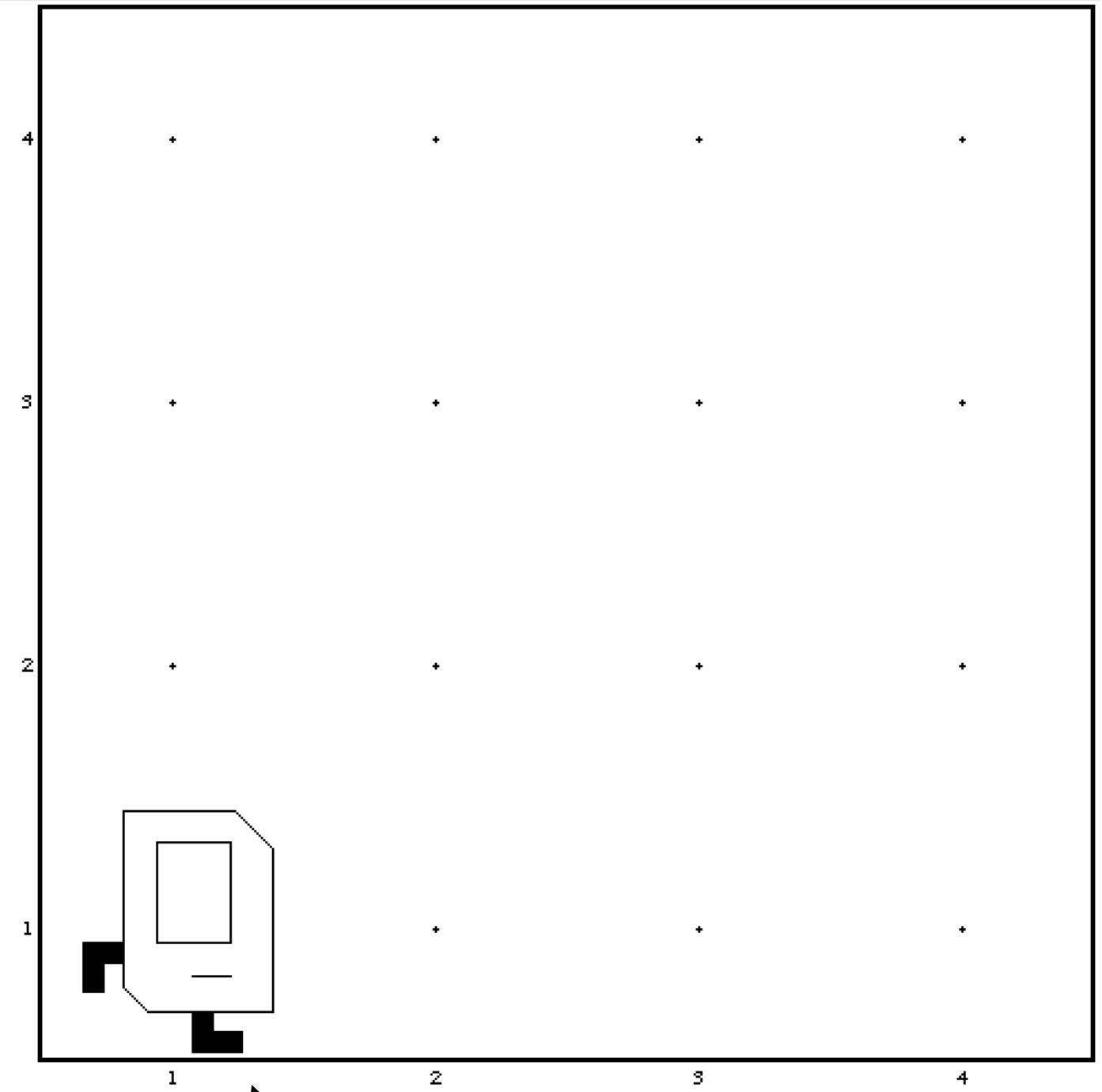


## Boucle "tant que"

Exemple: Aller au mur

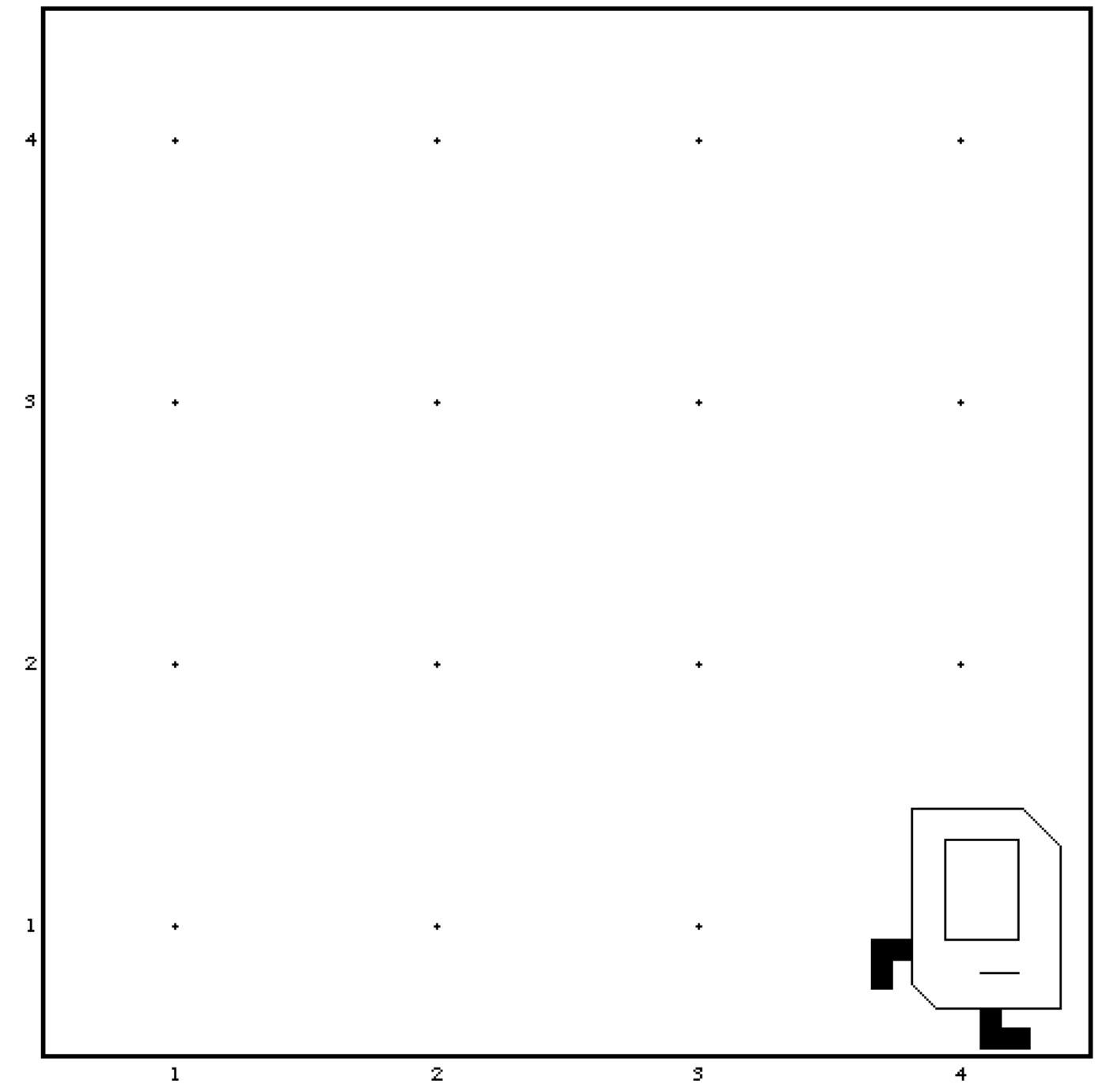
```
private void allerAuMur() {  
  
    while(frontIsClear()) {  
        move();  
    }  
}
```

Avant



frontIsClear()  
a la valeur true (vrai)

Après

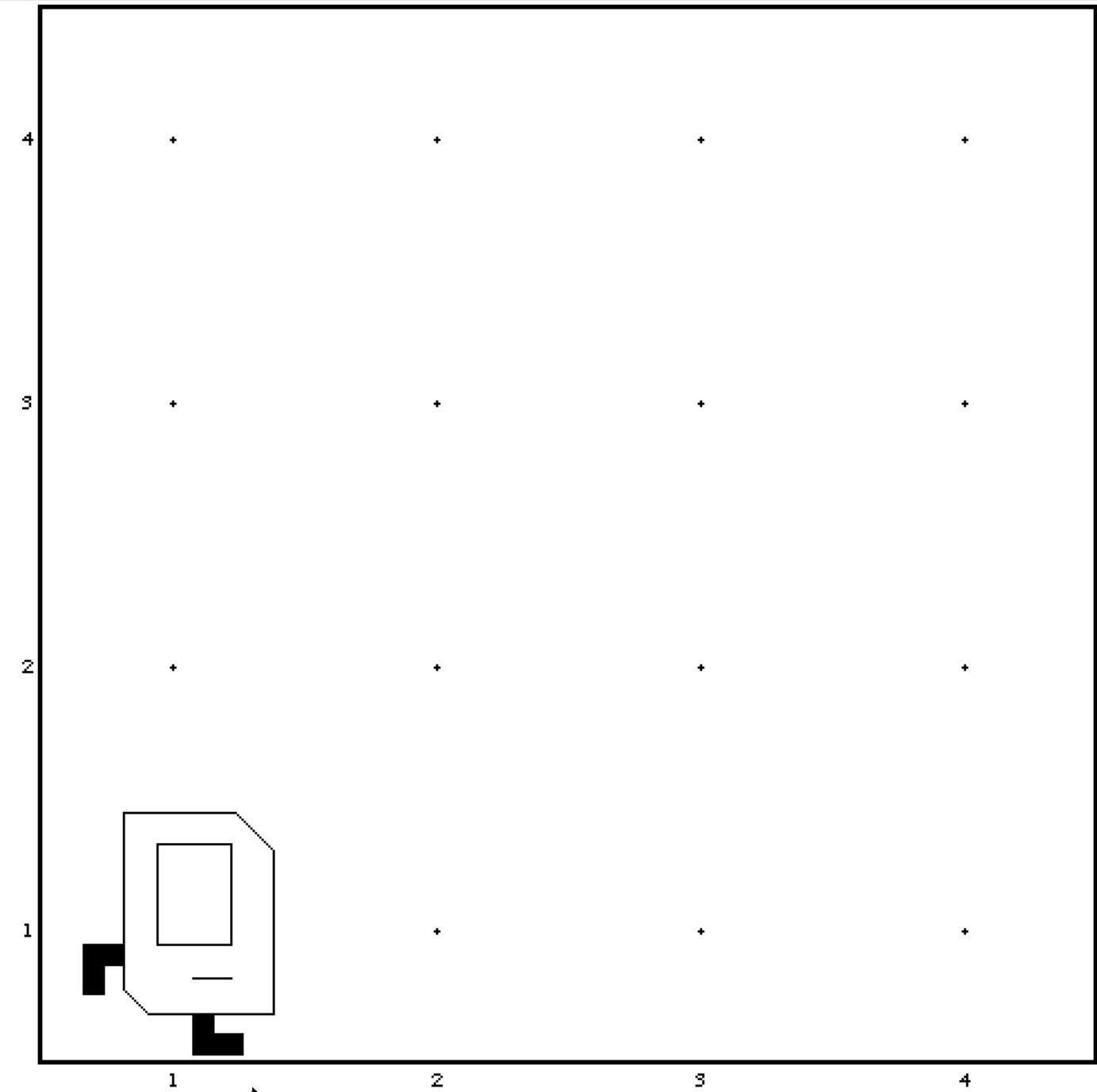


## Boucle "tant que"

Exemple: Aller au mur

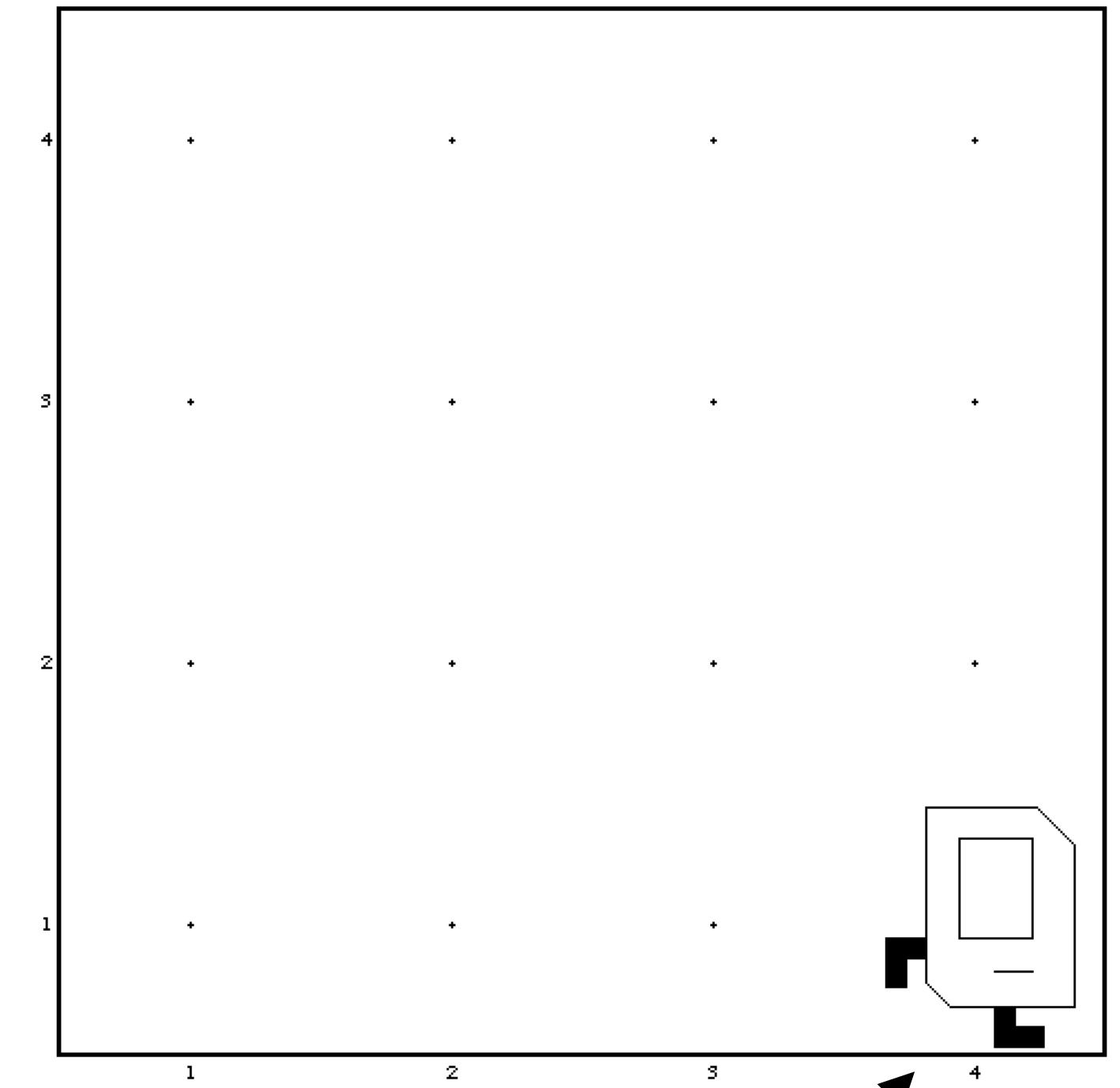
```
private void allerAuMur() {  
  
    while(frontIsClear()) {  
        move();  
    }  
}
```

Avant



**frontIsClear()**  
a la valeur **true** (vrai)

Après



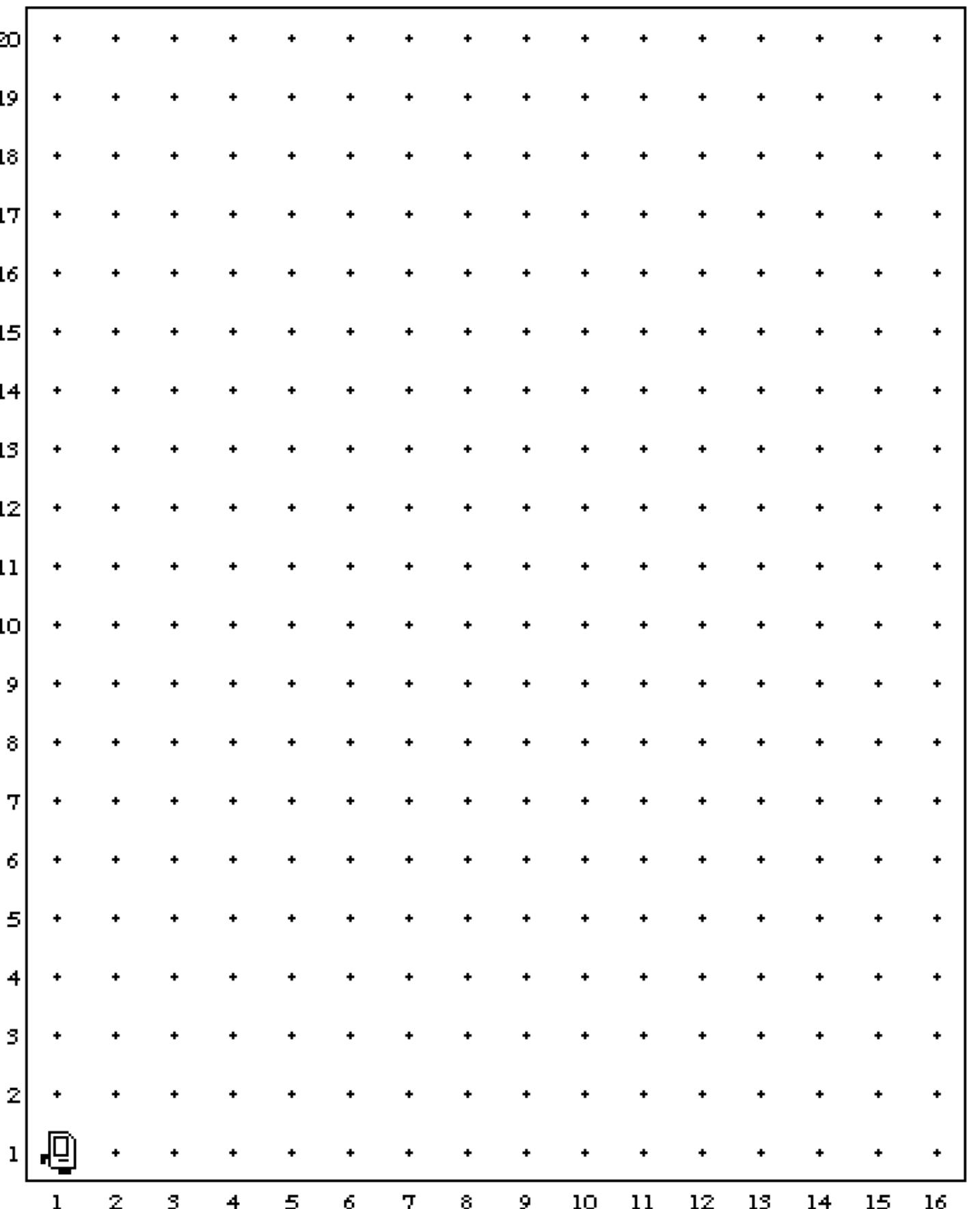
**frontIsClear()**  
a la valeur **false** (faux)

## Boucle "tant que"

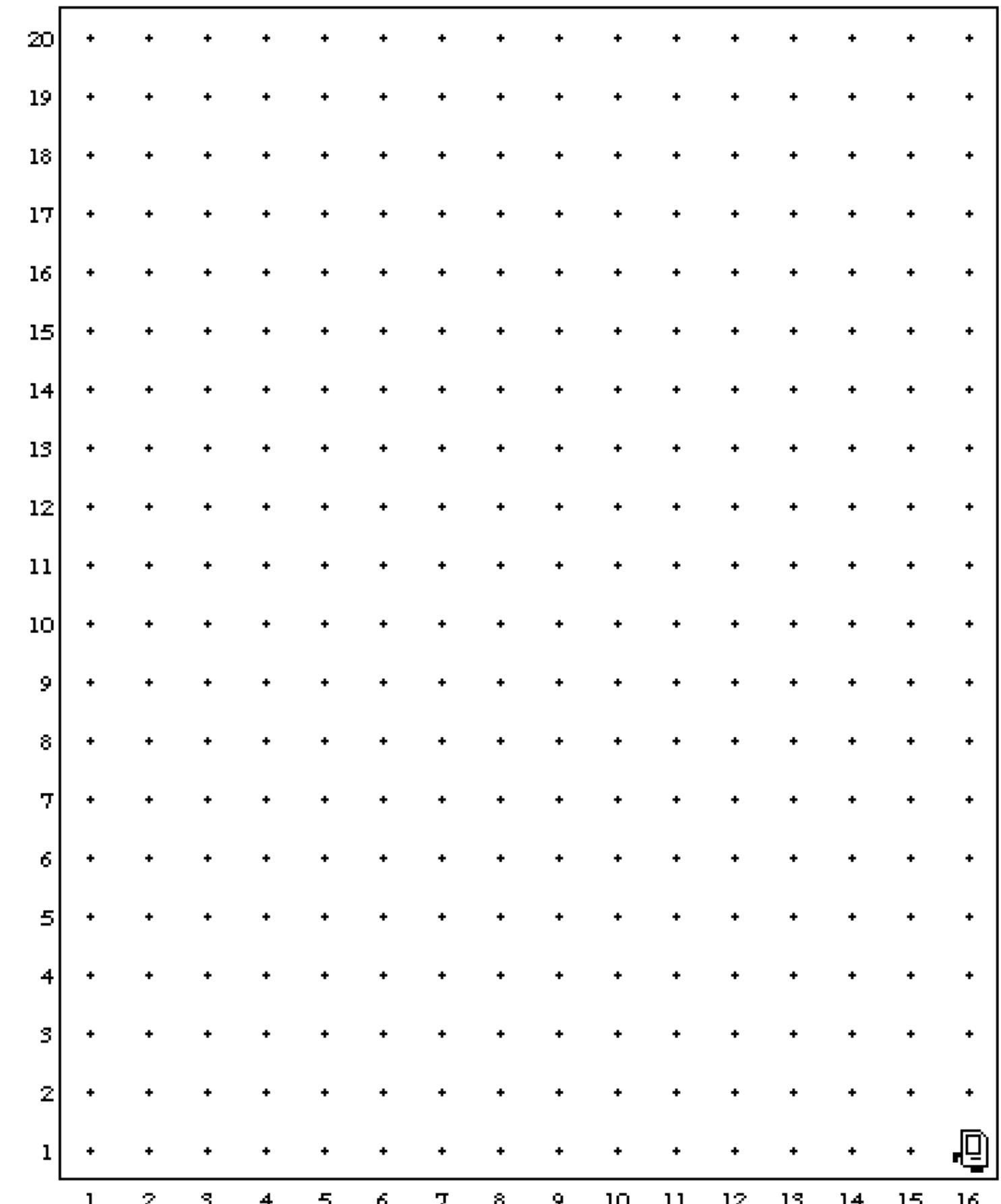
Exemple: Aller au mur

```
private void allerAuMur() {  
  
    while(frontIsClear()) {  
        move();  
    }  
  
}
```

Avant



Après

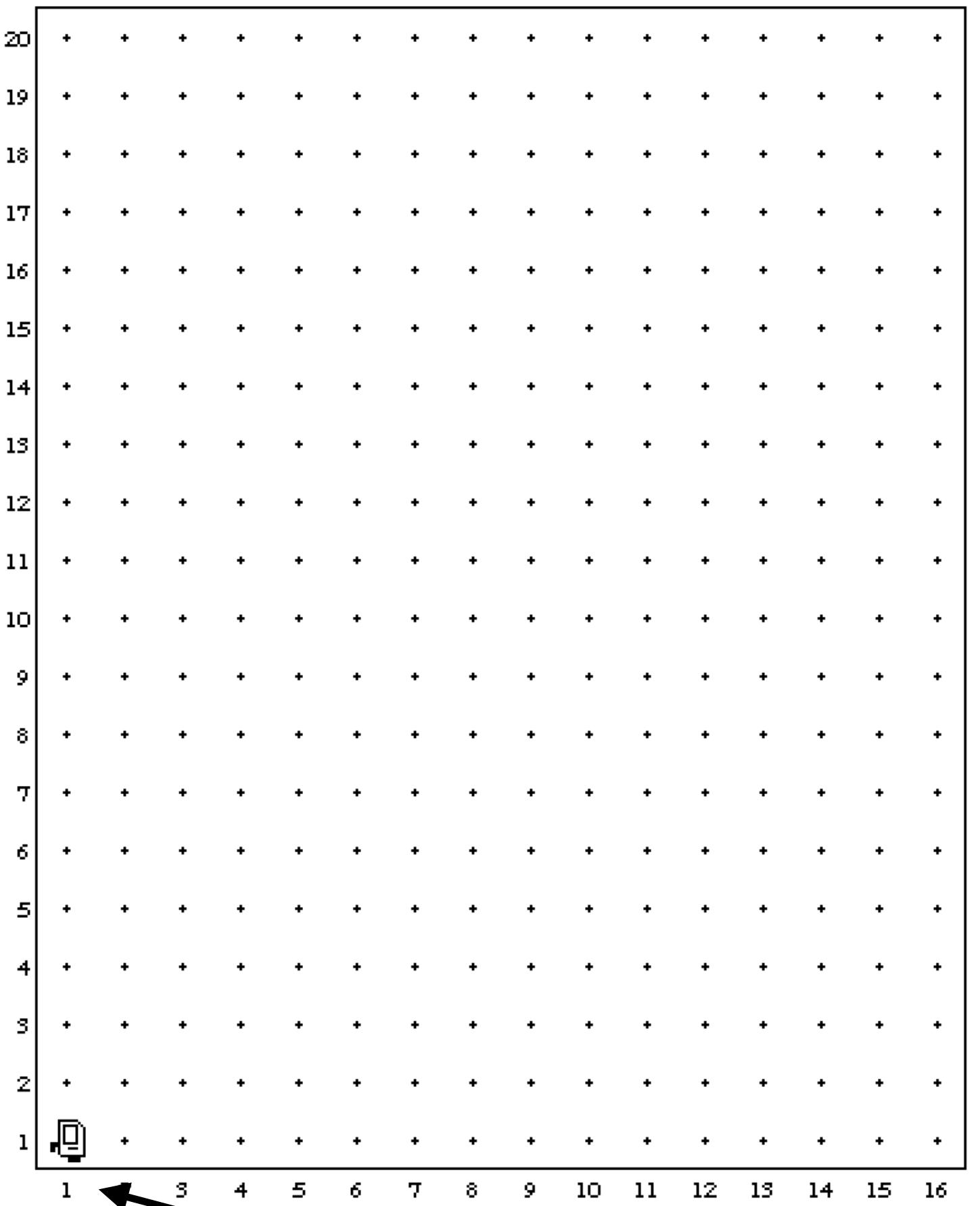


## Boucle "tant que"

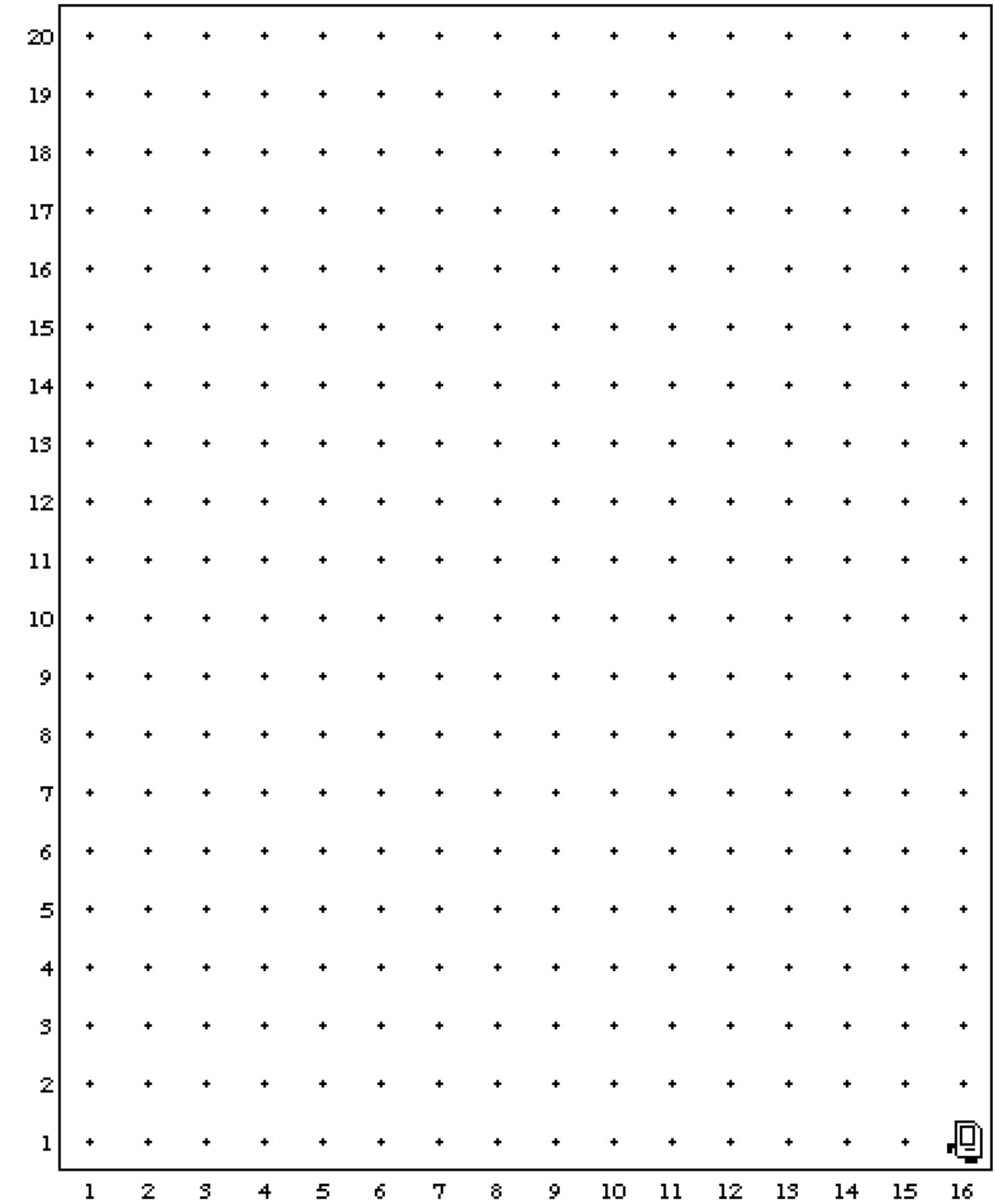
Exemple: Aller au mur

```
private void allerAuMur() {  
  
    while(frontIsClear()) {  
        move();  
    }  
  
}
```

Avant



Après



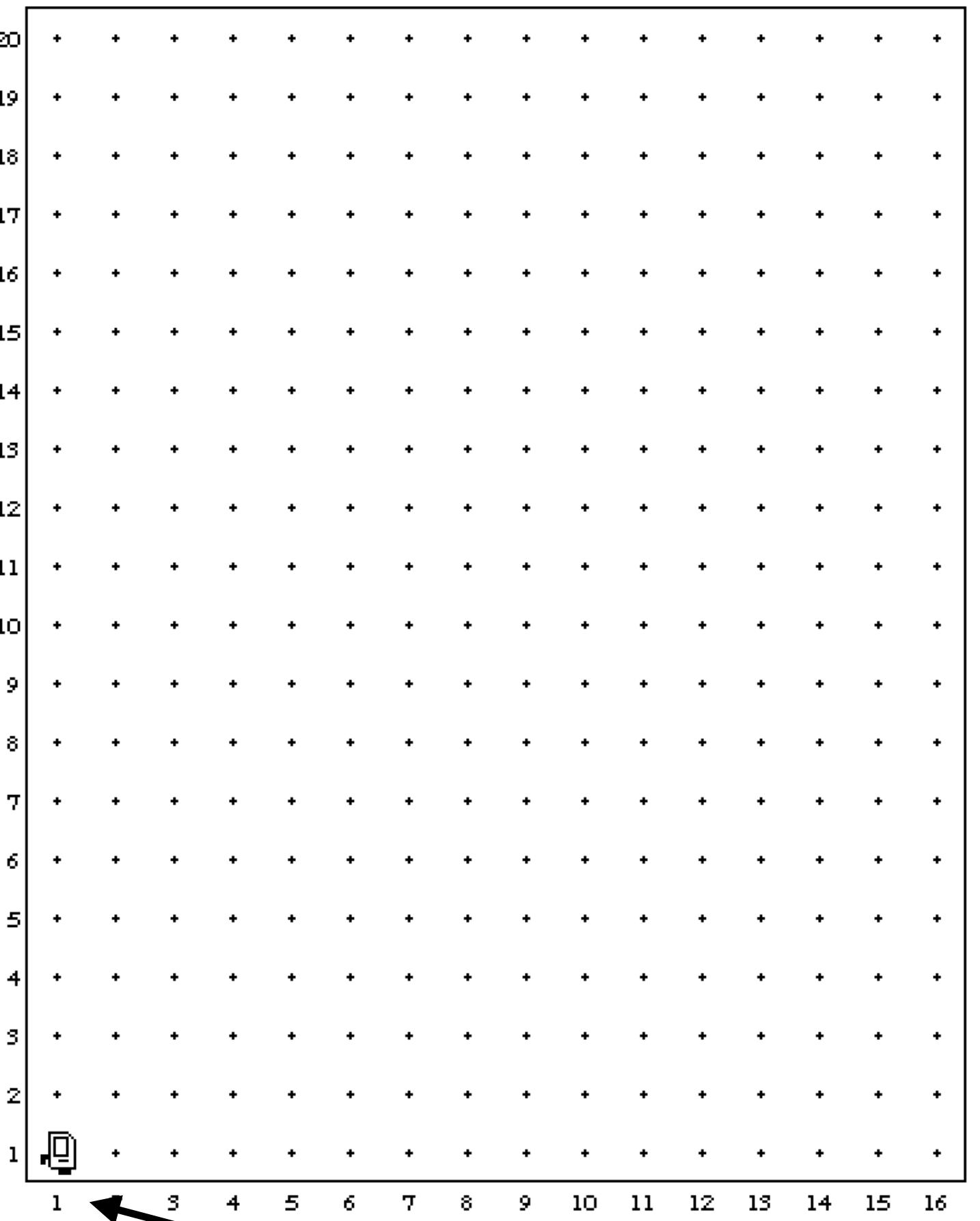
**frontIsClear()**  
a la valeur **true** (vrai)

## Boucle "tant que"

Exemple: Aller au mur

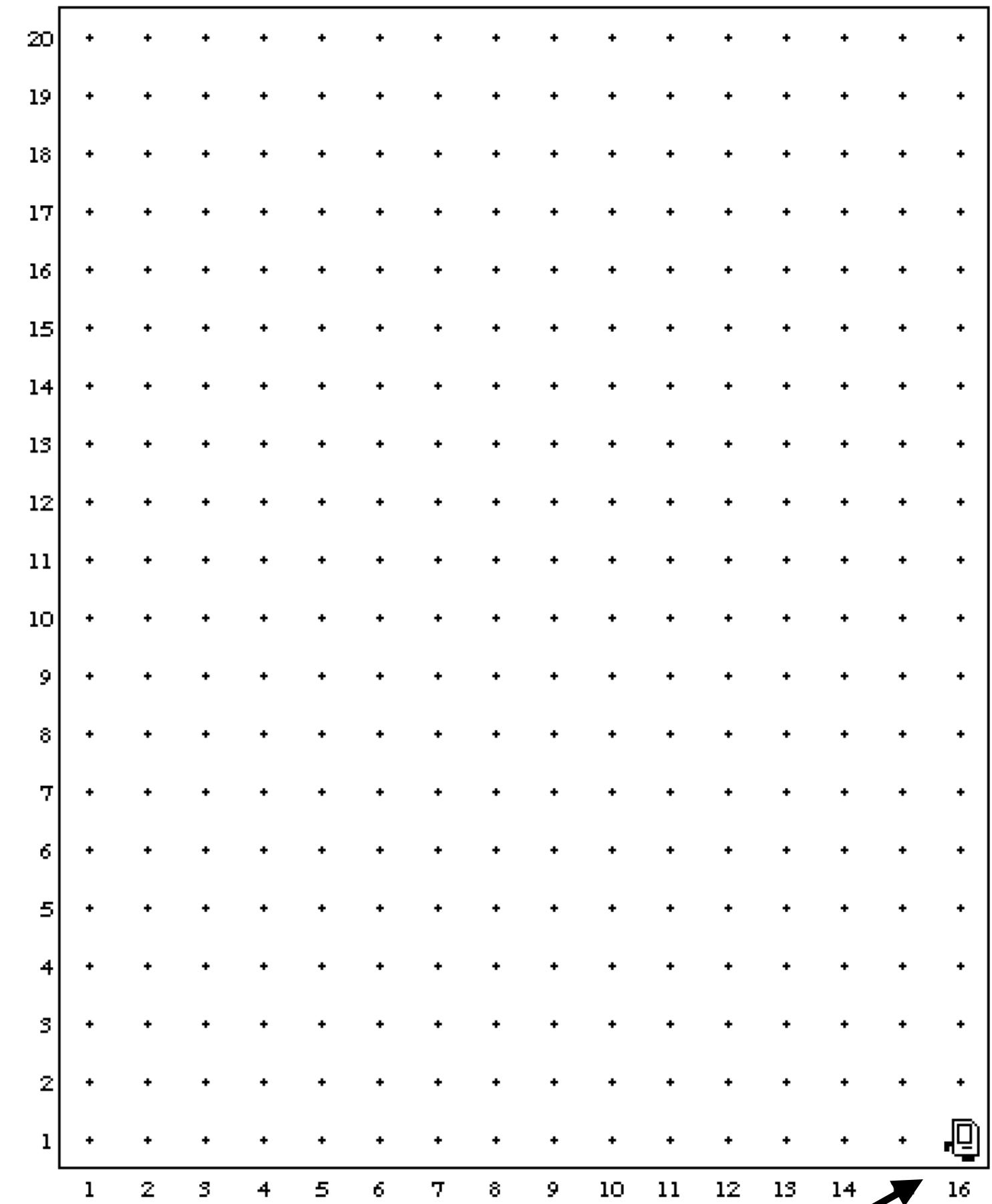
```
private void allerAuMur() {  
  
    while(frontIsClear()) {  
        move();  
    }  
  
}
```

Avant



**frontIsClear()**  
a la valeur **true** (vrai)

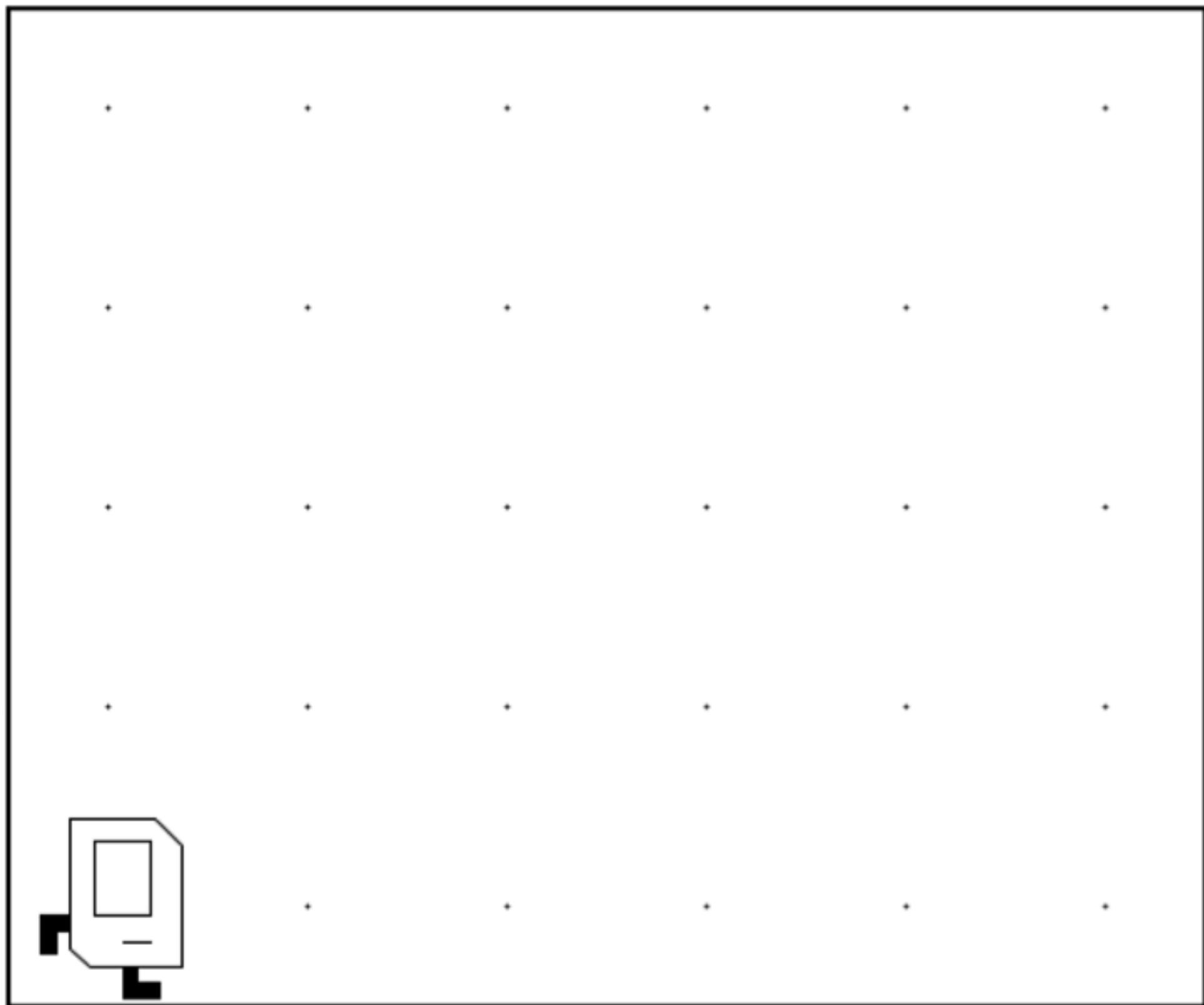
Après



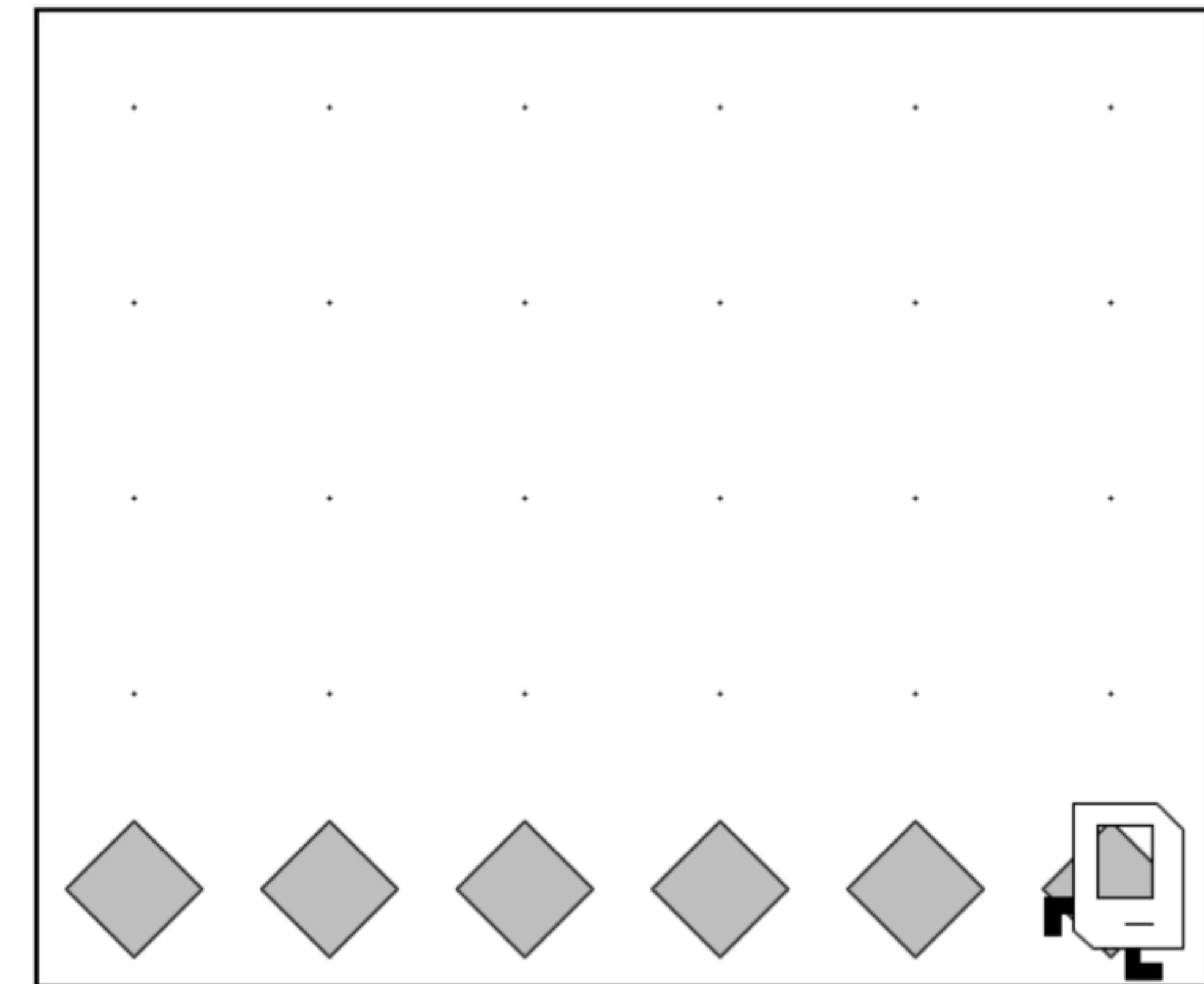
**frontIsClear()**  
a la valeur **false** (faux)

Placer une ligne de beepers:

Avant



Après



## Boucle "tant que"

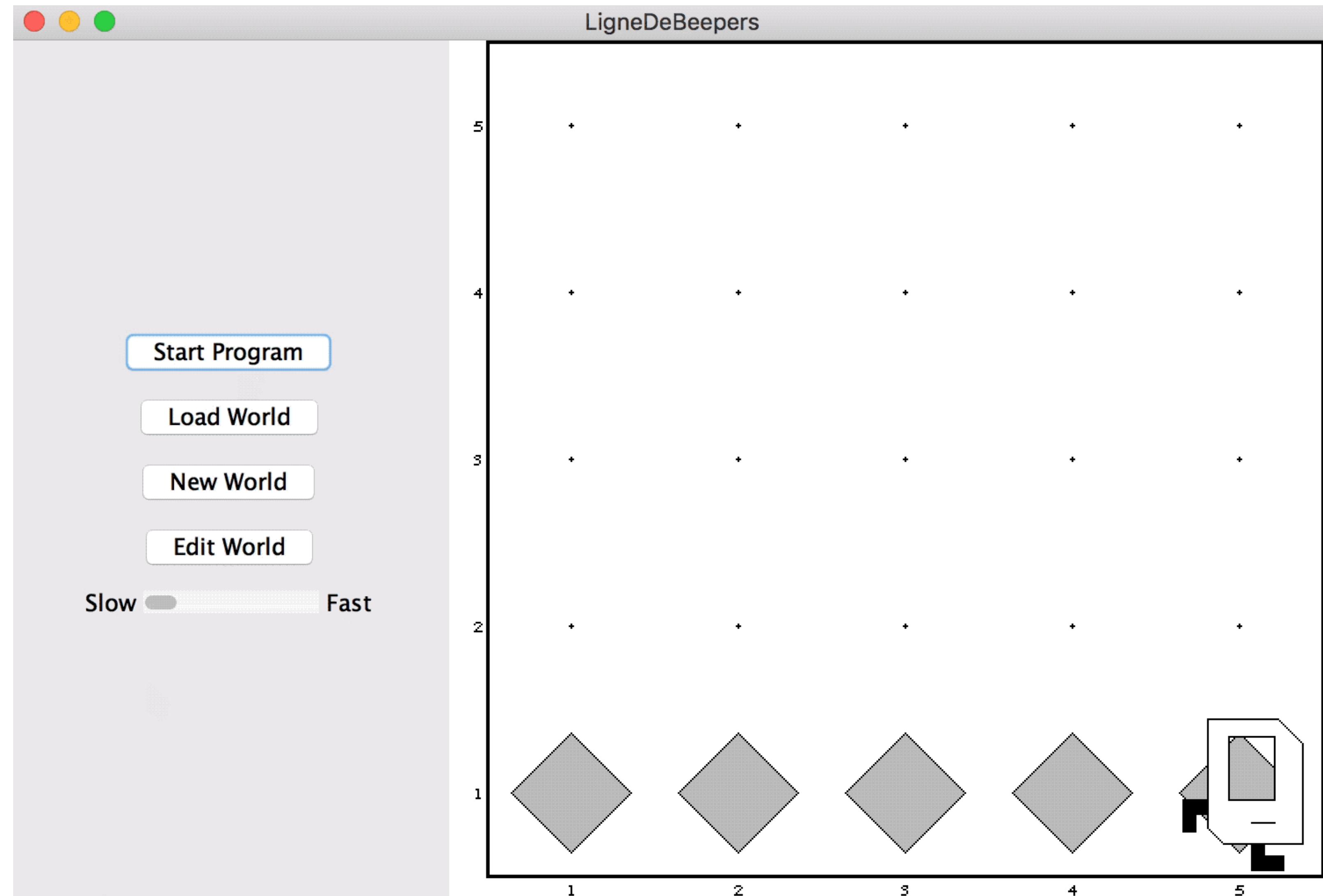
Exemple: Ligne de beepers

```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```

## Boucle "tant que"

Exemple: Ligne de beepers

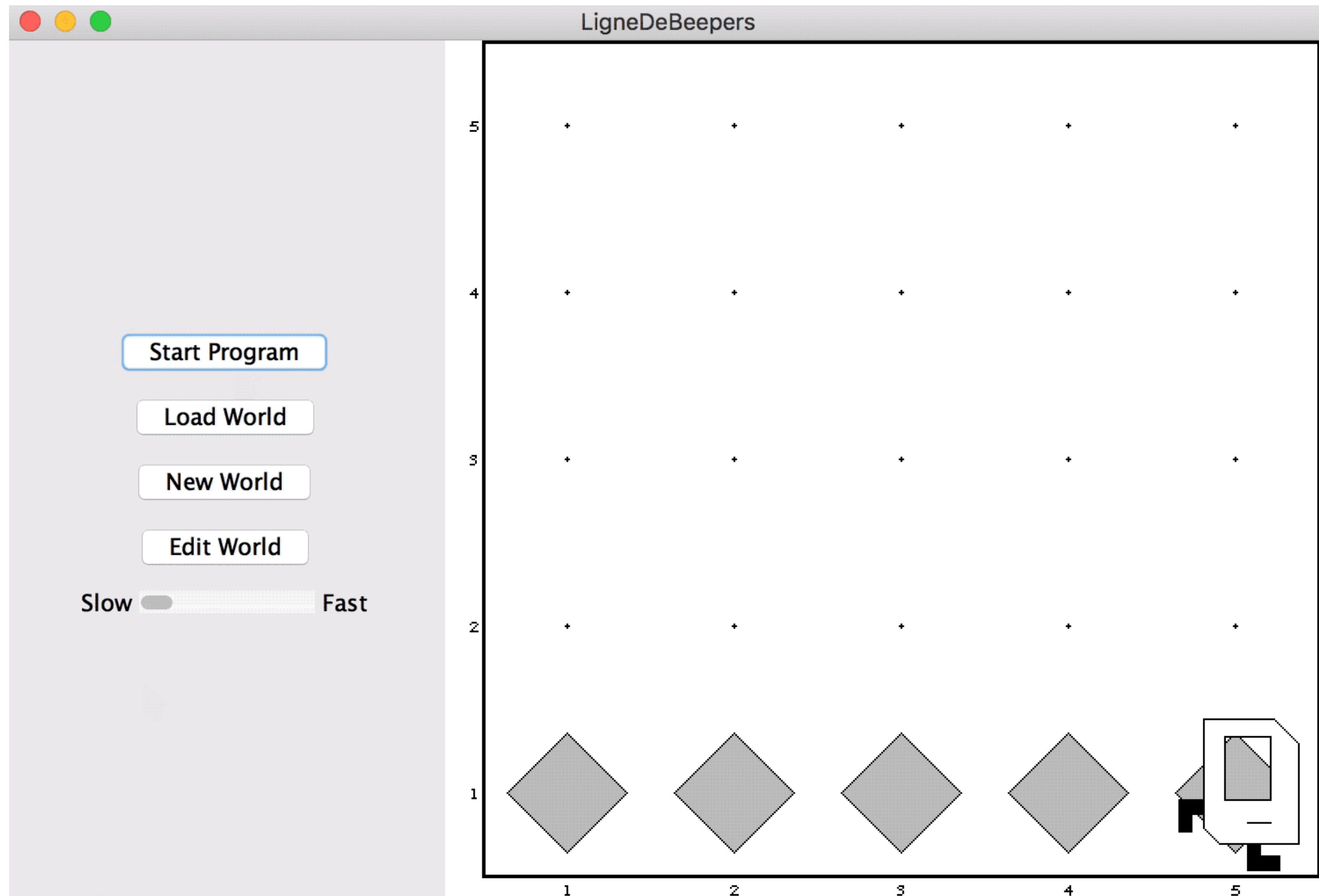
```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```



## Boucle "tant que"

Exemple: Ligne de beepers

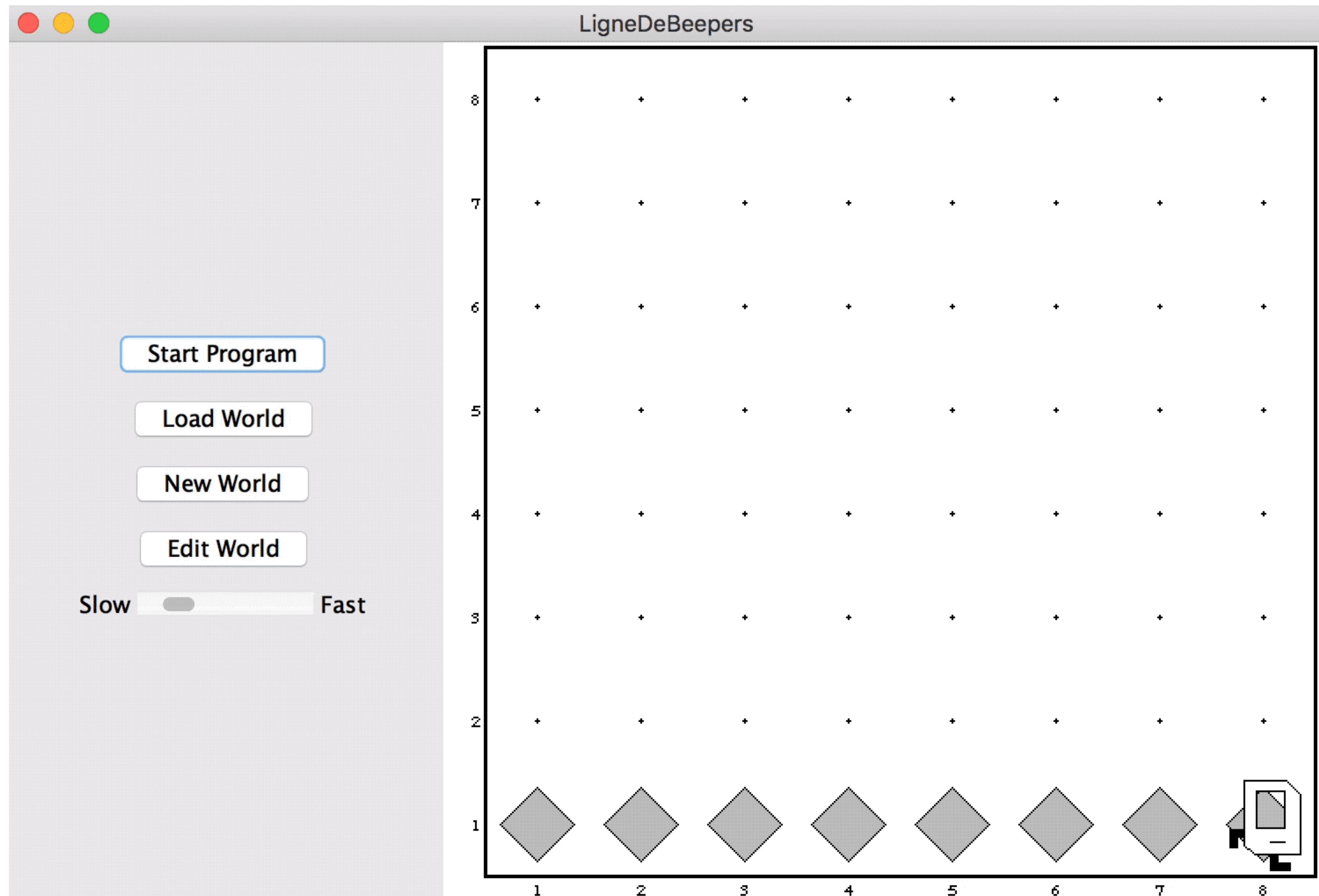
```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```



## Boucle "tant que"

Exemple: Ligne de beepers

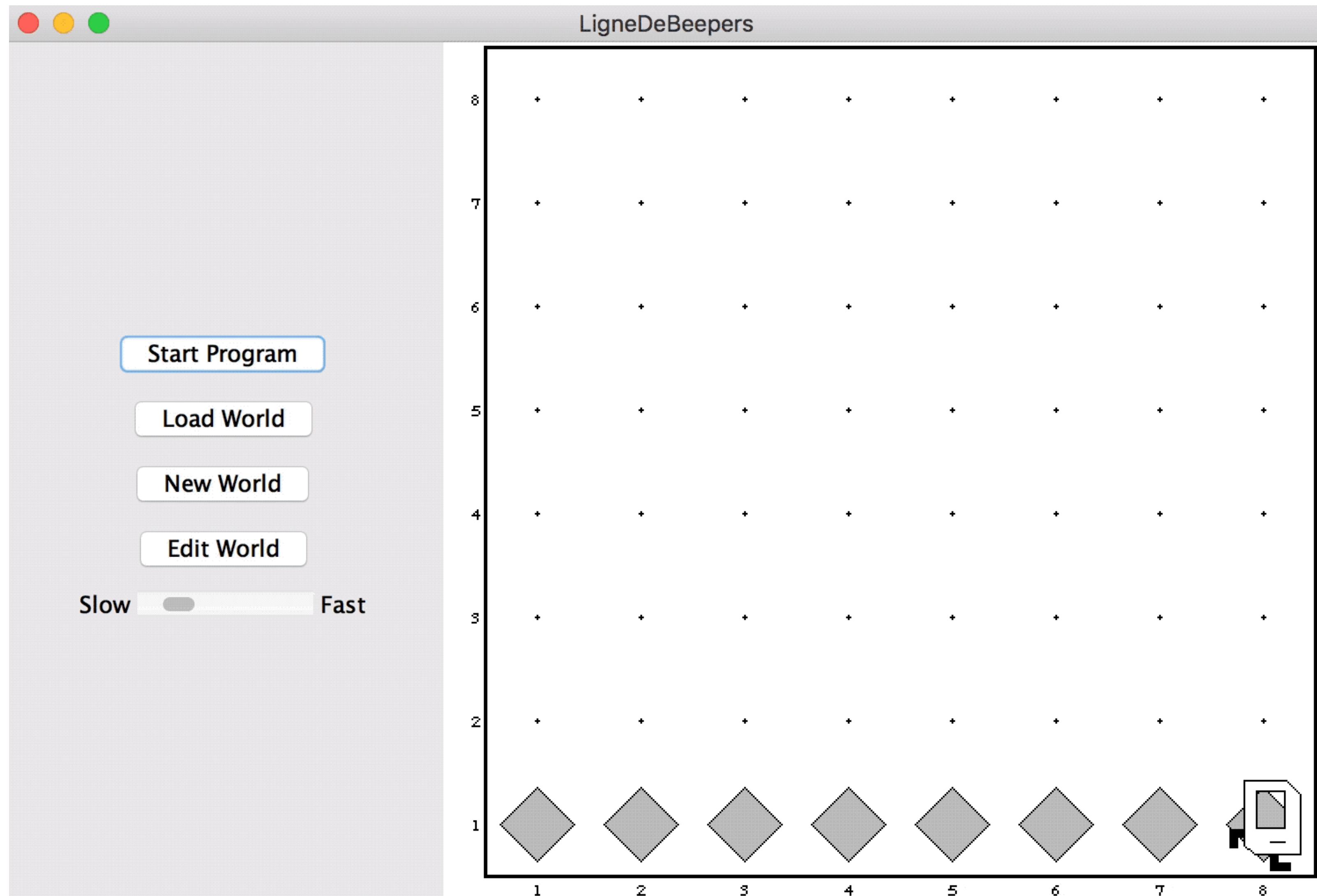
```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```



## Boucle "tant que"

Exemple: Ligne de beepers

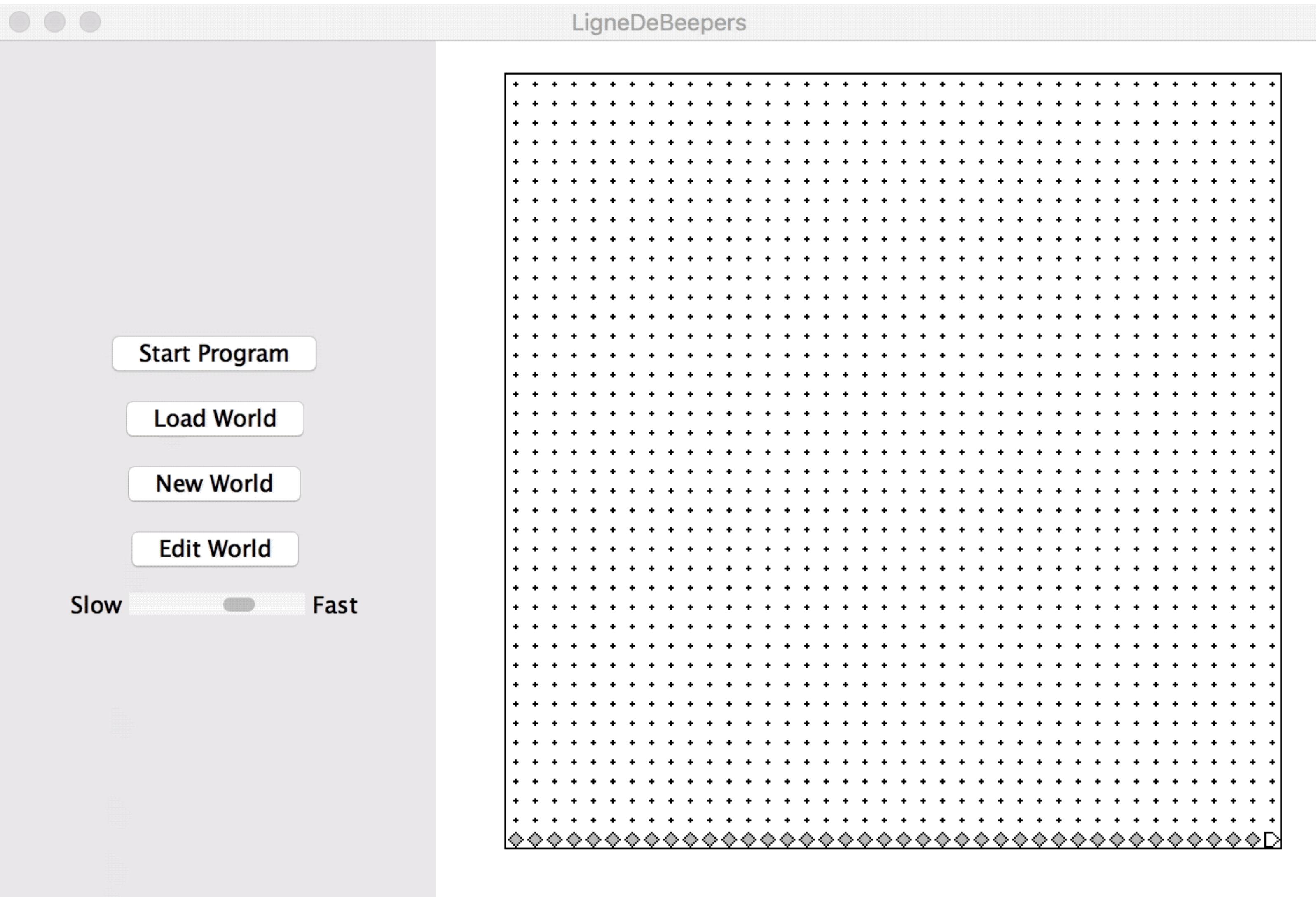
```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```



## Boucle "tant que"

Exemple: Ligne de beepers

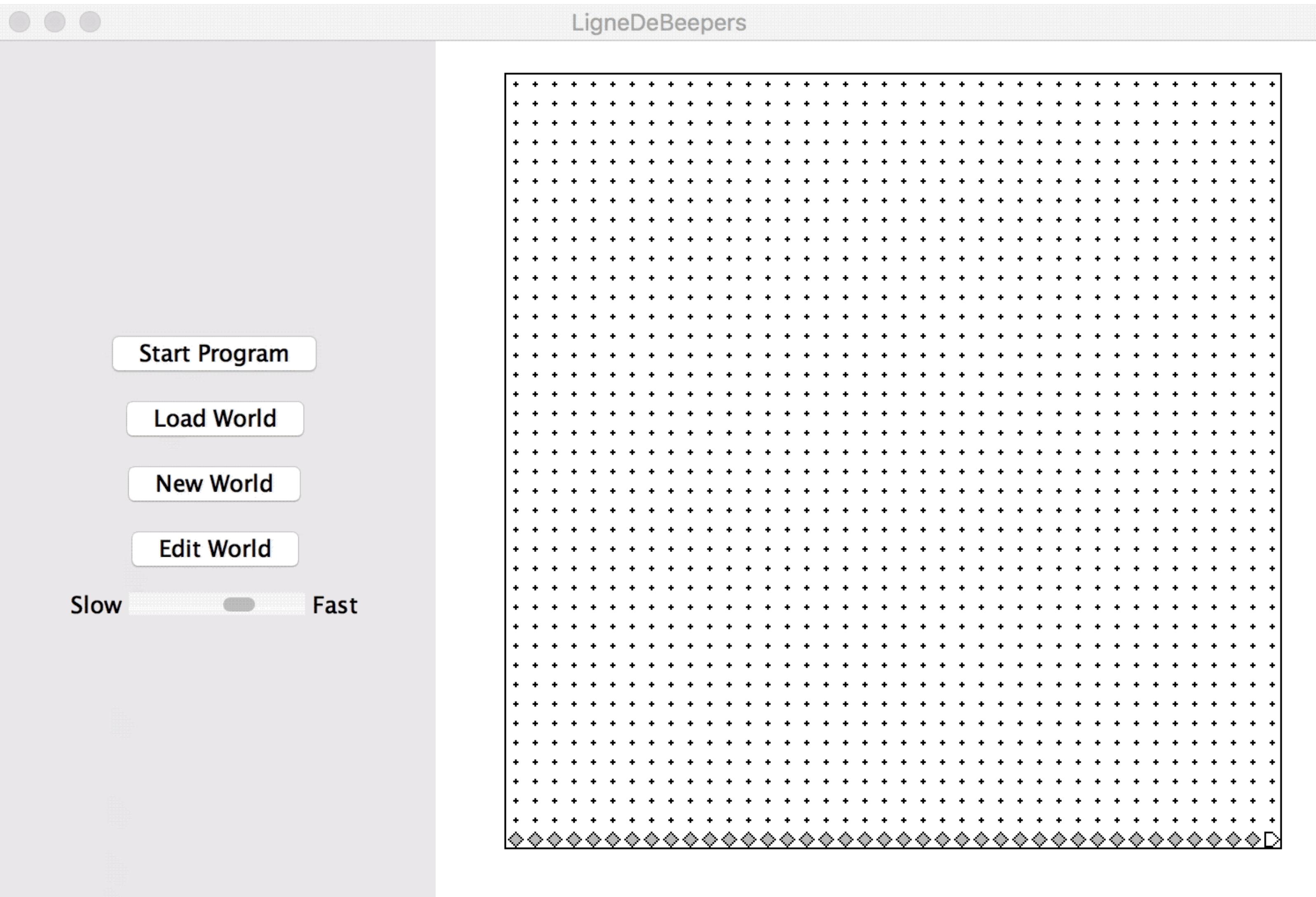
```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```



## Boucle "tant que"

Exemple: Ligne de beepers

```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```



## Boucle "tant que"

Exemple: Ligne de beepers

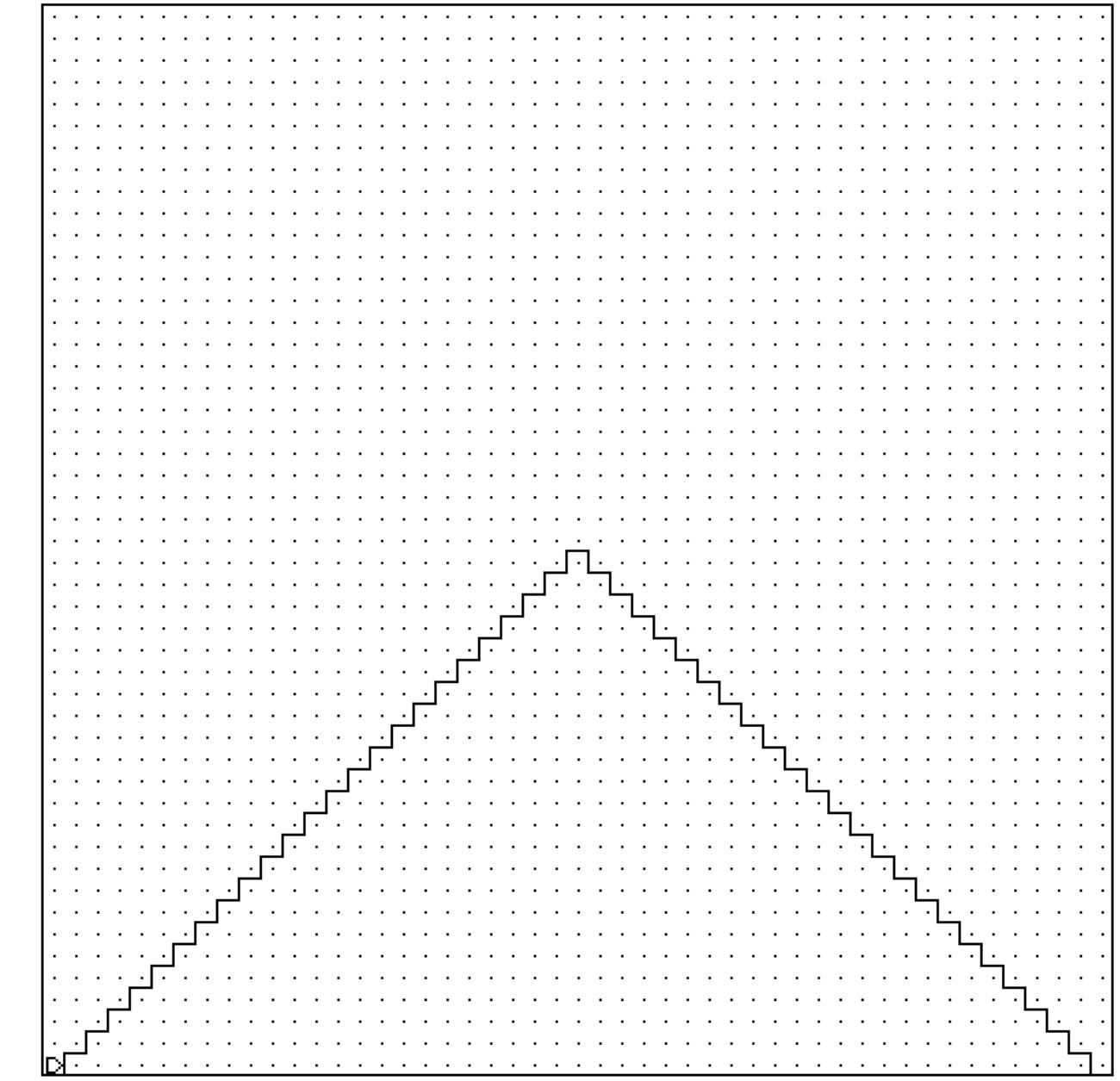
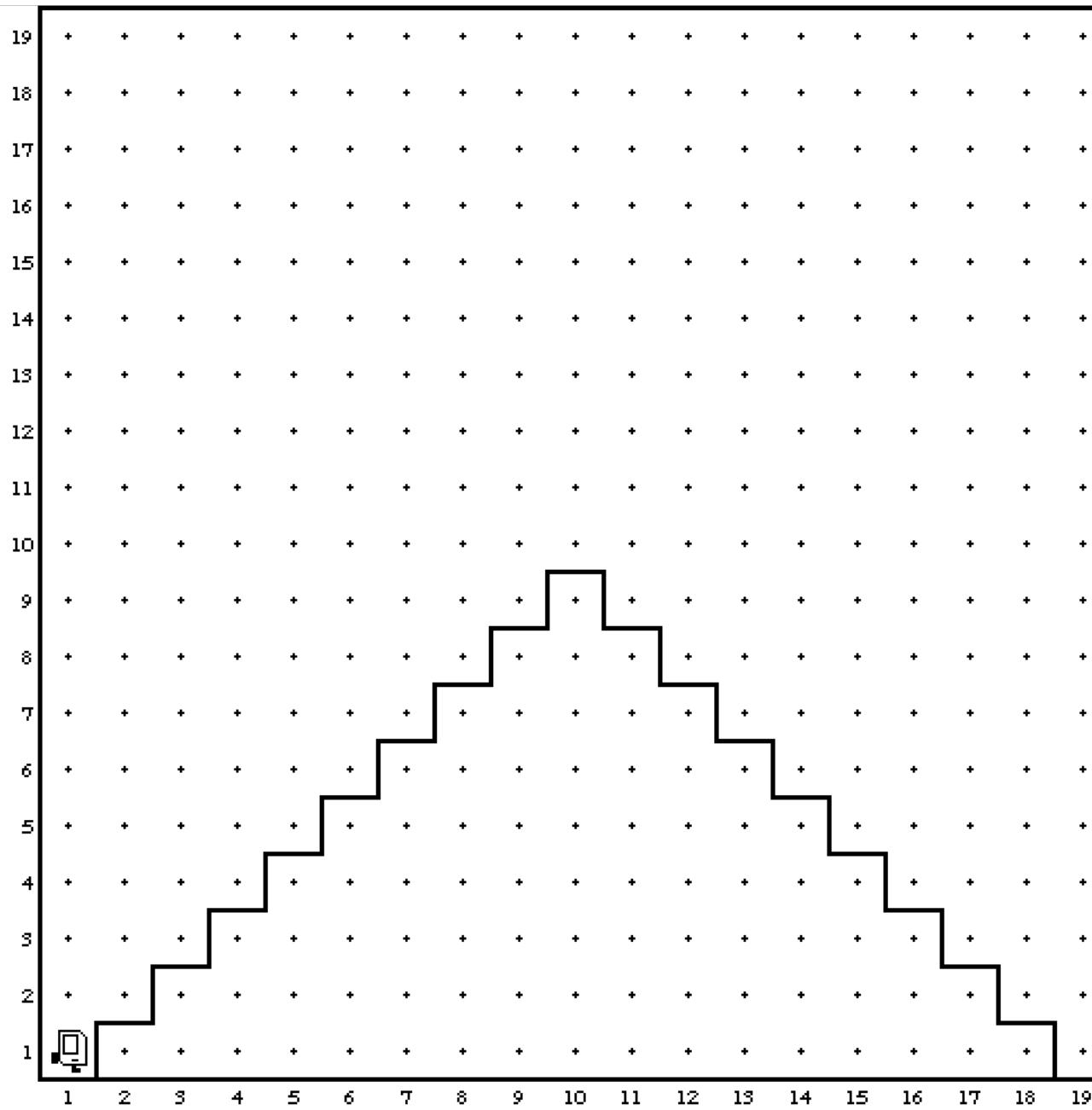
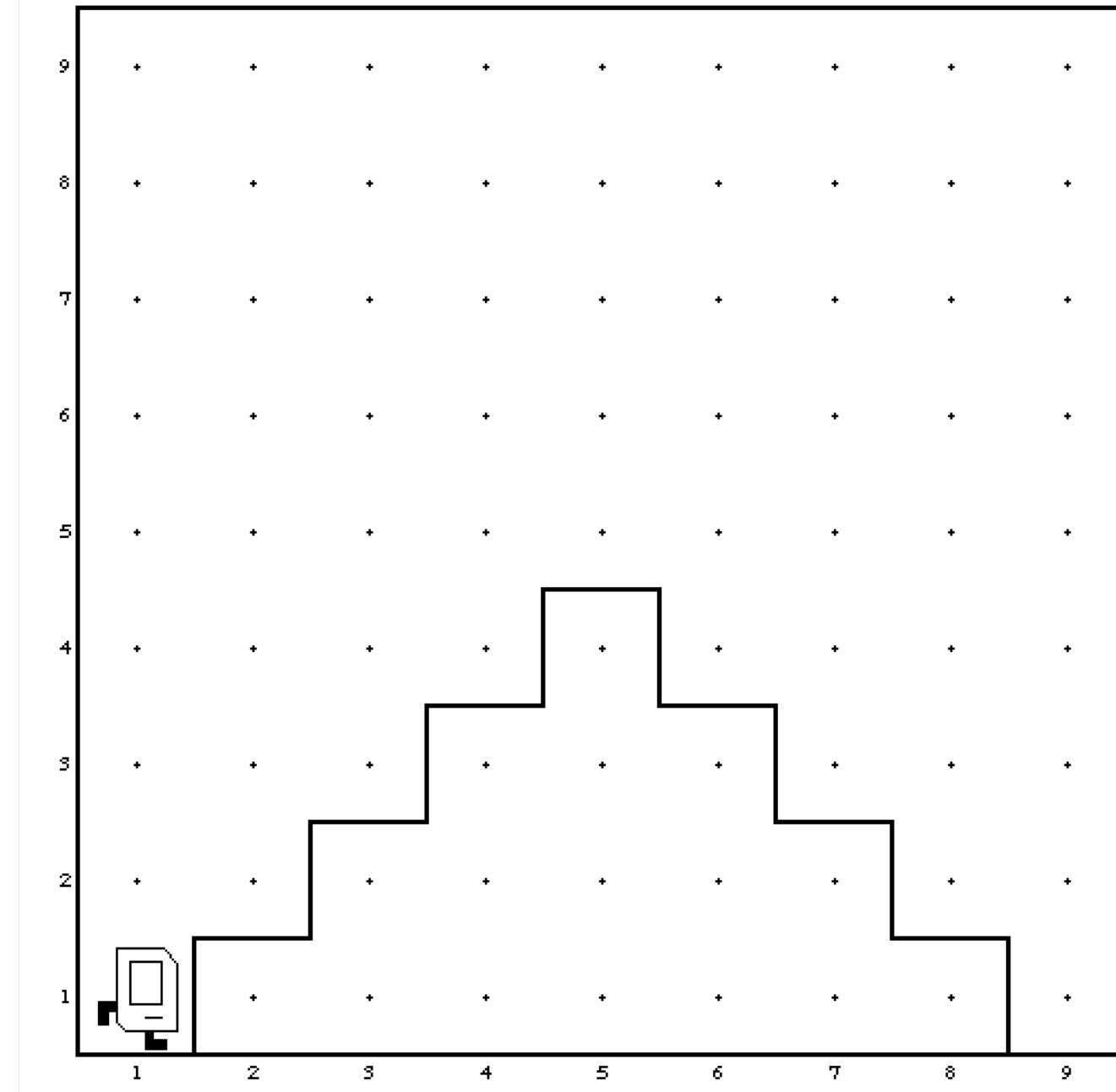
```
public void run() {  
  
    while(frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    putBeeper();  
}
```

```
public void run() {  
  
    putBeeper();  
  
    while(frontIsClear()) {  
        move();  
        putBeeper();  
    }  
}
```

# Conditions de Karel

<b>Test</b>	<b>Test Opposé</b>	<b>Description</b>
beepersPresent()	noBeepersPresent()	Au moins un beeper sur la cellule?
beepersInBag()	noBeepersInBag()	Au moins un beeper dans le sac?
frontIsClear()	frontIsBlocked()	Pas de mur devant?
leftIsClear()	leftIsBlocked()	Pas de mur à gauche?
rightIsClear()	rightIsBlocked()	Pas de mur à droite?
facingEast()	notFacingEast()	Est-ce que Karel fait face à l'Est?
facingWest()	notFacingWest()	Est-ce que Karel fait face à l'Ouest?
facingNorth()	notFacingNorth()	Est-ce que Karel fait face au Nord?
facingSouth()	notFacingSouth()	Est-ce que Karel fait face au Sud?

## Exercice: Karel des montagnes



## **Instruction Conditionnelle**

**Conditional statements; « if-else statements »**

## Instruction Conditionnelle

```
if (condition) {  
    // instructions a executer  
    // si condition est vrai  
}
```

## Instruction Conditionnelle

```
if(condition) {  
    // instruction a executer  
    // si condition est vrai  
}  
else  
{  
    // instructions a executer  
    // si condition est faux  
}
```

# **Exercices**

## Commandes additionnelles de SuperKarel

	<b>Commande</b>
<b>Tourner à droite</b>	<code>turnRight();</code>
<b>Faire demi-tour</b>	<code>turnAround();</code>
<b>Colorier cellule</b>	<code>paintCorner(<i>couleur</i>);</code>

## Commandes additionnelles de SuperKarel

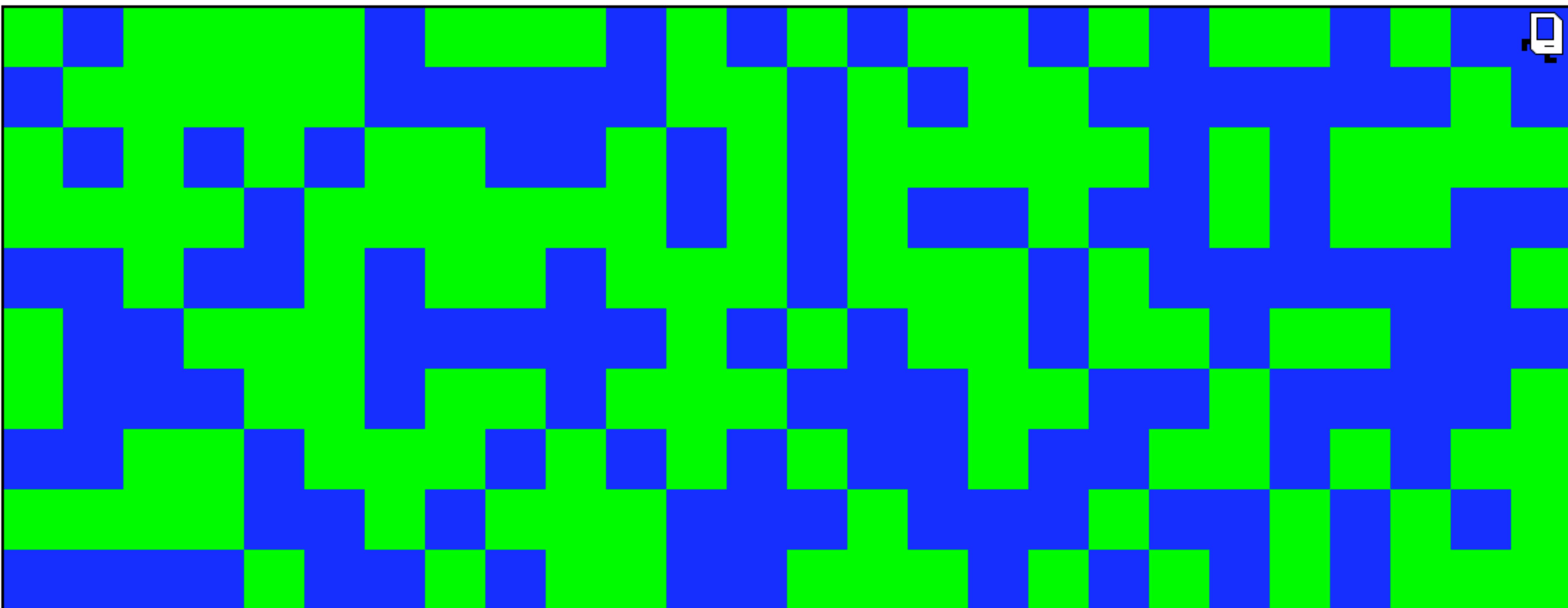
	<b>Commande</b>
<b>Tourner à droite</b>	<code>turnRight();</code>
<b>Faire demi-tour</b>	<code>turnAround();</code>
<b>Colorier cellule</b>	<code>paintCorner(<i>couleur</i>);</code>

SuperKarel sait aussi colorier!

## Conditions additionnelles de SuperKarel

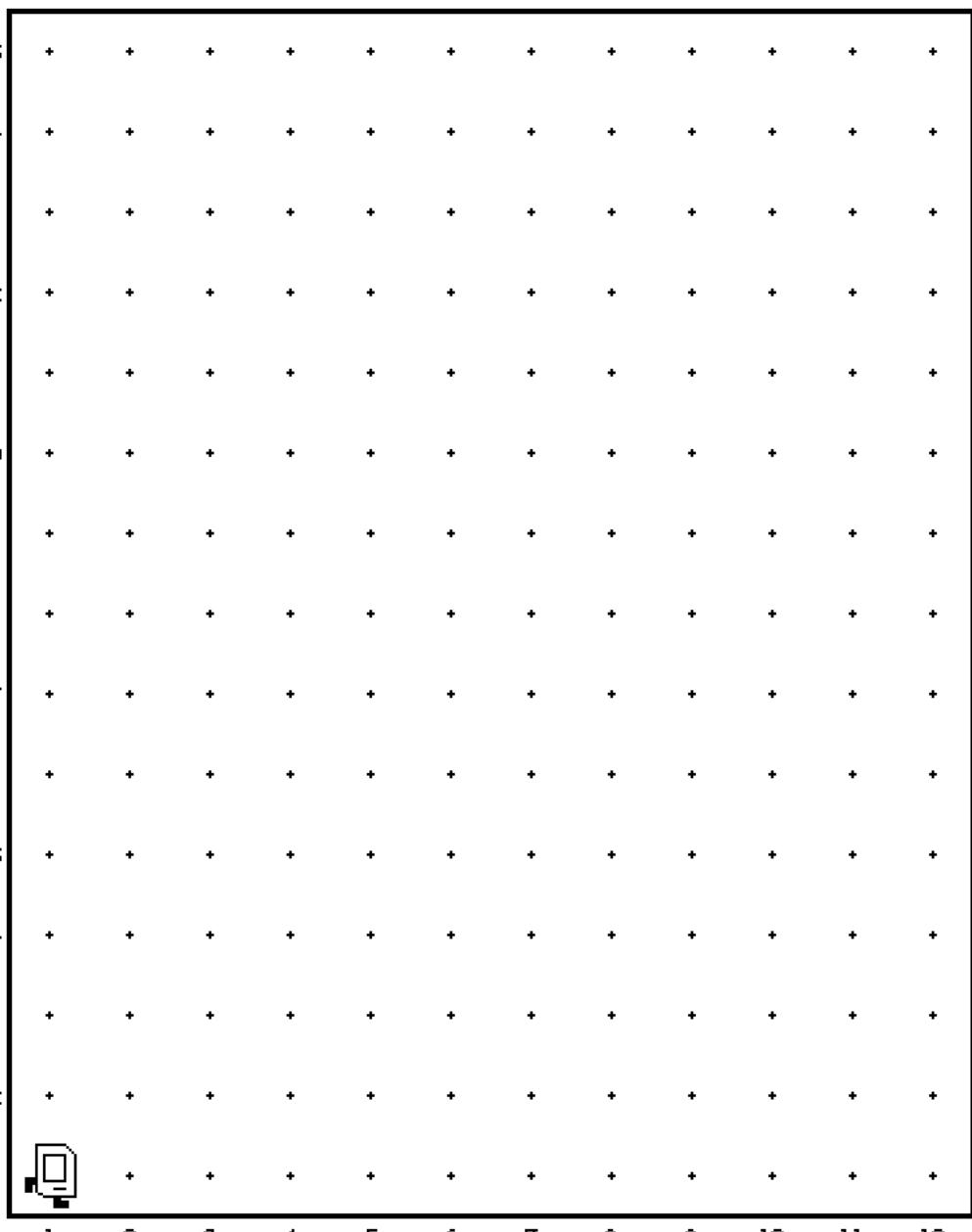
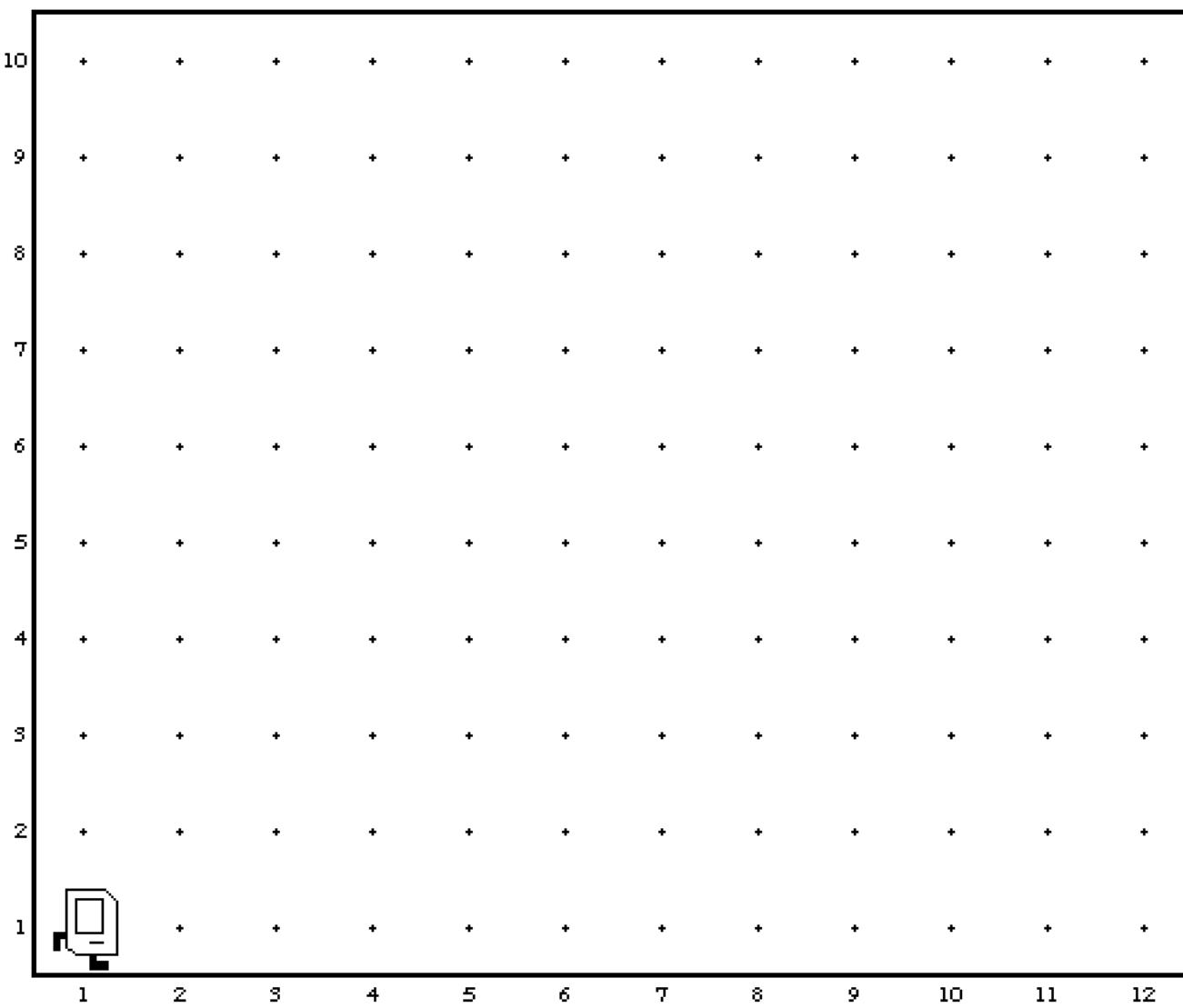
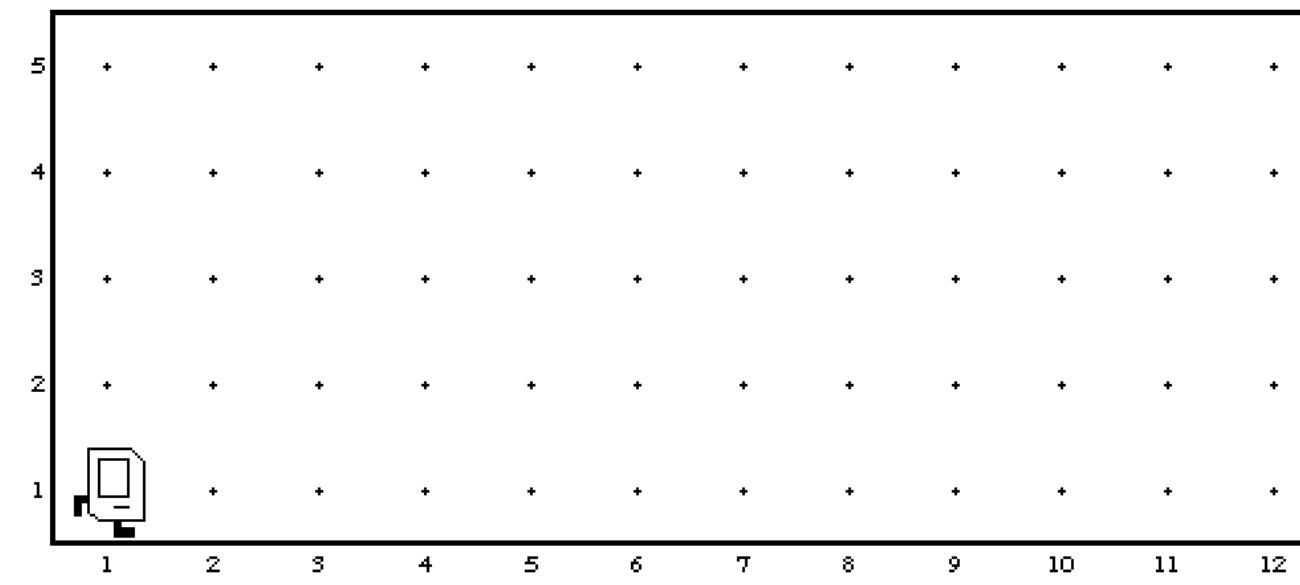
<b>Test</b>	<b>Description</b>
<code>random()</code>	Est vraie 50% du temps, mais de manière aléatoire
<code>random(<i>p</i>)</code>	Est vraie $p*100\%$ du temps, mais de manière aléatoire
<code>cornerColorIs(<i>c</i>)</code>	Est ce que la couleur de la cellule est <i>c</i> ?

## Exercice: Peintre Aléatoire



# Exercice: Drapeau Guinéen

# Avant



# Après

