Licence 3 Informatique
Module : théorie des langages
et compilation

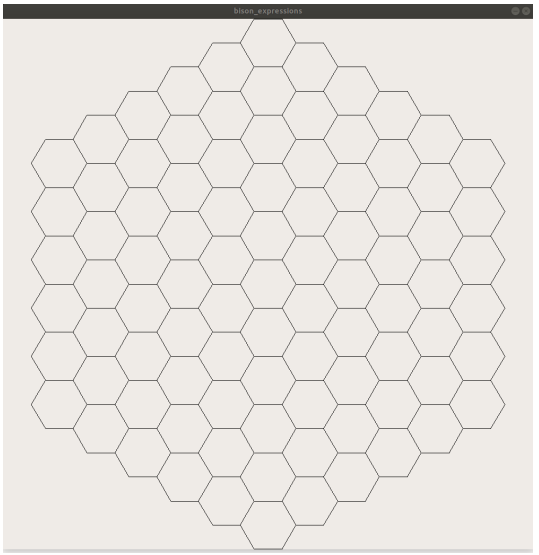
Projet

city (part 3) : Description des fichiers de tests

Test 1

```
1 Construire{
2
3 }
```

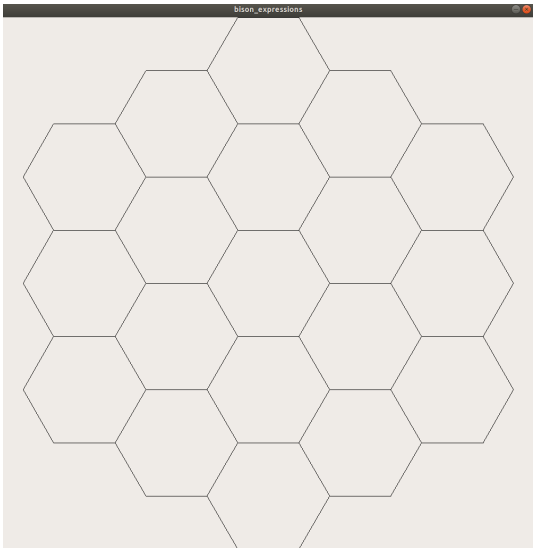
Ce test crée simplement une ville d'un rayon égal à 5 par défaut. Résultat attendu à droite.



Test 2

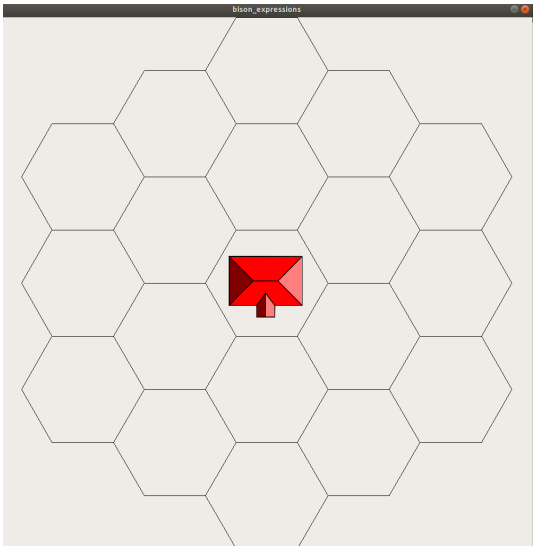
```
1 Construire (2){
2
3 }
```

Ce test crée simplement une ville d'un rayon égal à 2. Résultat attendu à droite.



Test 3

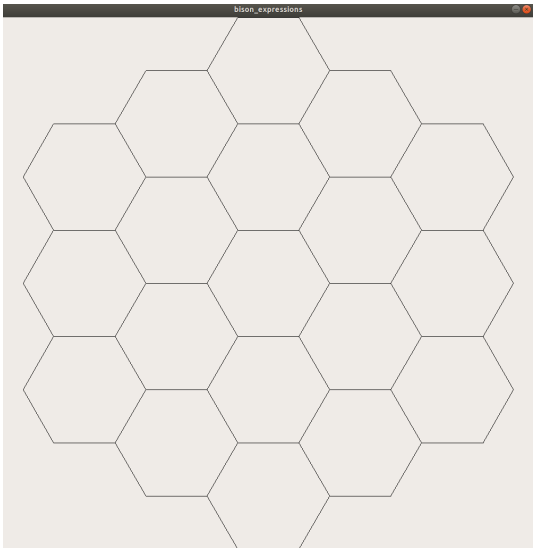
```
1 Construire (2){
2
3 }
4
5 Construire{
6   Maison(0,0,0)
7 }
```



Ce test crée simplement une maison aux coordonnées (0,0,0) dans une ville de rayon égal à 2. Résultat attendu à droite.

Test 3 bis

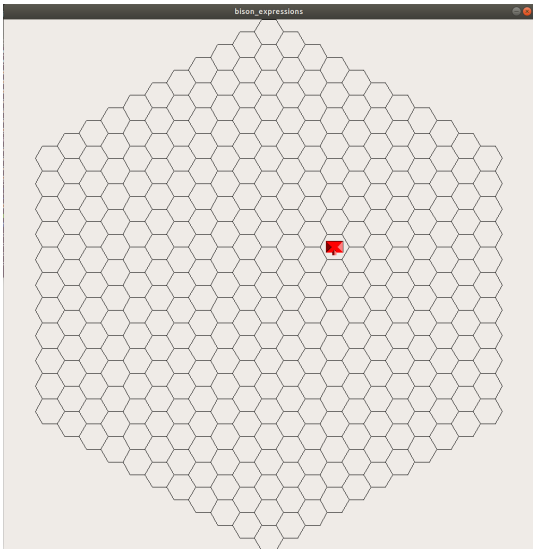
```
1 Construire (5){
2   Maison(3,0,-3)
3 }
4
5 Construire (2){
6
7 }
```



Ce test crée simplement une maison aux coordonnées (3,0,-3) dans une ville de rayon égal à 5 puis redimensionne la ville à un rayon égal à 2. La maison est hors du rayon de la nouvelle ville et doit être détruite. Résultat attendu à droite.

Test 3 ter

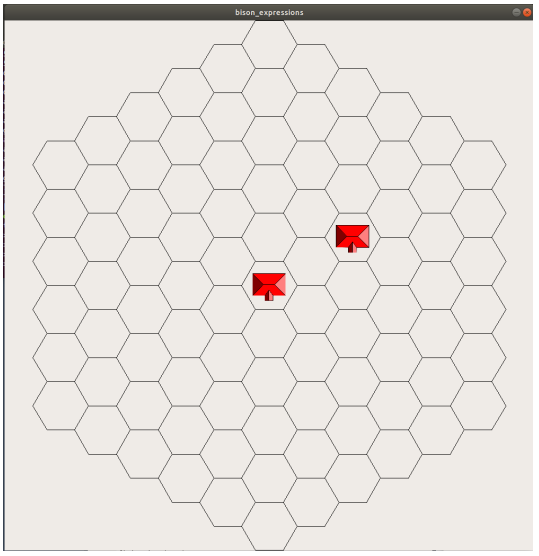
```
1 Construire (5){
2   Maison(3,0,-3)
3 }
4
5 Construire (10){
6
7 }
```



Ce test crée simplement une maison aux coordonnées (3,0,-3) dans une ville de rayon égal à 5 puis redimensionne la ville à un rayon égal à 10. La maison est à l'intérieur de la ville et n'est pas détruite. Résultat attendu à droite.

Test 4

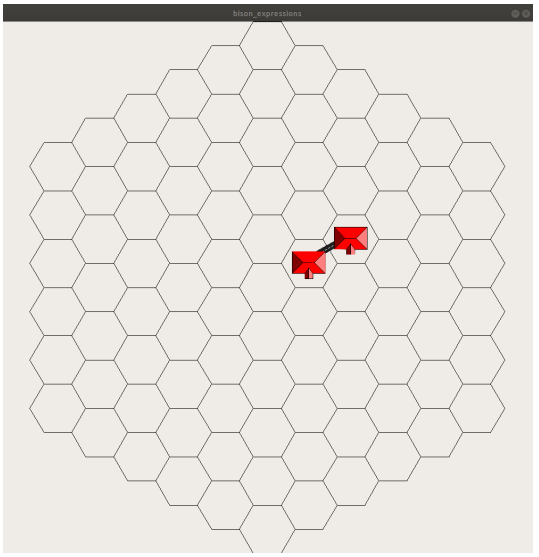
```
Construire{
  Maison(0,0,0)
  Maison
}
```



Ce test crée une maison aux coordonnées (0,0,0) et une seconde à des coordonnées aléatoires au sein de la ville. La seconde maison n'est pas forcément placée au même endroit que l'exemple. La position aléatoire générée ne doit pas être la même à chaque exécution. Un résultat possible à droite.

Test 5

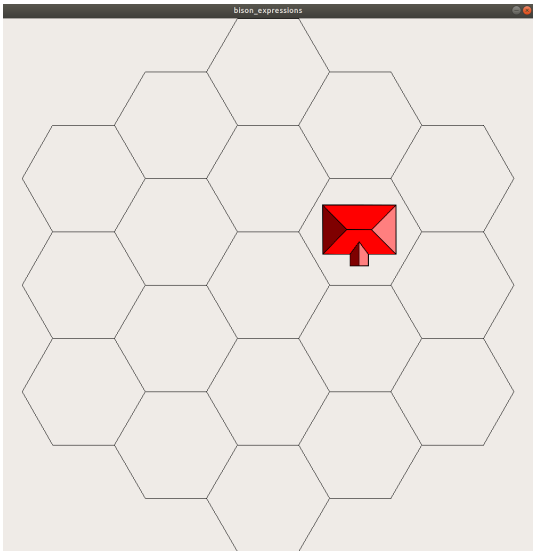
```
Construire{
  Maison(1,0,-1)
  Maison(2,0,-2)
  Route(1,0,-1) -> (2,0,-2)
}
```



Ce test crée simplement deux maisons aux coordonnées (1,0,-1) et (2,0,-2) ainsi qu'une route reliant les deux maisons. Résultat attendu à droite.

Test 5 bis

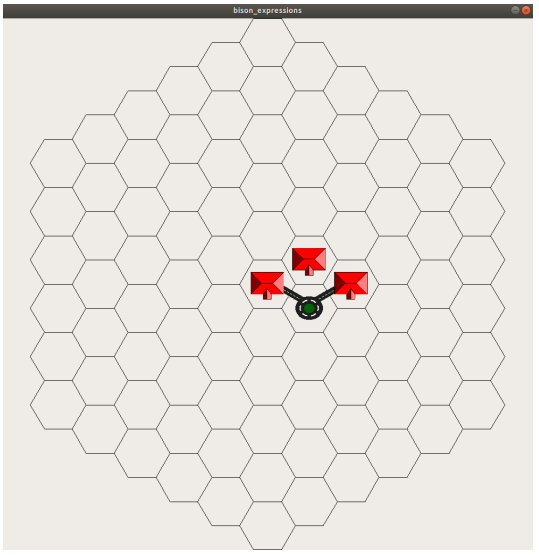
```
Construire (5){
  Maison(1,0,-1)
  Maison(5,0,-5)
  Route(1,0,-1) -> (5,0,-5)
}

Construire (2){

}
```



Ce test crée deux maisons aux coordonnées (1,0,-1) et (5,0,-5) ainsi qu'une route reliant les deux maisons puis redimensionne la ville à un rayon égal à 2. La maison hors périmètre de la ville et la route allant jusqu'à celle-ci doivent être détruites. Résultat attendu à droite.

Test 6

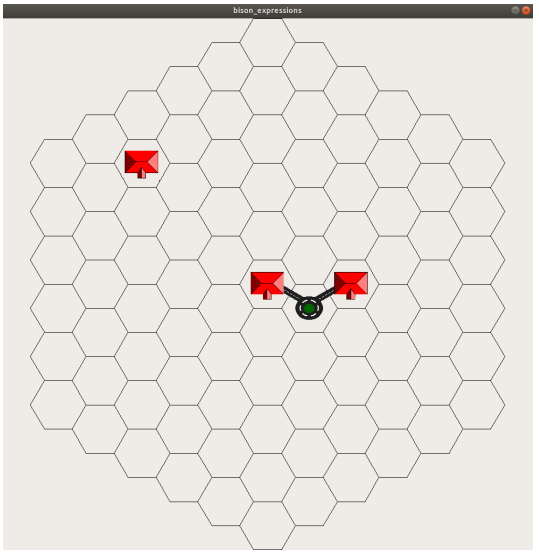
```
Construire (5) {
  %% Début du programme
  Maison
  %/
  Construit une maison à des coordonnées
  aléatoires dans la limite de la taille
  de la ville
  /%
  Maison (0, 0, 0)
  %% Construit une maison à l'emplacement de coordonné
      ↪es (0 ,0 ,0)
  Maison (2, -1, -1)
  Route (0, 0, 0) -> (2, -1, -1)
  %% Construit une route entre les maisons situées aux
      ↪coordonnées (0, 0, 0) et (2, -1, -1)
  %% fin du programme
}
```



Ce test est celui présenté dans le sujet du projet. Résultat attendu à droite.

Test 7

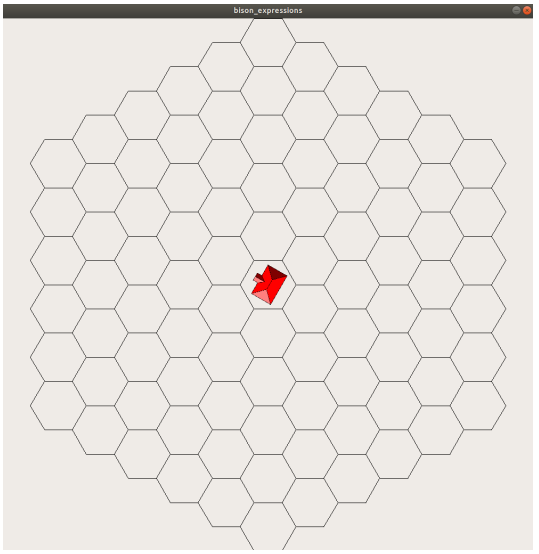
```
Construire (      5    ) {
  %% Début du programme
  Maison
  %/
  Construit une maison à des coordonnées
  aléatoires dans la limite de la taille
  de la ville
  /%
  Maison (    0    ,    0    ,    0)
  %% Construit une maison à l'emplacement de coordonné
      ↪es (0 ,0 ,0)
  Maison (    2    ,    -1    ,    -1)
  Route (    0    , 0,    0)        -> (        2    ,    -1
      ↪       ,  -1)
  %% Construit une route entre les maisons situées aux
      ↪coordonnées (0, 0, 0) et (2, -1, -1)
  %% fin du programme



}
```



Ce test est identique au précédent avec des espaces au milieu des instructions. Résultat attendu à droite.

Test 8

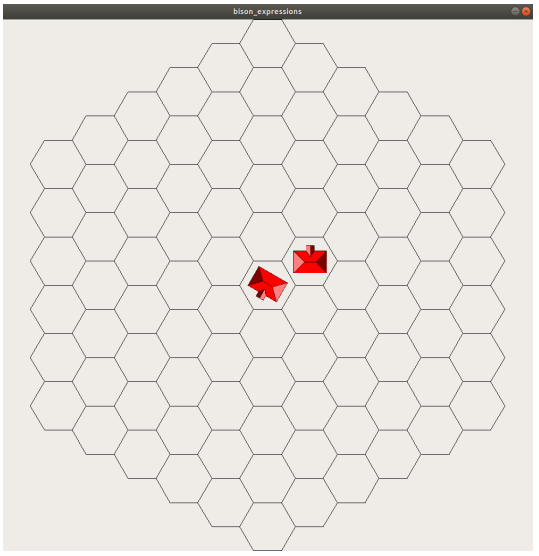
```
Construire ( 5 ) {
  Maison (0, 0, 0)
  %% Orienter la maison créée vers la gauche
  Tourner maison[1] horaire
  Tourner (0,0,0) horaire
}
```



Ce test construit une maison en (0,0,0) et la fait pivoter deux fois dans le sens horaire. Attention l'orientation de l'interface graphique et du projet ne sont pas les mêmes. 0° sur l'interface graphique correspond à 90° dans le projet. Résultat attendu à droite.

Test 9

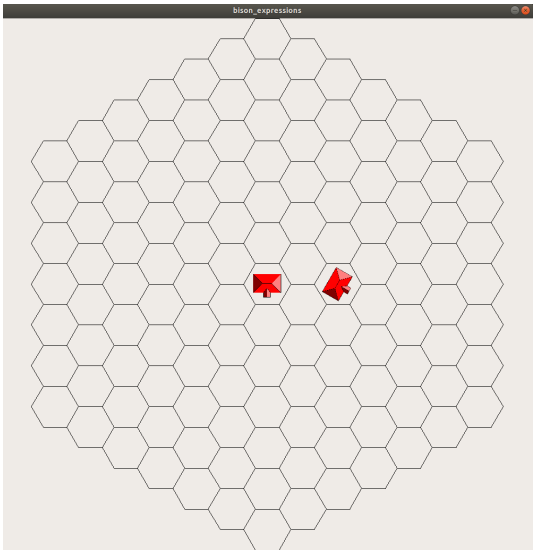
```
Construire (5) {
  Maison (0, 0, 0)
  Maison (1,0,-1)
  Orienter maison[1] 180°
  Tourner maison[1] !horaire
  Orienter (1,0,-1) -90°
}
```



Ce test construit deux maisons en (0,0,0) et (1,0,-1) et change l'orientation des maisons. Résultat attendu à droite.

Test 10

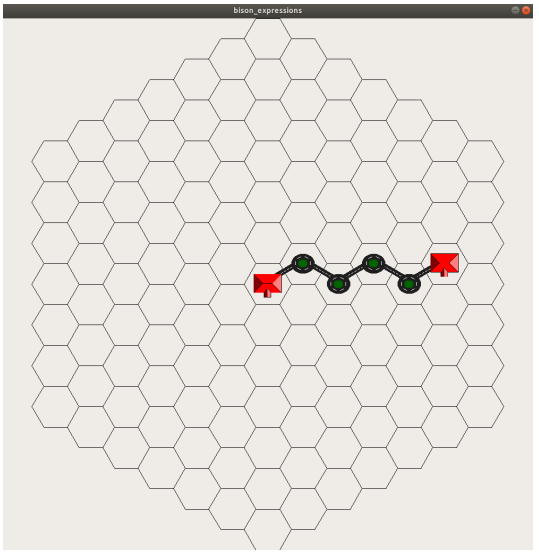
```
Construire (6) {
  Maison (0, 0, 0)
  Maison
  Tourner maison[2] !horaire
  Deplacer maison[2] -> (2,-1,-1)
}
```



Ce test construit une maison en (0,0,0) et une maison à une position aléatoire et change l'orientation et la position de la maison aléatoire. Résultat attendu à droite.

Test 11

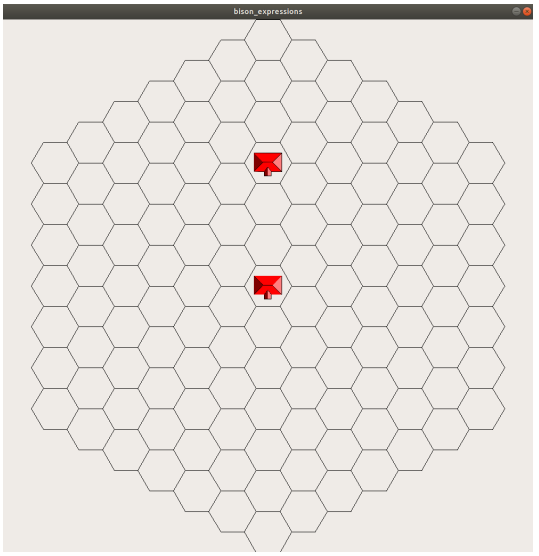
```
1 Construire (6) {
2   Maison (0, 0, 0)
3   Maison (3,0,-3)
4   Route maison[1] -> maison[2]
5   Deplacer maison[2] -> (5,-2,-3)
6 }
```



Ce test construit deux maisons en (0,0,0) et (3,0,-3) et trace une route entre ces deux maisons. Une fois la maison 2 déplacée en (5,-2,-3) la route doit suivre la nouvelle position. Résultat attendu à droite.

Test 12

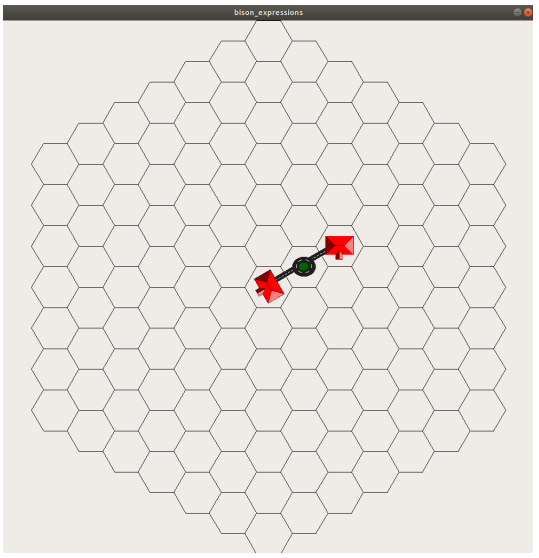
```
1 Construire (6) {
2   Maison (0, 0, 0)
3   Maison (2,0,-2)
4   Maison (0,3,-3)
5   Route maison[1] -> maison[2]
6   Route maison[1] -> maison[3]
7   Detruire maison[2]
8   Detruire maison[1] -> (0,3,-3)
9 }
```



Ce test construit trois maisons en (0,0,0), (2,0,-2) et (0,3,-3) et deux routes entre maison la première maison et les deux autres. On supprime ensuite la maison 2 et la route vers la maison 3. Lorsqu'on supprime une maison toutes les routes allant à cette maison doivent être supprimées. Résultat attendu à droite.

Test 13

```
Construire (6) {
  Maison (0, 0, 0)
  Maison (2,0,-2)
  Route maison[1] -> maison[2]
  Orienter maison[1] 150°
  Position maison[1]
  Orientation (0,0,0)
  Voisinage maison[1]
}
```



Ce test construit deux maisons en (0,0,0) et (2,0,-2) et une route entre ces deux maisons. On modifie la position de la maison 1 et on affiche les informations sur la position, l'orientation et le voisinage de la maison 1. Résultat graphique attendu à droite. Résultat attendu dans le terminal :

```
Position maison 1 : (0,0,0)
Orientation maison 1 : 150°
Maisons reliées à maison 1 :
    maison 2 : (2,0,-2)
```

Test 14

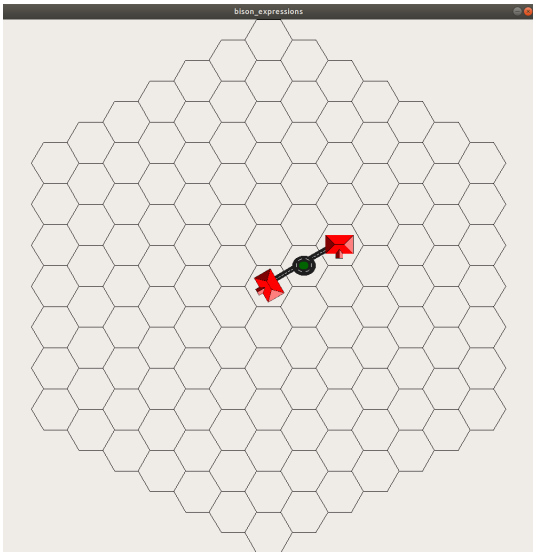
```
Construire {
  Maison M1 (0, 0, 0)
  Position M1
}
```

```
Position maison 1 : (0,0,0)
```

Ce test construit une maison en (0,0,0) avec un nom de variable M1 et afficha sa position. Résultat attendu dans le terminal à droite.

Test 14 bis

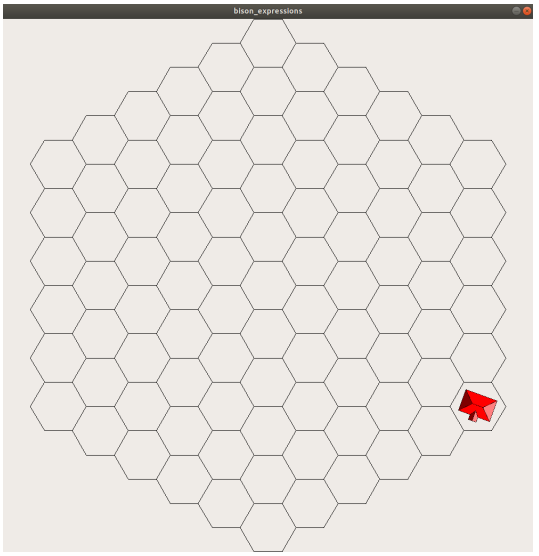
```
Construire {
  Maison M1 (0, 0, 0)
  Deplacer M1 -> (1,0,-1)
  Orienter M1 150°
  Maison M2 (0,2,-2)
  Route M1 -> M2
}
```



Ce test construit deux maisons M1 en (0,0,0) et M2 en (0,2,-2). On déplace M1 en (1, 0, -1) et oriente M1 à 150°. On trace une route entre ces deux maisons. Résultat attendu à droite.

Test 15

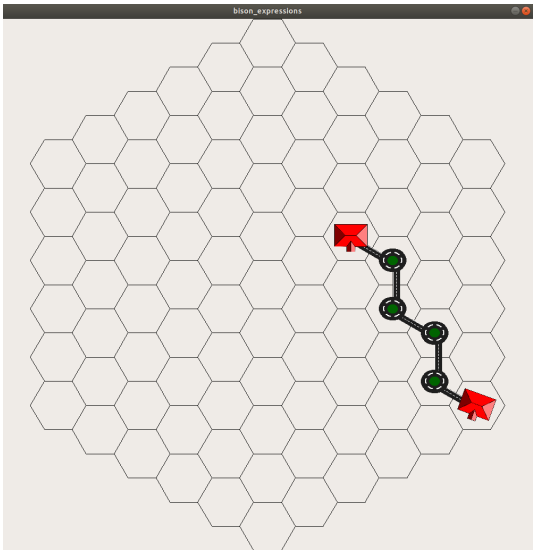
```
Construire {
  Maison (1+1, 4-5, 1*-1)
  Deplacer maison[1] -> (1+1+1+1+1,2+3*10-(15*2+7),0)
  Orienter (5,-5,0) 78+32°
}
```



Ce test utilise des expressions pour construire une maison en (2,-1,-1) et la déplace en (5,-5,0) et orientée à 110°. Résultat attendu à droite.

Test 15 bis

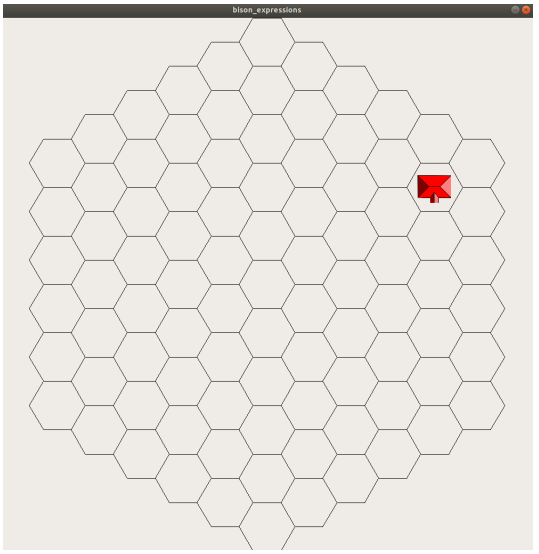
```
Construire (2+3) {
  Maison M1 (1+1, 4-5, 1*-1)
  Maison M2 (2,0,-2)
  Deplacer maison[1] -> (1+1+1+1+1,2+3*10-(15*2+7),0)
  Orienter M1 78+32°
  Route (4/2,65*0,---2) -> ((5),-5,0+0+0)
  Position M1
}
```



Ce test utilise des expressions pour construire une maison M1 en (2,-1,-1) et la déplace en (5,-5,0) et orientée à 110° et une maison M2 en (2,0,-2) plus une route entre les deux maisons. Résultat attendu à droite. Affichage de la position de M1.

Test 16

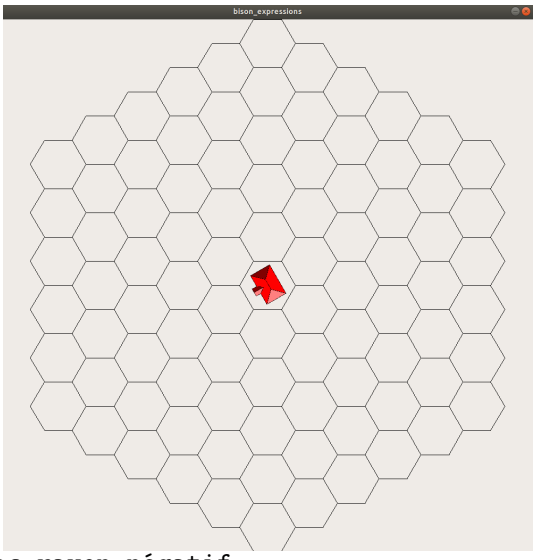
```
Construire {
  i=2
  j=1
  j=i+4
  Maison (j-i,0,-4)
}
```



Ce test utilise des variables entières i et j pour construire une maison en (4,0,-4). Résultat attendu à droite.

Test 17

```
Construire (-5) {

}

Construire {
  Maison (0,0,0)
  Tourner maison[1] horaire
  Maison (0,0,0)
  Route maison[1] (1,0,-1)
  Maison (5,2,3)
  Maison (6,0,-6)
}
```

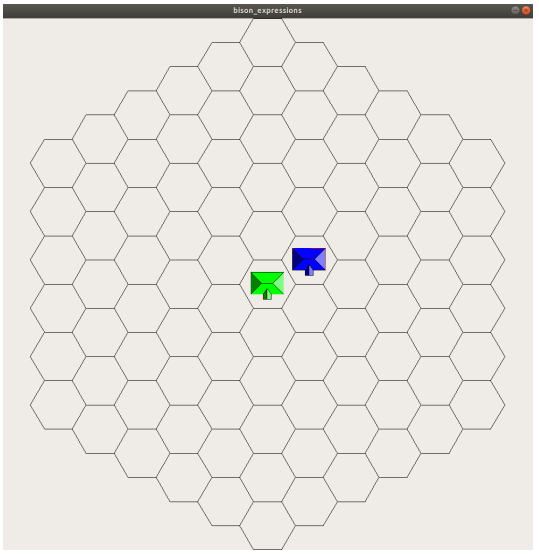


Ce test permet de tester des erreurs possibles. Construction d'une ville avec rayon négatif, construction d'une maison sur une position déjà existante, construction d'une route sur une position sans maison, construction d'une maison sur une position impossible et une construction de maison sur une position hors de la ville. Résultat graphique attendu à droite. Résultat possible dans le terminal :

```
Erreur (ligne 1): impossible de créer une ville avec rayon négatif.
Erreur (ligne 8): une maison existe déjà en (0,0,0).
Erreur (ligne 9): il n'y a pas de maison à relier en (1,0,-1).
Erreur (ligne 10): la position (5,2,3) n'est pas une position valide.
Erreur (ligne 11): La position (6,0,-6) est hors du périmètre de la ville.
```

Test 18

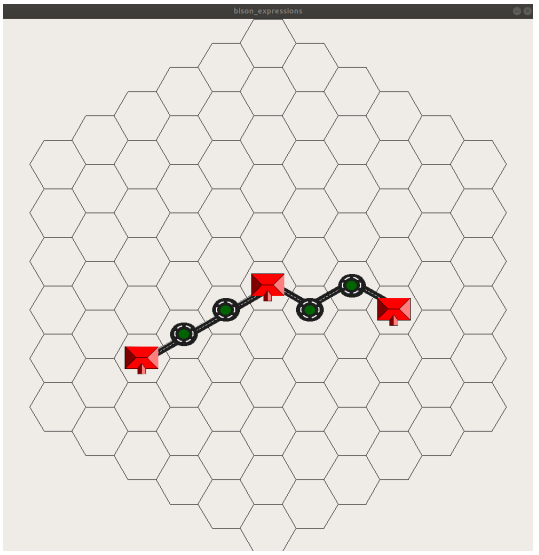
```
Construire{
  Maison(0,0,0)
  Coloriser maison[1] #00ff0
  Maison M2 (1,0,-1)
  Coloriser M2 (0,0,255)
  Couleur M2
}
```



Ce test permet de tester la colorisation des maisons. Construction de deux maisons M1 verte en (0,0,0) et M2 bleue en (1,0,-1). Résultat graphique attendu à droite. Affichage de la couleur de M1 sous la forme (0,255,0).

Test 19

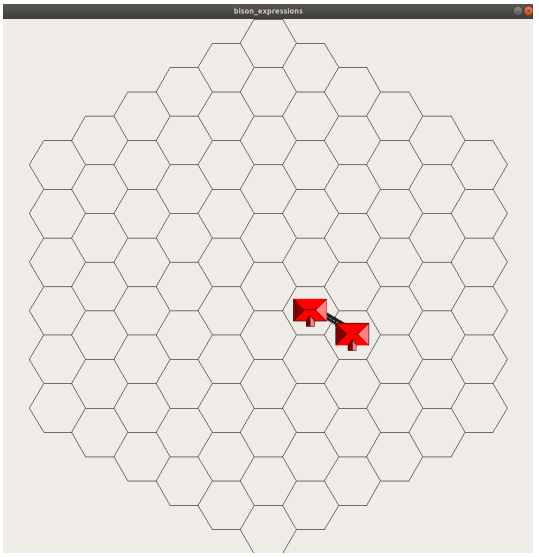
```
Construire {
  Maison (0,0,0)
  Voisin (0,0,0) 3
  Voisin (0,0,0) 3
}
```



Ce test permet de créer deux maisons au voisinage de maison 1. Les maisons sont placées à une position aléatoire à une distance de 3. Deux maisons ne doivent pas être superposées. S'il n'existe plus de place à une distance de 3 alors la maison n'est pas construite et on affiche une erreur. Résultat graphique attendu à droite.

Test 20

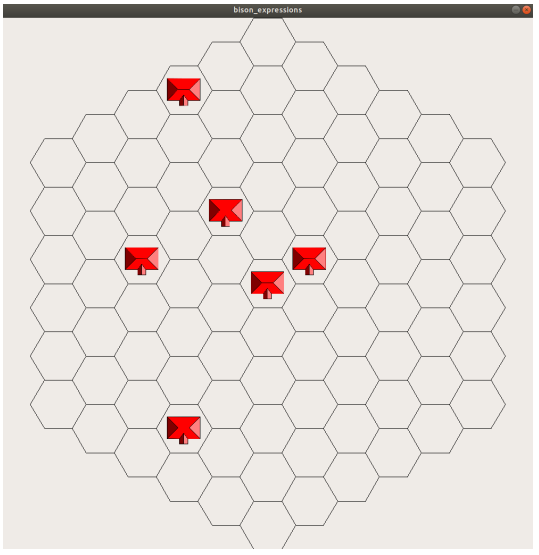
```
Construire {
  Maison M1 (1, -1, 0 )
  Si (occupe(2, -2, 0)) {
    Route M1 -> (2, -2, 0)
  }
  Sinon {
    Maison (2, -2, 0)
    Route M1 -> (2, -2, 0)
  }
}
```



Ce test permet de tester le bon fonctionnement des conditionnelles. Résultat graphique attendu à droite.

Test 21

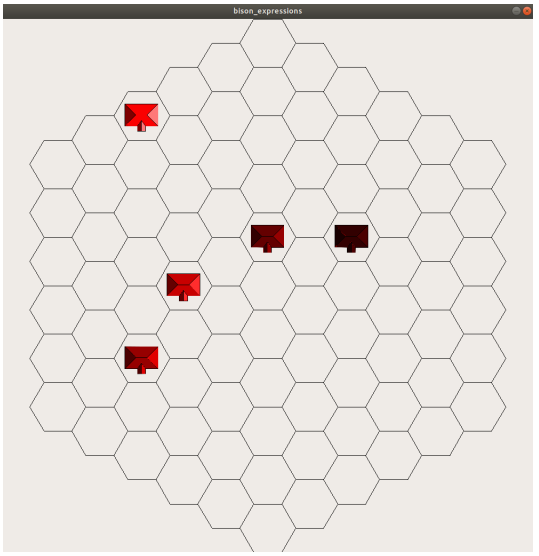
```
Construire {
  Maison M1 (0,0,0)
  i=0
  Tant que ( i < 5) {
    Voisin M1 1
    i=i+1
  }
}
```



Ce test permet de tester le bon fonctionnement de la boucle "Tant que". construction de 5 maisons à des positions aléatoires pour des distances allant de 0 à 5. Résultat graphique attendu à droite. Dans ce cas on doit avoir une erreur car on tente de construire une maison en (0,0,0) alors qu'il en existe déjà une.

Test 22

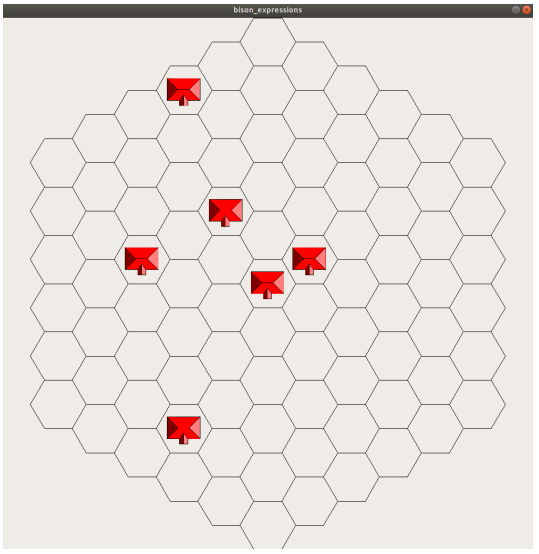
```
1 Construire {
2   i=5
3   j=1
4   Repeter i fois {
5     Maison
6     Coloriser maison[j] (j*50,0,0)
7   }
8 }
```



Ce test permet de tester le bon fonctionnement de la boucle "Repeter". Construit 5 maisons à des positions aléatoires et leur affecte une couleur de plus en plus claire. Résultat graphique attendu à droite.

Test 23

```
1 void nMaisons ( n ) {
2   Maison M1 (0,0,0)
3   i=0
4   Tant que ( i < n) {
5     Voisin M1 1
6     i=i+1
7   }
8 }
9
10 Construire {
11   nMaisons(5)
12 }
```



Ce test permet de tester le bon fonctionnement des fonctions. On déclare une fonction nMaisons qui crée n maisons autour d'une maison en (0,0,0) n étant un paramètre de la fonction. Résultat graphique attendu à droite.