

Projet de BTS Système Numérique  
– Informatique et Réseau

---

Éditeur de plan  
de maison

---

Lycée Antoine Bourdelle -  
Promotion 2018

## Table des matières

I . Introduction.....	3
II . Éditeur matriciel.....	4
1 . Principe de fonctionnement.....	4
2 . Problèmes de ce modèle.....	5
3 . Tests.....	6
a . Initialisation.....	6
b . Création et destruction d'étage.....	6
c . Navigation dans les étages.....	7
d . Ciblage du curseur.....	8
III . Éditeur vectoriel.....	9
1 . Principe de fonctionnement.....	9
2 . Raycast.....	11
3 . Tests.....	12
a . Initialisation.....	12
b . Création et destruction d'étage.....	12
c . Navigation dans les étages.....	13
d . Création des pièces.....	13
e . Sélection des pièces.....	15
f . Modification et destruction des pièces.....	16
g . Sauvegarde du plan.....	18
IV . Travail restant.....	18

## I . Introduction

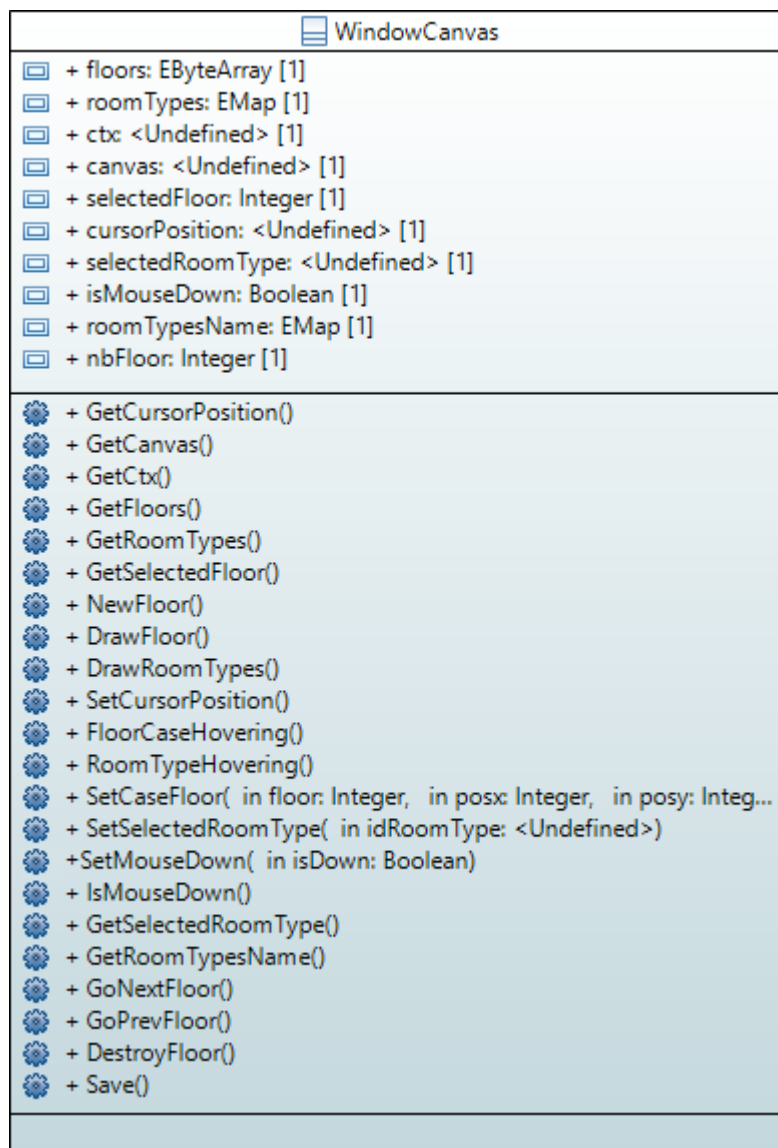
Dans le projet de visite virtuelle, il nous faut pouvoir créer les visites sur tout les points. Un des points est qu'il faut pouvoir créer un éditeur de plan de maison, de manière schématique, et faire en sorte que cette éditeur soit facile d'utilisation.

Il faut que l'éditeur soit réalisé avec du JavaScript pour qu'il soit directement utilisable par le site web. Le choix du JavaScript n'est pas anodin, car ce langage peut être interprété par n'importe quel navigateur, contrairement au flash, où il faut parfois installer un pilote puis l'activer au cas par cas, ce qui n'est pas pratique. Il doit aussi être simple d'utilisation, et que les données doivent être stocker de manière ordonné dans le programme, qui peut être fait facilement avec une approche objet. Et enfin l'application doit pouvoir sauvegarder le plan au format XML, pour que le plan puisse être réutiliser facilement.

Nous verrons que 2 méthodes ont été envisagé, la première avec une matrice, qui fut abandonné à cause des problèmes de sauvegarde, où les fichiers sont trop important, et une utilisation moins pratique. La seconde méthode est avec des tracés vectoriels, ce qui simplifie sont utilisation et sa sauvegarde, et permet de faire des pièces plus complexe et plus facilement.

## II . Éditeur matriciel

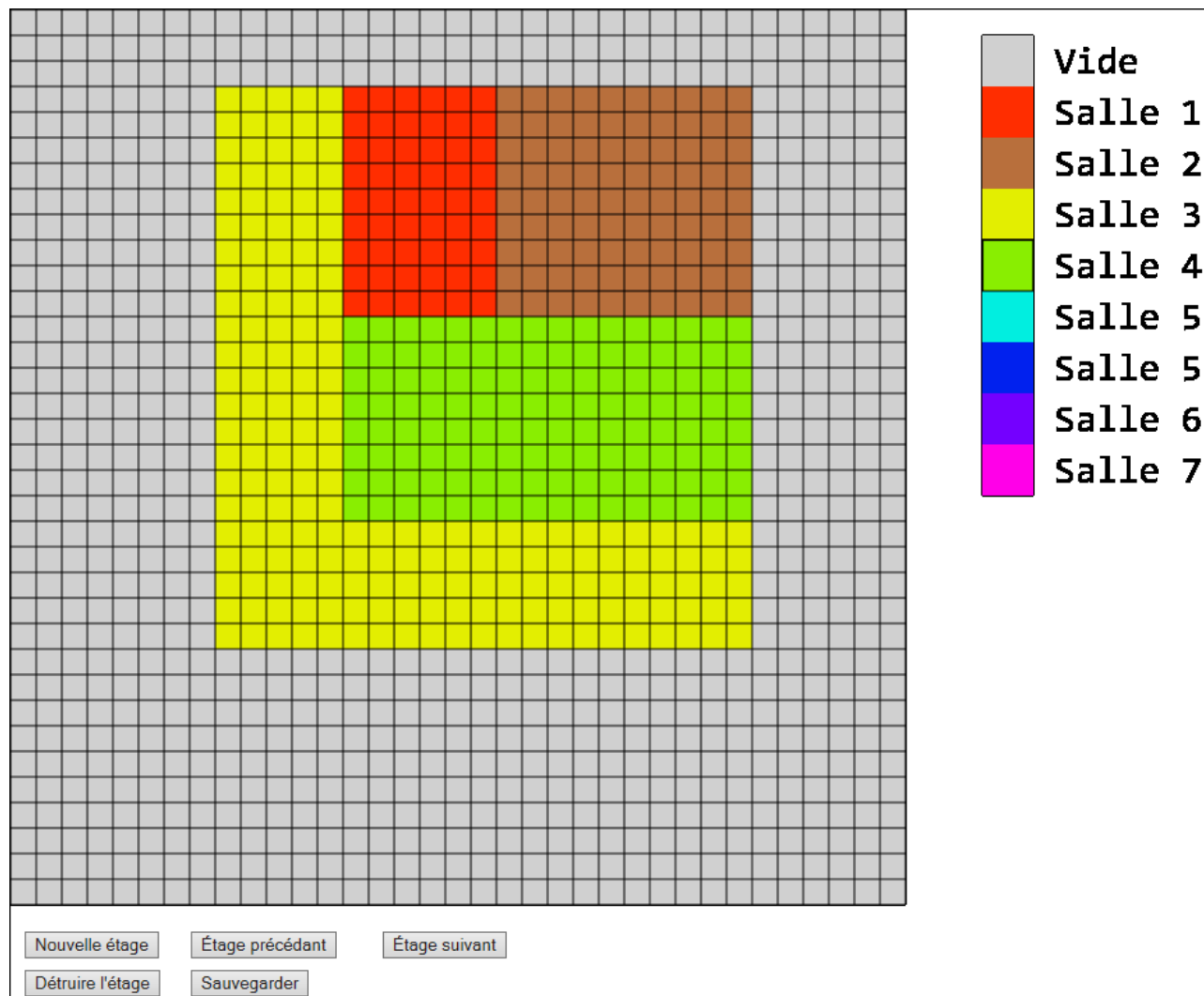
### 1 . Principe de fonctionnement



Le diagramme de classe suivant a été fait. Le problème de ce diagramme est qu'il n'y a qu'une seule classe ou tout est regroupé. Ça a rendu le développement de cette version très difficile.

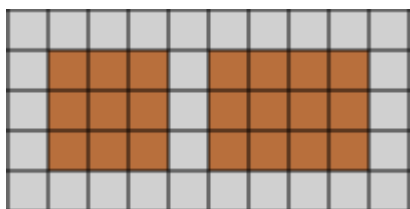
La première version de l'éditeur qui a été envisagé, et un éditeur matriciel, où nous sélectionnons un type de pièce grâce à un menu regroupant toutes les pièces différentes, et ensuite, une fois que le type de pièce est sélectionné, nous pouvons l'assigner à une case de la matrice. Nous pouvons créer plusieurs étages, et tout est sauvegardé en mémoire.

Voici le résultat où le développement s'est arrêté.



## 2 . Problèmes de ce modèle

Nous pouvions assigner des cases de manière fluide grâce à un clique puis glissé de la souris, mais cela était long et laborieux. Pour palier a ce problème, une sélection par zone a été envisagé, mais le développement s'est arrêté à ce moment. Le second problème est que nous pouvions avoir un type de pièce mais a plusieurs endroit, comme nous pouvons le voir ci dessous.



Ceci est une même pièce dissociée, ce qui n'a pas de sens.

La raison principale pour laquelle cette version a été abandonnée, est que si on devait enregistrer le plan, cela ferait des fichiers trop gros, due aux matrices, où des informations inutiles y seraient stockées, et rendrait la sauvegarde trop volumineuse.

### 3 . Tests

#### a . Initialisation

Au lancement de l'application, nous lui demandons de créer un étage automatiquement.

```
Étage créé : 0
editorTest.js (123,3)
└─ [object Array]      [Array[35]]
   editorTest.js (124,3)
   └─ 0                [object Array] [...]
      length            1
```

Ici nous avons le résultat du test, avec la ligne en surbrillance bleue qui témoigne de l'étage créé, qui est un tableau en deux dimensions initialisé à 0.

#### b . Création et destruction d'étage

Lorsque nous créons un nouvel étage manuellement par le biais du bouton, nous obtenons ceci.

```
Étage créé : 1
editorTest.js (123,3)
└─ [object Array]      [Array[35], Array[35]]
   editorTest.js (124,3)
   └─ 0                [object Array] [...]
      1                [object Array] [...]
      length            2
```

Avec la ligne en surbrillance bleue qui témoigne bien du nouvel étage créé.

Maintenant que nous savons créer un étage il faut savoir le détruire, et le bon, selon l'étage sélectionné. Nous avons 7 étages, et nous sommes sur l'étage 2.

```
étage détruit
editorTest.js (267,4)
└─ [object Array]      [Array[35], Array[35], Array[35], Array[35], Array[35], Array[35]]
   editorTest.js (268,4)
   └─ 0                [object Array] [...]
      1                [object Array] [...]
      2                [object Array] [...]
      3                [object Array] [...]
      4                [object Array] [...]
      5                [object Array] [...]
      length            6
étage sélectionné : 2
```

En supprimant l'étage nous remarquons qu'il ne reste plus que 6 étages, et notre étage sélectionné a été décrémenté.

Un problème qui pourrait survenir est que si nous sommes sur l'étage 0, et que nous le supprimons, le programme va décrémenter l'étage sélectionné et mettre l'indice -1, ce qui est impossible.

```
étage sélectionné : 0  
editorTest.js (280,4)  
étage détruit  
editorTest.js (267,4)  
▶ [object Array]      [Array[35], Array[35], Array[35], Array[35], Array[35]]  
editorTest.js (268,4)  
étage sélectionné : 0
```

Nous voyons ici que l'indice est resté à 0. Le dernier problème serait de supprimer tout les étages, sauf qu'il nous en faut au moins 1. Lorsque nous essayons de détruire ce dernier étage il nous est impossible de le supprimer.

```
impossible de détruire cette étage  
editorTest.js (272,4)
```

### c . Navigation dans les étages

Pour pouvoir utiliser le programme il nous faut pouvoir naviguer simplement dans les étages créés.

```
étage suivant : 1  
editorTest.js (290,4)  
étage suivant : 2  
editorTest.js (290,4)  
impossible d'accéder a l'étage suivant  
editorTest.js (292,4)
```

Nous voyons bien que l'incrémentation s'est bien déroulé, jusqu'au moment où nous sommes arrivé au dernier étage, et l'incrémentation s'est bloqué.

```
étage précédant : 1  
editorTest.js (280,4)  
étage précédant : 0  
editorTest.js (280,4)  
impossible d'accéder a l'étage précédant  
editorTest.js (282,4)
```

Même chose pour passer aux étages inférieurs.

### d . Ciblage du curseur

Pour la sélection des carrés, il nous faut savoir si nous nous trouvons dans la zone d'édition, quand nous sommes à l'extérieur de la zone nous obtenons ceci.

```
▷ [object Object]      {x: null, y: null}
```

Ce qui est le résultat attendu. Dans le cas contraire, nous obtenons bien les coordonnées de la bonne case.

```
▷ [object Object]      {x: 7, y: 18}
```

Nous avons la même chose pour le sélecteur de pièce, avec les valeurs attendus.

```
▷ [object Object]      {x: null, y: null}
```

```
▷ [object Object]      {x: 0, y: 4}
```

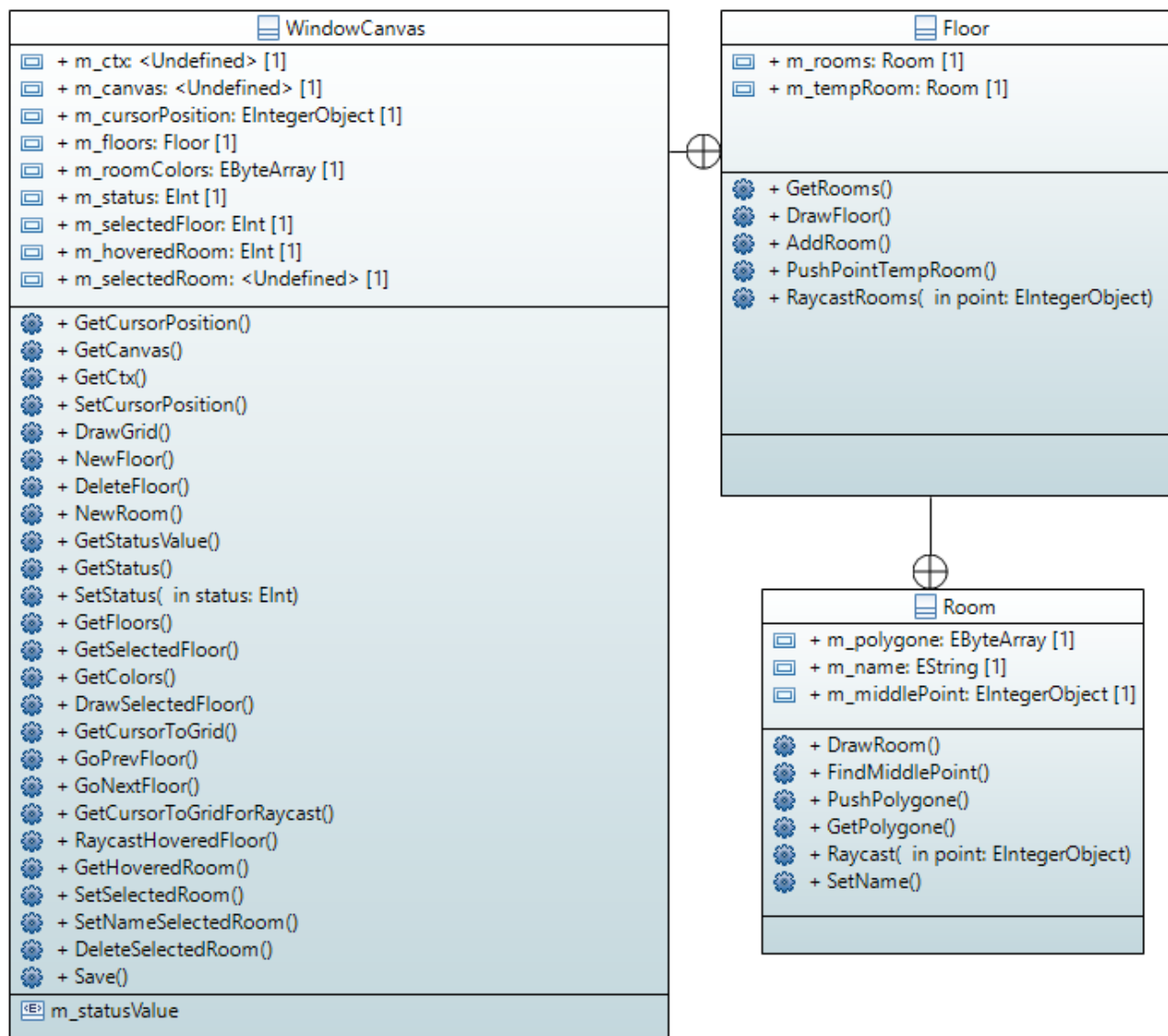
Le tableau du sélecteur étant en deux dimension, seul la coordonné y est prit en compte, mais il a était envisagé de le mettre en deux dimension, c'est pourquoi il y a une coordonné x, mais qui reste a 0, et n'est pas utilisé.



### III . Éditeur vectoriel

#### 1 . Principe de fonctionnement

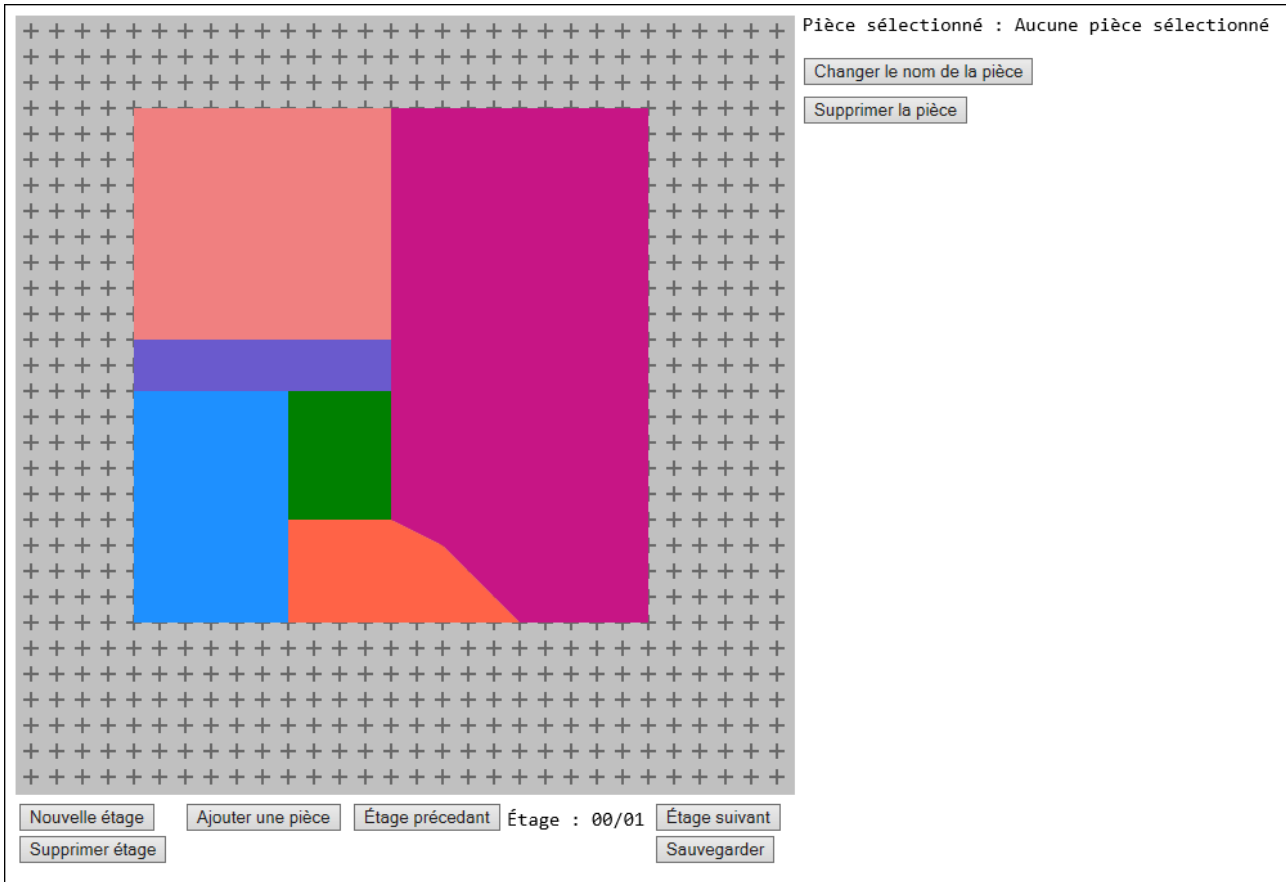
Ci dessous le diagramme de classe de l'éditeur de plan utilisant des vecteurs. Le précédent éditeur a été prit comme base, mais celui ci est beaucoup mieux organiser que le précédent.



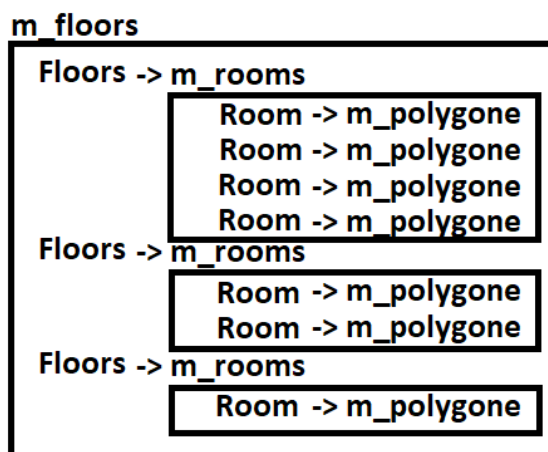
Le but de cette éditeur est de ne pas être limité par des carrés ou par un nombre limité de type de pièce. L'avantage est que les pièces sont des polygones créées par des cliques de souris, et les points sélectionnés viennent se coller à une grille. La création de pièce est simple et efficace.

# Rapport Technique du projet Visite Virtuelle – Éditeur de plan de maison

Voici le résultat final



Le stockage des informations a été imaginé de cette manière.

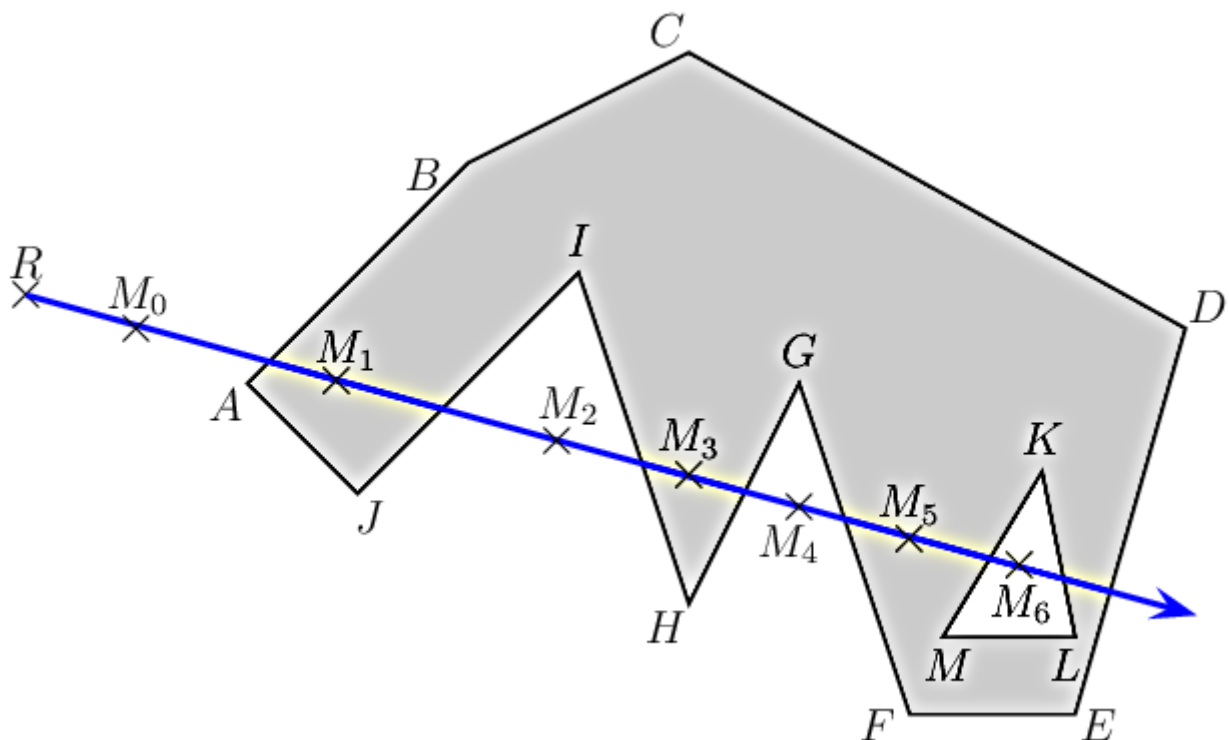


L'attribut `m_floors` contient les étages (floor) ayant l'attribut `m_rooms` contenant des pièces (room) contenant l'attribut `m_polygone` contenant des points et qui sert à afficher les pièces sur l'éditeur.

## 2. Raycast

Pour pouvoir modifier les pièces créées, il faut pouvoir les sélectionner, le plus simple d'utilisation est de cliquer sur la pièce en question, puis de modifier ses paramètres. Pour savoir si le curseur de la souris se trouve dans un polygone, nous devons utiliser la méthode du raycast.

Le principe est le suivant. Nous traçons une droite passant par deux points (R, et M, qui est le pointeur), puis nous comptons le nombre d'intersections entre le point M et R. Si ce nombre est impair, le point M est dans le polygone, et si il est pair, le point M se trouve à l'extérieur du polygones.



La mise en place d'un raycast a été une tâche très difficile, puisqu'il a fallut déterminer quand notre segment [RM] et un coté du polygone sont sécants.

### 3 . Tests

#### a . Initialisation

Lors du lancement de l'application nous devons créer une un étage vide.

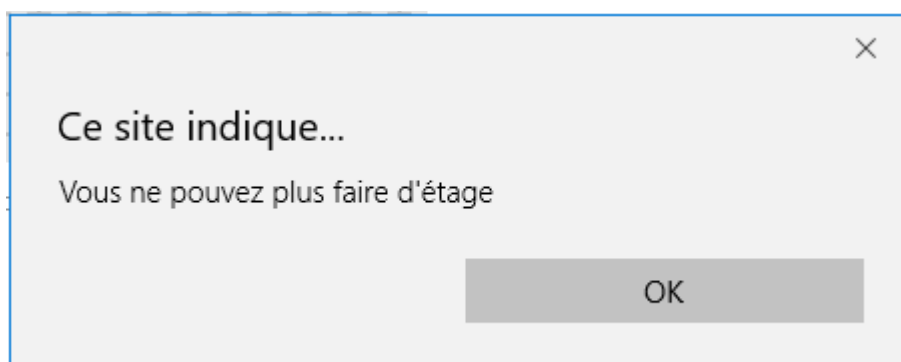
```
étage créé : 0
editorVectorTest.js (222,3)
└─ [object Array]      [Object {...}]
   └─ editorVectorTest.js (223,3)
      └─ 0              [object Object] {...]
         length          1
```

Nous voyons que l'étage a bien été créé, et que celui ci est bien vide.

#### b . Création et destruction d'étage

Nous pouvons utiliser le bouton qui créer un étage, et incrémente l'étage sélectionné.

La création d'étage est bloqué à 100, lorsque celui ci y arrive il fait apparaître un pop-up indiquant l'impossibilité d'en créer un de plus.

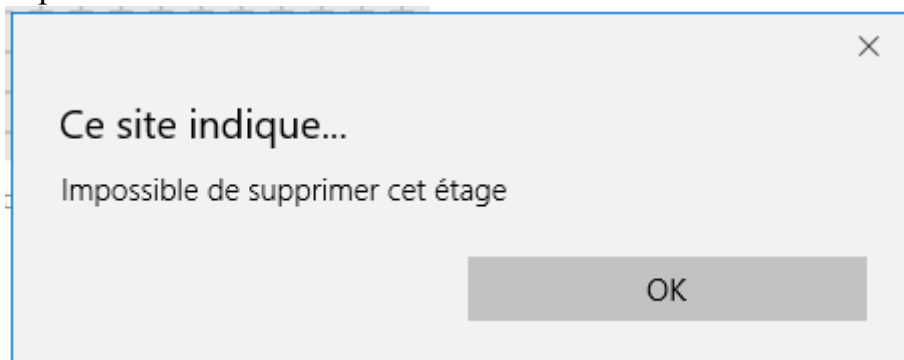


Lors de la suppression d'un étage, il décrémente l'étage sélectionné et supprime l'étage.

```
étage créé : 1
editorVectorTest.js (222,3)
└─ [object Array]      [Object {...}, Object {...}]
   └─ editorVectorTest.js (223,3)
      └─ étage détruit, étage sélectionné : 0
         editorVectorTest.js (236,3)
            └─ [object Array]      [Object {...}]
```

Si l'étage qui était sélectionné était l'étage 0, il ne décrémente pas et supprime l'étage.

De plus, si il ne reste qu'un seul étage, il est impossible de le supprimer, et un pop-up indique sont impossibilité.



### c . Navigation dans les étages

Nous pouvons passer d'un étage à l'autre en cliquant sur les boutons, le programme va alors incrémenter ou décrémenter l'étage sélectionné, mais ne va jamais dépasser le nombre d'étage présent

```
étage inférieur : 0
editorVectorTest.js (286,3)
impossible d'aller a l'étage inférieur
editorVectorTest.js (280,4)
étage supérieur : 1
editorVectorTest.js (298,3)
étage supérieur : 2
editorVectorTest.js (298,3)
impossible d'aller a l'étage supérieur
```

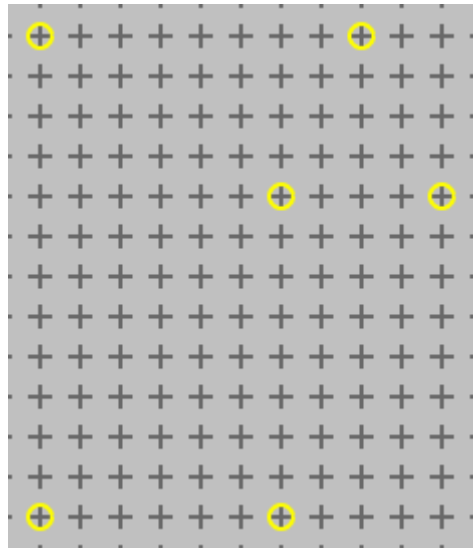
### d . Création des pièces

Pour créer une pièce il faut signaler à l'éditeur que nous passons en mode création de pièce. Nous modifions l'attribut "m\_status" de mainWindow, qui est là pour signaler en quel mode nous sommes, puis nous sommes capable de sélectionner des points sur la grille.

```
changement de l'état de l'éditeur vers le mode création de pièce
editorVectorTest.js (243,3)
```

Ici nous voyons que le mode a été changé.

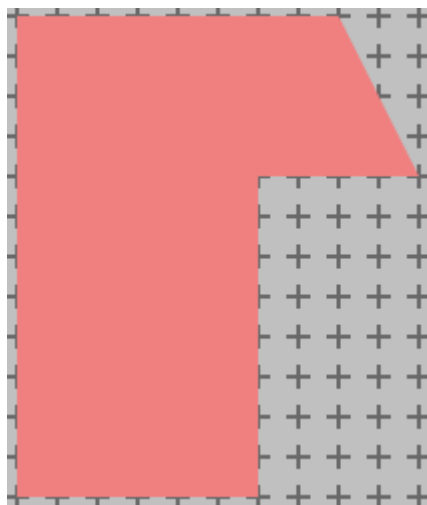
Ci dessous nous voyons les points sélectionnés.



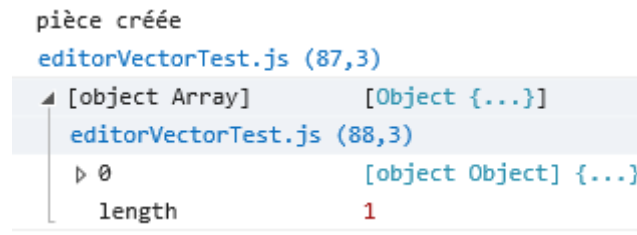
Lors de la validation des points en faisant un clique droit, l'éditeur nous demande d'entrer un nom de pièce.

A light grey dialog box with a blue border and a close button (X) in the top right corner. Inside the box, the text 'Entrez le nom de la pièce' is displayed. Below this text is a text input field with a blue border, containing the word 'Chambre'. To the right of the input field is a small 'X' button. At the bottom of the dialog box are two buttons: 'OK' and 'Annuler'.

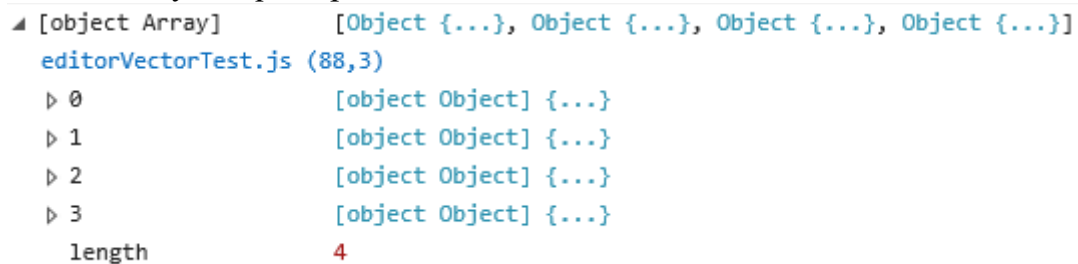
Une fois le nom saisi, la pièce s'affiche correctement.



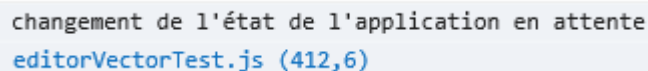
Ci dessous nous voyons l'attribut "m\_rooms" de la classe floor, qui témoigne de la création de la pièce.



Ci dessous nous voyons que 4 pièces ont été créées



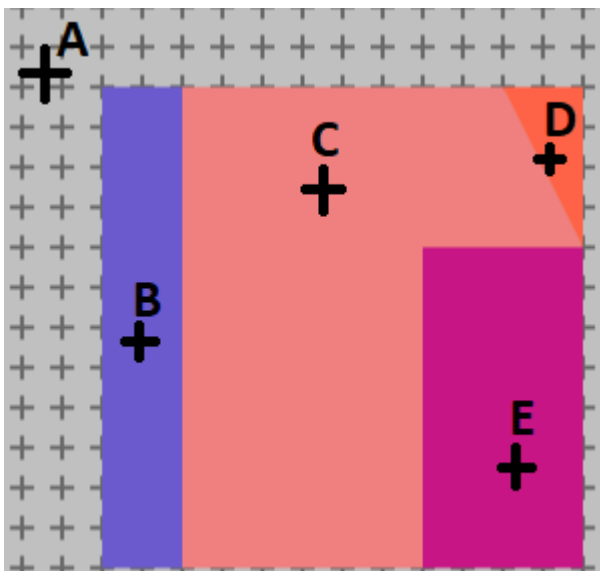
De plus l'attribut "m\_status" passe en mode attente, ce qui permet d'avoir le contrôle sur les autres fonctionnalités de l'application.



Si l'éditeur est en création de pièce, toutes les autres fonctionnalités de l'éditeur sont verrouillées, comme la création d'un étage, d'une pièce, le changement d'étage ou sa suppression, et la modification ou la suppression d'une pièce.

### e . Sélection des pièces

Pour sélectionner une pièce il nous faut utiliser notre raycast, pour le tester nous allons pointer différents endroits de l'éditeur contenant plusieurs pièces.



Le point A se trouve dans le vide, et devra afficher "vous ne survolez pas de pièce".

Le point B se trouve sur la pièce "Corridor" et devra afficher le nom puis l'indice dans le conteneur, donc "corridor : 0"

Le point C devra afficher "salle a manger : 1"

Le point D devra afficher "chambre : 2"

Et enfin le point E devra afficher "Salon : 3"

```
vous ne survolez pas de pièce
```

```
editorVectorTest.js (105,3)
```

```
corridor : 0
```

```
editorVectorTest.js (101,5)
```

```
salle a manger : 1
```

```
editorVectorTest.js (101,5)
```

```
chambre : 2
```

```
editorVectorTest.js (101,5)
```

```
Salon : 3
```

```
editorVectorTest.js (101,5)
```

Sur la console nous obtenons ceci, ce qui montre bien que notre raycast fonctionne.

Lorsque nous cliquons sur une pièce il faut pouvoir retenir que c'est celle ci sur laquelle nous avons cliqué, l'indice de la pièce sera donc stocké dans l'attribut "m\_selectedRoom", et s'affiche sur l'éditeur.

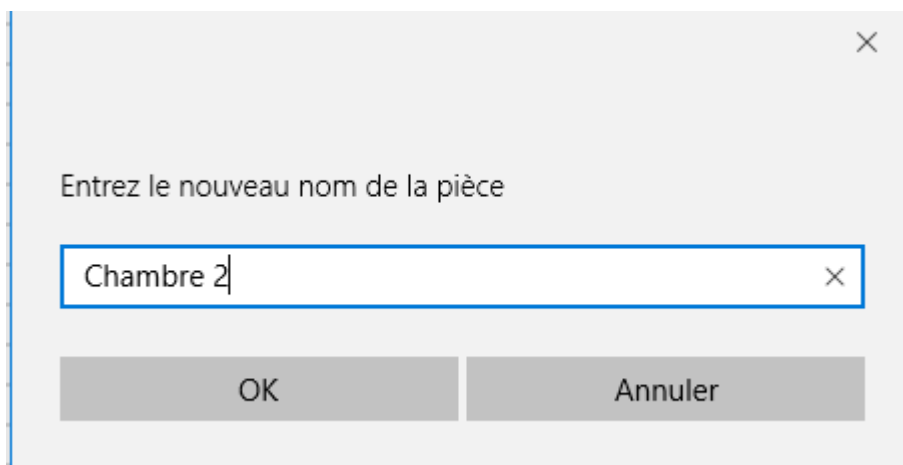
Pièce sélectionné : corridor

Changer le nom de la pièce

Supprimer la pièce

## f . Modification et destruction des pièces

Maintenant que nous pouvons sélectionner notre pièce, nous pouvons lui changer de nom, en cliquant sur le bouton nous le programme nous fait apparaître une pop-up pour que nous puissions changer de nom.



Après la validation le nom change, et affiche son nouveau nom.



Pièce sélectionné : Chambre 2

Changer le nom de la pièce

Supprimer la pièce

Si nous cliquons sur le bouton supprimer, la pièce est détruite et la pièce est désélectionnée.

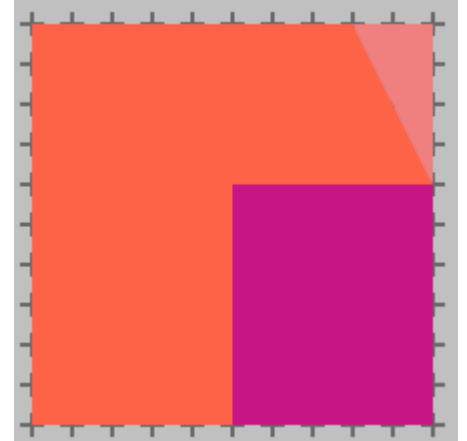
Ici nous voyons qu'il ne reste que trois pièces, que la pièce sélectionnée a été supprimée. Et qu'il n'y a plus de pièce sélectionnée.

```
▲ [object Array]      [Object {...}, Object {...}, Object {...}]  
  editorVectorTest.js (315,3)  
    ▶ 0                [object Object] {...}  
    ▶ 1                [object Object] {...}  
    ▶ 2                [object Object] {...}  
    length              3
```

Pièce sélectionné : Aucune pièce sélectionné

Changer le nom de la pièce

Supprimer la pièce



## g . Sauvegarde du plan

Lorsque nous sauvegardons, le programme nous donne une sauvegarde au format XML.

```
- <blueprint>
- <floor>
- <room>
  <name>pièce 1 étage 0</name>
  - <point>
    <x>9</x>
    <y>9</y>
  </point>
  - <point>
    <x>9</x>
    <y>15</y>
  </point>
  - <point>
    <x>14</x>
    <y>15</y>
  </point>
  - <point>
    <x>14</x>
    <y>9</y>
  </point>
</room>
</floor>
- <floor>
- <room>
  <name>pièce 1 étage 1</name>
  - <point>
    <x>9</x>
    <y>7</y>
  </point>
  - <point>
    <x>8</x>
    <y>17</y>
  </point>
  - <point>
    <x>12</x>
    <y>18</y>
  </point>
  - <point>
    <x>13</x>
    <y>13</y>
  </point>
  - <point>
    <x>17</x>
    <y>13</y>
  </point>
  - <point>
    <x>17</x>
    <y>7</y>
  </point>
</room>
</floor>
</blueprint>
```

Ici nous avons un exemple de sauvegarde avec deux étages, avec dans chacun d'eux une pièce où tout les points sont stocké ainsi que le nom.

## IV . Travail restant

À ce stade, il ne reste plus qu'à envoyer le le fichier XML à la base de donné du serveur, puis associer les noms des pièces déjà existant de la base de donné du à l'éditeur.