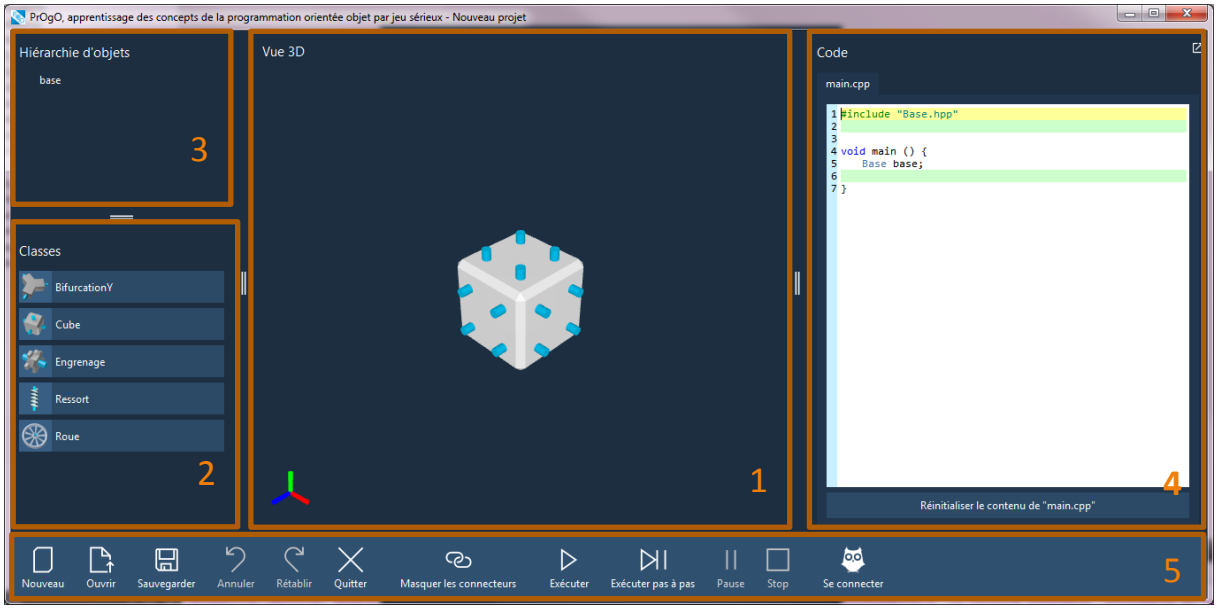
PrOgO : tutoriel d'utilisation d'objets

A l'IUT du Puy en Velay, dans le cadre du projet Tactileo, nous concevons PrOgO, un jeu sérieux pour l'apprentissage des concepts de base de la Programmation Orientée-Objet (POO) en C++.
L'objectif de la séance d'aujourd'hui est d'apprendre le concept d'*«Objet »*.

1. But du jeu et interface utilisateur

Le but du jeu dans PrOgO est de construire une structure robotique ou mécanique et de l'animer. PrOgO donne libre court à votre créativité et à votre imagination, et vous permet de réaliser un robot virtuel ou une machine virtuelle.



- **[1]** : L'interface de PrOgO dispose en son centre d'une scène 3D créative, dans laquelle se trouve en permanence un bloc 3D de base pour la construction du robot ou de la machine de votre choix.
  - Il est possible de tourner autour de l'objet en déplaçant la souris avec la molette enfoncée, et de zoomer sur la construction 3D en tournant la molette.
  - Si vous êtes sur un écran tactile, tourner autour de l'objet se fait à l'aide d'un contact tactile en déplaçant votre doigt. Vous pouvez également zoomer et dé-zoomer via un geste de pincement des doigts sur l'écran.
- **[2]** : En bas à gauche de l'interface se trouve l'onglet « *Classes*» qui liste les blocs de construction 3D que vous pouvez créer dans votre scène 3D.
- **[3]** : En haut à gauche de l'interface est affichée l'arborescence des objets venant d'être créés et assemblés à l'objet de base dans la scène 3D.
- **[4]** : L'onglet de droite abrite un fichier nommé « *main.cpp* », dans lequel est généré le code C++ correspondant à toutes vos actions directes sur les objets dans la scène 3D. L'édition du code n'est possible qu'aux lignes vertes, et un système d'autocomplétion vous assiste dans la rédaction des instructions. Le résultat de la modification du code dans cet éditeur est instantanément retranscrit sur les objets dans la scène 3D.

- Afin d'utiliser l'éditeur à autocomplétion, il suffit de faire un clic gauche (ou un toucher tactile) sur les lignes en surbrillance verte dans le fichier « *main.cpp* ». Les instructions de code possibles vous seront alors proposées en fonction de la ligne sélectionnée et du contenu courant du fichier *main.cpp*. Notez que le code n'est pas éditable au clavier, vous devez obligatoirement utiliser un des choix proposés dans la liste affichée. En revanche, il est possible d'annuler et de rétablir vos actions à l'aide des boutons « *Annuler* » et « *Rétablir* ». Cela concerne aussi bien les actions réalisées à l'intérieur de l'éditeur que dans la scène 3D.
- Il est possible de réinitialiser le contenu du fichier main.cpp en cliquant sur le bouton ***Réinitialiser le contenu de main.cpp***

[5] : La barre de menus permet l'ouverture et la sauvegarde de nouveaux projets, l'annulation et le rétablissement des actions réalisées à l'intérieur de la scène 3D **[1]** ou de l'éditeur à autocomplétion **[4]**, de masquer/démasquer les connecteurs de construction présents sur les blocs 3D, et de lancer l'exécution de l'animation programmée sur la structure 3D.

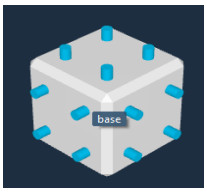
- Il est possible de visualiser la trace d'exécution de toutes les instructions présentes dans le fichier *main.cpp* en cliquant successivement sur le bouton ***Exécuter pas à pas***. Cela va entraîner l'affichage en surbrillance jaune des instructions en cours d'exécution, ainsi que la visualisation simultanée des résultats de leur exécution dans la scène 3D.
- La visualisation de l'animation de votre structure 3D se fait à l'aide du bouton ***Exécuter.***

2. Utilisation d'objets dans PrOgO

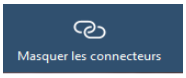
2.1 Un objet est une instance de classe

2.1.1 Création directe d'objets dans la scène 3D

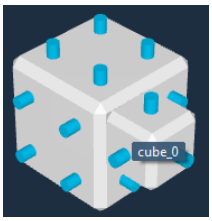
- Dans l'onglet *Classes*, vous pouvez choisir la classe à instancier à l'aide d'un simple clic gauche de la souris (ou d'un toucher tactile). Cela va créer un objet qu'il faudra connecter à l'objet de base nommé `base` en sélectionnant un de ses connecteurs à l'aide d'un clic gauche de la souris (ou d'un toucher tactile).



- Les connecteurs vous servent d'assistants visuels, et vous permettent d'assembler facilement vos composants 3D. Vous pouvez les masquer en cliquant sur le boutant ***Masquer les connecteurs***.



- A titre d'exemple, commencez par créer un objet cube en choisissant le composant *Cube*, puis sélectionnez un connecteur sur l'objet `base`. L'objet venant d'être créé se place alors à l'endroit du connecteur sélectionné comme sur l'image ci-après. Un nom par défaut `cube_0` lui est attribué, ce nom est l'identifiant unique de cet objet.



- Observez dans l'onglet de droite le code C++ généré suite à cette action :

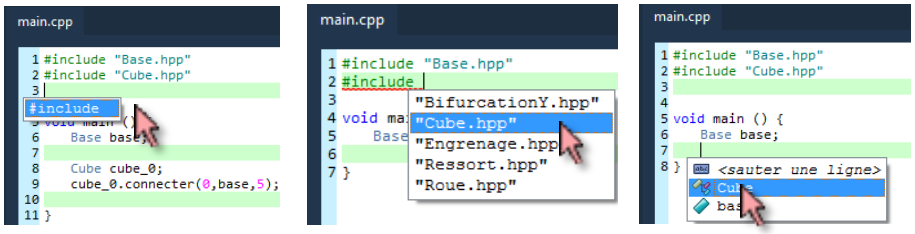
main.cpp

```
1 #include "Base.hpp"
2 #include "Cube.hpp"
3
4
5 void main () {
6     Base base;
7
8     Cube cube_0;
9     cube_0.connecter(0,base,5);
10
11 }
```

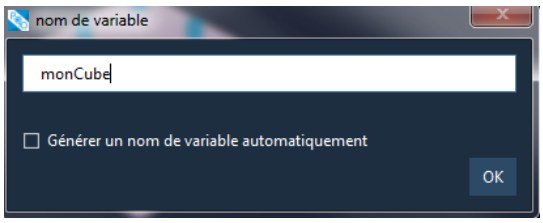
- L'instruction `#include "Cube.hpp"` est automatiquement ajoutée à l'entête du fichier `main.cpp`
- Le cube que vous avez créé est une instance de la classe `Cube` nommée `cube_0`. L'équivalent d'une telle déclaration en langage C++ s'écrit « `Cube cube_0;` » comme à la ligne 8 de l'éditeur.
- Dans PrOgO, les connecteurs des objets possèdent des identifiants numériques (0, 1, 2, …, 16). L'instruction de la ligne 9 « `cube_0.connecter(0,base,5);` » signifie que l'objet `cube_0` est rattaché par défaut par son connecteur 0 à l'objet `base`, et l'objet `base` se retrouve ainsi connecté par son connecteur 5 à l'objet `cube_0`.
  - Cette connexion se fait grâce à l'appel de la méthode `connecter()` sur l'objet `cube_0` prenant trois paramètres (le connecteur par défaut de l'objet `cube_0`, l'objet `base` et le connecteur 5 de l'objet `base`)
- Vous pouvez connecter d'autres objets à l'objet `cube_0` de la même manière que vous avez connectez `cube_0` à `base`.

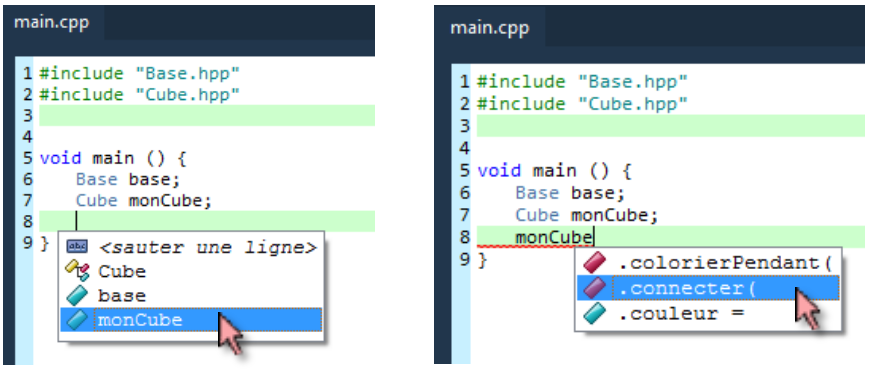
2.1.2 Création d'objets dans l'éditeur à autocomplétion

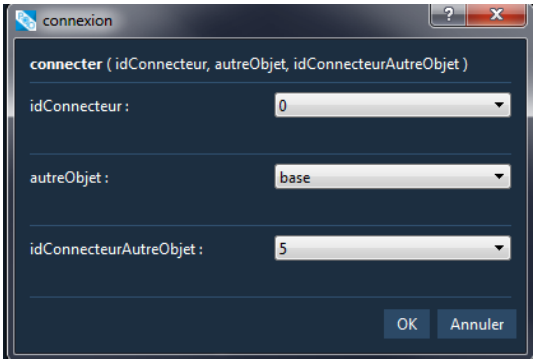
Dans l'éditeur à autocomplétion, faites un clic gauche (ou un contact tactile) sur la ligne verte sous l'instruction `#include "Base.hpp"`. L'option `#include` vous sera alors proposée. Une liste de fichiers de déclaration de classes apparait alors. Vous pouvez choisir à titre d'exemple `"Cube.hpp"`. L'instruction `#include "Cube.hpp"` s'ajoute tout de suite à l'entête du fichier. Vous avez ensuite, la possibilité d'instancier la classe `Cube` dans le corps de la fonction `main()`.



Laissez-vous guider par les propositions de l'assistant de contenu de l'éditeur. Vous pourrez créer un nouveau cube, que vous pourrez connecter à l'objet `base` via la méthode `connecter()`. Deux boîtes de dialogue vont alors apparaître vous invitant à saisir le nom de l'objet à créer, ainsi que les paramètres de la méthode `connecter()`.

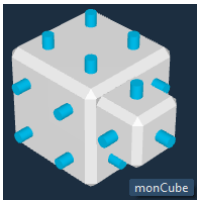






main.cpp

```
1 #include "Base.hpp"
2 #include "Cube.hpp"
3
4
5 void main () {
6     Base base;
7     Cube monCube;
8     monCube.connecter(0,base,5);
9
10 }
```



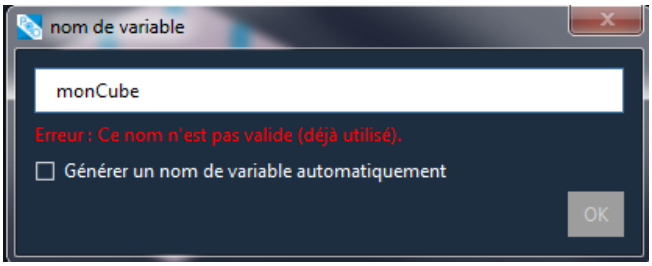
- Observez le résultat immédiat des instructions entrées sur les objets dans la scène 3D : Le cube venant d'être ainsi créé se place à l'endroit du connecteur choisi de l'objet `base`.

- Vous pouvez également visualiser la trace d'exécution de toutes ces instructions en cliquant successivement sur le boutant ***Exécuter pas à pas***. Cela va entraîner l'affichage en surbrillance jaune de ces instructions en cours d'exécution, ainsi que le résultat de leur exécution sur la construction 3D dans la scène.

main.cpp

```
1 #include "Base.hpp"
2 #include "Cube.hpp"
3
4
5 void main () {
6     Base base;
7     Cube monCube;
8     monCube.connecter(0,base,5);
9
10 }
```

- Observez que vous ne pouvez pas attribuer le même nom à deux objets différents :



- Vous pouvez également modifier le nom de votre objet monCube à laide d'un clic droit (ou une pression de doigt maintenue) :

```
4
5 void main () {
6     Base base;
7
8     Cube monCube;
9     monCube.c
10
11 }
```



```
4
5 void main () {
6     Base base;
7
8     Cube cube;
9     cube.connecter(0,base,5);
10
11 }
```

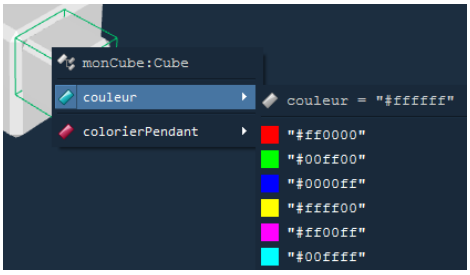
2.2 Un objet possède des attributs et des méthodes

2.2.1 Accès direct aux attributs et méthodes d'un objet dans la scène 3D :

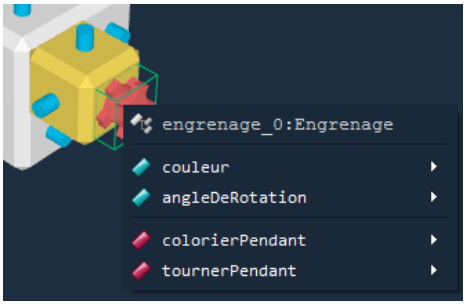
- Les attributs et méthodes se distinguent comme ceci :

- Une icône bleu identifie un attribut. Le nom de l'attribut est indiqué à côté de l'icône. Un attribut est une qualification de l'objet (la couleur de monCube est « `"#ffffff"` »)
- Une icône rouge identifie une méthode. Le nom de la méthode est indiqué à côté de l'icône. Une méthode est une action que l'objet peut réaliser (l'engrenage `engrenage_0` tourne de 45° pendant 3 secondes – `engrenage_0.tournerPendant(45,3)`)

- Un clic droit (ou un toucher maintenu) sur un objet dans la scène permet de le sélectionner. Une boite englobante s'affiche alors autour de l'objet indiquant que l'objet est bien sélectionné. Cela entraîne au même moment l'apparition d'un menu déroulant qui invite à sélectionner les caractéristiques de l'objet, c'est-à-dire ***ses attributs et ses méthodes***.



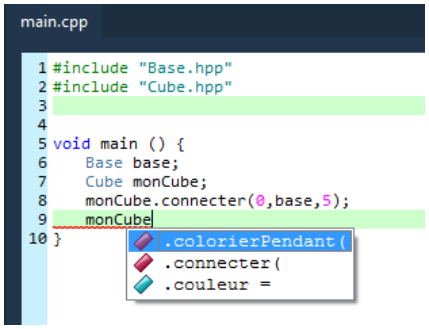
- Vous remarquerez également sur l'entête du menu déroulant l'affichage de l'identité de l'objet, c'est-à-dire son nom et sa classe : `monCube:Cube`
- La sélection de l'attribut `couleur` affiche un second menu invitant à choisir une nouvelle valeur. Vous remarquerez sur l'entête de ce sous-menu l'affichage de la valeur courante de l'attribut sélectionné. Dans l'exemple ci-dessous s'affiche la valeur initiale (par défaut) de l'attribut `couleur` : `couleur = "#ffffff"`

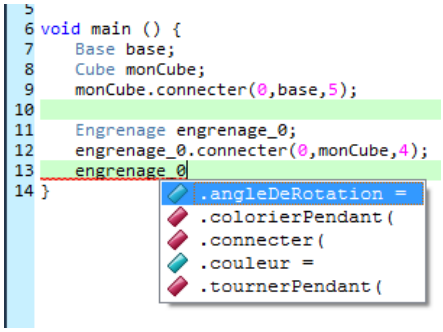


- Vous remarquerez que les objets ne comportent pas tous les mêmes attributs et les mêmes méthodes.

2.2.2 Accès aux attributs et méthodes d'un objet dans l'éditeur à autocomplétion

- Dans l'éditeur à autocomplétion, après avoir créé un objet, l'assistant va proposer dans un menu déroulant le nom de cet objet vous invitant à le sélectionner. Vous pourrez ensuite, accéder à la liste de ses attributs et méthodes afin d'agir sur cet objet.
- À titre d'exemple, après avoir créé l'objet `monCube`, vous pouvez accéder à ses attributs et méthodes à l'aide d'un clic gauche (ou contact tactile) sur la ligne verte après l'instruction « `Cube monCube;` ». L'assistant vous proposera dans un premier temps la liste des objets puis après avoir sélectionné l'objet `monCube`, il vous affichera la liste d'attributs et de méthodes de l'objet `monCube`.





- Remarquez que la liste d'attributs et de méthodes de l'objet `engrenage_0` est différente de celle de l'objet `monCube` étant donné que ces objets ne partagent pas la même classe.

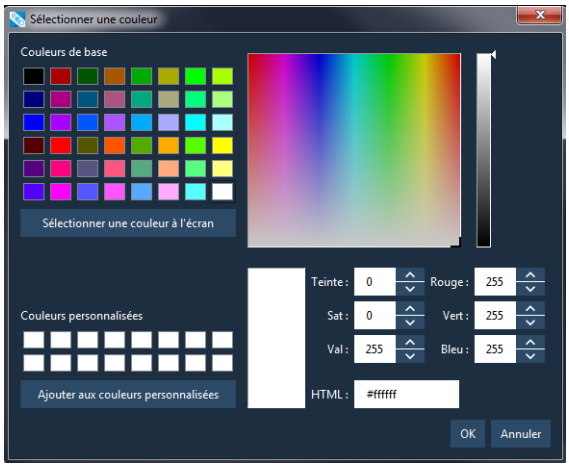
2.3 Modification de l'état d'un objet

2.3.1 Modification directe dans la scène 3D

Modification de l'état d'un objet par le biais de ses attributs

L'une des manières de modifier l'état d'un objet est de modifier la valeur d'au moins un de ses attributs.

À titre d'exemple, vous pouvez modifier la valeur de l'attribut `couleur` de l'objet `cube_0` en sélectionnant une couleur de votre choix. Vous pouvez également personnaliser cette couleur en choisissant *Autre*. La fenêtre ci-après doit alors apparaître. Le résultat est tout de suite visible sur l'objet dans la scène 3D.



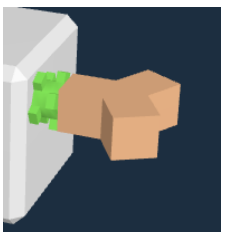
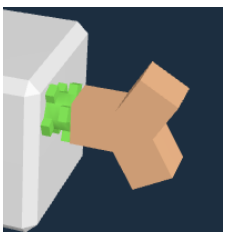
Observez le code généré suite à la modification de la valeur de l'attribut `couleur` de l'objet `cube_0` :

```
cube_0.couleur= "#ffff00";
```

Observez le code généré suite à la modification de la valeur de l'attribut `angleDeRotation` de l'objet `engrenage_0` :

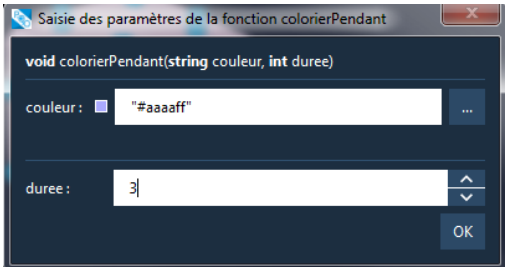
```
engrenage_0.angleDeRotation = 45;
```

Vous pouvez expérimenter la modification de l'état de plusieurs objets connectés grâce à l'attribut `angleDeRotation`. À titre d'exemple, vous remarquerez que quand vous connectez un objet `bifurcationy_0` de type `BiffurcationY` à l'objet `engrenage_0`, la modification de l'attribut `angleDeRotation` à `90°` de l'objet `engrenage_0` va tourner cet objet d'un angle de `90°` tout en entraînant avec lui l'objet `bifurcationy_0`. Cela va générer le code : `engrenage_0.angleDeRotation = 90;`

Modification de l'état d'un objet par le biais de ses méthodes

- Une autre manière de modifier l'état d'un objet est d'appeler l'une de ses méthodes.
- À titre d'exemple, vous pouvez appeler la méthode `colorierPendant()` afin de colorier l'objet `cube_0` pendant une durée définie en secondes que vous aurez saisi via la fenêtre ci-contre.



- Observez le code généré suite à l'appel de la méthode `colorierPendant()` de l'objet `cube_0` :

```
cube_0.colorierPendant("#aaaaff",3);
```

- Observez le code généré suite à l'appel de la méthode `tournerPendant()` de l'objet `engrenage_0` :

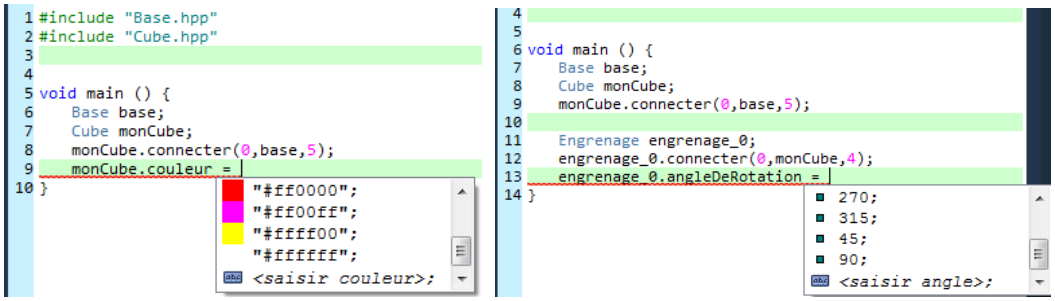
```
engrenage_0.tournerPendant(45,3);
```

- Vous pouvez expérimenter comme précédemment, la modification de l'état de plusieurs objets connectés grâce à la méthode `tournerPendant()`.

Modification dans l'éditeur à autocomplétion

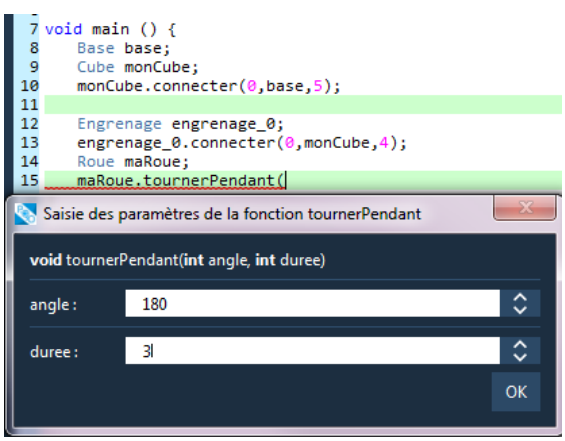
Modification de l'état d'un objet par le biais de ses attributs

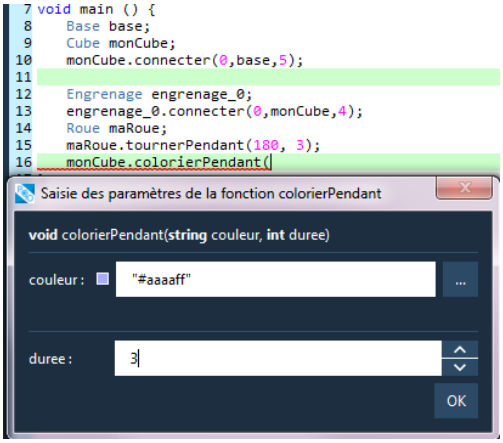
- Dans l'éditeur à autocomplétion, une fois que vous avez rentré le nom d'un objet préalablement créé, vous pourrez changer la valeur de l'un de ses attributs grâce au menu déroulant affichant ses attributs et méthodes. Vous pouvez alors, modifier la couleur de l'objet `monCube`, et choisir un angle de rotation pour l'objet `engrenage_0` comme sur les images ci-dessous.



Modification de l'état d'un objet par le bais de ses méthodes

- De la même manière que pour la modification des valeurs des attributs, vous pouvez changer l'état d'un objet préalablement créé en appelant ses méthodes. Vous pouvez alors appeler ses méthodes comme par exemple `tournerPendant()` et `colorierPendant()` sur les objets `maRoue` et `monCube` :





Remarquez que pour visualiser le résultat animé de l'appel de ces méthodes, vous devrez lancer son exécution à l'aide du bouton ***Exécuter***.

3. À vous de jouer !

Construisez et animez le robot ou la machine de votre choix dans le monde de PrOgO, donnez libre court à votre créativité, inspirez-vous d'objets que vous connaissez dans votre environnement réel, ou dans un monde imaginaire et surtout apprenez la POO.