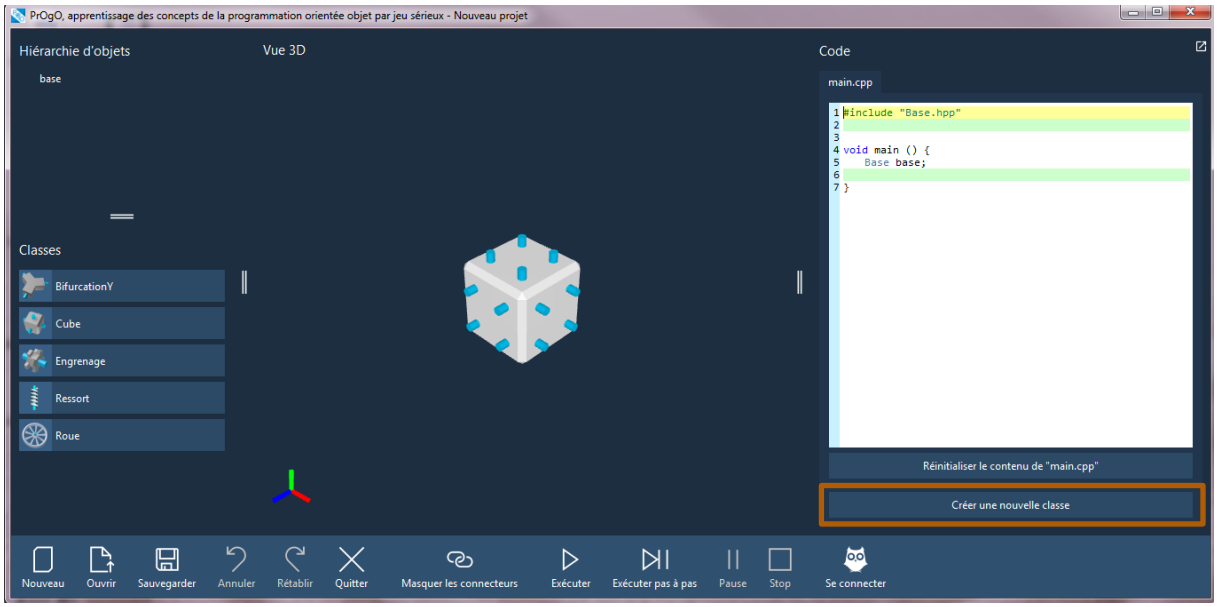
PrOgO : tutoriel de création de classes

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 de «*Classe*».

1. Interface utilisateur

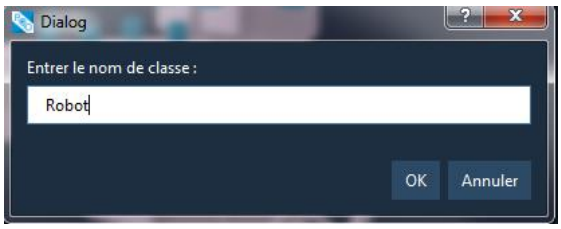


- Ouvrez votre ancien projet créé lors de la séance «***Utilisation d'objets***».
- Remarquer la présence d'un bouton ***Créer une nouvelle classe*** en bas de l'onglet contenant l'éditeur de code.

2. Création de classes avec PrOgO

2.1 Créer une classe revient à créer une nouvelle abstraction, ou un nouveau type de données

- Toute nouvelle construction dans PrOgO constitue une nouvelle classe que vous pouvez créer grâce au bouton ***Créer une nouvelle classe*** se trouvant en bas de l'éditeur de code.



- En cliquant sur ce bouton, une boite de dialogue s'affichera vous invitant à saisir le nom de la classe à créer. A titre d'exemple, vous pouvez la nommer `Robot`.

- Dès que la classe est créée, elle est ajoutée à un nouvel onglet «*Mes classes*» qui apparait sous l'onglet « *Classes*» pour être instanciée dans la scène 3D autant de fois que vous le souhaitiez.

C'est une nouvelle donnée qui existe désormais.

2.2 Une classe possède des données et fonctions membres

Dès que la classe `Robot` est créée, deux nouveaux fichiers de code s'ajoutent dans l'éditeur: `Robot.hpp et Robot.cpp` en plus du fichier `main.cpp`

- `Robot.hpp` contient la déclaration de la classe venant d'être créée, c'est-à-dire l'ensemble de ses membres (attributs et méthodes) ou caractéristiques. Son entête contient les inclusions des classes utilisées lors de la constitution de la structure robotique. Observez le code de déclaration de l'entête de la classe :

```
class Robot {
    …
} ;
```

- Le mot clé `class` suivi de `Robot` désigne que l'on déclare une classe nommée `Robot`.
- Le corps de la classe `Robot` liste l'ensemble de ses constituants : les objets utilisé lors de sa construction sont ses attributs membres. Les nouvelles fonctions propres à la classe Robot sont ses fonctions membres ou méthodes. Les nouvelles méthodes sont donc `initialiserAngles(),initialiserCouleurs(), animer() et reinitialiser()`.

2.3 Une classe encapsule ses données et fonctions membres

- Les membres précédés par le mot clé `public` sont accessibles à la fois à l'intérieur de la classe `Robot` et en dehors de cette classe (dans la fonction main(), par exemple).
- Les membres précédés par le mot clé `private` sont accessibles uniquement par les autres membres de la même classe. Ainsi les fonctions `initialiserAngles(), et initialiserCouleurs()` peuvent accéder à tous les attributs membres privés, mais elles même ne peuvent pas être appelées en dehors de la classe `Robot` (pas dans la fonction main(), par exemple). Elle sont en revanche appelées par la fonction `reinitialiser()`.
- Vous pouvez observer cela dans le fichier de définition Robot.cpp.

```
private:
  Base base;
  BifurcationY bifurcationy_0;
  BifurcationY bifurcationy_1;
  Cube cube_0;
   ……

   void initialiserAngles();
   void initialiserCouleurs();

public:
   Robot();
   void animer();
   void reinitialiser(bool b_angle, bool b_color);
```

- Le fichier `Robot.cpp` définit le comportement de la classe, c'est-à-dire le corps de ses fonctions membres. Vous pouvez observer par exemple, que la fonction `reinitialiser()` appelle les fonctions privées `initialiserAngles()`, et `initialiserCouleurs()` :

```
void Robot::reinitialiser(bool b_angle, bool b_color){
  if (b_angle)
      initialiserAngles();
  if (b_color)
      initialiserCouleurs();
}
```

- L'écriture `Robot::reinitialiser` signifie que l'on définit la méthode `reinitialiser()` de la classe `Robot`.
- Observez également que les fonctions privées `initialiserAngles()`, et `initialiserCouleurs()` accèdent aux membres privés de la classe :

```
void Robot::initialiserAngles(){
     engrenage_0.angleDeRotation = 0;
     engrenage_1.angleDeRotation = 0;
     …
}

void Robot::initialiserCouleurs(){
     base.couleur = "#ffffff";
     bifurcationy_0.couleur = "#ffffff";
     bifurcationy_1.couleur = "#ffffff";
     ……
}
```

- Vous pouvez observer que la méthode `animer()` contient toutes vos actions d'animation réalisées préalablement lors de construction de votre classe.

```
void Robot::animer(){
     engrenage_0.tournerPenadant (90,3);
     engrenage_1.colorienPendant("#ffffff",10)
     …
}
```

2.4 Une classe possède un constructeur

- Une classe possède un constructeur qui permet d'initialiser un objet lors de sa création (par l'instanciation de sa classe).
- Le constructeur doit porter le même nom que sa classe, doit être public et ne doit pas avoir de type de retour :
  - Observer la déclaration du constructeur de la classe `Robot` :

```
public:
      Robot();
```

➡ Observer sa définition dans le fichier `Robot.cpp`

```
Robot::Robot(){
   engrenage_0.connecter(0,base,13);
   engrenage_1.connecter(0,base,9);
   roue_0.connecter(0,engrenage_0,1);
   …
   initialiserAngles();
   initialiserCouleurs();
}
```

2.5 Utilisation d'une classe

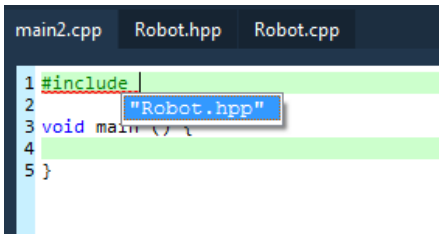
➡ Dans la fonction `main()` vous pouvez inclure le fichier `Robot.hpp` afin d'instancier la classe `Robot`.

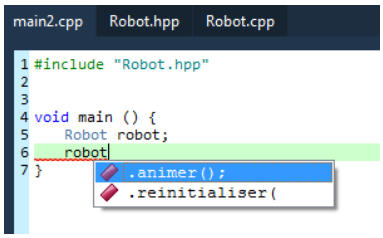
➡ observer le code généré. Le constructeur de la classe est invoqué suite à cette instruction :

```
Robot robot ;
```

➡ l'instanciation de la nouvelle classe peut être également réalisée directement en la sélectionnant dans l'onglet «*Mes classes*».

➡ A l'intérieur de la fonction `main()`, une fois l'instance de `Robot` est créée, vous pouvez invoquer les méthodes publiques, c'est-à-dire `animer()` et `reinitialiser()`.





➡ pour visualiser l'animation appuyez sur le boutant ***Exécuter.***

3. À vous de jouer !

Vous pouvez améliorer votre première réalisation en modifiant son apparence (en modifiant les anciennes données membres ou en lui ajoutant des nouvelles données membres), et en modifiant son comportement (en modifiant son animation).