

Développement d'un composant Delphi - Eléments de solution

Mise en œuvre : Delphi 6 version professionnelle

Remarques d'ordre général

Mise en garde

Tout au long de la réalisation de cette activité, les étudiants seront amenés à compiler, installer, utiliser et désinstaller des paquets de composants. La plupart de ces paquets auront en commun des noms de fichiers sources (horloge.pas notamment). De plus, la même machine pourra servir de station de travail à plusieurs étudiants n'étant pas parvenus à la même étape dans le développement (cas des groupes de TP).

Dans ces circonstances, il arrive fréquemment que de mauvaises manipulations soient effectuées et que la compilation ou l'exécution des projets utilisant les composants se passent mal...

En cas de problème apparemment insoluble, deux choses sont à examiner en priorité :

- Les chemins de bibliothèques indiqués dans les options d'environnement sont-ils corrects ? Supprimer les chemins incorrects ou inutiles pour l'exécution du projet.
- Les paquets d'exécution indiqués par l'option "Installer des paquets" sont-ils corrects ? Supprimer les paquets incorrects ou inutiles pour l'exécution du projet.

Test du composant

Il est souhaitable de tester chaque version à l'aide d'un programme de test (le projet fourni en corrigé dans chaque dossier solution) d'une part, et en temps que composant installé d'autre part. Il faut pour cela :

- Installer le composant.
- Créer un nouveau projet.
- Insérer une horloge sur la fiche principale et la paramétrer.
- Tester l'exécution.
- Désinstaller le composant pour permettre le test de la version suivante.

Recours à UML

Il m'a semblé intéressant de recourir à UML pour la notation de certains éléments de cette application. Je crois en effet que le problème s'y prête bien, en tous cas beaucoup mieux qu'un problème plus "classique" d'informatique de gestion.

Remarque : cet avis n'engage que moi et date de janvier 2002, les choses peuvent changer...

A/ Eléments de base

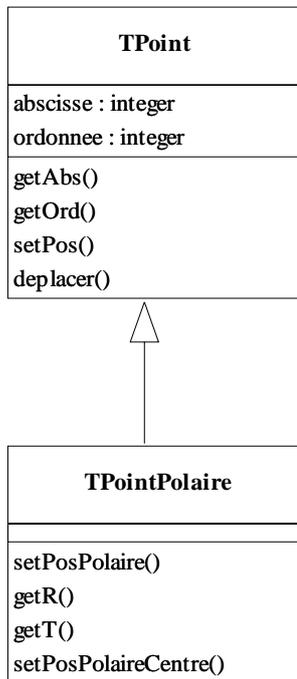
Une solution possible au problème posé se trouve dans le dossier solution "bases".

2. Un premier pas

Il ne faut pas oublier de créer le point manipulé par un appel au constructeur Create, et de le libérer par un appel à Free (il vaut toujours mieux appeler Free que Destroy car il y a vérification systématique de l'existence de l'objet avant la destruction).

3. Points polaires

Diagramme de classes



Remarque : la méthode `setPosPolaire` est ajoutée à la troisième question.
La solution programmée figure dans le projet.

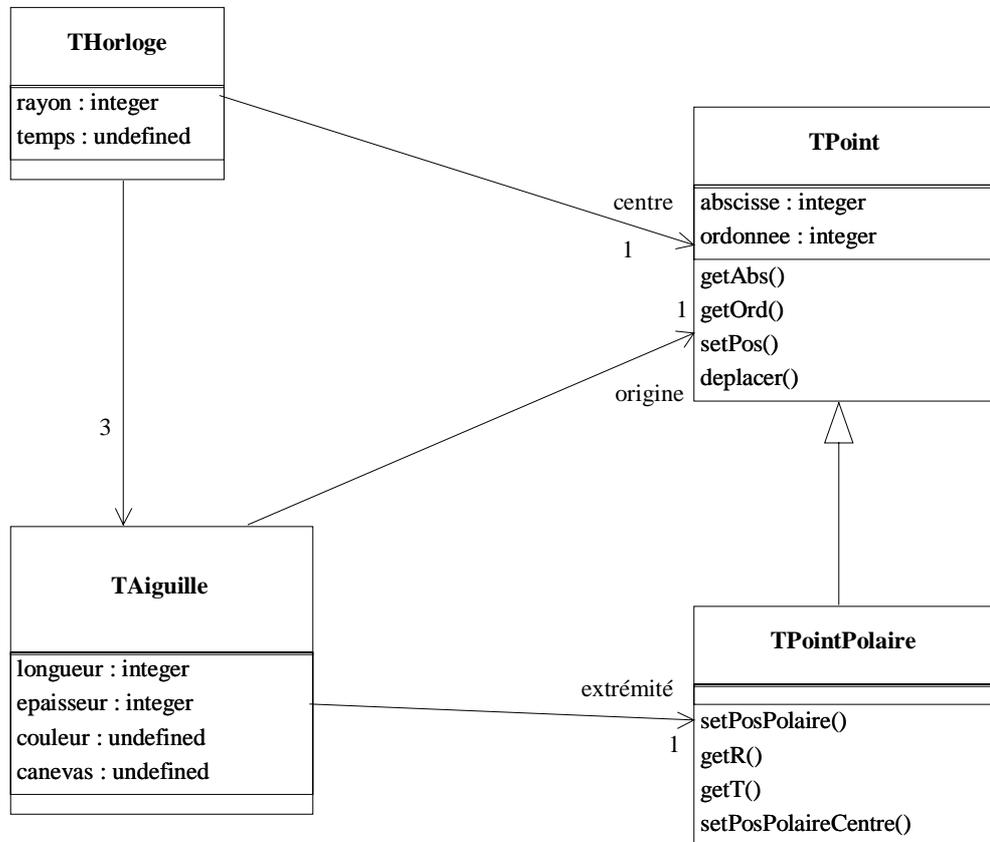
Remarque : les fonctions `getAbs`, `getOrd`, `getR` et `getT` sont réellement similaires pour les clients de la classe. On peut donc se demander si la notion de "fonction d'accès" souvent mentionnée est pertinente...

6. Installation du composant

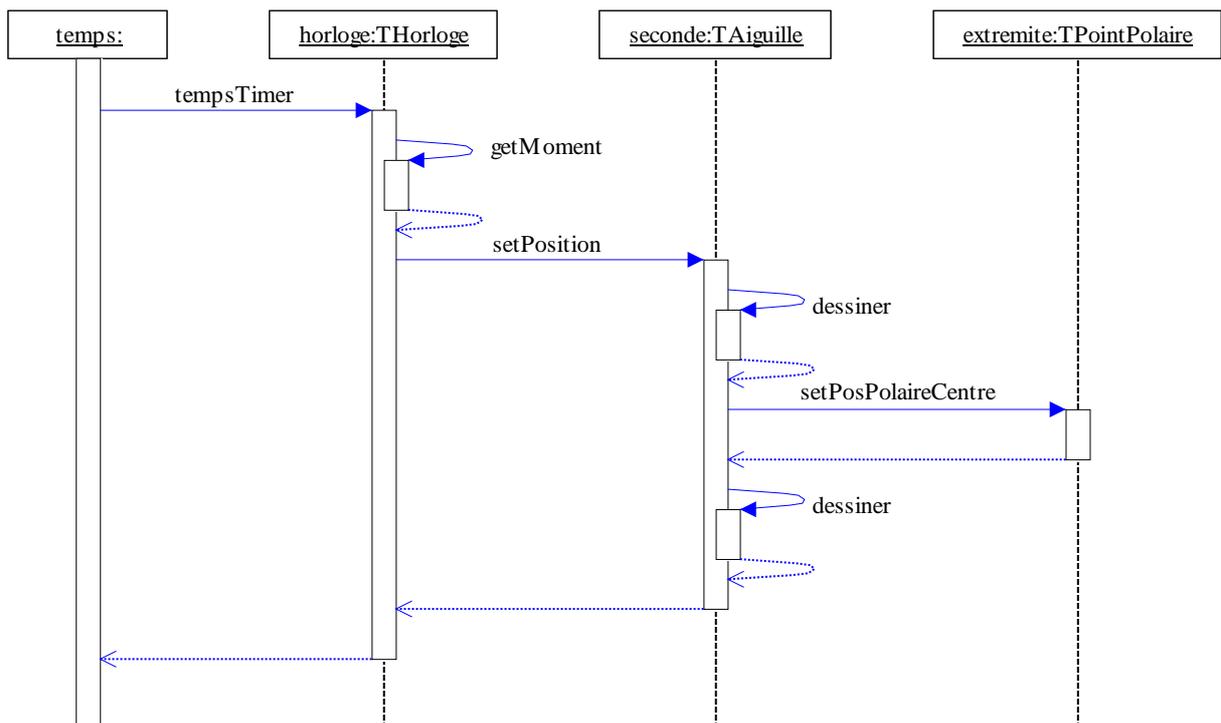
Le bitmap se trouve dans le fichier "Horloge.dcr".

B/ Une horloge simple

1. Les classes



2. Réalisation du composant



Remarque : on pourrait également faire figurer le détail de la méthode setPosPolaireCentre.

La réalisation complète figure dans le dossier "simple".

C/ Amélioration du composant

La version "améliorée" de l'horloge figure dans le dossier "amelioration".

D/ Ajout d'une fonction alarme

La version finale de l'horloge se trouve dans le dossier "alarme". Avant l'ajout de la possibilité de paramétrer la réaction de l'alarme, la gestion événementielle du timer est plus simple : il suffit d'afficher le message.

Remarque : l'appel d'invalidate après le paramétrage et le déclenchement de l'alarme permet d'en rafraîchir l'affichage.