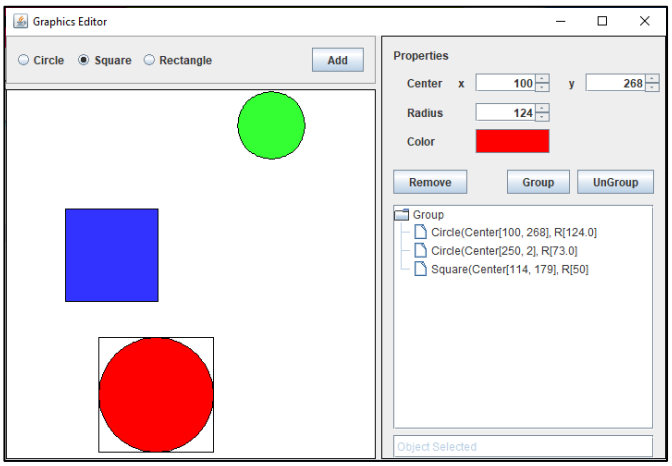
DEPARTEMENT INFORMATIQUE
71 rue Peter Fink
01000 Bourg-en-Bresse
04 74 45 50 59

TP4 – Virtual World

R3.04 QUALITE DE DEVELOPPEMENT

I. INTRODUCTION

Dans ce TP, vous allez utiliser les designs patterns *Composite* et *Factory* pour créer et gérer vos modèles. Le design pattern composite permet de gérer une hiérarchie d'éléments tandis que le design pattern *Factory* va permettre de créer à la volée différents objets pour notre scène. Le projet est structuré avec le design pattern MVC que vous avez utilisé jusqu'à présent.



II. PRESENTATION DU PROJET

Nous souhaitons créer un éditeur de scènes virtuelles. Cet éditeur doit permettre d'ajouter différents éléments ou groupes d'éléments à la scène, de pouvoir les modifier et les supprimer comme tout éditeur vectoriel. Nous nous intéresserons à n'importe quelle donnée qui peut être représentée comme des arbres, des forêts, des maisons, voitures, etc… Pour commencer, vous allez travailler sur des formes géométriques simples afin de définir le cœur du programme et les différentes interactions. Ensuite, vous adapterez les formes géométriques afin de proposer un éditeur de scènes virtuelles.

Le projet de base est composé de plusieurs vues : une vue de dessin *GraphicsPainter*, une vue arborescente listant l'ensemble des modèles *JTree*, un formulaire d'ajout de formes, une zone permettant de changer les propriétés des formes, des boutons de modification de l'arborescence et une zone d'affichage de texte *JTextPane*. Pour simplifier le diagramme de classes, toutes les vues sont accessibles dans la *JFrame Window.*

II.1 VUE ARBORESCENTE

Nous utilisons la classe *JTree* afin d'afficher l'ensemble de l'arborescence de notre scène. Une seule liste de cercles est définie dans le projet de base. La vue arborescente devra permettre d'effectuer les opérations génériques pour modifier les géométries, comme la sélection, la suppression ou permettre de grouper ou dissocier des objets.

Adrien Peytavie – Eric Duchene

II.2 APPLICATION

L'application MVC de départ permet d'ajouter des cercles de tailles fixes et de positions aléatoires dans la scène. Le bouton *Add* est implémenté en suivant l'architecture *MVC*. L'ajout va utiliser un *contrôleur* pour ajouter une forme au gestionnaire de formes *ShapeManager*.

Lors d'une modification du modèle, une notification est envoyée à la vue globale afin de la mettre à jour l'ensemble des composants.

III. TRAVAIL A REALISER

Avant de se lancer dans la création de formes complexes, vous devez modifier le projet pour ajouter le design pattern *Composite* qui permettra de gérer l'arborescence de différents objets en groupant par paquets plusieurs objets. Ensuite, vous devez ajouter le design pattern *Factory* qui permettra de créer des formes à la volée en fonction de la forme sélectionnée dans l'interface. Vous pourrez ajouter n'importe quel objet par la suite.

De nombreuses fonctionnalités sont à développer, vous pouvez les effectuer dans l'ordre souhaité.

III.1 DESIGN PATTERN COMPOSITE

L'application de base permet de créer uniquement des cercles placés aléatoirement.

Vous devez modifier le code afin de mettre en place le design pattern *Composite* :

- Développer la classe abstraite *Shape* qui permettra de gérer génériquement l'ensemble des formes.
- Mettre en place une classe *Group* pour regrouper un ensemble d'objets et ainsi de permettre la création d'une hiérarchie d'objets.
- Ajouter de nouvelles formes géométriques à afficher, par exemple : carré et rectangle. Vous ajouterez des formes complexes dans un deuxième temps.
- Adapter le code du *ShapeManager* afin de ne pas gérer un ensemble de cercles mais la racine de l'arborescence donc en définissant une variable de type *Shape*.
- Adapter les fonctions du *ShapeManager* afin de pouvoir faire afficher une hiérarchie d'objet. Vous pouvez utiliser la fonction *init()* afin de prédéfinir une hiérarchie. Attention le nœud racine doit être un *Shape Group* afin de pouvoir ajouter plusieurs objets sinon lors de l'ajout d'un deuxième objet il faut modifier la hiérarchie afin d'insérer les 2 objets dans un groupe.
- Valider dans un premier temps l'arborescence de votre structure avec la méthode *toString(int)* (pensez à ajouter un paramètre pour compter le nombre de tabulations à prendre en compte).
- Valider l'affichage de votre *Shape* dans l'interface avec l'affichage des cercles et l'affichage du *JTree*.

III.2 DESIGN PATTERN FACTORY

L'application ne contient qu'un seul bouton pour ajouter des formes à la scène. L'utilisateur doit sélectionner son choix dans les *jRadio* proposés afin d'ajouter le type d'objet souhaité à la scène. Afin de gérer facilement la création des objets, vous devez mettre en place le design pattern *Factory*.

- Mettre en place la classe *ShapeFactory* permettant de créer les formes sélectionnées par l'utilisateur dans l'interface. La fonctionnalité de création sera utilisée dans le contrôleur d'ajout de forme.
- Ajouter l'ensemble des formes créées et mettre à jour le code à chaque création de nouvelles formes.

III.3 Interactions

Les interactions actuelles sont limitées à l'ajout de formes dans l'arborescence. Voici la liste des fonctionnalités à mettre en place, pour chaque fonction vous devez utiliser le modèle MVC donc passer par un contrôleur pour modifier les données :

- **Informations :** Faire afficher dans le *jTextPane* toutes les informations utiles (Messages d'erreur, réalisation d'une opération, …)
- **Suppression :** Suppression d'un ou plusieurs nœuds sélectionnés dans la *jTree* dans le modèle MVC. Aide : possibilite de recuperer les index dans l'arborescence avec la fonction getSelectionRows().
- **Grouper** : Grouper un ensemble de nœuds qui ont été sélectionnés en utilisant le modèle MVC.
- **Dissocier** : Supprimer le groupe sélectionné en redistribuant les objets au niveau inférieur. Attention il n'est pas possible de dissocier la racine.
- **Selection d'un nœud** : Implémenter le double clic sur l'arborescence afin de sélectionner un nœud et d'afficher ses propriétés dans la zone prévue à cet effet. Il est necessaire d'ajouter dans le modele le nœud qui a ete selectionne.
- **Modification d'un nœud** : Rajouter l'ensemble des interactions/contrôleurs afin de modifier dynamiquement les propriétés d'un nœud (position, rayon, couleur).

IV. AMELIORATION DE L'APPLICATION

Maintenant que le cœur de l'application est opérationnel, vous pouvez compléter l'application de fonctionnalités et modèles complémentaires afin de pouvoir gérer une scène virtuelle. Vous êtes libre d'avancer sur les fonctionnalités qui vous semblent pertinentes.

- **Factory :** Ajouter des formes complexes à votre librairie d'objet comme des arbres, des maisons, ou voitures qui peuvent être une composition d'objets. Vous pouvez également avoir des boutons permettant de créer directement des groupes d'objets comme un groupe d'arbre pour définir une forêt automatiquement.
- **Decoration :** Ajouter des boutons à l'interface et des attributs aux formes afin de modifier les apparences des objets comme les couleurs d'affichage d'une forme avec son contour en utilisant le QPen ou son remplissage (QBrush), vous pouvez également permettre la modification de la taille des objets, etc.