

INF2990

Projet de logiciel graphique interactif

Tests logiciels

Version 16.3

**Fait par : Équipe 11**

10 Novembre 2016

# Exemple

|  |  |  |  |
| --- | --- | --- | --- |
| **Suite de cas de test # 1** | | | |
| **Classe testée** | NoeudComposite (NoeudCompositeTest) | **Branche** | Master |
|  | | | |
| **Justification** | | | |
| Comme une grande partie de notre architecture se base sur les nœuds composites et ses méthodes, donc l’idée de tester cette classe peut être pertinent.  On se base dans cette classe de test sur la classe « nœudTable » puisque la table est l’objet qui va contenir le plus d’objets dans l’application. | | | |

|  |  |
| --- | --- |
| **Cas de test # 1** | |
| **Méthode testée** | NoeudComposite::vider (testVider()) |
|  | |
| **Justification** | |
| Il est primordial d’initialiser la scène de jeu à l’ouverture de notre application, à travers la méthode « initialiser » qui fait appel à son tour à vider () du nœud Composite.  La méthode « vider » permet donc d’effacer tous les nœuds fils d’un nœud composite (Nœud Table dans notre cas). | |
|  | |
| **Explication du cas de test** | |
| Dans ce test, on ajoute un nœud enfant (un bonus accélérateur) au nœudTable, on assure que le nombre d’enfant de la table est correct (1 dans notre cas), après on appelle la méthode vider () et revérifie encore le nombre des enfants (doit être 0) | |
|  | |
| **Cas de test # 2** | |
| **Méthode testée** | NoeudComposite::ajouter (testAjout()) |
|  | |
| **Justification** | |
| L’ajout des objets est un élément essentiel pour le fonctionnement de l’application, puisque la plupart des fonctionnalités manipulent des objets dans la scène. Donc on a jugé qu’il sera une fonctionnalité pertinente à tester. | |
| **Explication du cas de test** | |
| On prend le nœud Table comme père, on vérifie d’abord qu’il n’a pas d’enfants, on crée ensuite deux nœuds Muret et Bonus et on fait appel à la méthode « ajouter ». Finalement on vérifie que le nœud Table a le bon nombre de descendance. | |
|  | |

|  |  |
| --- | --- |
| **Cas de test # 3** | |
| **Méthode testée** | NoeudComposite::effacerSelection (testSuppression()) |
|  | |
| **Justification** | |
| Puisqu’on a testé l’ajout des objets, il est intéressant de vérifier si la suppression des objets est bien fonctionnelle, on a pris on compte aussi le fait qu’il n’est pas possible de supprimer un seul portail. | |
| **Explication du cas de test** | |
| On commence par ajouter un noeudBonus et une paire des portails, on vérifie que le nombre des enfants du Table est exacte, puis on sélectionne un seul portail et on appelle la méthode effacerSelection.  On vérifier finalement, que la fonction n’a enlever que le portail sélectionné et son frère. | |
|  | |
| **Cas de test # 4** | |
| **Méthode testée** | NoeudComposite::Chercher (testTrouverObjet();) |
|  | |
| **Justification** | |
| Vu le nombre illimité de nœuds fils qu’on peut avoir, le test de la méthode Chercher (2 surcharges : indice et type) sera nécessaire.  Cette méthode peut être appeler en passant comme paramètres l’indice ou le type du nœud cherché et retourne le nœud en question s’il est trouvé. | |
|  | |
| **Explication du cas de test** | |
| On commence par ajouter trois nœuds au nœud Table, on vérifie que le nombre d’enfants est correct, puis on appelle la fonction chercher avec le type du nœud comme paramètre (bonus par exemple) et l’élément retournée par cette fonction correspond à l’élément crée au début.  De même avec Chercher (avec indice comme paramètre).  On a testé aussi le cas où l’indice ne correspond à aucun élément (out of range) | |
|  | |

|  |  |
| --- | --- |
| **Cas de test # 5** | |
| **Méthode testée** | NoeudComposite::selectionnerTout(testSelectionMultiple()) |
|  | |
| **Justification** | |
| Avec un grand nombre des enfants, il sera plus efficace d’appeler une méthode qui fait sélectionner tous les enfants du nœud composite au lieu de parcourir l’arbre et faire la sélection de chaque nœud. | |
|  | |
| **Explication du cas de test** | |
| On insère deux éléments au nœud Table, on appelle la fonction selectionnerTout sur le nœud père puis on vérifie que les trois nœuds (table + 2 fils) sont sélectionnés. | |
|  | |

|  |  |
| --- | --- |
| **Cas de test # 6** | |
| **Méthode testée** | NoeudComposite::setScale (testScale()) |
|  | |
| **Justification** | |
| Une des fonctionnalités du mode édition est la mise en échelle des objets, cette fonctionnalité permet d’agrandir ou diminuer un objet. | |
|  | |
| **Explication du cas de test** | |
| On crée un objet Bonus qui a initialement le scale (1,1,1), on a agrandi et on vérifie la valeur de Scale (on essaie aussi diminuer l’objet) | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Suite de cas de test # 2** | | | |
| **Classe testée** | PointsControle(PointControleTest) | **Branche** | Master |
|  | | | |
| **Justification** | | | |
| Cette classe permet de modifier la position des points de contrôle, il permet de sélectionner un point de contrôle et modifier sa position, puis, on modifier la position du point de contrôle symétrique. | | | |

|  |  |
| --- | --- |
| **Cas de test # 1** | |
| **Méthode testée** | NoeudTable::getPc(testPositionPoint()) |
|  | |
| **Justification** | |
| La position des points de contrôle se change en paire, c’est-à-dire, on chaque fois qu’on modifie la position d’un nœud, son nœud opposé doit changer sa position en conséquence. | |
|  | |
| **Explication du cas de test** | |
| On crée un nœud table avec ses 8 points de contrôle, puis on prend la position du point de contrôle (haut-gauche) et et on vérifie que le point de contrôle (haut-droite) a les mêmes coordonnées en y et z, et x inversé. | |
|  | |
| **Cas de test # 2** | |
| **Méthode testée** |  |
|  | |
| **Justification** | |
|  | |
|  | |
| **Explication du cas de test** | |
|  | |
|  | |
| **Cas de test # 3** | |
| **Méthode testée** |  |
|  | |
| **Justification** | |
|  | |
|  | |
| **Explication du cas de test** | |
|  | |
|  | |
| **Cas de test # 4** | |
| **Méthode testée** |  |
|  | |
| **Justification** | |
|  | |
|  | |
| **Explication du cas de test** | |
|  | |
|  | |