

FloorTile objects implemented with the Composite Structural Design Pattern

### Nobody (Team 3: Caleb Corlett, Ryan Nodarse, Patrick Storer, Ethan Wyman)

The Composite Design Pattern is a Structural Design Pattern that is handy when you need to work with an instance of a class that might contain other instances of the same class, which might have more instances of the same class attached to each, creating a tree-like structure that grows wider and more complex the further you get from the original starting instance. Composite Structures make use of a Composite class and a Leaf class inheriting from a Component class to achieve this tree-like structure, with a Leaf object holding a reference to a Composite object which will store references to any other attached Leaf objects within it. Another important property of the Composite Structural Design Pattern is that the Composite object contains unique versions of the Leaf class’ methods which, by themselves, do not perform any actual behavior, but instead will recursively call every child Leaf’s method of the same name to perform the actual behavior, until the edge of the object’s tree is reached and no more Leafs exist to traverse.

In Betrayal Online, there will be many square floor tiles placed around a grid during a game, with some special items and rooms having unique effects that apply to all rooms a set number of spaces away. For these situations, the recursive tree-like nature of the Composite Design Pattern makes these particular operations incredibly simple to implement.

The Betrayal Online implementation of the Composite Design Pattern involves a general FloorTileComponent class with both a NeighborComposite class and a FloorTileLeaf class inheriting from it to represent the Component, Composite and Leaf classes of the Composite Design Pattern.

The general FloorTileComponent class contains a public method room\_logic() which is a simple method that acts as a default placeholder method if the floor tile does not contain any special behavior. For floor tiles with their own behavior, this default room\_logic() method would be overwritten with the proper behavior. The FloorTileComponent class also contains a public method execute(func: function, \*\*kwargs) which executes the passed function from within an instance of FloorTileComponent.

Inheriting from the FloorTileComponent class, the FloorTileLeaf class contains a private String identifier, as well as an attribute called neighbors which holds a reference to an instance of the sibling class, NeighborComposite, which will hold the references to neighboring FloorTileComponent objects on the game board. If the specific FloorTileLeaf instance has unique behavior attached to it, the default room\_logic() method, inherited from FloorTileComponent, would be overwritten with a new room\_logic() method containing the special room logic.

Also inheriting from the FloorTIleComponent class is the NeighborComposite class which contains a private attribute called neighbors which contains a list of the FloorTileComponent objects neighboring a FloorTileLeaf object, giving the structure its tree-like shape. Located in the NeighborComposite class, is an add(a: FloorTileComponent) and remove(r: FloorTileComponent) method which adds and removes FloorTileComponent instances to the neighbors attribute and returning them for any use or processing, as well as a get\_neighbors(recursion\_cap: int = 0) method which returns a Python set of neighboring FloorTileComponents up to a certain number of spaces away, with a recursion\_cap parameter used to limit the number of spaces from the original FloorTileLeaf instance to include, defaulting to 0 to only return the directly adjacent FloorTileComponent neighbors. The NeighborComposite class also contains a public room\_logic(recursion\_cap: int = -1), overwriting the inherited room\_logic() method, to instead recursively call every attached FloorTileComponent object’s room\_logic() method, stopping when recursion reaches the attached recursion\_cap, or until all FloorTileComponent objects on a floor have activated their room\_logic() if the default value for recursion\_cap is left at its default value of -1. And finally, the last method in the NeighborComposite class is an overwritten version of the FloorTileComponent class’ execute(func: function, \*\*kwargs), overwriting it with execute(func: function, recursion\_cap:int = -1, \*\*kwargs) which will now recursively call every attached FloorTileComponent object’s execute(func: function, \*\*kwargs) method and pass along the function and arguments to be executed within every single attached FloorTileComponent object, stopping when the recursion\_cap parameter has been reached or until all FloorTIleComponent objects on a floor have had their execute(func: function, \*\*kwargs) methods called if recursion\_cap was left at its default value of -1.

**Design Class Diagram:**

