Lab 7
Scene Graph Development

Description: In this lab, you will complete and test various method of the VisibleObject class to support the creation of a scene graph data structure. Your starting point will be what we having been working on during the last few class sessions.

Getting Started

1. Download and extract the zip archive that contains the `CSE387Lab7` solution.

2. Use MSVC 2013 to open the extracted solution.

Initialization Traversal

In class we implemented a rendering traversal of our scene graph by adding a `draw` method to the `VisibleObject` class. This super class method included a `for` loop that called the draw method on each of the children in the `children` vector. We then added a call to this method at the end of `draw` methods of all the `VisibleObject` sub-classes.

3. Implement an initialization traversal for the scene graph by adding a concrete initialize method to the `VisibleObject` class. The method should call the initialize method of all the children. Call this method at the end of the initialize methods of all the VisibleObject sub-classes. Remove all the individual explicit calls to initialize in the MyScene initialize method and replace them with a single call to the VisibleObject initialize method.

Update Traversal

4. Change the declaration of the `update` method in the `VisibleObject` class to the following:

```
virtual bool update(float deltaTime);
```

5. Implement an update traversal for the scene graph by adding a concrete `update` method to the `VisibleObject` class. In the method, call the `update` method all the children. If the returned Boolean value of the `update` method is `false`, then that `child` should be removed from the children vector and deleted. Use the `erase` method of the vector class to remove the `child`. Call delete to deallocate the memory associated with the `child`. You might want to use a `while` loop when going through the `children` vector. Realize that all the children will shift when one is removed. You want to make sure that no children do not get updated.

```
delete children[i];
children.erase(children.begin() + i);
```

6. Add update methods to the `Cube`, `Sphere` and `AssimpModel` classes that call the `VisibleObject` update method. Put print statements before these calls to verify that each is being called during the repeated traversals scene graph. Don't forget to call the `VisibleObject update` method at the end of the myScene update method.

   For now, we will let the update method of the `MyScene` class continue to do all the actual updating of the objects in the scene.

detachFromParent Method

7. Complete the `detachFromParent` method of the `VisibleObject` class. It should search the `children` vector of the `parent` of the object thru which the method was called and remove the "this" object from the vector. The `parent` of the object thru which the method was called should be set to `null`. The method should return `null` if "this" object could not be removed from its parent or the address of the removed object if the detach was successful. You can use a `for` loop to search through the parent's children vector and break out once you have found and removed the appropriate `child`.

8. Test this method by having the 'm' key toggle attaching and detaching the moon to the earth.

reparent Method

9. Complete the `reparent` method. It should attach the `newChild` argument to the object thru which the method was called. The position and orientation of the child relative to World coordinates should not be changed when this occurs. `newChild` needs to be removed from the `children` vector of its old `parent` and added to the `children` vector of its new parent. You will have to do some transformation arithmetic to figure out what the new local transformation should be for `newChild`.

   Test by adding an additional planet orbiting the sun when the 'l' and 'w' keys are pressed. When the 'l' is pressed, reparent the planet to the sun only once. This should result in the planet starting to more with the sun. When the 'w' key is pressed, reparent the planet to the sun every frame. This should result in the planet staying in same position relative to World coordinates as the sun orbits the cube.

detachAndDeleteChild Method

10. Write a method that will detach and delete a specified child from a VisibleObject object. Have it return true if the child was found and deleted and false otherwise.

```
bool detachAndDeleteChild (VisibleObject* childToBeDeleted);
```

Comments

11. Add Javadoc style header comments to `addChild`, `getParentWorldTransform`, `getLocalTransformation`, `getWorldTransfomation`, `detachFromParent`, and `reparent` methods of the `VisibleObject` class. Put the comments in `VisibleObject.h`. This is to

reinforce in your mine what each of the methods accomplishes. Feel free to copy and paste from this document for the methods that were implemented as part of this lab.

Turn it in

12. Copy the folder containing your solution to the desktop.

13. Change the name of the folder to `CSE387LabSeven` followed by your unique identifier. For instance “CSE387LabFSevenBachmaer.”

14. Open the solution. Make sure it still runs.

15. Clean the solution by selecting `Build->Clean Solution`. If there is an .sdf file in the solution directory, delete it.

16. Zip up the solution folder using the standard windows compression tool. (No 7zips, rars, etc.)

17. Submit your zip archive of the solution through canvas.