Adapter

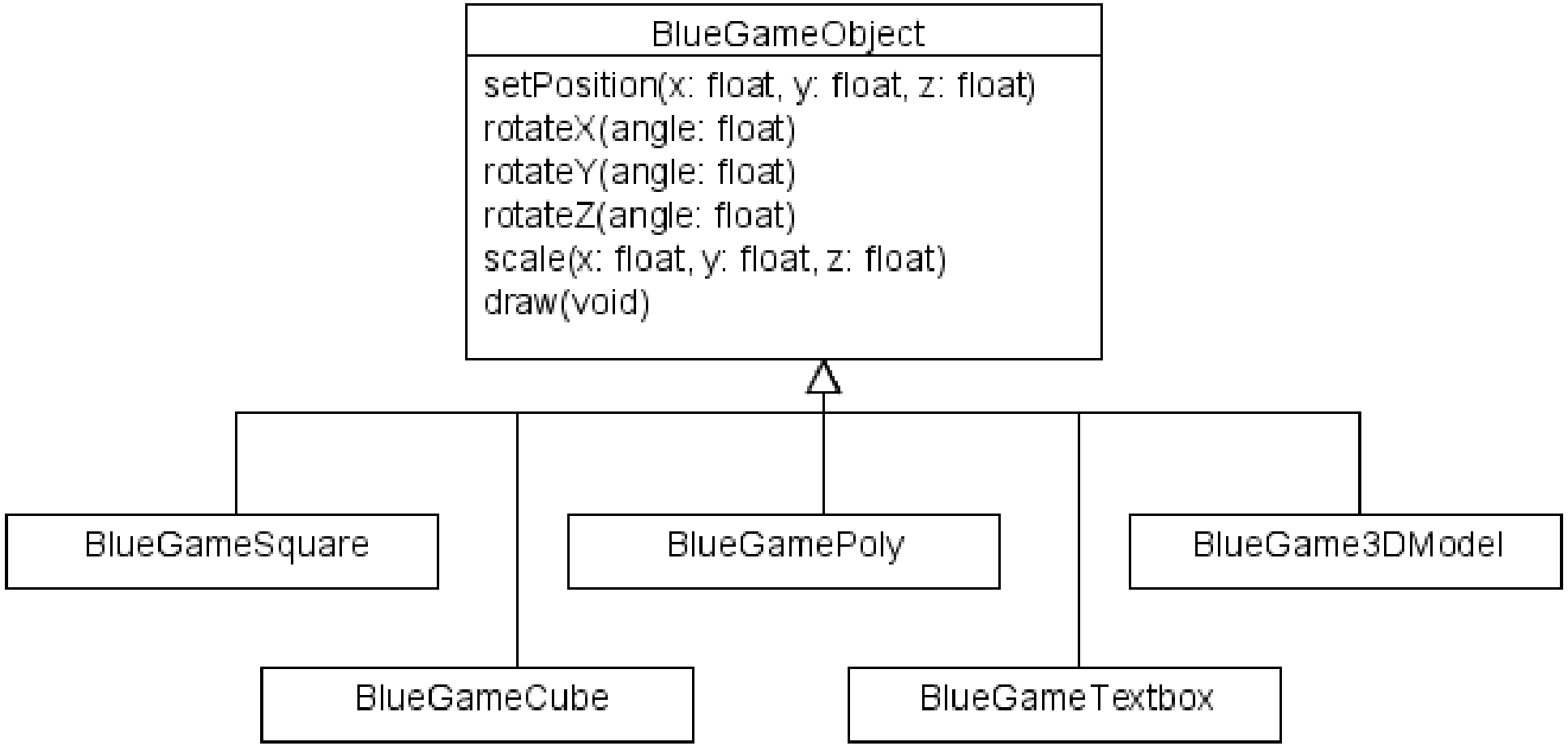
Designing a Simple Game Engine

We want to design a simple 3D game engine.

Lets call it the “Blue Game Engine”.

In this simple engine, every object displayed on the screen is an instance of a BlueGameObject.

BlueGameObject



Implementation Concerns

So far, implementing most of the BlueGameObject is fairly straightforward using any 3D library

Draw geometric shapes in 3D is easy.

But what about the TextBox?

GUI elements (buttons, labels, forms, ...) are inherently difficult to develop in 2D/3D game libraries.

Most game companies buy specialized libraries for this.

Introducing GreenGUI

Let's now introduce a new library GreenGUI, which specializes in GUIs for 3D engines.

Like all GUI systems, GreenGUI does have a class for text boxes.

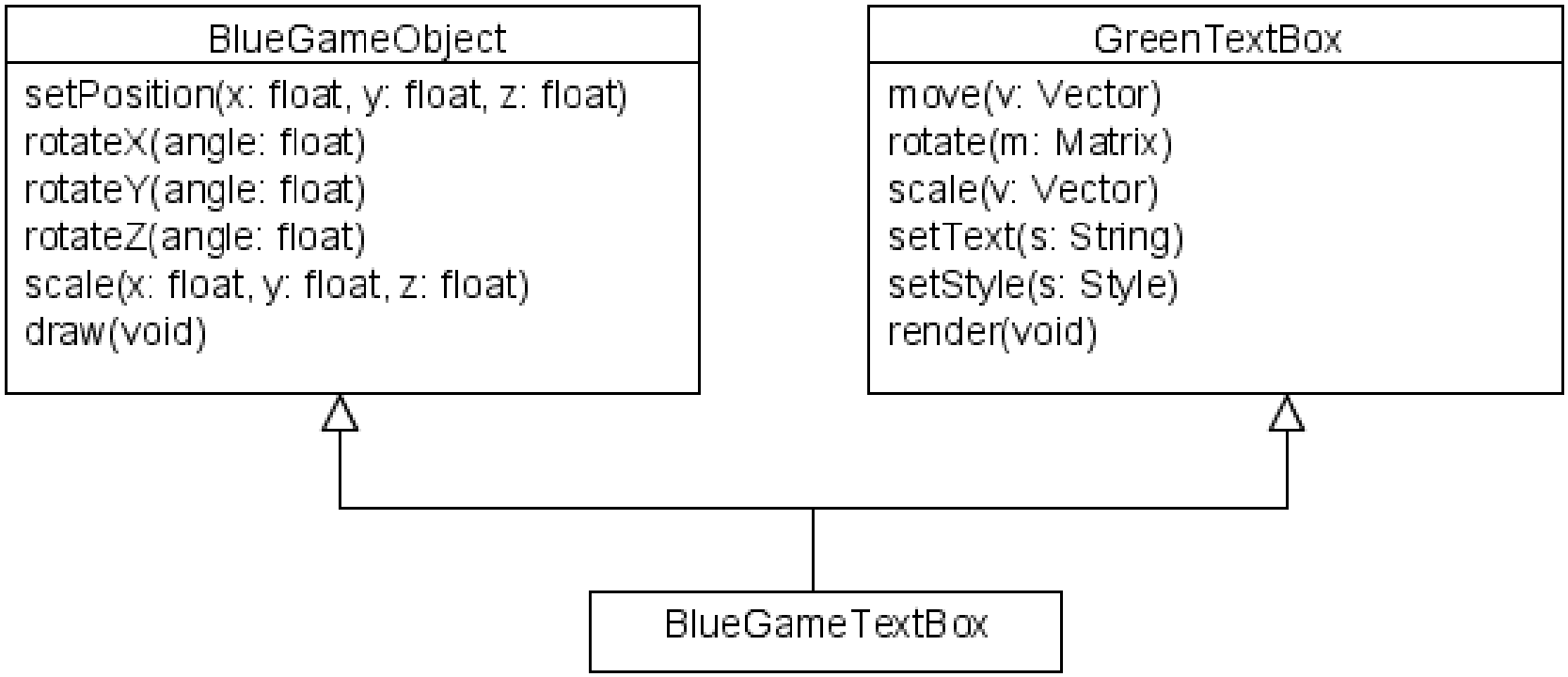
However, GreenGUI obviously has a different API.

GreenTextBox vs. BGO incompatible?

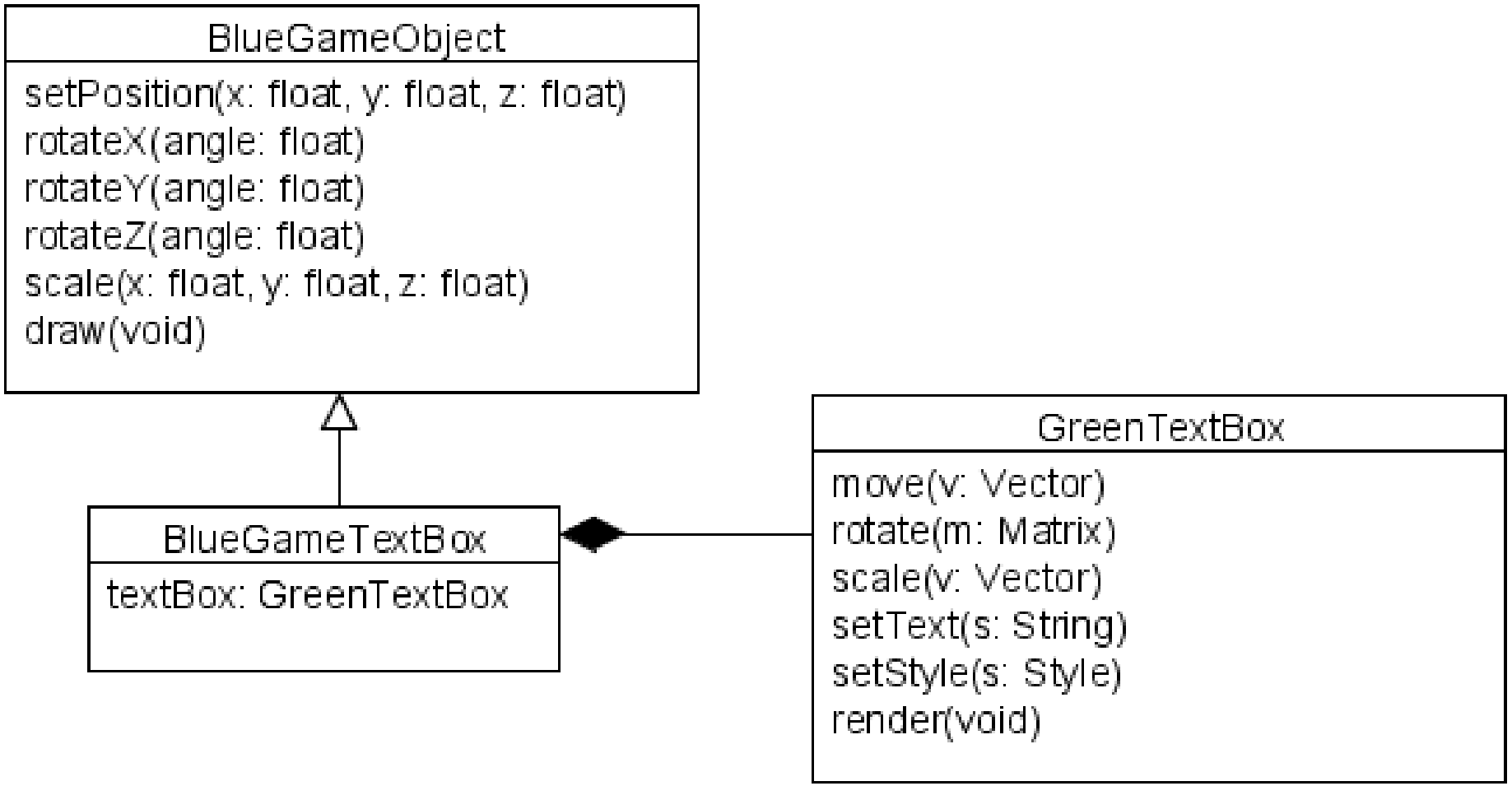
BlueGameObject
setPosition(x: float, y: float, z: float) rotateX(angle: float) rotateY(angle: float) rotateZ(angle: float) scale(x: float, y: float, z: float) draw(void)

GreenTextBox
move(v: Vector) rotate(m: Matrix) scale(v: Vector) setText(s: String) setStyle(s: Style) render(void)

Inheritance



Composition



Adapter

Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

Aka: Wrapper

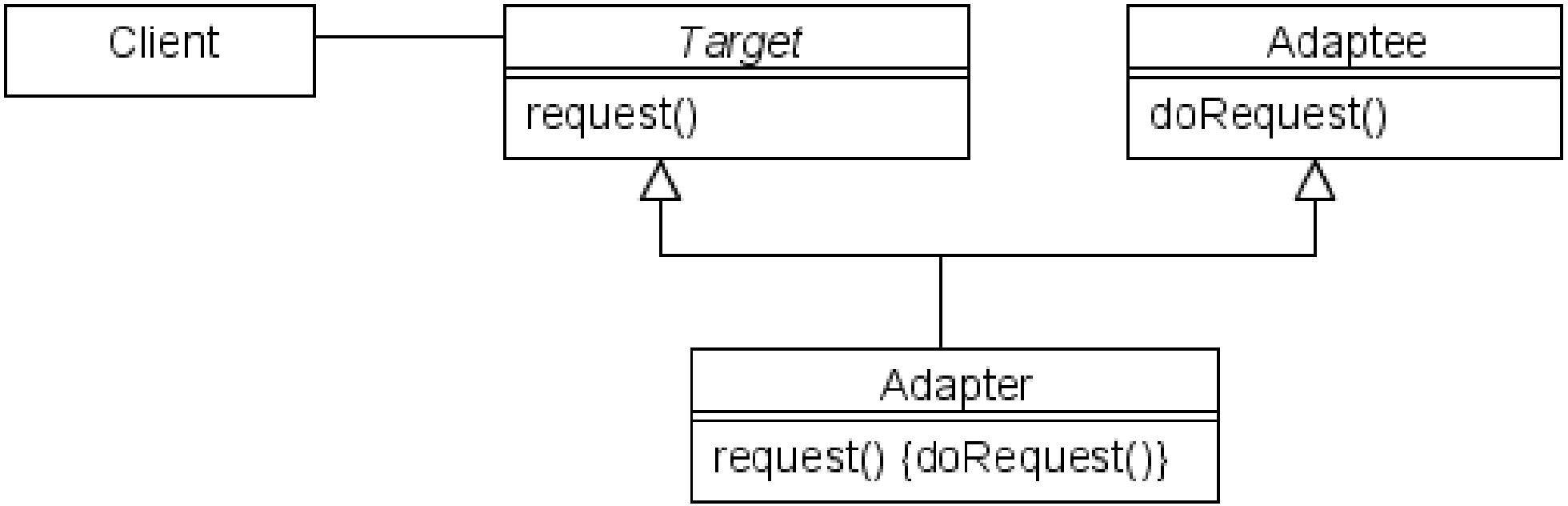
Motivation

Sometimes a toolkit or class library can not be used because its interface is incompatible with the interface required by an application.

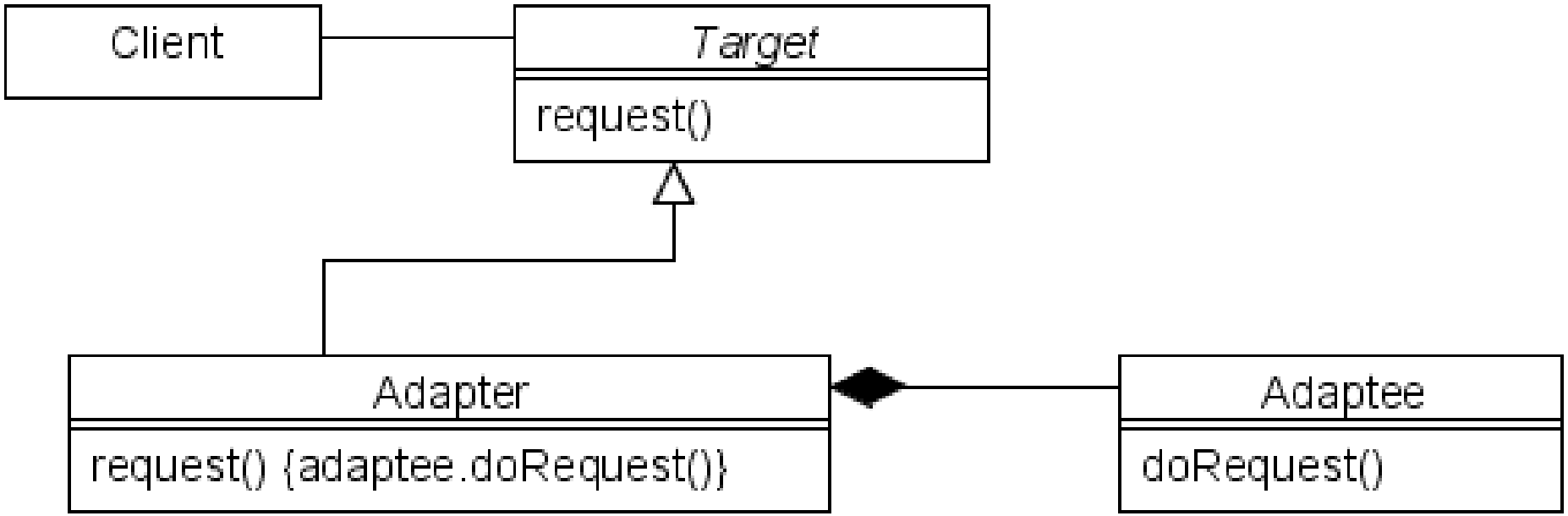
We can not change the library interface, since we may not have its source code.

Even if we did have the source code, we probably should not change the library for each domain-specific application.

Class Adapter



Object Adapter



When to use?

Use the Adapter pattern when

You want to re-use an existing class, and its interface does not match the one you need

You want to create a reusable class that cooperates with unrelated classes with incompatible interfaces

Implementation Issues

How much adapting should be done?

Simple interface conversion that just changes operation names and order of arguments

Totally different set of operations

Does the adapter provide two-way transparency?

A two-way adapter supports both the Target and the Adaptee interface. It allows an adapted object to appear as an Adaptee object or a Target object