Comp-304 : Adapter
Lecture 23

Alexandre Denault
Computer Science
McGill University
Fall 2007

Transactions

- **A**tomicity : Either all the tasks in the transactions are done, or none of them are.
- **C**onsistency : You application will be at a legal state at the beginning and the end of the transaction.
- **I**solation : The tasks done in the transaction will be isolated from other operations.
- **D**urability : Once the transaction is completed, it will persist and cannot be undone.

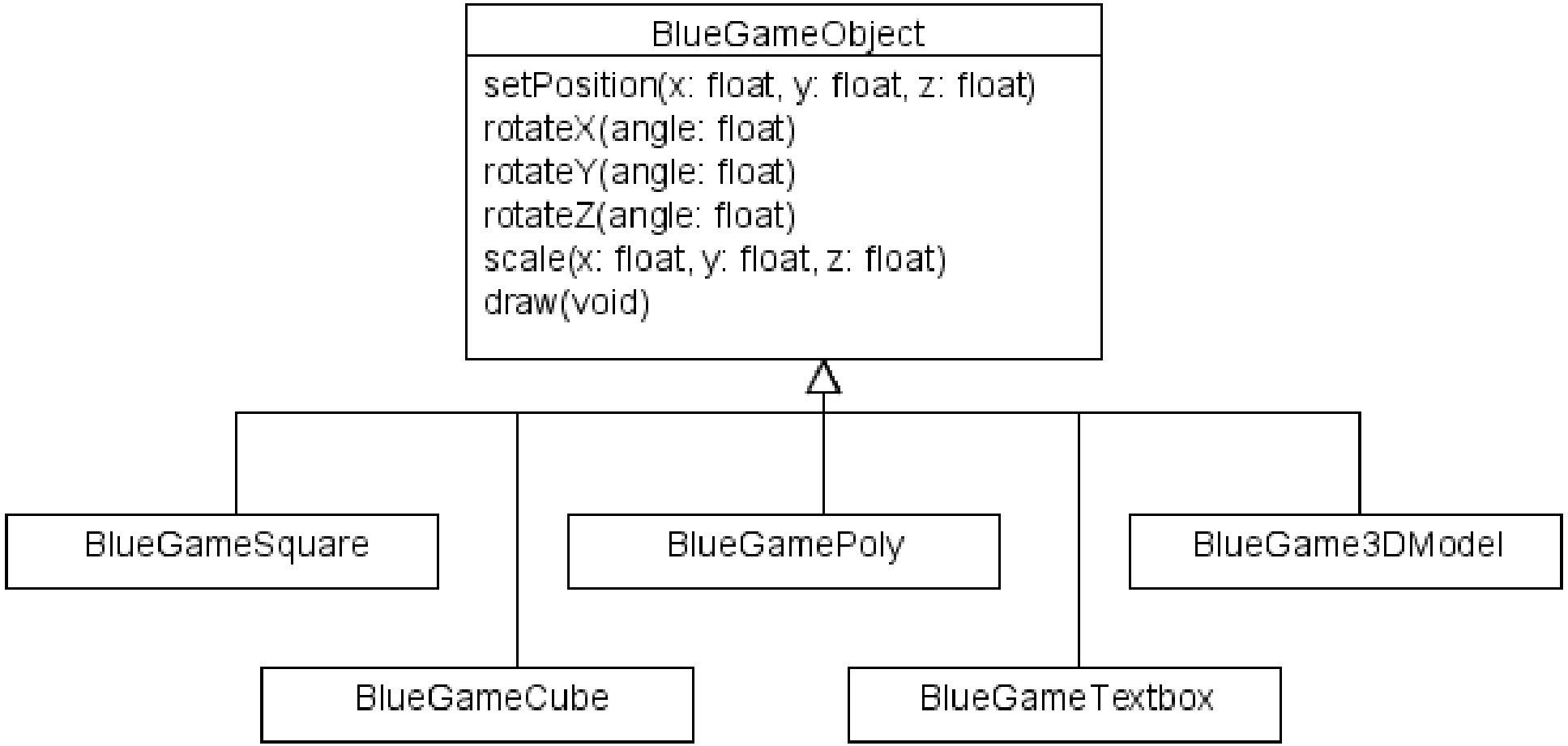
Command

- We finished the command pattern pretty fast last class.
- Lets take a minute to review it.

Designing a Simple Game Engine

- 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 the most of the BlueGameObject is fairly straight forward using any 3d library
  - Draw geometric shapes in 3d is easy.
- But what about the TextBox?
  - GUI elements are inherently difficult to develop in 2D/3D game libraries.
  - Most game companies will buy specialized tools for this.

Introducing GreenGUI

- Now, lets introduce a new library, GreenGUI, which specializes in developing GUI systems for 3D engines.
- Like all GUI system, GreenGUI does have a class to does text boxes.
- However, GreenGUI obviously has a different API.

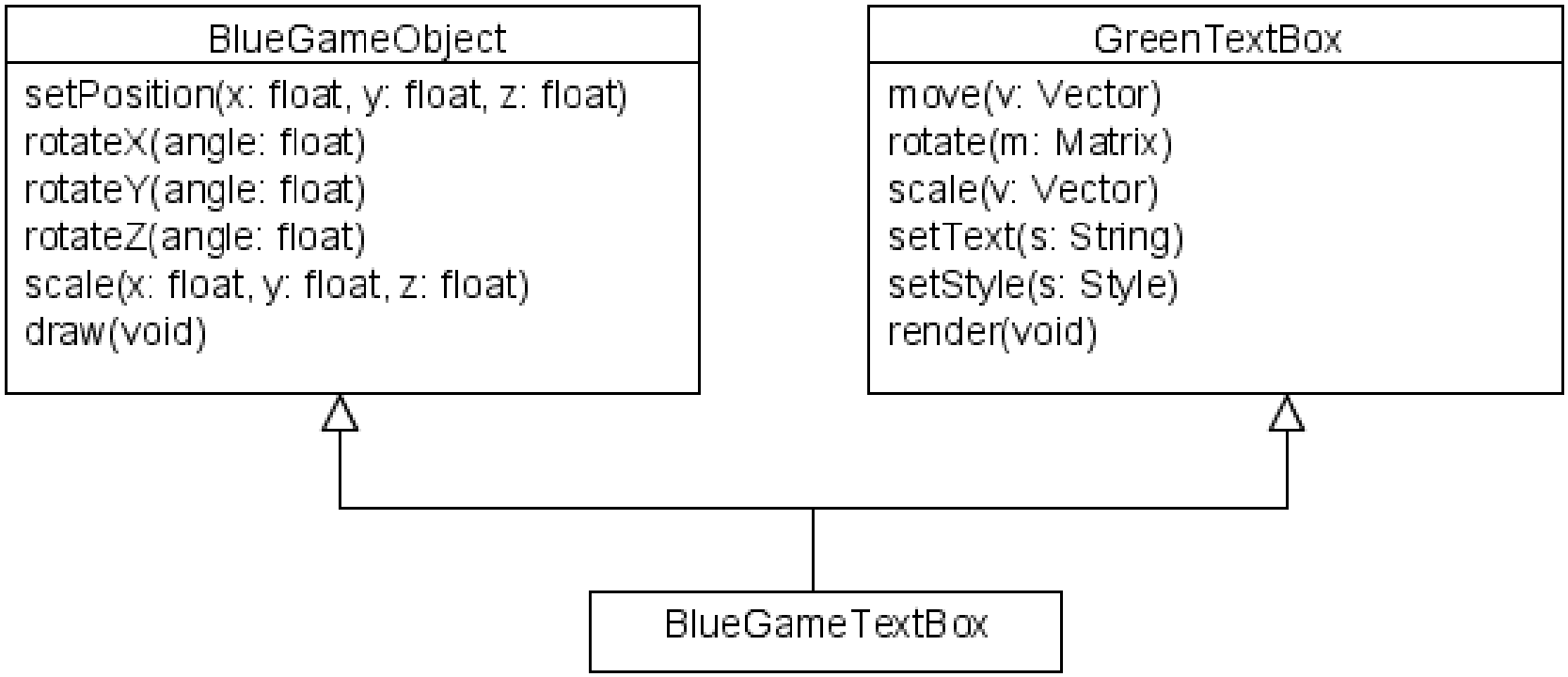
GreenTextBox as a BGO

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)

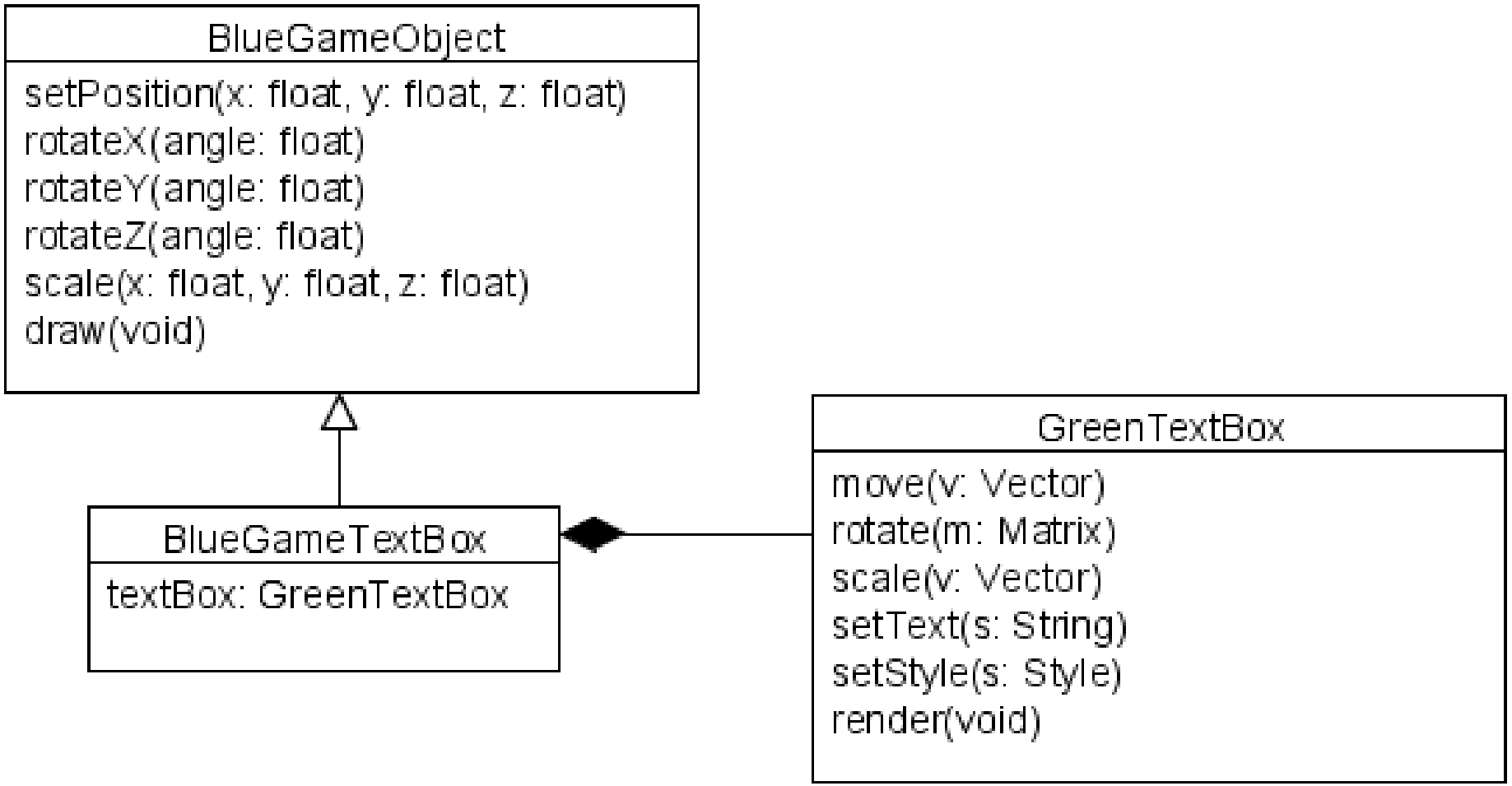
GreenTextBox
move(v: Vector) rotate(m: Matrix) scale(v: Vector) setText(s: String) setStyle(s: Style) render(void)

What to do?

Inheritance



Composition



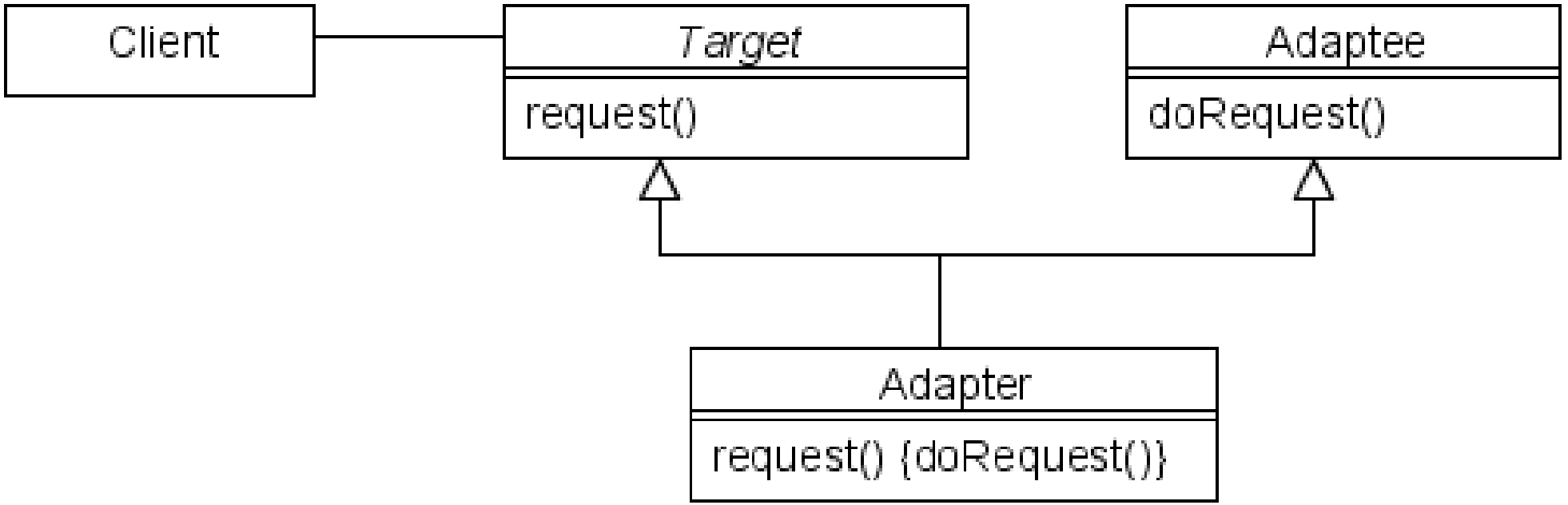
Adapter

- 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

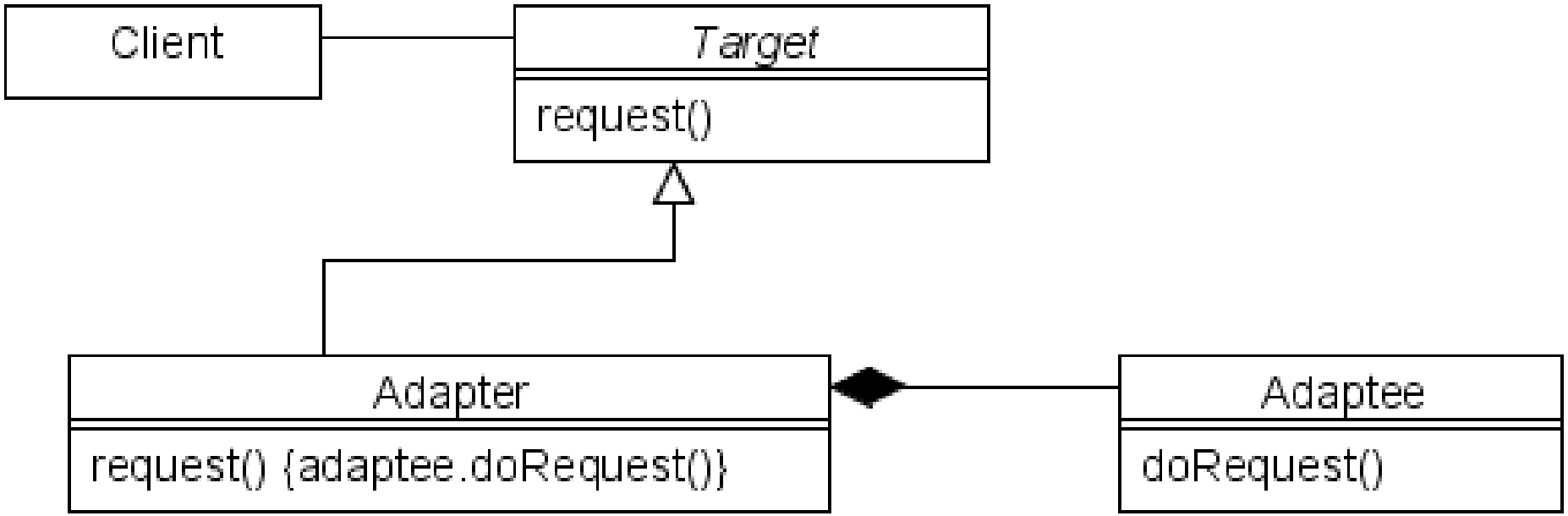
Motivation

- 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 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 (Adapter) to appear as an Adaptee object or a Target object

FengGUI

- Now, for a more formal example of Adapter pattern in action.
- FengGUI is a Java OpenGL library for drawing GUI's
- It is compatible with all the major tool set:
  - JOGL
  - LWJGL
  - Xith 3D
  - JMonkey

Input Handling in FengGUI

- Since FenGUI is a GUI tool set, it needs to know about keyboard and mouse input.
- The main class in FengGUI is Display.

Display
+Display(binding: Binding) +fireKeyPressedEvent(keyValue: char, keyClass: Key): boolean +fireKeyReleasedEvent(keyValue: char, keyClass: Key): boolean +fireMouseDraggedEvent(mouseX: int, mouseY: int, mouseButton: MouseButton): boolean +fireMouseMovedEvent(displayX: int, displayY displayY): boolean +fireMousePressedEvent(mouseX: int, mouseY: int, mouseButton: MouseButton, clickCount: int): boolean +fireMouseReleasedEvent(mouseX: int, mouseY: int, mouseButton: MouseButton, clickCount: int): boolean +fireMouseWheel(mouseX: int, mouseY: int, up: boolean): boolean

JOGL

- The JOGL project hosts the development version of the Java™ Binding for the OpenGL® API (JSR-231).
- It is designed to provide hardware-supported 3D graphics to applications written in Java.
- JOGL provides full access to the APIs in the OpenGL 2.0 specification as well as nearly all vendor extensions, and “integrates” with the AWT and Swing widget sets.

Dealing with JOGL

- JOGL functions over AWT, thus uses the regular MouseListener for mouse input.

Adapter

```
public class FengMouseListener implements MouseListener {
    Display display;

    mousePressed(e: MouseEvent) {
        this.display.fireMousePressedEvent(e.getX(), e.getY(),
            e.getMouseButton(), e.getClickCount());
    }
    mouseReleased(e: MouseEvent) {
        this.display.fireMouseReleasedEvent(e.getX(), e.getY(),
            e.getMouseButton(), e.getClickCount());
    }
    mouseClicked(e: MouseEvent) {}
    mouseEntered(e: MouseEvent) {}
    mouseExited(e: MouseEvent) {}

}
```