УДК 681.3

ШАБЛОН ПРОЕКТИРОВАНИЯ ГРАФИЧЕСКОГО РЕДАКТОРА

А. В. Шалиткин

Воронежский государственный университет

Поступила в редакцию 01.03.2010 г.

Аннотация. Целью данной статьи является описания шаблона проектирования, который можно будет использовать для создания легко расширяемого, модульного графического редактора в объектно-ориентированных средах.
Ключевые слова: шаблон, графический редактор

Abstract. The aim of this article is to describe the design pattern of a graphical editor. The graphical system must be extensible and must have modular structure.
Keywords: pattern, graphical editor.

ВВЕДЕНИЕ

Компьютерная графика используется повсеместно в различного рода системах, начиная от простых редакторов графики и заканчивая средами разработки и, хотя, уже реализованных приложений довольно много, эта область продолжает бурно развиваться. Каждый раз при разработке такого приложения приходится разрабатывать архитектуру, применять различные приемы для достижения хорошего уровня расширяемости, упрощения поддержки.

Применение шаблонов проектирование может облегчить эту задачу.

Шаблон проектирования, Паттерн (англ. design pattern) — это многократно применяемая архитектурная конструкция, предоставляющая решение общей проблемы проектирования в рамках конкретного контекста и описывающая значимость этого решения. Паттерн не является законченным образцом проекта, который может быть прямо преобразован в код [1].

Описать шаблон означает описать предметную область, то есть ситуацию, проблему, соответствующие термины и рекомендации по созданию решения. Рекомендации включают в себя UML диаграммы и комментарии по реализации.

1. ОПРЕДЕЛЕНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

С позиции шаблонного проектирования графический редактор рассматривается в данной статье впервые, и, учитывая возможность применения шаблона во многих ситуациях, функционал описан в общем виде.

Независимо от приложения, будь это расширение текстового редактора, встроенный редактор UML или комплекс для трехмерного моделирования, графический редактор должен иметь следующие элементы:

- Модель данных.
- Средства для отображения модели.
- Средства для изменения модели.
- Средства для реакции на пользовательские события.

Кроме того, должна существовать возможность отображать модель в нескольких режимах или на нескольких экранах одновременно и обновлять экраны сразу же при изменении модели.

Исходя из вышесказанного и [2], можно определить графический редактор следующим образом.

Определение 1: Графический редактор представляет собой модель и операции по изменению модели. Кроме того, существует один или несколько экранов, которые позволяют отобразить модель в том или ином виде. При применении какой-либо операции и обновлении модели необходимо обновлять экраны.

Шаблон должен быть спроектирован таким образом, чтобы в наибольшей степени достигались loosely-coupling и cohesion. Мы не рассматриваем granularity, так как в данном случае для этого необходимо более детальное проектирование. Термины loosely-coupling, cohesion и granularity будут описаны ниже.

2. ОПИСАНИЕ ШАБЛОНА

Основные характеристиками приложений, на которые влияет архитектура системы, являются: Расширяемость, масштабируемость, затраты на реализацию и поддержку. Отказоустойчивость или производительность относятся скорее к стадии реализации, чем к проектированию.

С точки зрения проектирования влиять на вышеизложенные характеристики можно, выполняя набор некоторых принципов [5]. Рассмотрим их ниже.

• Достижения loosely-coupling. То есть уменьшение количества связей между подсистемами, в частности классами.

• Достижение высокого уровня cohesion. Каждый класс должен выполнять вполне определенные функции, непосредственно связанные с соответствующей сущностью предметной области. Например, если класс инкапсулирует механизм обработки пользовательских событий, то он не должен нести в себе функциональность рисования или построения графических фигур.

• Выбор правильного уровня granularity. Granulatity определяет детализацию интерфейса, то есть сколько методов класса необходимо вызвать для выполнения одной операции предметной области.

В самом общем виде графический редактор имеет модель данных и графический интерфейс, а поэтому к нему применима модель MVC2.

Даная модель представляет общее решение для систем, имеющих графический интерфейс, разделяя уровень модели данных и уровень представления посредствам контроллера. К преимуществам такой модели можно отнести:

• Loosely-coupling между моделью и представлением, а следовательно, удобство разработки и тестирования, расширяемость, модульность, возможность быстрой смены реализации одного из уровней или использование нескольких реализаций одновременно.

• Cohesion. Исходя из общего представления системы, три элемента модели четко сфокусированы на бизнес логике, интерфейсе и координации между ними.

Схема функционирования такой модели представлена на рис. 1.

Исходя из определения 1 и MVC2 модели, общая схема функционирования графического редактора может быть представлена следующим образом (рис. 2).

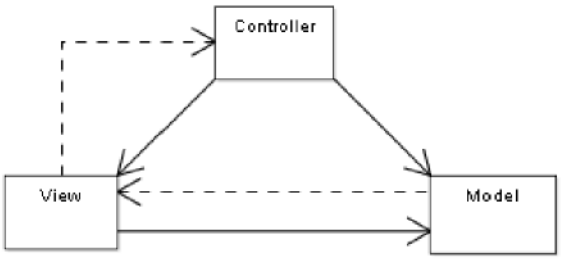

Рис. 1.

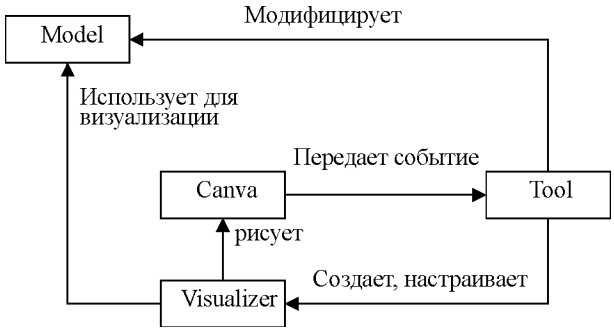

Рис. 2.

MVC2 модели соответствует Model. MVC2 контроллеру — Tool. MVC2 View — Visualizer и Canva

Model — внутренне модель данных. Например, одна из ER моделей, описанных в [4].

Canva — контейнер для графического представления. В случае трехмерного редактора, будем иметь 4 canvas — для каждой из проекций и для перспективы. Canva передает пользовательский события в Tool. Canva не инкапсулирует никакой бизнес логики или логики конкретного режима отображения, что позволяет использовать одну и ту же Canva в комбинации с различными Visualizer.

Tool — текущий инструмент, который предназначен для изменения модели. Tool реагирует на события от Canva, изменяет модель и перерисовывает, в случае необходимости, Cannva. Вся бизнес-логика пользовательских операций (создание, изменение, удаление графических объектов, оверлейные операции над ними) должна быть инкапсулирована в Tool.

Vizualizer — визуализатор модели. Отображает модель на Canva. Каждый Tool может использовать как свои собственные визуализаторы, так и общие, отображающие, например, координатные оси. В общем, в системе существует упорядоченный список или несколько списков (для различных Canva) визуализаторов.

Разделение подсистемы отображения на Canva и Vizualizer позволяет достичь более высокого уровня loosely-coupling и, как следствие, легче создать несколько экранов, требуемых в определении 1.

3. РЕАЛИЗАЦИЯ

Ниже более детально описано взаимодействие элементов общего шаблона. Введение классов менеджеров и контейнеров необходимо для достижения большего уровня loosely-coupling.

3.1. CANVA — VIZUALIZER

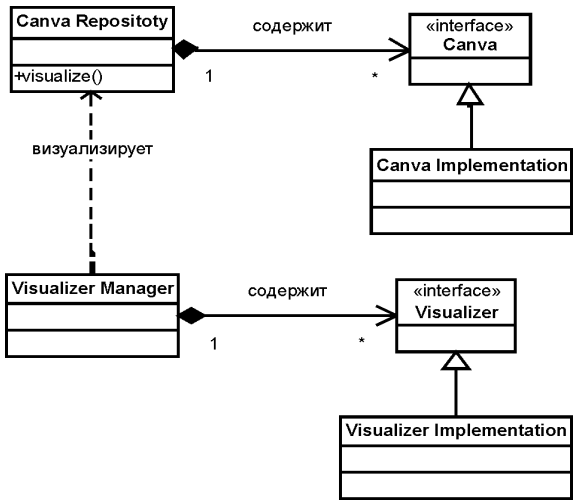

Рис. 3.

Все Canva должны быть зарегистрированы в Canva Repository. Visualizer Manager содержит списки визуализаторов и их соответствие различным Canva. Таким, образом, Visualizer не привязан непосредственно к Canva, а визуализация осуществляется по средством вызова соответствующего метода у Visualizer Manager.

3.2. TOOL — CANVA

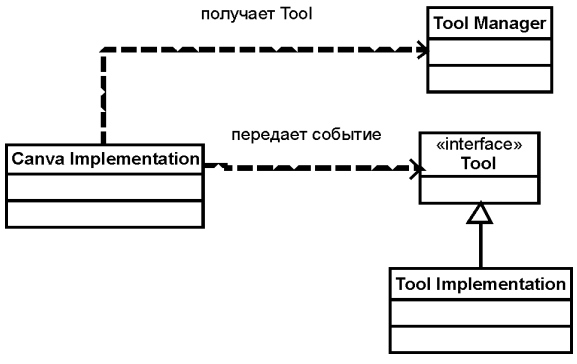

Рис. 4.

В случае наступления какого-либо пользовательского события, конкретная реализация Canva использует Tool Manager для получения текущего Tool и передает ему информацию о произошедшем событии.

3.3. VISUALIZER — TOOL

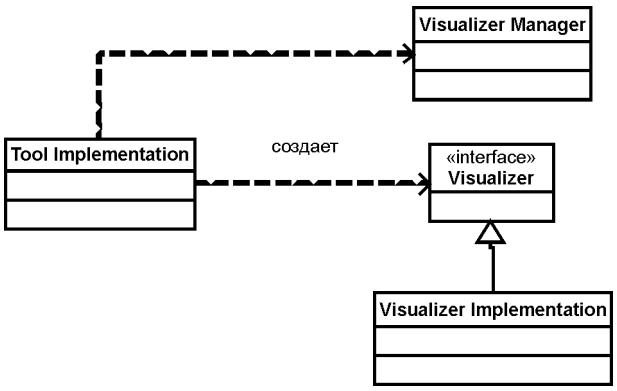

Рис. 5.

В отличии от взаимодействия Tool—Canva, Tool может сам создавать и регистрировать Visualizer, что позволяет ему инкапсулировать бизнес логику отображения того или иного инструмента, а добавление нового инструмента не требует внесение изменений в остальные подсистемы.

4. ЗАКЛЮЧЕНИЕ

Использование вышеописанного шаблона позволят достичь высокого уровня расширяемости, уменьшить затраты на поддержку и реализацию. Как и любой шаблон проектирования, он предоставляет общее решения, которое может быть использовано при возникновении ситуации, описанной в предметной области.

Необходимо заметить, что шаблон описывает лишь рекомендации и жесткое ему следование не является обязательным. В каждом конкретном случае предметная область может накладывать свои ограничения или давать возможность альтернативного решения.

Следуя этому шаблону, был реализован простой двухмерный графический редактор, который затем был расширен до трехмерного с минимальным изменением существующего кода.

СПИСОК ЛИТЕРАТУРЫ

1. *Википедия*. Шаблоны проектирования. — http://ru.wikipedia.org/wiki/Шаблоны_проектирования

2. *Тюкачев Н. А., Илларионов И. В., Хлебостроев В. Г.* Компьютерная графика и мультемедиа. Воронеж: Воронежский госуниверситет, 2008.

3. *Википелия*. Model—View—Controller. — http://ru.wikipedia.org/wiki/MVC

4. *Тюкачев Н. А.* Разработка алгоритмов вычислительной геометрии и их реализация в трехмерной геоинформационной системе. Воронежский госуниверситет. 2009.

5. *ErichGamma, RichardHelm, RalphJohnson, JohnVlissides*. Design Patterns: Elements of Reusable Object-Oriented Software. AddisonWesley Professional. 1994.

Шалиткин Андрей Владимирович — аспирант кафедры Программирования и информационных технологий, Воронежский государственный университет. Тел. (4732) 208-470.

Shalitkin A. V. — postgraduate student , the dept. of the Programming and Information Technologies, Voronezh State University. Tel. (4732) 208-470.