Создание сложных графических систем с использованием подключаемых компонент

Е.Ю. Денисов
Институт Прикладной Математики им. М.В.Келдыша РАН, Москва
e-mail: eed@spp.keldysh.ru

Современные графические системы настолько сложны, что для обеспечения низкой стоимости их разработки и дальнейшей поддержки в течение жизненного цикла целесообразно создавать такие системы на основе объектно-ориентированных компонент (модулей, динамических библиотек). Такие компоненты, будучи минимально-зависимыми друг от друга, позволяют решить следующие задачи:

- Упрощение, ускорение и удешевление разработки (разные компоненты могут разрабатываться разными командами программистов).
- Возможность сборки из набора компонент системы любой сложности, ориентированной на решение конкретного набора задач («LEGO»-принцип).
- Простота подключения новых модулей, в том числе созданных независимыми программистами.

Предлагаемое решение позволяет оформить графическую систему в виде подключаемых компонент.

Все подключаемые компоненты имеют единый общий интерфейс: функции запуска и завершения, которые гарантированно вызываются механизмом загрузки компонент при подключении / отключении компоненты.

Помимо функций запуска и завершения компонент реализованы и другие операции, такие как автоматическая загрузка и инициализация необходимых компонент в нужном порядке, а также возможность загрузить желательные компоненты, наличие которых не является необходимым (и отсутствие которых не приведет к ошибке работы системы). Кроме того, реализован контроль версий для исключения возможности загрузки компоненты, не рассчитанной на работу с данной версией системы (конфликт версий).

Необходимо отметить, что сами компоненты, в свою очередь, могут состоять из набора более мелких компонент. Предлагаемая реализация предоставляет такую возможность и корректно загружает набор подкомпонент в правильном порядке, гарантирующем их корректную инициализацию и дальнейшее функционирование.

Важным элементом является возможность загрузить компоненту не в процессе старта графической системы, а в любой момент в течение работы системы, например, как только необходимая функциональность понадобилась («горячая загрузка»).

В процессе загрузки компонента должна зарегистрировать себя в системе и предоставить свою функциональность для использования другими компонентами. Для этого имеются набор правил и регламентированных процедур, позволяющие компонентам предоставить свои структуры данных и методы в общее пользование другим компонентам системы. Это делается при помощи вызова соответствующих функций, предоставляемых SDK. Конкретная специфика подключаемой компоненты определяется тем, что она делает при подключении. Компоненты ядра графической системы, как правило, вводят новые классы и объекты, необходимые для реализации некоторой функциональности, и регистрируют их в специальных древовидных структурах, обеспечивающих объектно-ориентированное представление всего набора данных, с которым оперирует графическая система.

Компоненты пользовательского интерфейса, как правило, регистрируют исполняющие объекты в структурах, соответствующих меню и инструментальным линейкам графической системы, и в дальнейшем осуществляют вызов функций из компонент ядра при выборе пользователем пунктов меню. Тем самым они обеспечивают появление новых функциональных элементов (предоставляемых компонентами ядра) в пользовательском интерфейсе графической системы.

Реализована так же независимость от платформы. У программиста нет необходимости изменять исходный код компоненты ядра под каждую новую платформу. Нуждаются в изменении только компоненты пользовательского интерфейса. Однако, если они в свою очередь были реализованы с применением платформенно-независимой среды разработки пользовательского интерфейса (например Qt), такие компоненты также не нуждаются в адаптации под другую платформу. Таким образом, графическая система минимальными усилиями может быть адаптирована для работы на другой платформе.

Концепция подключаемых компонент используется как во внутренней архитектуре графической системы, так и для поддержки внешних пользователей-программистов, обеспечивая им возможность расширения графической системы подключаемыми компонентами собственного производства. Используя готовый SDK, они могут создавать собственные компоненты, которые, подчиняясь определенному набору правил, будут вести себя точно так же как и остальные компоненты системы.

Предложенный подход был использован в реализации нескольких систем синтеза реалистичных изображений и оптического моделирования [1], а также в системе моделирования освещения тонкого красящего слоя [2], основанного на решении волновой задачи дифракции. Использование описанного подхода позволило ускорить разработку этих систем и одновременно снизить трудозатраты на их создание и тестирование.

Работа поддержана грантами РФФИ № 08-01-00649, 09-01-00472, 10-01-00302, а также компанией Integra Inc.

Список литературы

1. Волобой А.Г., Галактионов В.А. Машинная графика в задачах автоматизированного проектирования // "Информационные технологии в проектировании и производстве", № 1, 2006, с. 64-73.
2. Волобой А.Г., Ершов С.В., Поздняков С.Г. Решение дифракционной задачи для моделирования освещения тонкого красящего слоя / Препринт ИПМ им. М.В. Келдыша РАН, № 75, 2009, 22 с.