- *поддержка оконных систем.* В оконных системах стандарты внешнего облика обычно различаются. По возможности дизайн Lexi должен быть независимым от оконной системы;
- *операции пользователя.* Пользователи управляют работой Lexi с помощью элементов интерфейса, в том числе кнопок и выпадающих меню. Функции, которые вызываются из интерфейса, разбросаны по всей программе. Разработать единообразный механизм для доступа к таким «рассеянным» функциям и для отмены уже выполненных операций довольно трудно;
- *проверка правописания и расстановка переносов.* Поддержка в Lexi таких аналитических операций, как проверка правописания и определение мест переноса. Как минимизировать число классов, которые придется модифицировать при добавлении новой аналитической операции?

Ниже обсуждаются указанные проблемы проектирования. Для каждой из них определены некоторые цели и ограничения на способы их достижения. Прежде чем предлагать решение, мы подробно остановимся на целях и ограничениях. На примере проблемы и ее решения демонстрируется применение одного или нескольких паттернов проектирования. Обсуждение каждой проблемы завершается краткой характеристикой паттерна.

2.2. Структура документа

Документ – это всего лишь организованное некоторым способом множество базовых графических элементов: символов, линий, многоугольников и других геометрических фигур. Все они несут в себе полную информацию о содержании документа. И все же автор часто представляет себе эти элементы не в графическом виде, а в терминах физической структуры документа – строк, колонок, рисунков, таблиц и других подструктур.[1] Эти подструктуры, в свою очередь, составлены из более мелких и т.д.

Пользовательский интерфейс Lexi должен позволять пользователям непосредственно манипулировать такими подструктурами. Например, пользователю следует дать возможность обращаться с диаграммой как с неделимой единицей, а не как с набором отдельных графических примитивов, предоставить средства ссылаться на таблицу как на единое целое, а не как на неструктурированное хранилище текста и графики. Это делает интерфейс простым и интуитивно понятным. Чтобы придать реализации Lexi аналогичные свойства, мы выберем такое внутреннее представление, которое в точности соответствует физической структуре документа.

В частности, внутреннее представление должно поддерживать:

- отслеживание физической структуры документа, то есть разбиение текста и графики на строки, колонки, таблицы и т.д.;

[1] Авторы часто рассматривают документы и в терминах их *логической* структуры: предложений, абзацев, разделов, подразделов и глав. Чтобы не слишком усложнять пример, мы не будем явно хранить во внутреннем представлении информацию о логической структуре. Но то проектное решение, которое мы опишем, вполне пригодно для представления и такой информации.

- генерирование визуального представления документа;
- отображение позиций экрана на элементы внутреннего представления. Это позволит определить, что имел в виду пользователь, когда указал на что-то в визуальном представлении.

Помимо данных целей имеются и ограничения. Во-первых, текст и графику следует трактовать единообразно. Интерфейс приложения должен позволять свободно помещать текст внутрь графики и наоборот. Не следует считать графику частным случаем текста или текст – частным случаем графики, поскольку это в конечном итоге привело бы к появлению избыточных механизмов форматирования и манипулирования. Одного набора механизмов должно хватить и для текста, и для графики.

Во-вторых, в нашей реализации не может быть различий во внутреннем представлении отдельного элемента и группы элементов. При одинаковой работе Lexi с простыми и сложными элементами можно будет создавать документы со структурой любой сложности. Например, десятым элементом на пересечении пятой строки и второй колонки мог бы быть как один символ, так и сложно устроенная диаграмма со многими внутренними компонентами. Но, коль скоро мы уверены, что этот элемент имеет возможность изображать себя на экране и сообщать свои размеры, его внутренняя сложность не имеет никакого отношения к тому, как и в каком месте страницы он появляется.

Однако второе ограничение противоречит необходимости анализировать текст на предмет выявления орфографических ошибок и расстановки переносов. Во многих случаях нам безразлично, является ли элемент строки простым или сложным объектом. Но иногда вид анализа зависит от анализируемого объекта. Так, вряд ли имеет смысл проверять орфографию многоугольника или пытаться переносить его с одной строки на другую. При проектировании внутреннего представлении надо учитывать эти и другие потенциально конфликтующие ограничения.

Рекурсивная композиция

На практике для представления иерархически структурированной информации часто применяется прием, называемый *рекурсивной композицией*. Он позволяет строить все более сложные элементы из простых. Рекурсивная композиция дает нам способ составить документ из простых графических элементов. Сначала мы можем линейно расположить множество символов и графики слева направо для формирования одной строки документа. Затем несколько строк можно объединить в колонку, несколько колонок – в страницу и т.д. (см. рис. 2.2.).

Данную физическую структуру можно представить, введя отдельный объект для каждого существенного элемента. К таковым относятся не только видимые элементы вроде символов и графики, но и структурные элементы – строки и колонки. В результате получается структура объекта, изображенная на рис. 2.3.

Представляя объектом каждый символ и графический элемент документа, мы обеспечиваем гибкость на самых нижних уровнях дизайна Lexi. С точки зрения отображения, форматирования и вкладывания друг в друга единообразно трактуются текст и графика. Мы сможем расширить Lexi для поддержки новых наборов