Титульник

Защищается студент группы Б8403а Куцелабский Егор Сергеевич по теме «Текстовый процессор для open-source движка Citrus»

Руководитель – старший преподаватель кафедры информатики, математического и компьютерного моделирования, Кленин Александр Сергеевич

Game Forest и текст

Студия Game Forest занимается созданием игр. Текст занимает важную часть в информационных сообщениях в играх. Так же текст используется в визуальных редакторах игрового движка, разрабатываемого компанией. Текущая реализация текстового процессора в движке не обладает достаточной эффективностью, поэтому, при увеличении количества текста, изображаемого на экране, частота кадров в приложениях падает.

Текстовые процессоры

Для редактирования текстовых данных существуют текстовые процессоры.

Они хранят текст, используя при этом различные по эффективности структуры данных, вычисляют его размеры, отрисовывают, некоторые редакторы (например, Word), позволяют, помимо редактирования текста, использовать в документе изображения, таблицы и прочие не относящиеся непосредственно к тексту элементы.

Цели и задачи работы

Цель работы – создание нового текстового процессора для движка.

Для достижения цели были поставлены следующие задачи:

* Изучить эффективные методы представления текста
* Реализовать текстовый процессор
* Оценить выигрыш в производительности

Структуры данных

Мною были рассмотрены различные подходы к реализации текстовых редакторов. Сравнительную таблицу алгоритмических сложностей подходов для базовых операций вы можете увидеть на экране. Стоит отметить, что она не вполне отражает реальные характеристики подходов, поскольку типичная производительность подходов значительно отличается от производительности в худшем случае и её сложно верно оценить. При наполнении таблицы использовались данные для худших возможных случаев для соответствующих подходов. Подробную информацию о перечисленных в таблице подходах вы можете прочитать в отчете.

Функциональные требования

Эффективная обработка внутреннего представления текста -

Текст должен храниться в эффективной структуре данных, все базовые операции должны выполняться быстро, в том числе для больших объемов текстовых данных (порядка нескольких десятков мегабайт) при этом структура не должна требовать значительных объемов оперативной памяти.

Отрисовка текста с заданными параметрами –

Текст должен отрисовываться в окне заданного размера, с заданными параметрами шрифта, при этом отрисовываться должна только та часть текста, которая попадает в видимую область окна.

Обработка пользовательского ввода –

Процессор должен обрабатывать пользовательский ввод – ввод текстовых данных, перемещение курсора с помощью мыши и клавиш клавиатуры или касаний экрана выполняя при этом соответствующие операции по редактированию текста.

Архитектура системы: предыдущая версия

На слайде представлена диаграмма классов предыдущей версии текстового процессора.

Эта реализация была признана неудачной. При оптимизации этой версии процессора пришлось бы совершить двойную работу, переписывая похожие участки кода в двух классах, в то время как функциональность RichText и SimpleText можно объединить в одном классе.

Архитектура системы: представление текста

Система состоит из трех модулей:

Модуль, отвечающий за обработку внутреннего представления текста – обработка базовых команд, выделение, перемещение курсора, undo/redo.

На слайде изображена диаграмма классов данного модуля.

Архитектура системы: отрисовка и ввод

В основу модуля, отвечающего за отрисовку, взят существовавший ранее код, однако он был значительно модифицирован для оптимизации (т. е. для отрисовки только видимой части текста).

Модуль, отвечающий за обработку пользовательского ввода, принимает поддерживаемые команды и передаёт их соответствующим обработчикам.

На слайде изображены диаграммы классов перечисленных систем.

Piece Table

В качестве подхода для реализации был выбран метод Piece Table. Суть Piece Table состоит в том, что вместо хранения символов в памяти хранится информация о позиции текстового фрагмента в потоке, его длине, а также о том, к какому именно потоку относится фрагмент. В Piece Table используется два потока: первый – исходный файл, этот поток используется только для чтения. Второй – добавочный файл, весь новый текст записывается в него. При необходимости вывести текст на экран выполняется пробег по структуре.

На слайде представлен пример работы Piece Table.

Splay Tree

Piece table это структура для эффективного редактирования текста, необходима также структура и для навигации по тексту.

Splay tree – сбалансированное бинарное дерево поиска, основное свойство которого – элементы, к которым обращались в последний раз, переносятся в корень дерева. Это свойство очень полезно при работе с текстовыми редакторами, поскольку большая часть операций вставки и удаления производится в позиции курсора, а частота её смены на значительную величину невелика по сравнению с частотой прочих операций.

Алгоритм балансировки с переносом элемента в корень имеет логарифмическую амортизированную сложность. После переноса необходимого элемента в корень, любая операция вставки, удаления или обращения к элементу выполняется за O(1). Подробнее об алгоритме балансировки можно прочесть в отчете, там же приведена ссылка на статью с доказательством сложности алгоритма.

На слайде приведён пример операции переноса элемента в корень.

Интерфейс

На экране представлен пример интерфейса. Представлен многострочный текст, часть строк достаточно длинна, чтобы к ним применялся soft wrap. Видно, что текст можно прокручивать. Это эффективная реализация прокрутки, в которой отрисовываются только строки, в данный момент помещающиеся в видимую область.

Тестирование

Функциональность модуля внутренней обработки текста была проверена с помощью юнит-тестирования.

Сравнение скорости работы предыдущей версии редактора и моей реализации показало, что при работе с текстом объемом 100 тысяч строк новая версия процессора при вертикальной прокрутке отрисовывается без задержек на высокой частоте кадров, в то время как предыдущая версия процессора имеет задержку в 2-3 секунды перед отрисовкой нового кадра. То же относится и к изменению размеров виджета. Операции вставки, по сравнению с обычным текстом, выполняются примерно на 30% быстрее, при этом во время вставок в новой версии процессора пересчитываются позиции символов в пикселях, что даёт значительный прирост производительности на этапе отрисовки.

Реализация

Физические характеристики системы приведены на слайде.

Заключение

* Выполнен анализ структур данных для работы с текстом
* Реализована новая версия текстового процессора
* Реализованные модули показывают значительное улучшение производительности по сравнению с предыдущей версией процессора
* В настоящий момент код находится на этапе опытной эксплуатации

СПАСИБО ЗА ВНИМАНИЕ