УДК 004.415

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ API ДЛЯ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Теплов С.В., студент гр. ПИб-151, IV курс
Научный руководитель: Пимонов А.Г., д.т.н., профессор
Кузбасский государственный технический университет
имени Т.Ф. Горбачева
г. Кемерово

Графический пользовательский интерфейс (англ. graphical user interface, GUI) не менее важная часть программы, чем её функционал, так как идея перехода от интерфейса командной строки состояла в том, чтобы упростить использование ЭВМ для конечных пользователей [1]. Решение было верным. У каждого из нас есть телефон, компьютер или другое цифровое устройство явно не с консольным интерфейсом. Операционная система без программ для такого устройства мало функциональна, потому было создано множество программ разной направленности: от простых текстовых редакторов до систем управления предприятием. Каждая из этих программ имеет разнообразный интерфейс и многие из них непохожи друг на друга. Это многообразие возможно благодаря обширному GUI API, предоставляемому операционной системой.

API (англ. application programming interface) – интерфейс прикладного программирования, по сути набор функций, констант, классов и прочего для написания программистами всевозможных программ [2]. В контексте данной статьи это классы и методы для создания графических элементов и настройки их внешнего вида.

Существует особый вид интерактивных программ – компьютерные игры. Для каждой игры пишется её ядро, называемое движком (англ. engine). В него входит множество функциональных элементов, отвечающих за разные аспекты игры: физику, графику, звук и прочее. К таким элементам относится собственный GUI API.

Взявшись за очередной проект на движке игры MTA (англ. multi theft auto, мультиплеерная версия игры GTA San Andreas), я обнаружил, что предоставляемый GUI API функционал очень скуден. Встал вопрос проектирования и реализации собственного интерфейса, так как на рынке высокая конкуренция и необходимо переманивать игроков чем-то новым и уникальным.

Для начала необходимо было проанализировать доступные инструменты. Движок предоставлял мне:

1. язык программирования Lua;
2. обработчик нажатия клавиш;
3. обработчик мыши;

4. вывод изображений на экран.

Lua – скриптовый язык программирования, без строгого типизирования данных. Основная его мощь – таблицы со свободной индексацией, а также метатаблицы – некое подобие классов из объектно-ориентированного программирования (ООП). На основе последних и было принято решение реализовывать GUI API. Метатаблицы предоставляют доступ к таким технологиям ООП как компиляция, агрегирование, а также возможность пользоваться полями и методами [3].

Кратко процесс работы компьютера можно охарактеризовать следующим образом: ввод, хранение, обработка и вывод информации. Клавиатура – мощный инструмент по вводу текстовой информации. Обработчик клавиш в движке MTA позволяет получить индекс клавиши и её состояние (нажата или отпущена) – этого более чем достаточно для реализации GUI.

Компьютерная мышь – ещё один способ ввода информации. Без графического интерфейса польза курсора не столь велика, но вместе с ним удобство управления компьютером увеличилось в разы. Игровые движки на персональном компьютере предоставляют доступ к позиции курсора и обработке кликов мыши. Остаётся последний по списку, но не по значению инструмент.

Вывод изображения на экран – то, без чего невозможно создать GUI по определению. MTA обладает графическим движком, основанным на библиотеке DirectX. Это позволяет выводить картинки из большого спектра форматов, а также выводить текст любого шрифта. Так как вывод графики обрабатывается на видеокарте, то существует возможность использовать шейдеры, но об этом позже.

От анализа инструментов переходим к аналитике необходимых элементов интерфейса. Их можно разделить на три основных группы: 1) ввод данных; 2) вывод данных; 3) контейнеры. К первой группе относятся: поле для ввода текста, переключатели, кнопки. Ко второй – поле вывода текста и статичное изображение. К контейнерам относятся поля с прокруткой и «главное окно», предоставляющие доступ к взаимодействию с интерфейсом. Последний элемент имеет смысл только в играх, где мышь используется для ориентации в игровом мире. Тут не описаны более сложные элементы, такие как списки и комбинированные списки, но они конструируются из простых элементов и не всегда нужны в играх.

Все элементы обладают ссылкой на родителя, именем, для обращения к элементу, позицией и размером, также флагом отображения и методами показать\скрыть. Это параметры, которые необходимо вынести в базовый класс, но Lua не предоставляет наследования, потому они просто были продублированы в каждом из классов. Следующий шаг: реализация основных элементов вывода.

Класс для текстового элемента. Он должен обладать полями базового класса и параметрами самого текста, такими как шрифт, размер шрифта, цвет текста и прочими. Как в любой хорошей API все изменения параметров делаем через методы (get/set на каждое поле). Далее проектируем класс статично-

го изображения. Отличия от базового класса – это наличие объекта класса текстового элемента и выводимое изображение, а также положение элемента относительно координат. Два вышеописанных класса очень важны. На их основании будут строиться остальные элементы.

Класс кнопки. В основе элемента статичное изображение. Кнопка должна подсвечиваться при наведении курсора. Это позволит интуитивно понять пользователю, что на элемент можно кликнуть. К методам изменения параметров добавляются методы подсветки\затухания и включить\выключить.

Класс поля для ввода. Элемент также базируется на статическом изображении. Основные состояния: включен\выключен\элемент в фокусе. Вывести элемент в фокус можно при помощи мыши, на той же основе, что и с кнопкой. Когда элемент в фокусе, управление над текстом переходит к клавиатуре.

Все остальные элементы базируются на вышеуказанных в различных комбинациях и модификациях. От требований GUI зависит то, какие ещё элементы необходимо включить в API.

Несколько замечаний про программирование GUI в графических движках. Как уже говорилось выше, обработка графики на видеокарте предоставляет возможность использовать шейдеры. Шейдер – это программа, исполняемая на процессоре видеокарты и помогающая доводить графику до невероятных высот. В нашем случае шейдеры могут помочь в создании эффектов для таких элементов, как использование масок, вращение изображений\текста, изменение тональности, наложения различных искажений и прочего. Ограничения только в возможностях фантазии автора.

В заключение хочется сказать, что подход к проектированию GUI «от простого к сложному» весьма эффективен. На его основе в проекте было сначала спроектировано, а позже и программно реализовано 10 элементов пользовательского интерфейса, которые активно используются. API на их основе предоставляет возможность гибкой настройки, что в свою очередь предполагает многоразовое использование данного программного продукта.

Список литературы:

1. Графический интерфейс пользователя. Википедия. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Графический_интерфейс_пользователя, свободный (дата обращения: 31.03.2019).

2. API. Википедия. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/API, свободный (дата обращения: 31.03.2019).

3. Lua: about. [Электронный ресурс]. – Режим доступа: URL: http://www.lua.org/about.html, свободный (дата обращения: 31.03.2019).