УДК 004.453.2

В. В. Войтко, к. т. н., доцент; А. В. Денисюк; Г. Л. Луцишин

УНИВЕРСАЛЬНЫЙ ГРАФИЧЕСКИЙ КОМПОНЕНТНЫЙ РЕДАКТОР ИГРОВЫХ КАРТ

В статье предложена разработка графического редактора игровых карт, реализованного по архитектурному принципу управления объектами с использованием компонентной технологии для расширения своих функциональных возможностей в системах открытого типа. Редактор характеризуется независимостью от сюжетов и жанров игр и от конкретных игровых каркасов, что обуславливает возможность его универсального использования не только в области создания игровых технологий, но и как средства в дизайнерских и инженерных проектах.

***Ключевые слова**: игровые каркасы, автоматизация разработки игр, игровые сущности, компонентная архитектура, универсальные характеристики, структура графических игр.*

Введение

Сегодня, в эпоху стремительного развития компьютерных технологий, мультимедийные продукты, направленные на достижение научно-исследовательской, научно-просветительской, учебной, популяризаторской, развлекательной и коммерческой целей, широко используются. Разработке современных компьютерных игр предшествует создание редактора игровых карт, который требует дополнительного времени в пределах прогнозируемого срока разработки конечного программного продукта [1]. Разработанные фирмами-производителями игровые карты реализуются в виде отдельных файлов, форматы которых не являются унифицированными и не разглашаются. Редактор игровой карты может входить в базовую игровую поставку для обеспечения возможности создания и использования собственных карт в рамках отдельного мультимедийного продукта с целью реализации авторских идей в графической среде (такая возможность обеспечивается в играх Termins, Lode Runner, Serious Sam и др.).

Однако современные редакторы игровых карт характеризуются узкоспециализированными характеристиками, ограниченными и жестко определенными функциональными возможностями и ориентацией на конкретное программное обеспечение для создания компьютерного приложения, которое ограничивает перспективу использования существующих графических редакторов даже для написания новых версий игровых мультимедийных продуктов. Использование имеющихся редакторов игровых карт другими разработчиками вообще видится невозможным в связи с ориентированием редакторов на разные игровые каркасы. Поэтому актуальной сегодня является разработка графических редакторов игровых карт, специализированные характеристики которых носили бы универсальный характер, что обусловит перспективность их использования в средствах создания компьютерных игр.

Следовательно, целью работы является автоматизация процесса создания игровых карт средствами универсального графического редактора. Объект исследования – методы построения графических редакторов. Предметом исследования являются принципы создания игровых карт и методы реализации графических эффектов в компьютерных играх. Главными задачами работы считаем разработку графического редактора игровых карт и обеспечение его универсальных характеристик.

Анализ современных графических редакторов игровых карт

Рассмотрим возможности и ограничения некоторых существующих редакторов. Игра Termins – двумерная графическая стратегия реального времени, которая поставляется со встроенным редактором карт. Он позволяет редактировать и создавать собственные карты с целью их

последующего использования в игре. Редактор содержит стандартный набор игровых объектов, ориентированных на обеспечение сюжета игры. Этот набор не подлежит дополнению собственными объектами и является конечным. Кроме того, существуют определённые правила относительно размещения объектов на карте. Таким образом, по назначению редактор носит узкоспециализированный характер, имеет скрытый формат карты, не обеспеченный возможностью видоизменения или дополнения библиотеки объектов, что делает невозможным его использование другими разработчиками в процессе создания новых игр.

Рассмотрим более мощный редактор, который поставляется вместе с игрой Heroes 3. В редакторе позволяется строить карту игры из заранее определенных наборов объектов – набора дорог, препятствий, деревьев, травы и т. п. Объекты каждого набора имеют фиксированные свойства и четко определенное назначение. Создание собственных объектов не предусмотрено. Редактор поддерживает лишь несколько фиксированных размеров карт. Формат файла карты является скрытым. Поэтому приложение редактора для создания собственных игр также является невозможным.

Следует отдельно обратить внимание на то, что сегодня появились системы автоматизированной разработки игр. К ним принадлежит Microsoft XNA Game Studio, которая предназначена для разработки игр под консоли XBox и ОС Windows. Система имеет развитой редактор уровней. Уникальной в своем роде является система SkyStudio фирмы Skyfallen Entertainment, которая представляет собой универсальную интегрированную среду разработки игр, не зависящую от платформы и игрового каркаса; генерирует программный код и имеет широкий спектр возможностей. Однако подобные современные среды еще не приобрели популярность и распространение, а подавляющее большинство игр пишется традиционно, с использованием самостоятельно разработанных редакторов уровней. Автоматизированные системы имеют собственные языки скриптов, методы программирования физических явлений, логики и графики, однако имеют и определенные ограничения на расширение и усовершенствование библиотеки типовых объектов. Таким образом, актуальной видим разработку графического редактора игровых карт, модель которого будет носить открытый характер, что обеспечит возможность расширения набора объектов собственными COM-компонентами и обеспечит универсальность характеристик редактора.

Постановка задачи

Под универсальностью разрабатываемого редактора понимаем: расширение функциональных возможностей; независимость от жанров и сюжетов игр; возможность настройки под конкретные частные ситуации специализированного характера; независимость от игровых каркасов. Скрытость формата исходного файла карты реализуем с целью защиты авторских прав, однако редактор должен быть обеспечен средствами считывания файла карты.

Поддержка расширения функциональных возможностей редактора заключается в том, что редактор являет собой набор COM-компонентов [2], которые взаимодействуют между собой через документируемые интерфейсы. Таким образом, любой компонент может быть легко заменен другим, более модернизированным или специализированным. Кроме того, обеспечивается возможность добавления новых компонентов, созданных разработчиками игр, с целью настройки редактора под свои потребности. Добавленные компоненты заносятся в актив библиотеки для повторного использования. Поэтому разрабатываемый редактор должен иметь полностью компонентную архитектуру.

Все современные игры объединяет их общая структура [3]. Игра – это набор объектов, которые взаимодействуют между собой. Каждый из объектов имеет определенный набор свойств и определенную логику поведения. Независимость разрабатываемого редактора от жанров и сюжетов игр заключается в возможности средствами редактора создавать новые игровые объекты, задавать им определенные свойства и логику поведения и использовать в процессе составления игровых карт (даже объект «трава» на карте может иметь свойства: цвет, опасность, не говоря о более сложных объектах) [3].

Возможность настройки редактора на описание ситуаций с последующим превращением универсальных возможностей в специализированные считаем главной характеристикой разрабатываемого редактора. С одной стороны редактор обеспечивает возможность создания разных игровых объектов, но это допускается лишь на этапе разработки моделей игры. Конечный вариант

редактора должен поставляться в конкретно специализированном виде с целью обеспечения корректности режимов работы игры. Эта проблема может быть разрешена двумя путями: обеспечением возможности замены компонентов (такой подход предусматривает создание двух копий компонента: универсальной – для этапа разработки игры и специализированной – для конечного программного продукта), что, в свою очередь, требует избыточных системных ресурсов; созданием компонентов, которые через файлы конфигурации можно настраивать на разные режимы работы, причём разработчик игры свободен в определении синтаксиса подобных конфигурационных файлов.

Сегодня существует много игровых каркасов, отличающиеся принципами функционирования и использования, а также своими специализированными возможностями [4]. Графические эффекты, которые поддерживает один каркас, могут вообще не поддерживаться другими каркасами или выглядеть в них по-другому. Если графический редактор будет работать на определенном игровом каркасе, то этот же каркас его будет и ограничивать. Если разработчик использует другой игровой каркас, то он будет вынужден создавать графические эффекты дважды на разных каркасах: для редактора и для самой игры. Поэтому было решено, что редактор не должен быть ориентированным по умолчанию на конкретный каркас. Разработчик один раз создает для редактора компонент инициализации каркаса средствами собственного каркаса. Причём такой компонент станет единственным не только для конкретного разработчика, но и для всех разработчиков, использующих этот игровой каркас. Таким образом, можно утверждать, что разработанный редактор будет представлять собой надстройку над определенным игровым каркасом. Следует заметить, что игровые каркасы являются высокоуровневыми средствами разработки игр, поэтому создание компонента инициализации не является сложной задачей.

Формат исходного файла карты должен быть скрытым. Это позволит дизайнерам создавать игровые карты для существующих игр без нарушения авторских прав. В редакторе предусмотрено два обязательных системных компонента, один из которых сохраняет карту в файле, а другой считывает информацию из него. Конечно, как и любой другой компонент редактора, эти компоненты можно дополнительно усовершенствовать или заменить собственными. Компонент считывания файла карты редактора будет использоваться непосредственно в игре. Аналогично возможно также использование любых других компонентов редактора. Таким образом, разработчик игры лишь один раз должен реализовать необходимые графические и другие возможности карты средствами своего каркаса, программная поддержка их использования обеспечивается средствами автоматизации графического редактора.

Разработка структуры редактора

Базовым компонентом графического редактора является регистратор других компонентов UCWERegistrar. Этот компонент содержит динамический список компонентов, которые на время работы редактора находятся в активном режиме. Это обеспечивает принцип: «каждый компонент может зарегистрироваться в списке и связаться с каждым активным компонентом библиотеки». Благодаря такому принципу возможности редактора можно расширять в любом направлении. Интерфейсы компонента UCWERegistrar приведены в табл. 1.

Таблица 1

Интерфейсы компонента UCWERegistrar

Название интерфейса	Название функции	Назначение функции
ICUWERegistrar	Registering	Вызывается компонентом для регистрации себя в списке компонентов
	UnRegistering	Вызывается компонентом для удаления себя из списка зарегистрированных компонентов
	GetComponent	Зарегистрированный компонент всегда содержит ссылку на интерфейс ICUWERegistrar. Через эту функцию он может запросить интерфейс IUnknown любого другого зарегистрированного компонента.
	GetComponentsList	Возвращает список всех активных компонентов редактора

Главным компонентом редактора является компонент UCWEEditor, создающий окно главного меню программы и отвечает за загрузку других компонентов, список которых указан в конфигурационном файле. Одним из интерфейсов этого компонента является интерфейс IUCWEComponent. Это главный интерфейс, который должен бать реализован всеми компонентами, которые поддерживаются редактором. Интерфейс IUCWEComponent включает основные функции управления компонентом.

Еще одним поддерживаемым интерфейсом является IUCWEMenu. Его наличие указывает на поддержку компонентом главного меню, а другие компоненты могут добавлять в это меню собственные опции, используя интерфейс IUCWEMenuItem. Интерфейс IUCEMenu хранит список интерфейсов IUCWEMenuItem компонентов редактора и обеспечивает вызов их функций в соответствии с выбранным элементом меню. Список интерфейсов, которые поддерживаются компонентом UCWEEditor приведен в табл. 2.

Таблица 2

Список интерфейсов компонента UCWEEditor

Название интерфейса	Название функции	Назначение функции
IUCWEComponent (функция этого компонента является одинаковой для каждого компонента редактора)	Registering	Любой компонент-загрузчик должен вызывать эту функцию загружаемого компонента, чтобы загружаемый компонент мог зарегистрироваться, вызвав функцию ICUWERegistrar.Registering
	Init	После вызова Registering компонент-загрузчик должен вызывать эту функцию загружаемого компонента, чтобы последний мог инициализироваться
	Do	Функция вызывается компонентами-загрузчиками, чтобы загруженные ими компоненты могли выполнять свои функции
	UnInit	Функция деинициализации. Вызывается компонентом-загрузчиком.
	UnRegistering	Функция дерегистрации. Вызывается компонентом-загрузчиком, чтобы загружаемый компонент мог освободить раньше занятый компонент UCWERegistrar.
	GetData	Функция вызывается компонентом для сохранения обновленных данных карты на диске
	SetData	Функция вызывается, когда загрузчик карты считал данные соответствующего компонента и передает последнему его сохраненные на диске с помощью функции GetData данные карты
IUCWEMenu	AddItem	Любой компонент, имея указатель интерфейса IUCWEMenu другого компонента, может добавить в меню последнего собственную опцию, передав в функцию AddItem реализованный интерфейс IUCWEMenuItem
	DeleteItem	Компонент, который добавил опцию в меню другого компонента, может удалить свою опцию, вызвав данную функцию

Компонент UCWEMap добавляет опции открытия и создания карты в меню компонента UCWEEditor, реализует эти опции и может загружать другие компоненты. UCWEMap в начале своей работы загружает три компонента: UCWEEngine, UCWESaver и UCWEOpener. UCWEEngine – это компонент инициализации игрового каркаса и обработки введения данных, возможности которого пользователь будет использовать в редакторе в процессе создания карт. UCWESaver обеспечивает сохранение игровой карты в файле на диске, а UCWEOpener отвечает за корректную загрузку. UCWEOpener используется непосредственно в игре для загрузки карт. UCWEMap поддерживает

интерфейс IUCWEComponent, как и компонент UCWEEditor, и имеет аналогичное назначение функций интерфейса.

При выборе в меню компонента UCWEEditor опции создания или открытия карты UCWEMap загружает компоненты UCWEObjectsManager и UCWELayersManager. UCWEObjectsManager предназначен для создания, хранения и удаления игровых объектов. Компонент также сохраняет список экземпляров игровых объектов, которые пользователь может выбирать и размещать на поле карты. Рис. 1 иллюстрирует возможности диалогового окна компонента UCWEObjectsManager.

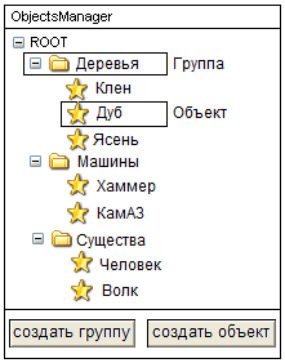

Рис. 1. Диалоговое окно компонента UCWEObjectsManager

Все объекты в списке структурно разделены на группы. Группа не несет конечной информации и не является компонентом, а служит для удобного визуального упорядочивания объектов в списке. Интерфейсы, которые поддерживает компонент UCWEObjectsManager, приведены в табл. 3.

Таблица 3

Интерфейсы компонента UCWEObjectsManager

Название интерфейса	Название функции	Назначение функции
IUCWEComponent	Функции интерфейса описаны в табл. 2	
IUCWEObjectsManager	CopyCurrentObject	Возвращает копию выделенного объекта в списке экземпляров объектов
	GetCurrentObject	Возвращает указание на выделенный объект в списке экземпляров объектов
	GetObjectsList	Возвращает список экземпляров объектов

Каждый объект будет инициализироваться в редакторе компонентом UCWEObject, то есть UCWEObjectsManager сохраняет список компонентов UCWEObject. Пользователь в процессе создания игровой карты выбирает необходимый объект и располагает его на карте. Структура карты состоит из слоев. В определённый момент времени существует лишь один активный слой, поэтому любой добавленный объект обязательно будет принадлежать одному конкретному слою. Слои служат средством группирования добавленных на карту объектов с целью организации удобной схемы управления ими. Слой в редакторе подан компонентом UCWELayer. За редактирование слоев отвечает компонент UCWELayersManager, который сохраняет список всех созданных слоев. Диалоговое окно компонента UCWELayersManager изображено на рис. 2.

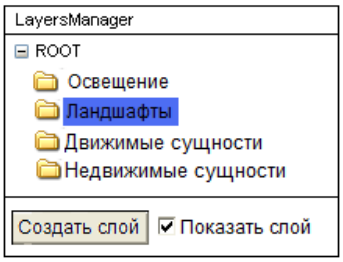

Рис. 2. Диалоговое окно компонента UCWELayersManager

Интерфейсы, которые поддерживает компонент UCWELayersManager, поданы в табл. 4ю

Таблица 4

Интерфейсы компонента UCWELayersManager

Название интерфейса	Название функции	Назначение функции
IUCWEComponent	Функции интерфейса описаны в табл. 2	
IUCWELayersManager	GetCurrentLayer	Возвращает указатель на интерфейс IUCWELayer компонента UCWELayer, который будет инициализировать текущий рабочий слой карты

Компонент UCWELayersManager управляет списком компонентов UCWELayer. Компонент UCWELayer будет инициализировать конкретный слой карты. Он содержит список компонентов UCWEObject – копии объектов из списка в UCWEObjectsManager, добавленных на карту. Действие пользователя на карте перехватывается компонентом UCWEEngine, который обращается к соответствующей функции UCWELayer, а она, в свою очередь, добавляет в свой список новый объект, который передается ей как аргумент. Когда пользователь нажимает клавишу Delete, то компонент UCWEEngine вызывает нужную функцию компонента UCWELayer, которая удаляет из активного списка конкретный (выделенный) объект. Аналогично происходят все действия над объектами карты. Компонент UCWEEngine отслеживает введение данных пользователем и, вызывая функции соответствующих компонентов, оперирует объектами на карте. Список интерфейсов компонента UCWELayer подан в табл. 5.

Таблица 5

Список интерфейсов компонента UCWELayer

Название интерфейса	Название функции	Назначение функции
IUCWEComponent	Функции интерфейса описаны в табл. 2	
IUCWELayer	AddObject	Добавить новый объект к слою
	DeleteObject	Удалить объект из списка
	SetCurrentObject	Сделать объект на карте текущим
	GetCurrentObject	Возвратить указатель на выделенный объект

Компонент UCWEObject представляет собой игровой объект карты. Каждый объект – это сущность, которая аккумулирует определенные свойства. Свойство будет инициализироваться в редакторе компонентами UCWEProperty. Таким образом, компонент UCWEObject содержит набор компонентов UCWEProperty. Компоненты UCWEProperty самостоятельно создает пользователь редактора в соответствии с сюжетом разрабатываемой игры средствами выбранного игрового каркаса. Инициализируемые компоненты автоматически используются в игре. Интерфейсы компонента UCWEObject поданы в табл. 6.

Таблица 6

Список интерфейсов компонента UCWEObject

Название интерфейса	Название функции	Назначение функции
IUCWEComponent	Функции интерфейса описаны в табл. 2	
IUCWEObject	GetPropertiesList	Возвращает список компонентов UCWEProperty как свойств объекта

Для каждой конкретной игры компоненты UCWEProperty могут быть своими. Одним из компонентов UCWEProperty можно реализовать внешний вид объекта, другим – числовое значение параметра и т. д. Компонент обязательно поддерживает стандартный интерфейс IUCWEComponent. Следует отметить, что каждый из компонентов редактора может поддерживать другие собственные интерфейсы. Особенно этот принцип касается компонентов UCWEProperty в процессе реализации тесного взаимодействия с компонентом UCWEEngine, поскольку они создаются средствами одного игрового каркаса. Общая модель графического компонентного редактора изображена на рис. 3.

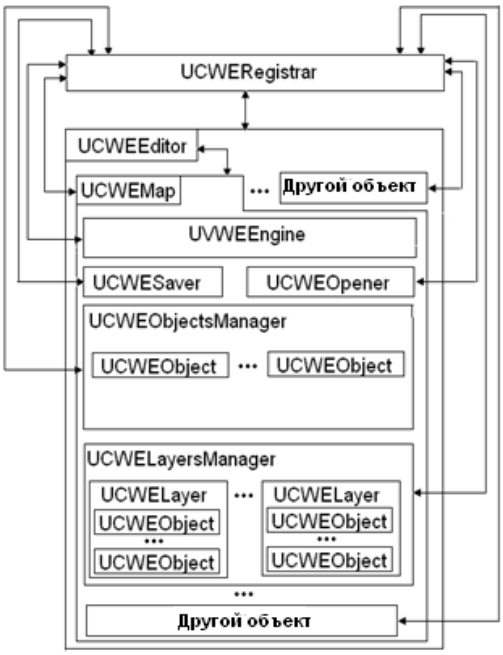

Рис. 3. Общая модель графического компонентного редактора

Заметим, что под вложенностью одного компонента в другой понимаем возможность первого компонента создавать (загружать) следующий. Таким образом, время жизни загружаемого компонента является вложенным во время жизни компонента-загрузчика. Компоненты взаимодействуют между собой по технологии COM, реализуя архитектурный принцип управления объектами. Так, запуск к выполнению конечного файла обеспечивает загрузку компонентов UCWERegistrar и UCWEEditor. UCWEEditor загружает непосредственно компонент UCWEMap и другие компоненты, которые перечислены в конфигурационном файле компонента UCWEEditor. UCWEMap, в свою очередь, загружает соответствующие компоненты UCWESaver, UCWEOpener, UCWEEngine. Рабочий файл UCWE.exe вызывает функцию IUCWEComponent::Registering компонента UCWEEditor, который, в свою очередь, вызывает эти же функции во всех своих загруженных компонентах, побуждая их к выполнению аналогичных действий вплоть до самого низкого уровня вложенности. Аналогично файл UZWE.exe вызывает функцию IUCWEComponent::Init. Компонент UCWEEngine, создав собственными средствами окно рендеринга, циклически вызывает функцию IUCWEComponent::Do компонента UCWEEditor. Последний обеспечивает вызовы этих же функций во всех активных компонентах, двигаясь последовательно к

последнему уровню вложенности. Если пользователь выбирает опцию меню создания карты, то UCWEMap дополнительно догружает компоненты UCWEObjectsManager и UCWELayersManager, и циклические вызовы функции Do повторяются снова: когда управление доходит до функции Do компонентов UCWEProrerty, то, в зависимости от назначения, они реализуют свои эффекты: прорисовку самого объекта одним из компонентов, предоставление ему оттенка другим, графическую реализацию определенного эффекта третьим и т. д. Так пошагово происходит создание объектов, которые образуют карту. Когда пользователь закрывает редактор, то сначала вызывается функция UnInit, а затем функции UnRegistering всех компонентов редактора в последовательности обратной загрузке. Если пользователь выбирает опцию сохранения файла, то компонент UCWEMap вызывает функции сохранения компонента UCWESaver, обеспечивая цепной вызов функций GetData всех вложенных компонентов редактора, и в файле сохраняются все данные, которые будут предоставлены этими функциями. По аналогичной схеме работает и алгоритм открытия файла карты с вызовом компонента UCWEOpener.

Компоненты редактора могут создавать рабочие потоки, следовательно все компоненты должны создаваться с учетом многопоточности. Стандартной для разработанного редактора является модель свободных потоков COM (Free threaded model).

Использование цепной схемы реализации архитектурного принципа управления объектами обеспечивает надежную работу графического редактора и позволяет избегать образования коллизий и конфликтных ситуаций в процессе обработки данных.

Вывод

Разработан универсальный графический компонентный редактор игровых карт, который обобщил возможности существующих редакторов, обеспечив таким образом независимость от сюжетов и жанров игр. Решена проблема независимости редактора от конкретных игровых каркасов. Благодаря применению компонентной технологии, редактор обеспечивает расширение своих функциональных возможностей в системе открытого типа. Имеющиеся средства защиты файла карты позволяют дизайнерам реализовать авторскую идею. Редактор может быть использован для моделирования систем объектов в дизайнерских и инженерных разработках разных направлений.

СПИСОК ЛИТЕРАТУРЫ

1. Todd Barron. Strategy game programming with DirectX 9.0. – Wordware Publishing Inc.: 2003. – 350 с.
2. Дональд Бокс. Сущность технологии COM. Библиотека программиста. – СПб.: Питер, 2001. – 400 с.
3. Ламот Андре. Программирование игр для Windows. Советы профессионала, 2-е изд. : Пер. с англ. – М. : Издательский дом "Вильямс", 2004. – 880 с. : ил. – Парал. тит. англ.
4. Марк Зальцман. Компьютерные игры: как это делается.: Издательский Дом “Логрус”, 2000. – 540 с.

Войтко Виктория Владимировна **–** к. т. н., доцент кафедры программного обеспечения, тел. +38(0432) 598 483.

Денисюк Алла Васильевна **–** асистент кафедры программного обеспечения, тел. +380961104084.

Луцишин Геннадий Леонидович **–** студент 4-го курса факультета компьютерного интеллекта института информационной технологии и компьютерной инженерии, тел. +380969721736, e-mail: Hennadiy@bk.ru.

Винницкий национальный технический университет.