Безопасность в чрезвычайных ситуациях (05.26.02, технические науки)

УДК 614.842.65: 004.428

О разработке программного ядра системы динамического моделирования боевых действий по тушению пожаров

About program core of firefighting actions dynamic modeling system

О.С. Малютин
ФГБОУ ВО Сибирская пожарно-спасательная академия ГПС МЧС России

O.S. Malyutin
FSBEE HE Siberian Fire and Rescue Academy EMERCOM of Russia

Аннотация:

В статье рассказывается о разработке программного обеспечения системы динамического моделирования боевых действий по тушению пожаров. Приводится краткое описание существующей системы изучения пожаров и тенденции ее развития с внедрением современных компьютерных технологий. Даются определения объектно-информационной модели и моделирования боевых действий по тушению пожара. Отмечаются достоинства и недостатки статического и динамического моделирования боевых действий. В качестве инструмента динамического моделирования боевых действий по тушению пожаров предлагается разработка программного обеспечения и приводится описание структуры и принципов построения программного ядра такого решения. Описываются предлагаемые паттерны проектирования системы и алгоритмы. Приводится пример использования программного ядра для моделирования динамики в объектно-ориентированных моделях. В заключении приводятся примеры использования предлагаемой системы.

Ключевые слова: пожарная охрана, пожарная тактика, компьютерное моделирование, программирование, управление, алгоритмы.

Abstract:

In this paper described software for firefighting actions dynamic modeling system development. A brief description of the existing fire examination system and its development trends with the introduction of modern computer technologies is given. Definitions of object-information model and modeling of firefighting actions are given. Pros and cons of static and dynamic modelling methods is marked. As a tool for dynamic modelling offers special software development and provides structure and principles of program core development for such solution. Describes proposed development patterns and work algorithms. Showed examples of program core usage for dynamic computer modelling in object-oriented models. In conclusion, examples of the use of the proposed system are given.

Key words: ifire service, fire tactic, computer modelling, programming, management, algorithms.

Введение

Существующая в настоящее время система управления силами и средствами на пожаре складывалась в середине XX столетия. В виду трудоемкости описания и анализа действий по тушению крупных пожаров, в этот период специалисты пожарной охраны были вынуждены прибегать к упрощению методов работы с информацией о пожарах. Это отразилось и на методах проведения пожарно-тактических расчетов и на системе описания пожаров.

В настоящее время, по мере всеобщей компьютеризации пожарных подразделений, методы работы с описаниями пожаров изменяются в сторону увеличения обрабатываемой информации. Применение компьютерных технологий в работе с разнообразными данными позволило расширить перечень информации, включаемой в описания пожаров. Сами данные приняли более формализованный вид. С одной стороны это позволило включать в сферу изучения пожаров больший объем данных – при этом, акцент в основном делается на внешней стороне работы, на оформлении результатов анализа пожаров. С другой стороны, методы работы с такими данными в целом остались прежними, что привело к увеличению трудоемкости работы в целом.

Можно отметить два основных направления развития системы изучения пожаров: повышение степени формализации используемых данных и поиск новых инструментов для описания и анализа пожаров.

Повышение степени формализации используемых в описании пожаров данных проявляется в стремлении придать им единые, универсальные формы, облечь их в строгий, буквально понимаемый и однозначно трактуемый не зависимо от личности изучающего пожар специалиста вид. Например, с этой целью в последнее время в перечень документов описания пожара включаются карточки учета пожара составленные согласно требованиям изложенным в Приказ МЧС РФ от 21 ноября 2008 г. N 714 "Об утверждении Порядка учета пожаров и их последствий" [1].

Поиск новых инструментов можно заметить в имеющих место попытках создания новых инструментов представления и анализа сведений о пожаре. В качестве примера, иллюстрирующего данное направление развития системы изучения пожаров можно привести созданную в ФГБОУ ВО Сибирская пожарно-спасательная академия ГПС МЧС России автоматизированную информационно-графическую систему ГраФиС-Тактик (далее – ГраФиС), оперирующую описаниями оперативной обстановки на пожаре в виде компьютерных объектно-информационных моделей.

Под объектно-информационной моделью боевых действий по тушению пожара (далее – ОИМП) в рамках ГраФиС понимается совокупность любых объектов, расположенных на территории, на которой осуществляются боевые действия по тушению пожара, с присущими им свойствами, отражающая состояние оперативной обстановки на месте пожара. Ключевой особенностью практической реализации ОИМП в ГраФиС является составление модели исходя из пространственного положения объектов, что делает данную систему схожей по методам составления с системами автоматизированного проектирования.

Под объектно-информационным моделированием боевых действий по тушению пожара в ГраФиС понимается подход к описанию боевых действий по тушению пожара, который предполагает сбор и комплексную обработку в процессе моделирования всей архитектурной, пожарно-технической, пожарно-тактической, управленческой и иной информации о пожаре и ходе его тушения со всеми её взаимосвязями и зависимостями, когда пожар, ход его тушения и всё, что имеет к этому отношение, рассматриваются как единый объект.

В настоящее время модели, создаваемые при помощи ГраФиС являются статическими. Это означает, что они позволяют рассматривать оперативную обстановку на пожаре только в том состоянии, в каком она находилась в конкретный момент времени. С точки зрения пользователя системы ГраФиС получаемые модели выглядят как схемы расстановки сил и средств при тушении пожара (рис.1). Однако информационная составляющая моделей позволяет существенно более подробно описывать текущее состояние боевых действий, что при использовании возможностей как самой системы ГраФиС, так и MS Visio, на базе которой она разработана, существенно расширяет возможности анализа содержащихся в ней данных. Такой подход делает возможным проведение полного комплекса пожарно-тактических расчетов для моделируемой обстановки. Сама же модель фактически формируется параллельно с составлением схемы расстановки сил и средств.

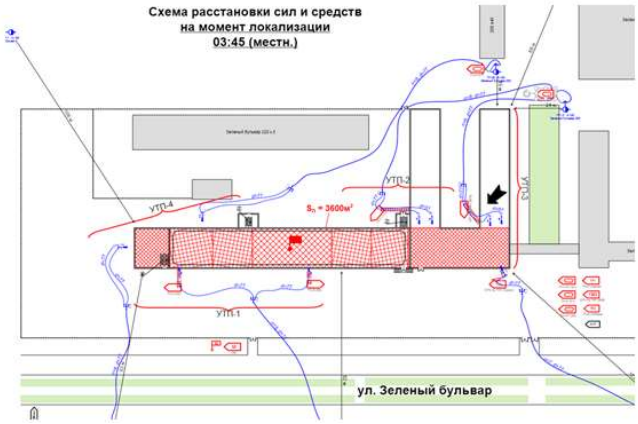

Рис. 1 - Схема тушения пожара составленная при помощи АИГС ГраФиС-Тактик

Применение статических объектно-информационных моделей боевых действий по тушению пожаров позволяет более подробно описывать реально складывающуюся обстановку и хорошо подходит для изучения произошедших пожаров по их окончании, либо для отображения ожидаемой обстановки при составлении планов тушения пожаров или методических разработок на проведение занятий с личным составом. Вместе с тем, реконструкция обстановки на пожаре при помощи статических моделей не позволяет осуществлять моделирование боевых действий по тушению пожаров как процесса – в динамике. При этом возможность моделирования боевых действий и процесса развития самого пожара бывает очень востребованной при прогнозировании обстановки на месте реального пожара, в деятельности органов управления силами и средствами. Возможность спрогнозировать развитие обстановки на пожаре и оценить таким образом принимаемые решения полезна при тренировке лиц, выполняющих функции руководителя тушения пожара (далее – РТП) в процессе профессиональной подготовки.

Таким образом, задача создания системы динамического моделирования боевых действий по тушению пожаров представляется актуальной. Систему объектно-информационного моделирования можно рассматривать как основу интерактивной среды, в рамках которой динамическое моделирование может осуществляться.

Структура ядра системы динамического моделирования

Система ГраФиС разработана на платформе офисного программного обеспечения MS Visio. Данное программное обеспечение располагает обширным инструментарием для пользовательского расширения функционала, включающим среду разработки VBA (Visual Basic for Application) и средства взаимодействия между приложениями COM (Component Object Model). Вместе с тем разработчиками изначально не предполагалось наличие возможностей какой-либо анимации в приложении. Задача создания системы динамического моделирования боевых действий по тушению пожаров, которая могла бы отображать изменения ОИМП в реальном времени предполагает создание собственно системы динамического моделирования в широком смысле. Поскольку задача создания системы динамического моделирования наиболее близка задаче разработке компьютерных игр, то за основу избран подход широко используемый в игровой индустрии, подразумевающий разделение ядра системы, отвечающего собственно за моделирование физических процессов (игрового движка) и оболочки, обеспечивающей решение частных задач конкретной игры (пользовательский интерфейс, логика игры, условия победы и т.д.).

Изучение опыта разработки программных решений для различных платформ и целей показало, что задача создания систем анимации в средах, изначально для этого не предназначенных успешно решается. Наиболее ярким примером может послужить библиотека JQuery превращающая язык программирования Java script в мощный инструмент анимации страниц HTML.

Стандартной основой создания игровых платформ является использование потока кадров. Каждый кадр является игровым событием, во время которого реализуется логика игры. Например, производятся вычисления переменных, изменяется положение объектов, проверяются условия выполнения команд и т.д.

Обычно запуск потока кадров, а так же выполнение общей логики приложения осуществляется специальным объектом, отвечающим за работу программы в целом. Для этого был реализован базовый класс c_Game.

В целом логика работы программы представлена на рисунке 2.

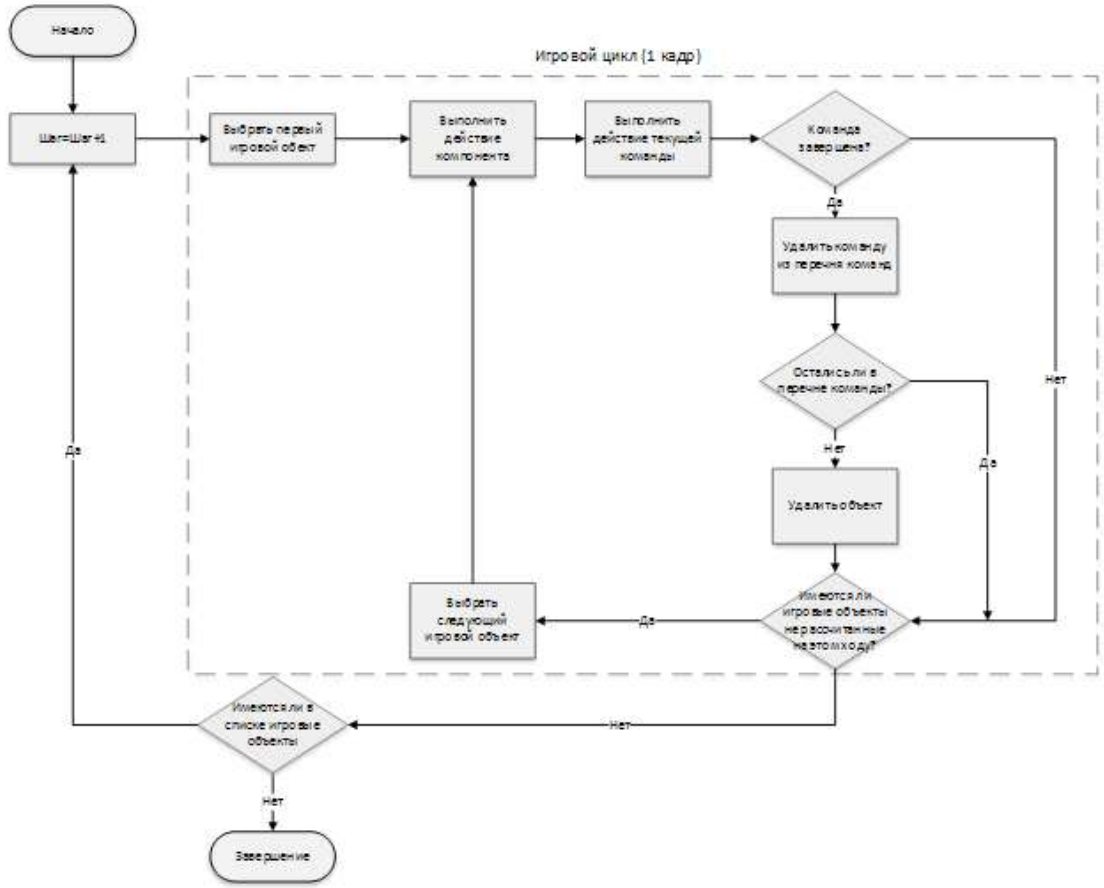

Рис. 2 - Алгоритм работы программного ядра

Класс c_Game содержит коллекцию объектов gameObjects, в которой содержатся ссылки на все активные игровые объекты. Каждый игровой объект содержит коллекцию команд и коллекцию компонентов. При запуске игры, на каждой игровой итерации (на каждом кадре) для каждого игрового объекта выполняется активная для него в текущий момент команда. Активной может быть только одна команда. В случае, если после выполнения команды достигнуто условие означающее, что команда выполнила свою задачу, она удаляется из списка команд для данного объекта, активной становится следующая команда. В случае, если список команд пуст – объекту возвращается логическое значение означающее, что он выполнил все команды и может прекратить свое существование, после этого объект удаляется из списка игровых объектов. Если список игровых объектов оказался пуст, игровой процесс останавливается.

На каждой игровой итерации, помимо команды выполняются действия компонентов игровых объектов. В отличие от команд, компоненты существуют в течение всего жизненного цикла игрового объекта и срабатывают каждый ход. Компоненты могут быть включены или выключены.

Пользователь на любой из итераций может выполнить любые действия с игрой – добавить или удалить игровые объекты, их команды и компоненты. На следующей итерации все изменения внесенные пользователем будут отражены в ходе выполнения программы.

Архитектура программного кода основана на применении объектно-ориентированного подхода к разработке.

В качестве базового шаблона приложения выбран паттерн проектирования «Команда». Паттерн «Команда» (Command) позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. Этот объект запроса на действие и называется командой. При этом объекты, инициирующие запросы на выполнение действия, отделяются от объектов, которые выполняют это действие. Команды могут использовать параметры, которые передают ассоциированную с командой информацию. Кроме того, команды могут ставиться в очередь и также могут быть отменены [2].

В разрабатываемом решении паттерн «Команда» разработан в виде интерфейса ICommand, который реализуют непосредственно все конкретные классы команд приложения (рис. 3).

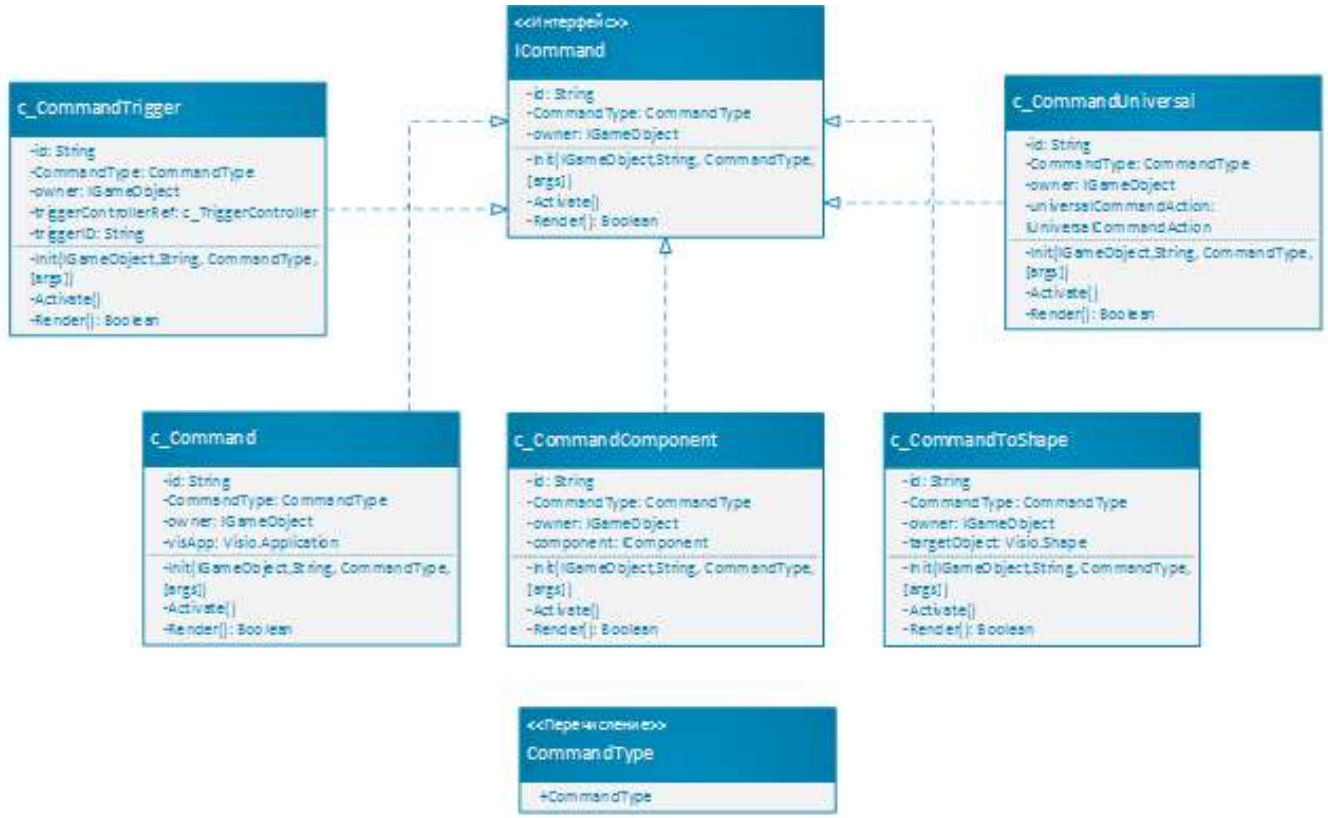

Рис. 3 - UML-диаграмма реализации паттерна «Команда»

Применение паттерна Команда позволяет не только составлять список команд, пользуясь инструментами пользовательского интерфейса оболочки, но и генерировать их непосредственно в процессе выполнения программы, как ответ на изменение состояния ОИМП, что открывает путь к применению в работе с ОИМП программных инструментов анализа вплоть до применения систем искусственного интеллекта. Другими словами, программная надстройка использующая разрабатываемое ядро, сможет самостоятельно управлять элементами оперативной обстановки на схеме пожара – раздавать команды на передислокацию, подачу стволов, применение звеньев ГДЗС и т.д.

Особенностью команд в текущей реализации является тот факт, что для каждого игрового объекта они выстраиваются в очередь выполнения. При выполнении команды возвращается логический результат ее действия. В случае, если результат выполнения команды достигнут – например, объект достиг некой конечной точки движения – команда удаляется из списка и очередь действия переходит к следующей команде.

Такой подход позволяет логически выстраивать последовательность выполнения действий объекта. Однако в ряде случаев может возникнуть ситуация когда какое-либо действие должно выполняться объектом постоянно в течение всего жизненного цикла, и при этом команды так же должны выполняться в обычном порядке. Для реализации этой задачи использован паттерн «Компонент».

Паттерн «Компонент» позволяет инкапсулировать некие свойства родительского объекта в отдельном классе, что делает возможным модифицирование его поведения без необходимости внесения изменений в программный код. Реализация компонентов в рамках системы представлена на рисунке 4.

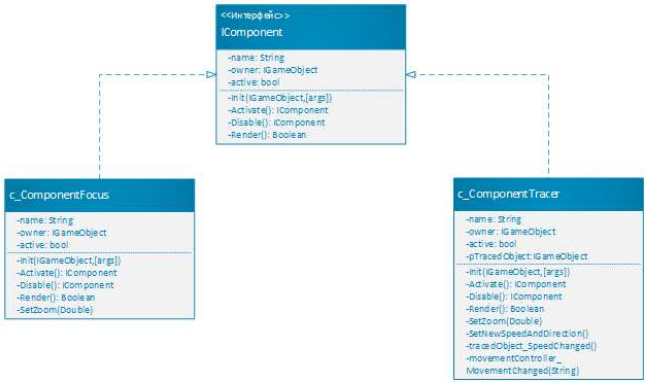

Рис. 4 - UML-диаграмма реализации паттерна «Компонент»

Собственно абстрактный игровой объект создан как реализация интерфейса IGameObject. Использованный при этом подход подразумевает, что конкретные игровые объекты являются моделями, не содержащими собственной логики. Вся обработка их состояний (перемещение, изменение данных и т.д.) осуществляется командами и компонентами, выступающим здесь в роли контроллеров. Учитывая, что визуальное представление объектов осуществляется инструментальными средствами MS Visio, можно говорить о использовании в данном случае паттерна MVC (Model View Controller – модель, представление, контроллер). Схема интерфейса IGameObject и его реализации c_GameObject представлена на рисунке 5.

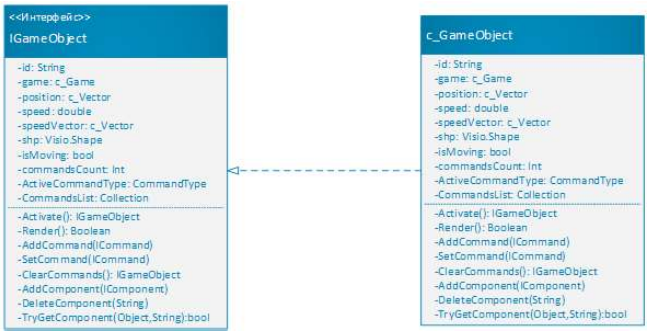

Рис. 5 - UML-диаграмма реализации игрового объекта

Создание новых игровых объектов, команд и компонентов осуществляется посредством специально разработанных методов. Это позволяет упростить программный код и свести последовательность создания новых объектов к одной строке.

Работа программного ядра

Запуск программного ядра заключается в создании экземпляра объекта c_Game. В дальнейшем прописывается перечень объектов, команд и компонентов для каждого из них. Наконец, производится запуск.

На рисунке 6 приведен пример кода VBA, содержащий инициализацию ядра, игрового объекта и указание для него перечня команд заставляющих объект перемещаться по периметру прямоугольника.

```
1   Public Sub runCommands()
2   Dim game As c_Game
3   Dim go As IGameObject
4
5       Set game = NewGame(0.01)
6
7       Set go = game.GetGameObject(1).Activate
8
9       go.AddCommand NewCommand(go, ct_SetSpeed, "10")
10      go.AddCommand NewCommand(go, ct_MoveTo, "1000;1000")
11      go.AddCommand NewCommand(go, ct_MoveTo, "1000;7000")
12      go.AddCommand NewCommand(go, ct_MoveTo, "7000;7000")
13      go.AddCommand NewCommand(go, ct_MoveTo, "7000;1000")
14      go.AddCommand NewCommand(go, ct_MoveTo, "1000;1000")
15
16      game.PlayGame
17  End Sub
```

Рис. 6 - Листинг кода запуска движения объекта.

В строке 2 объявляется переменная game являющаяся реализацией главного класса программного ядра – c_Game. В строке 3 объявляется переменна go реализующая интерфейс IGameObject. Это и есть игровой объект, который будет выполнять команды в Строке 5 инициируется программное ядро. Для этого в качестве аргумента функции NewGame передается значение количества секунд в одном кадре. В данном случае указано значение 0,01 секунда/кадр, означающее, что каждая секунда игрового времени будет представлена 100 итерациями игрового процесса.

В строке 7 переменной go присваивается ссылка на необходимый игровой объект. Для этого используется функция game.GetGameObject, в качестве единственного аргумента которой передается уникальный идентификатор (id) игрового объекта. Программное ядро проверяет, имеется ли объект с таким id в перечне активных объектов и если не находит, выбирает в качестве такового фигуру Visio, расположенную на активном рабочем листе, имеющую такой id. Далее объект активируется при помощи функции Activate.

В строках 9-14 игровому объекту go задаются команды. Создание новых команд осуществляется при помощи функции NewCommand. Первым аргументом, которой передается ссылка на текущий игровой объект, вторым аргументом передается тип команды, третьим аргументом передается строка с перечнем параметров команды.

В строке 9 задается команда типа ct_SetSpeed – «Установить скорость движения равной…». В данном случае скорость устанавливается равной 10 дюймов/кадр.

В строках 10-14 следуют команды типа ct_MoveTo – «Двигаться к…». В качестве параметров передаются координаты точек (в дюймах), в формате «x;y».

Все размеры, используемые программой, измеряются в дюймах, т.к. для Visio эти единицы явля-

ются используемыми по умолчанию. Однако для удобства пользователя разработаны специальные функции, переводящие величины из метрической системы мер в дюймовую и обратно.

Основные механизмы создания надстройки для системы динамического моделирования боевых действий по тушению пожаров

Согласно концепции разработки приложения для динамического моделирования боевых действий по тушению пожаров подразумевается, что программное ядро является лишь основой системы изменения состояния ОИМП во времени. Всю работу по взаимодействию с пользователем, учет пожарно-технических характеристик техники и их влияния на расчеты, и прочий специфический функционал должна взять на себя так называемая программная обертка. Разделение базовых функций динамического моделирования и специфических функций задачи позволит разрабатывать обертку отдельно от ядра, что сделает код более понятным, а необходимость обновления его частей независимой друг от друга.

Создание обертки подразумевает написание программной части использующей функционал программного ядра для решения частных задач. В нашем случае таковыми являются задачи моделирования боевых действий по тушению пожаров в динамике. Программное ядро выступает как обособленная часть, связанная с оберткой посредством COM.

Важно отметить, что язык VBA не позволяет в полной мере реализовать парадигму объектно-ориентированного программирования. Поэтому, вместо абстрактных классов в разработке использованы интерфейсы. Основной смысл интерфейсов заключается в том, чтобы предоставить разработчикам возможность однозначного описания функционала, которым должен обладать объект, для того, чтобы иметь возможность быть использованным в том или ином контексте. В нашем случае это подразумевает, что в качестве команд могут быть использованы любые объекты, реализующие интерфейс ICommand, в качестве компонентов – IComponent, и в качестве игровых объектов IGameObject.

Такой подход хоть и не столь гибок, как использование абстрактных классов, но все же позволяет реализовать механизм создания программной обертки. Подразумевается, что пользователь может как использовать уже имеющиеся в ядре классы реализующие интерфейсы ядра, так и создавать собственные классы реализующие эти интерфейсы. Это необходимо в тех случаях, когда базового функционала классов ядра не достаточно для реализации частных задач моделирования. Например, это необходимо при разработке системы прокладки рукавных линий.

Заключение

Опыт создания ядра системы динамического моделирования боевых действий по тушению пожаров показал обширные перспективы создания такого программного обеспечения. Например, система может быть использована для визуализации хода тушения пожаров, как инструмент повышения наглядности описаний реальных пожаров. Так же система может быть использована в качестве интерактивного тренажера для РТП, позволяя вырабатывать и закреплять у них навыки принятия решений по тушению пожаров в различных условиях. Наконец, подобная система может быть применена как инструмент прогнозирования обстановки на пожарах при составлении документов предварительного планирования боевых действий по тушению пожаров.

Кроме того, полученная система с легкостью может быть использована и в задачах, никак не связанных с пожарной охраной. Например, при анимировании данных в диаграммах Visio. В настоящее время подобные механизмы для данного приложения отсутствуют. Вместе с тем, существующие тенденции разработки программного обеспечения для визуализации данных говорят о том, что анимация является одним из трендов в инфографике и разработке пользовательских интерфейсов. А значит, программное ядро, позволяющее решать подобные задачи может пользоваться спросом среди пользователей системы Visio в целом.

Литература

1. Приказ МЧС РФ от 21 ноября 2008 г. N 714 "Об утверждении Порядка учета пожаров и их последствий"
2. Metanit.com Сайт о программировании, Команда (Command) [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/patterns/3.3.php, свободный – (20.10.2019) Геометрическая модель фазовых переходов (задача о перколяции) Виноградов Д.В., Поршнев С.В. http://www. exponenta.ru/educat/systemat/porshnev/perokol/main.asp