КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

УДК 004.923

И.И.ВЕРЕЩАГИН

ОРГАНИЗАЦИЯ АНИМАЦИОННЫХ ФИЛЬМОВ В ГИБКИХ ТРЕНАЖЁРАХ

Показана актуальность организации анимационных фильмов в гибких, т.е. перепрограммируемых, тренажёрах для развития у обучаемых интеллектуальных навыков. Рассмотрены основы денотационной модели организации анимационных фильмов в качестве универсальной альтернативы обычным видеофильмам.

В течение ряда лет коллектив отдела №160 Международного научно-учебного центра информационных технологий и систем НАН Украины проводит исследования в рамках комплексной проблемы синтеза гибких тренажёров для подготовки оперативного персонала сферы управления. Целью синтеза гибких, т.е. перепрограммируемых, тренажёров является привитие навыков принятия оперативных управленческих решений в обстановке “виртуальной” реальности, “максимально” схожей с той, с которой обучаемый может столкнуться в реальности действительной. За рубежом привитию обучаемым подобных “интеллектуальных” навыков, например, навыков управления фирмой в кризисных ситуациях, в виртуальных образовательных средах никакого внимания не уделяется, не смотря на чрезвычайно большой интерес к виртуальным средам обучения вообще в связи с глобализацией сети Internet [1, 2]. Данный факт объясняется тем, что такие образовательные среды, по мнению зарубежных экспертов, слишком специфичны, а потому требуют неоправданно больших капиталовложений на реализацию и продвижение на рынок при сомнительном потенциальном спросе. Следует также принять во внимание и относительно низкие скоростные возможности сети Internet по сравнению с автономной работой большинства персональных компьютеров, что на сегодняшний день принципиально не позволяет сделать виртуальную ситуацию на экране действительно “живой”. Тем не менее, проведённые нами поисковые исследования и эксперимен-

тальное программирование позволяет утверждать, что общие решения при синтезе гибких тренажёров всё-таки могут быть получены, если ограничиться сферой развития "интеллектуальных" навыков. В этом случае от скорости и реактивности процедур допустимо отказаться в пользу их универсальности и гибкости, что невозможно в тренажёрах, требующих от обучаемого высокой моторики – например, в тренажёрах для подготовки водителей транспортных средств [3].

При разработке инвариантных к реализации принципов структуры и управления гибких тренажёров большое внимание, по нашему мнению, следует уделить организации применения в них так называемых "анимационных" фильмов, как универсальной альтернативе обычным видеофильмам. В настоящей работе приводятся основные результаты исследований данного вопроса. Инструментом формализации была выбрана денотационная семантика Венского метода разработки программ (VDM). Именно для семантики этого типа была убедительно продемонстрирована, на наш взгляд, применимость денотационного подхода для описания совершенно различных компонентов программного обеспечения [4,5], в то время как другие денотационные семантики характеризуются в качестве средств описания лишь языков программирования [6,7].

Организация анимационных фильмов в гибких тренажёрах предполагает разработку двух компонентов программного обеспечения – визуального программатора сценариев фильмов и визуального интерпретатора этих сценариев.

Определение доменов визуального программатора (ВизПр) сценариев анимационных фильмов представлено ниже предложениями (1) – (3):

вп : ВизПр = (эс_сц : Эпиз+)* × фр_эт:ФрЭт – **set** ×
редукц : ФрЭт – m→(ф_редукц:ФрЭс→Ком) (1)

Эпиз ⊂ **union rng** (ФрЭт – m→ФрЭс – **set**) (2)

ком: Ком = иэ : $\mathbf{N_1}$ × ик : Ик × тк : Тк ×
<ип : Ип × пар : Пар – **set**> * (3)

Здесь нет места устанавливать необходимые и достаточные инварианты определённых выше доменов. Неформально отметим только самые основные моменты. Визуальный программатор (1) служит графическим инструментом, посредством которого разработчик курса тренажёра рисует эскиз сценария (эс_сц) анимационного фильма. Эскиз сценария представляет собой непустой список эпизодов (Эпиз+). Каждый эпизод – это один заполненный экран программатора, который создаётся размещением на экране эталонных фрагментов эпизода (ФрЭт) и изменением параметров эталонов (Пар–**set**) в нужном направлении. Пусть предполагаемый эпизод включает разворачивающееся из точки графическое окно с заголовком. Для задания такого фрагмента разработчик курса выбирает из множества эталонных фрагментов эталон "Окно с текстом". Затем эталон мышью размещается в требуемом месте экрана, набирается текст заголовка, задаётся стиль окна, цвет его фона и текста, а также шрифт и кегль заголовка. В заключение определяется анимационный эффект – "Развернуть окно из точки" и скорость, например, 100 мм в секунду. Таким путём эталонный фрагмент эпизода

(ФрЭт) преобразуется в эскиз фрагмента эпизода (ФрЭс) анимационного фильма (2). К визуальному программатору динамически подключается транслятор эскиза сценария, преобразующий эскиз в собственно сценарий анимационного фильма. Такой сценарий представляет собой список команд анимации, записанных в двоичном формате. Для управления генерацией сценария из его эскиза предусматривается отображение эталонных фрагментов эпизодов (ФрЭт) на функции редукции (ф_редукц), каждая из которых преобразует эскиз фрагмента (ФрЭс), созданного на основе соответствующего эталонного фрагмента (ФрЭт), в команду анимации (Ком).

Семантика команды анимации Ком (3) зависит от её имени Ик. Имя команды определяет состав её подкоманд <ип : Ип × пар : Пар **– set**> ∗. Каждая подкоманда, в свою очередь, обладает уникальным именем Ип и набором параметров Пар **– set.** Тип команды – системная, обычная и циклическая – служит только уточняющей характеристикой анимационной команды. В пределах сценария анимационного фильма всякая команда обладает своим номером, который мы называем именем экземпляра команды (иэ : $\mathbf{N_1}$). Экземпляр команды при визуальной интерпретации сценария должен в динамике представить то, что при визуальном программировании называлось эскизом фрагмента эпизода (ФрЭс).

Визуальный интерпретатор сценариев анимационных фильмов служит вторым компонентом, которым в отличие от первого, инструментального, компонента обладает любая версия программного обеспечения гибкого тренажёра. Входной информацией для него является произвольный сценарий анимационного фильма, наглядно интерпретируемый на экране рабочей станцией (автономным компьютером) тренируемого. Интерпретатор начинает работать в определённые моменты времени, назначенные в общем сценарии тренажа, иными словами, по одной из команд общего сценария тренировки.

В программном обеспечении визуального интерпретатора предусматривается создание буфера анимационных команд, принятых к исполнению (Буфер). Для отражения текущей фазы обработки команды анимации интерпретатор создаёт отдельный объект управления (Упр) для каждой команды в буфере. Такой объект может находиться в одном из 5-и состояний (5):

буф : Буфер = Ком **– set** (4)

упр : Упр = Буфер ← m →{СОЗДАТЬ_ОБЪЕКТ,
СОЗДАТЬ_ПОТОК, ВЫПОЛНИТЬ_АНИМ,
ПОДДЕРЖАТЬ_ЦИКЛ, РАЗРУШИТЬ_ОБЪЕКТ}. (5)

Если буфер пуст или в нём находятся только циклически исполняемые команды (о чём более подробно будет сказано далее), визуальный интерпретатор создаёт поток управления для помещения очередной команды анимации (i) в свой буфер. Затем конструируется объект управления для этой команды в состоянии “СОЗДАТЬ_ОБЪЕКТ”:

let ком = <Ком+> [i],
упр = упр ∪ [ком ← m → СОЗДАТЬ_ОБЪЕКТ]
in Буфер := Буфер ∪ ком

Следующий шаг заключается в создании необходимых структур в памяти компьютера и потока управления для команды, только что прочитанной в буфер. Подробно рассмотрим этот вопрос. С семантической точки зрения на множестве анимационных структур (С) визуального интерпретатора задан строгий частичный порядок, при котором утверждение $C_i < C_j$ ($i \neq j$) свидетельствует, что структура C_j включает все функции обработчики для подкоманд <ип : Ип × пар : Пар **- set**> ∗ команды сценария $Ик_j$ (3) – имена экземпляров команд иэ : $\mathbf{N_1}$ для данного имени команды $Ик_j$ во внимание не принимаются. Структура C_j может включать новые функции обработчики, нужные для команды $Ик_j$, или переопределять некоторые из обработчиков структуры C_i, хотя возможны и оба варианта. Таким образом, множество функций обработчиков структуры C_j для команды анимации $Ик_j$ содержит только новые или переопределённые обработчики, тогда как в сценариях анимационных фильмов команды Ком (3) содержат все подкоманды, требуемые для независимого функционирования каждой отдельной анимационной команды сценария. Все сказанное можно формально описать следующими определениями доменов:

$$c_j : C_j = \text{табл} : (\text{ип:Ип} \text{ — } m \rightarrow \text{иф:Иф}) \times \Phi\text{–}\mathbf{set} \times \phi_j , \quad (6)$$

$$\text{Иф} = \text{Пар}\text{–}\mathbf{set} \times \text{Х}\text{–}\mathbf{set} \rightarrow \text{Х}\text{–}\mathbf{set} \in \mathbf{union}\ \Phi\text{–}\mathbf{set} , \quad (7)$$

$$\phi_j = \varnothing \rightarrow \text{табл}(C_{j-1}). \quad (8)$$

Здесь Ф–**set** есть множество обработчиков для команды анимации $Ик_j$. Центральное место в структуре занимает таблица **табл.**, ставящая в соответствие каждому имени подкоманды (Ип) уникальное имя её обработчика (Иф). Функция ϕ_j позволяет получать аналогичную таблицу для низшей структуры в иерархии структур обработки анимационных команд, так что в цепи $Ик_1 < \ldots < Ик_i < Ик_j$ цепь структур обработки $C_0 < C_1 < \ldots < C_i < C_j$ содержит j+1 членов и структура C_0 является минимумом для всех цепей, а также включает “пустую” функцию перехода $\phi_0 = \varnothing \rightarrow \varnothing$, достижение которой сигнализирует о том, что, например, подкоманда $Ип_x$ не предусмотрена в данной команде анимации.

Пусть теперь команда анимации имеет имя экземпляра команды $иэ_j$, а её объект управления носит имя $упр_j$. Тогда визуальный интерпретатор должен сначала создать структуру в памяти ($C_j^{п}$), подчиняясь требованию объекта управления о её создании $C_j \leftarrow m \text{ — } C_j^{п}$, и перевести сам объект управления в очередное состояние $упр_j(иэ_j) \rightarrow$ СОЗДАТЬ_ПОТОК. Подчеркнём, что в памяти компьютера создаются все структуры цепи, а именно: $C_0^{п} < C_1^{п} < \ldots < C_i^{п} < C_j^{п}$.

Недавно упоминавшаяся минимальная структура C_0, кроме “пустой” функции перехода, включает стартовую функцию ($\phi_{старт}$) потока управления исполнением анимационной команды. Поэтому сам поток управления создаётся какой-либо гипотетической процедурой операционной системы (обозначим её знаками @@) с передачей ей имени стартовой функции и списка параметров для всех подкоманд экземпляра анимационной команды $иэ_j$:

@@Создать_поток($\phi_{старт}$, <ип : Ип × пар : Пар **- set**> ∗).

Стартовая функция начинает работу с просмотра таблиц **табл** (6), беря за основу структуру $C_j^п$. Найдя нужную подкоманду, функция $ф_{старт}$ вызывает её обработчик из набора Ф – **set**. Если анимационная команда не является циклической, её объект управления переводится в состояние ВЫПОЛНИТЬ_АНИМ. В этом случае после обработки всех подкоманд команды $иэ_j$ объект управления переводится в состояние РАЗРУШИТЬ_ОБЪЕКТ, и визуальный интерпретатор выполняет такое действие, а затем создаёт поток управления для чтения новой команды сценария в буфер. Если команда циклическая, например, бегущая строка, то её объект управления переводится в состояние ПОДДЕРЖАТЬ_ЦИКЛ. С этого момента визуальный интерпретатор также создаёт поток управления для помещения очередной команды в свой буфер. Чтобы разрушить объект циклической команды, требуется отдельная команда анимации. Таким образом, в визуальном интерпретаторе одновременно выполняются несколько потоков циклических команд, объекты управления которых находятся в состоянии ПОДДЕРЖАТЬ_ЦИКЛ, и только один поток управления иного рода. Он либо читает очередную команду сценария в буфер интерпретатора, либо исполняет подкоманды анимационной команды до перевода объекта управления в состояние РАЗРУШИТЬ_ОБЪЕКТ или ПОДДЕРЖАТЬ_ЦИКЛ.

В заключение сделаем несколько разрозненных замечаний, уточняющих основные положения работы.

Во-первых, визуальный интерпретатор поддерживает список копий команд для восстановления своего экрана. Всего выделяется три типа команд анимации Тк (3): системная команда, обычная и циклическая команда. В список копий не включаются лишь системные команды анимации. К ним, например, относятся команда задержки исполнения и команда очистки буфера визуального интерпретатора.

Во-вторых, поясним более подробно сигнатуру функций обработчиков подкоманд анимации (7). Вторым параметром обработчиков и возвращаемым значением является домен Х – **set**. Под этим доменом понимается либо структуры в памяти компьютера, либо экранные структуры, то есть фрагменты эпизодов анимационного фильма.

Следующее замечание относится к линейным размерам, задаваемым при визуальном программировании сценария анимационного фильма. Такие размеры должны вноситься в параметры подкоманд анимации в нормированном виде, например, относительно величины 1024 или 2048, чтобы в ходе интерпретации можно было учесть реальное разрешение экрана компьютера тренируемого и для возможности масштабирования окна с анимационным фильмом относительно полного экрана компьютера.

И последнее замечание касается скорости эффектов анимации. Мы рекомендуем задавать их в метрических размерах относительно секунды. В этом случае скорость анимации можно сделать мало зависимой от быстродействия компьютера и видеоадаптера, если перед началом визуальной интерпретации

оценивать скорости основных графических операций, реализующих анимационные команды, и производить соответствующие поправки.

ОСНОВНЫЕ ВЫВОДЫ:

1. Организация анимационных фильмов в гибких тренажёрах предполагает разработку программного обеспечения двух компонентов – визуального программатора сценариев фильмов и визуального интерпретатора этих сценариев.

2. Рассмотрены основы денотационной модели организации анимационных фильмов в качестве универсальной альтернативы обычным видеофильмом в обучающих системах, ориентированных на тренировку.

3. Изложенные принципы построения денотационной модели организации анимационных фильмов могут найти применение не только в гибких тренажёрах для развития у обучаемых интеллектуальных навыков, но и в системах, где анимация является важным, но тем не менее вспомогательным средством функционирования таких систем.

4. Все расстояния, задаваемые при визуальном программировании, целесообразно представлять в сценариях анимационных фильмов в нормированном виде относительно величин, выбранных в качестве стандарта.

5. Скорости анимационных эффектов целесообразно задавать в метрических размерах относительно секунды, а при интерпретации необходима корректировка с учётом реального быстродействия компьютера и видеоадаптера.

1. *Piet Kommers, Lora Aroyo.* Agents for Constuctivistic Learning in Virtual Realities // УСиМ. – 2002. – №3/4. – С.92 – 102.
2. *Шереметов Л., Усков В.* Виртуальные образовательные среды. Приложение // Информационные технологии. – 2002. – №5. – 24 с.
3. *Верещагін І.І., Динисовець В.Д.* Принципи архітектури гнучких тренажерів для вищих ланок управління // УкрІНТЕІ. – 2002. – №1/2. – С.10 – 13.
4. *Бьернер Д.* Формальное специфирование как экспериментальная наука // Программирование. – 1991. – №6. – С.24 – 43.
5. *Jones C.B.* Systematic Software Development Using VDM, 3-rd ed., Prentice-Hall, 1997, xxii+365 pp.
6. *Вольфенгаген В.Э.* Конструкции языков программирования. Приёмы описания. – М.: АО "Центр ЮрИнфоР", 2001. – 276 с.
7. *Лавров С.С.* Программирование. Математические основы, средства, теория. – СПб.: БХВ–Петербург, 2001. – 320 с.

Получено 15. 06. 2003