Колесов Юрий Борисович
Сениченков Юрий Борисович

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ В КАРТИНКАХ
ИЛИ
РИСУЕМ ПОВЕДЕНИЕ ДИНАМИЧЕСКИХ СИСТЕМ С ПОМОЩЬЮ "MODEL VISION"

Краткое руководство для тех, кто верит, что стоит прочесть книжку "Когнитивная графика - это очень просто!", и можно за один вечер, на "286-компьютере", нарисовать полуторачасовой мультфильм. Если Вы из их числа, то, как говорят теле- и радиоведущие, "Оставайтесь вместе с нами! Мы за пять минут научим Вас рисовать динамические образы сложных физических и технических систем, позволяющие глубже проникать в тайны естествознания". Впрочем, кто его знает, может быть, у Вас и получится. Мы, авторы, ведь тоже из числа тех, кто верит, что можно создать простой и удобный инструмент для графической иллюстрации сложного динамического поведения.

Напомним, что программный комплекс MODEL VISION 2.1 состоит из двух компонентов: "Редактора моделей" и "Аниматора". Как работать с "Редактором моделей", мы "проходили" на прошлом уроке (см. предыдущий выпуск журнала, № 3/4). Сегодня мы рассмотрим несколько простых примеров, позволяющих понять, как можно нарисовать динамический образ изучаемых в школьной физике объектов, используя средства "Аниматора", и как, с его же помощью, нарисовать "ручки", "движки", "кнопки", с помощью которых можно будет влиять на поведение объекта, изменяя значения требуемых параметров модели.

Мы намеренно используем термин "динамический образ", так как хотим подчеркнуть, что главное для таких программ, как MODEL VISION, в отличии от программ, помогающих создавать игровой мультфильм, это умение рисовать простой *графический образ*, движение которого во времени и пространстве подчинено предопределенным моделью *законам движения*. Иными словами, мы хотим наблюдать *динамику* графического *образа*. С одной стороны, это более простая задача по сравнению с рисованием мультфильма. Графические образы изучаемых объектов чаще всего носят эскизный характер. С другой стороны, мы хотели бы как можно больше автоматизировать процесс рисования и не заставлять пользователя работать со сложным графическим языком, а это трудная задача. Необходимо признать, что ни нам, ни авторам других аналогичных программных продуктов еще не удалось найти удовлетворительного решения.

Для того, чтобы понять, как работать с "Аниматором", воспользуемся следующей аналогией. Представьте себя театральным режиссером, решившим зафиксировать в картинках свой спектакль. Спектакль разбит на действия, действия на сцены, сцены на мизансцены. Сфотографируем или зарисуем основные мизансцены, то есть зафиксируем положение действующих лиц в некоторой сцене. Последовательность таких рисунков может рассматриваться как инструкция актерам - как и куда двигаться в данной сцене во время спектакля. Разложим все картинки друг за другом, действие за действием,

сцена за сценой, дополним указанием, что следует говорить, и получим сценарий. Во входном языке "Аниматора" вместо слов "сценарий, дополненный рисунками ключевых мизансцен" используется термин "сценограмма". "Аниматор", используя рисунки мизансцен, оживляет их в соответствии с поведением модели, или, другими словами, связывает движение графического образа с изменением фазовых переменных. Говорить наш пакет не может, но может пояснять поведение титрами, как в немом кино.

СОЗДАНИЕ ДИНАМИЧЕСКОГО ОБРАЗА (ДОПУСТИМЫЕ ЭВОЛЮЦИИ)

При математическом моделировании обычно не применяется весь арсенал графических средств, используемых сегодня для рисования. Введем ряд упрощений, позволяющих называть метод рисования движения в "Model Vision" технической или эскизной анимацией.

- Первое упрощение связано с разделением будущей "живой" картинки на "фон" (декорации) и "образ" (действующие лица). Все, что должно оставаться неподвижным - фон. Фон можно нарисовать, используя любой графический редактор. Все, что движется - образ.

- Вторым упрощением является предположение, что образ - это либо прямоугольный статический графический фрагмент, который можно различными способами "перемещать" относительно фона, либо простой графический примитив - линия, круг, многоугольник. Образ можно нарисовать в том же редакторе, что был использован для изготовления фона, или "вырезать" из уже готовой картинки. Как рисовать графические примитивы, знает "Аниматор".

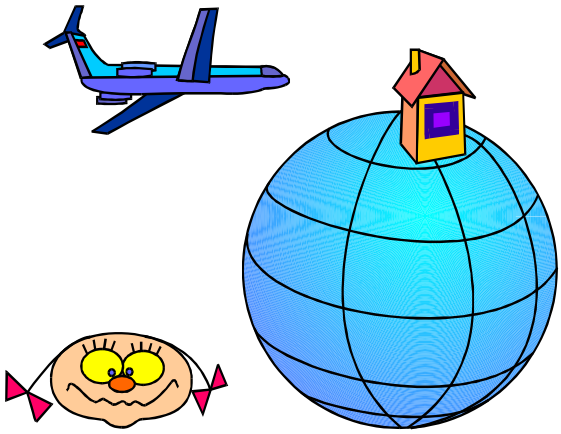

Рисунок 1

- Третье упрощение связано с ограничением числа возможных типов перемещения образов относительно фона (эволюций). Под перемещением будем понимать непосредственное перемещение образа (без вращения) в прямоугольной системе координат, связанной с фоном; появление или исчезновение образа в заданной точке плоскости в зависимости от значения управляющих переменных булевского типа; появление или исчезновение нескольких образов в зависимости от значения управляющих переменных перечислимого типа. Графические примитивы представлены только одним элементом - отрезком.

УПРАВЛЕНИЕ ПОВЕДЕНИЕМ ОБЪЕКТА ПО ХОДУ ЭКСПЕРИМЕНТА (ДОПУСТИМЫЕ ИНТЕРПРЕТАЦИИ)

Когда говорят, что программный продукт создает интерактивную модель, подразумевают, что пользователь получает возможность менять параметры модели непосредственно по ходу эксперимента. Проще всего (с точки зрения реализации) такую возможность предусмотреть заранее, на этапе проектирования модели. Например, предположим, что Вы изучаете закон Ома, и "собрали" (нарисовали с помощью "Аниматора") на экране простейшую схему, состоящую из сопротивления, источника питания, выключателя и измерительного прибора.

Если параметры схемы заданы заранее и используются только эволюции, то Ваши изобразительные возможности весьма ограничены. Создаваемый Вами научно-популярный фильм будет приблизительно таким. Кадр первый - схема выключена, стрелка прибора стоит на нулевой отметке. Кадр второй - после вклю-

чения в заданный заранее момент времени стрелка прибора занимает требуемое положение. Для создания таких фильмов можно использовать средства, предоставляемые "Model Vision" для планирования эксперимента. Очень простой язык планирования эксперимента позволяет менять параметры, входные воздействия, включать и выключать отдельные устройства в заданные заранее моменты. Таким способом можно создавать "пассивные" лабораторные работы, иллюстрирующие поведение объекта в различных условиях, или готовить демонстрационный материал для уроков.

Однако, согласитесь, что если бы Вы предусмотрели возможность менять и сопротивление, и напряжение источника по ходу эксперимента, то созданная Вами модель уже представляла бы почти настоящую лабораторную установку. Можно спорить, могут ли компьютерные лабораторные работы заменить настоящие, но то, что они удачно дополняют их и помогают ученику самостоятельно подготовиться к проведению реальных лабораторных работ - несомненно. Более того, если бы существовал компьютерный банк таких лабораторных работ и эффективно работали глобальные и локальные сети, то насколько эффективнее стал бы процесс обучения!

В "Model Vision" предусмотрены следующие возможности передавать информацию модели (интерпретации) в процессе выполнения:

- создание активных зон экрана, где можно вводить значения вещественных и целых чисел и присваивать их нужным переменным;
- создание различных типов кнопок и связывание их с переменными булевского и перечислимого типа;
- создание аналогов "движков" и связывание положения мыши в таких активных зонах со значениями переменных вещественного типа;
- создание кнопок, управляющих сменой сценограмм.

Специальные окна, где может появляться заранее подготовленный текст, создаются при планировании эксперимента (см. команду "План" в окне созданной модели и соответствующую главу в Руководстве Пользователя).

СОЗДАНИЕ ПРОСТЕЙШЕЙ СЦЕНОГРАММЫ

Тем, у кого установлена полная версия "Model Vision", достаточно вызвать "Аниматор". Это можно сделать, не покидая "Редактора моделей", щелкнув мышью по полю соответствующей команды. Пользователям же журнальной версии необходимо переустановить пакет. Прочтите инструкцию, содержащуюся в файле "Read me", сделайте все, что в ней написано, и Вы получите возможность создавать графические приложения.

Прежде чем начать работу по оживлению картинок, следует создать модель и нарисовать образы и фон. Выберем простейшую модель. Например, давайте перемещать точку по окружности, считая, что это самолет, облетающий планету.

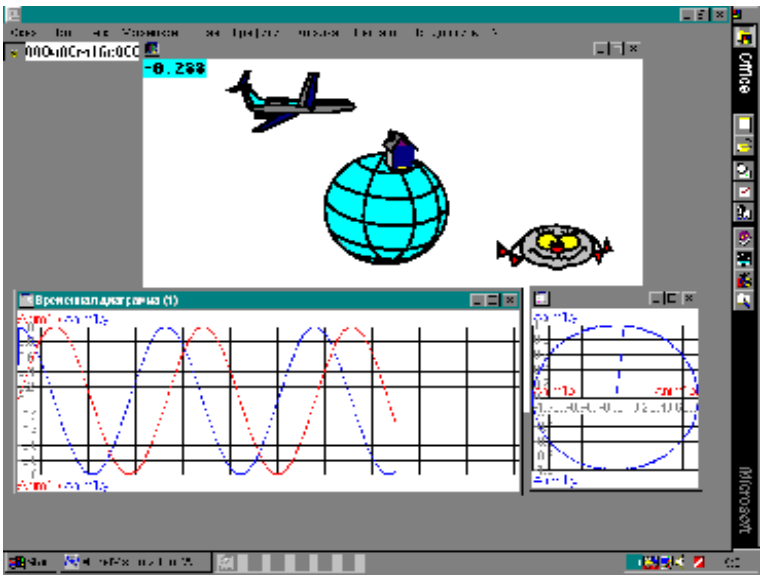

Рисунок 2

Координаты точки на плоскости зададим в виде $x(t)=A*cos(t)$ и $y(t)=A*sin(t)$. Вызовите любой графический редактор, нарисуйте фон (планету) и несколько образов, например, самолет и всплывающий образ, который нам понадобится в дальнейшем (рисунок 1). Поместите файлы, их содержащие, в папку "Idx". Папка "Idx" автоматически создается в тот момент, когда Вы начинаете работать с "Редактором моделей" над новым проектом и дали ему конкретное имя. Она располагается в папке, где хранится Ваш конкретный проект вместе с папками "DOC" и "U".

Для того, чтобы увидеть, как наш самолет облетает планету, создаем сценограмму, базирующуюся на эволюции "Перемещение образа". В главном окне "Аниматора" выбираем команду "Изображения", затем "Образ" и "Фон". В папке "Idx" хранятся все образы для всех сценограмм, и наша цель - выбрать нужную. Выберите требуемый фон (пока у нас он один) и образ самолета. Две дополнительные команды этого же меню, "Координаты" и "Границы", дают возможность увидеть значения координат мыши, когда Вы перемещаете ее в системе координат, расположенной на плоскости "Фон", и размеры прямоугольников, содержащих образы. Начало координат (0,0) расположено в левом верхнем углу, горизонтальная ось *X* совпадает с верхней горизонтальной границей, а вертикальная - с левой вертикальной и направлена вниз. В результате Ваших действий, в главном окне "Аниматора" появятся три дочерних окна: "Эволюции", "Интерпретации" и "Фон". Сделайте активным окно "Эволюции" и выполните команды "Элементы" и "Создать". В открывшемся диалоговом окне последовательно заполните поле "Название" (дайте любое имя создаваемой эволюции) и выберите тип эволюции из предлагаемых в поле "Тип" (в нашем случае "Передвижение образа"). Нажмите на кнопку "Формуляр". Появится окно, которое поможет нам связать значения фазовых переменных с координатами образа. В левой части окна собраны все во-просы о фазовых переменных модели, интерпретируемых как координаты *x* и *y*: имена, их минимальное и максимальное значения. Последние могут быть неизвест-ны заранее, на стадии создания модели. В этом случае необходимо провести эксперимент и нарисовать, например, графики. Окно, рисующее графики, снабжено механизмом автомасштабирования. Можно нарисовать фазовую диаграмму (зависимость *x* от *y*), и тем самым создать "макет" будущего движения образа. В правой части окна расположена соответствующая информация о движении точки (образа) по плоскости "Фон". Выберите требуемый образ и задайте минимальное и максимальное значения координат (другими словами, задайте координаты левого верхнего и правого нижнего угла прямоугольника, в котором будет двигаться образ). Задать коор-

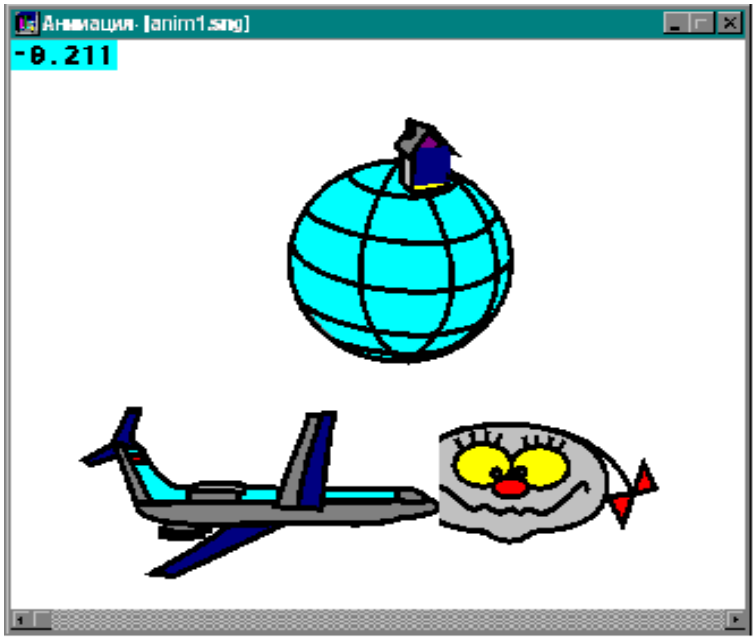

Рисунок 3

динаты углов проще всего так. Нажмите на кнопку "*X,Y* - min", и Вы увидите перемещаемый образ в начале координат плоскости "Фон". Отбуксируйте его мышью в нужную точку экрана. Проделайте то же самое для правого нижнего угла прямоугольника.

Сохраните созданную сценограмму в файле с расширением ".sng" и вернитесь в "Редактор". Поставьте "Редактор" в известность, что Вы создали сценограмму и хотите ей воспользоваться. Для этого последовательно выполняем команды "Установки", "Модель", в появившемся диалоговом окне отмечаем, что созданный рисунок надо оживить, и указываем, какой именно. Далее действуем как обычно: создаем модель ("Создать и запустить") и в главном окне модели выполняем "Окна", "Открыть", а затем помечаем "галочкой" поле "Анимация". Все. Можно нажать кнопку "Пуск".

На рисунке 2 Вы видите изображение экрана с анимационным окном, графиком и фазовой диаграммой. Поле в верхнем левом углу анимационного окна - это еще один вид допустимой эволюции: "Вывод значения" или "живая" таблица. В ней отображаются значения координаты *x*. Цвет фона и шрифт выбираются пользователем.

На рисунке 3 Вы видите моментальный снимок анимационного окна с двумя образами. Второй образ используется эволюцией "Вывод образа" - она аналогична эволюции "Вывод значения", только вывод заданного образа в заданную точку экрана осуществляется, когда некоторая булевская переменная принимает значение "Истина". Введем дополнительную переменную x_negative, булевского типа, принимающую истинное значение при отрицательных значениях координаты *x*. Для этого придется либо определить новую функцию, либо создать дискретный процесс, с двумя состояниями "Неотрицательные_x" и "Отрицательные_x", условиями перехода выбрать выражения $x>=0$ и $x<0$, и мгновенными действиями x_negative:=false и x_negative:=true.

Обратите внимание, что мы "сфотографировали" окно в момент, когда один образ перекрыл другой и стер его часть. Это расплата за простоту. Образы независимы и перерисовывается только фон. В окне хорошо видно также, что перемещается целиком прямоугольник, содержащий образ.

Теми же средствами можно сделать видимым полет шарика, летящего по следующему закону (рисунок 4). Сначала шарик летит с постоянной скоростью V_X на высоте $H=y$, а затем, в заданный момент времени, горизонтальная составляющая скорости V_X становится равной нулю и начинается свободное падение. Долетев до наклонной плоскости (угол наклона задан), шарик упруго отскакивает от нее и дальше начинается полет тела, брошенного под углом к горизонту. Меняя момент времени, когда начинается свободное падение, надо добиться, чтобы шарик попал в “лунку”. В перехваченном с эк-

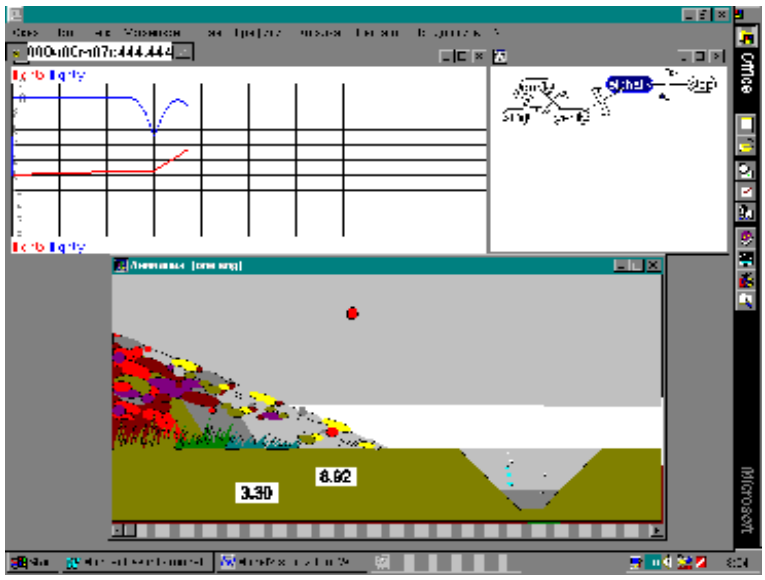

Рисунок 4

рана окне вы видите график (координаты *x* и *y*), карту состояний с состояниями “Старт”, “Горизонтальный полет”, “Свободное падение”, “Полет под углом к горизонту” с выделенным другим цветом состоянием “Полет под углом к горизонту”, что означает, что мы наблюдаем данную фазу моделирования, и анимационное окно.

Обсудим сложности, возникающие при таком способе создания рисованных фильмов. Первая - это согласование систем координат. Обычно координаты, для которых написаны уравнения движения, не совпадают с координатными осями, связанными с фоном. Необходимо вводить дополнительные переменные для согласования. Вторая - все границы изменения переменных должны быть известны заранее. Третья связана с тем, что образ - это прямоугольник, и его нельзя поворачивать, поэтому наш спутник летит “то вперед головой, то вперед ногами”. Можно обратить внимание и на другие трудности, но есть и неоспоримое преимущество - скорость изготовления фильмов. Если не учитывать времени рисования фона и образов или рисовать их так же грубо, как это намерено сделали мы, то на изготовление анимационных окон, приведенных в качестве иллюстраций, уйдет меньше часа. Это много по сравнению со временем создания модели только с графиками, которые тоже, вообще говоря, являются примером анимационных окон, так как изображающая точка движется и оставляет видимый след. Это мало, если сравнивать со временем программирования на каком-либо процедурном языке. Заметим, что если средств, предоставляемых аниматором, не хватает, то можно подключить к модели и “внешнюю” анимацию.

Заканчивая рассмотрение возможностей “Аниматора”, приведем пример интерпретации.

Начнем с простейшего случая. В рассмотренном нами примере полета шарика свободный полет начинается в заданный заранее момент времени. Предположим, что мы хотим управлять моментом начала свободного падения. Для этого создадим в анимационном окне кнопку, нажатие которой прерывает горизонтальный полет.

Прежде всего внесем изменения в модель. В карте состояния дискретного процесса изменим условие перехода из состояния “Горизонтальный полет” в “Свободное падение” с Stime>t1 на *go=two*. Новая переменная *go*, перечислимого типа, имеет два значения: *one* и *two*. Значение *two* соответствует свободному полету. Теперь можно перейти к созданию кнопки. Подготовительная работа заключается в том, что надо внести изменения в фон. Нарисуем на старом фоне кнопку, желательно в виде прямоугольника, в нужном месте. Создадим еще два образа - один совпадающий по размерам с кнопкой на фоне, изображающий ее нажатой, а второй - отпущенной. Делаем активным окно интерпретаций, выбираем новую интерпретацию, даем ей имя, выбираем тип “Кнопка”, связываем с переменной *go* (указываем, что при нажатии кнопка должна принять значение *two*) и совмещаем координаты активного поля, первого и второго образов.

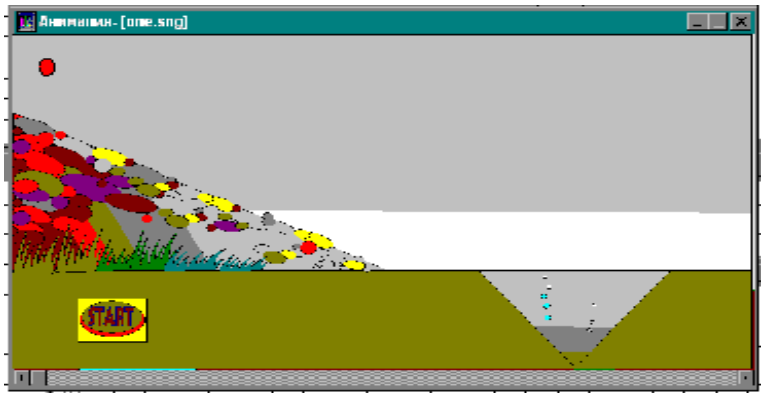

Рисунок 5

Чуть более сложным является способ создания “движков”, то есть актив-

ных зон экрана, где значения выбранных переменных пропорциональны значениям координат курсора. Стандартная линейка прокрутки на рисунке 5 - пример такого движка. Поставим в соответствие крайнему левому положению движка минимальное положение переменной, крайнему правому - максимальное (этого достаточно, чтобы определить линейное отображение). Можно сказать, что существующая интерпретация “Изменение переменной” является обратной операцией для эволюции “Перемещение образа”, в том смысле, что теперь координаты курсора пересчитываются в координаты выбранных переменных. Именно это и позволяет создать видимость передвижения движка. Необходимо одновременно создать и интерпретацию “Изменение переменной”, и эволюцию “Перемещение образа”, перемещающую флажок движка в виде выбранного образа относительно фона, на котором нарисован движок. Как мы видим, наша интерпретация позволяет создавать и горизонтально расположенные движки, и вертикально, и вообще использовать для этой цели всю площадь активного прямоугольника.

В нашем случае движок можно применить для управления значением величины *g*, ускорения свободного падения. (Не забудьте изменить статус *g*, в предыдущем случае естественно было считать *g* константой, теперь она должна стать входной переменной). Красть грешно. Заимствовать, ссылаясь на источник, и лучше имея разрешение, можно. Поэтому наш Вам совет - берите в качестве прообраза стандартные кнопки и движки, перехватывая понравившиеся вам изображения с экрана, и модифицируйте их по своему вкусу. На рисунке 6 показан пример сложного движка для управления тремя переменными с одновременным отображением и положения движка, и значения переменных. Здесь дополнительно использована эволюция “Вывод значения”.

Мы же можем ограничиться простейшим движком, созданным по описанной выше технологии (рисунок 6).

ЗАКЛЮЧЕНИЕ

Вернемся к нашему шуточному введению. В нем, как мы видим теперь, оказалось достаточно много горькой правды. Мы стремились показать, что отдельные приемы оживления картинок достаточно просты, если рисунки носят эскизный характер. И число таких приемов можно было бы увеличивать и увеличивать, их уже разработано достаточно много. Однако, как хорошо видно из примера, демонстрирующего полет шарика, простого преумножения числа приемов недостаточно. Важно научиться связывать различные способы оживления в единое целое. Хотелось бы, чтобы шарик действительно отскакивал от наклонной плоскости и не “съедал” ее кусок, чтобы рисунок, как и график, содержал бы оси, и отмеренные в этих осях расстояния совпадали бы с расчетными, и многое другое. Все это требует достаточно кропотливых расчетов, что уже превращается в проблему.

В то же время хорошо видно, что достаточно просто решается задача создания интерактивных приложений. Выполняемая модель содержит разнообразные по форме органы управления, и это делает компьютерные лабораторные работы очень похожими на настоящие.

ПОСЛЕСЛОВИЕ

Заканчивая нашу трилогию о математическом моделировании, вычислительном эксперименте и технической анима-

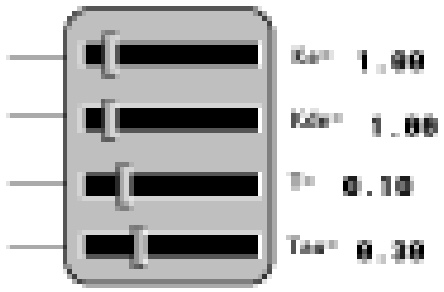

Рисунок 6

ции, хотелось бы еще раз вернуться к главному, ради чего эта работа затевалась. Мы искренне убеждены, что уровень сложности современных программных средств для моделирования физических и технических объектов достиг того уровня простоты эксплуатации, когда ими могут воспользоваться преподаватели как высшей, так и средней школы для решения своих конкретных повседневных задач. Дело по применению интерактивных моделей в обучении сдвинулось бы с мертвой точки (а сейчас реально работающих приложений очень мало), если бы либо появился полноценный учебник с примерами по одному из разделов физики, либо, как крайний случай, появился банк таких приложений, созданных различными авторами. Мы, со своей стороны, готовы помочь, научить и объяснить, как создавать приложения, модифицировать или видоизменить технологию создания приложений, но основная нагрузка должна все-таки лечь на *практикующего* учителя. Мы были бы чрезвычайно рады, если по прочтении этих статей кому-либо захотелось бы самому создать свою собственную компьютерную лабораторную работу (с помощью MODEL VISION, конечно же!).

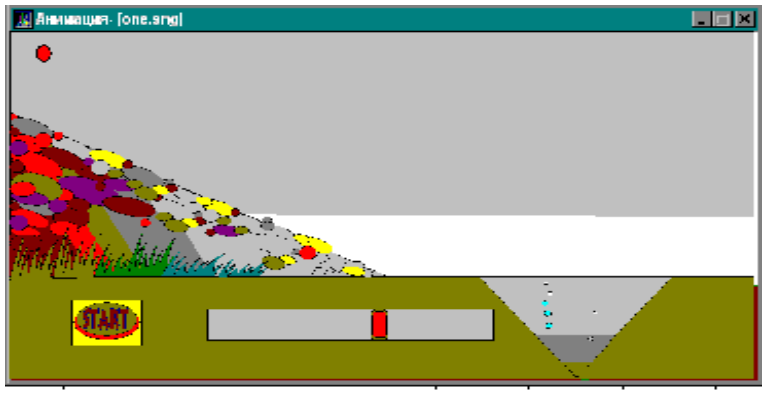

Рисунок 7

НАШИ АВТОРЫ

Колесов Юрий Борисович, кандидат технических наук, руководитель проекта MODEL VISION 2.1, старший научный сотрудник Центра Учебного и научного программного обеспечения ФТК СПбГТУ.

Сениченков Юрий Борисович, кандидат физ.-мат. наук, доцент кафедры РВКС ФТК СПбГТУ.