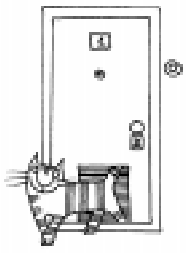
Колесов Юрий Борисович,
Сениченков Юрий Борисович

ШКОЛА МОДЕЛИРОВАНИЯ-2003. ЗАНЯТИЕ 1.



ВМЕСТО ВВЕДЕНИЯ

Авторский коллектив учебника «Информатика» и задачника «Информатика. Задачник по моделированию» под руководством Макаровой Н.В.[1, 2] и разработчики графической среды Model Vision Studium [5] приступают к совместной публикации серии статей, посвященных проблемам школьного обучения. Материалы, которые мы собираемся опубликовать, можно рассматривать как методические указания по использованию и созданию виртуальных лабораторий, разработанных на базе пакета Model Vision Studium. Эти лабораторные работы смогут, как мы надеемся, найти свое применение не только на уроках информатики при изучении раздела «Моделирование», но и на уроках физики и биологии.

Процесс создания любой компьютерной модели условно можно разделить на два этапа. На первом этапе формулируются цели моделирования, и будущая модель описывается с помощью, с одной стороны, достаточно формализованного, с другой стороны, – образного и интуитивно понятного человеку языка. На втором этапе выбирается программное средство, с помощью которого эту модель можно реализовать наиболее эффективным способом, и интуитивное описание переводится на язык пакета. В задачнике [2] его авторы предлагают педагогу на втором этапе использовать для разработки компьютерных моделей средства, предоставляемые Microsoft Office, справедливо полагая, что этот программный продукт доступен практически всем. В то же время, основное назначение Microsoft Office совсем другое, и его «нецелевое» использование может вызывать методические трудности. Поэтому мы и предлагаем использовать на втором этапе Model Vision Studium. Таким образом, преподаватель сможет сравнить два подхода и выбрать любой из них для практических занятий.

Сегодня мы представляем первую статью из этого цикла, посвященную возможностям пакета Model Vision Studium.

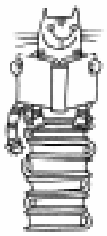
ЧТО НУЖНО ИМЕТЬ К СЛЕДУЮЩЕМУ ЗАНЯТИЮ

Желание и возможность читать книги и одновременно иметь доступ к достаточно неприхотливому компьютеру.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Информатика. 7–9 класс. Базовый курс. Авторский коллектив: *Макарова Н.В., Волкова И.В., Николайчук Г.С., Нилова Г.С., Потягайло А.Ю., Титова Ю.Фр.*, под редакцией *Макаровой Н.В.* СПб.: Изд. «Питер», 2002.

2. Информатика. Задачник по моделированию 7–9 класс. Базовый курс. Авторский коллектив: *Макарова Н.В., Николайчук Г.С., Титова Ю.Фр.,* под редакцией *Макаровой Н.В.* СПб.: Изд. «Питер», 2003.



ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

3. *Горстко А.Б.* Познакомьтесь с математическим моделированием. М.: Знание, 1991.

4. *Самарский А.А., Михайлов А.П.* Математическое моделирование: Идеи. Методы. Примеры. М.: Наука, Физматлит. 1997.

5. *Бенькович Е.С., Колесов Ю.Б., Сениченков Ю.Б.* Практическое моделирование сложных динамических систем. С. Петербург: БХВ, 2001.

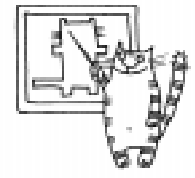
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

1. На сайте http://www.exponenta.ru/soft/Others/mvs/mvs.asp вы всегда сможете найти свободно распространяемую и постоянно обновляемую экспериментальную версию графической среды визуального моделирования Model Vision Studium Free 3.XX (MVS). Размер дистрибутива около 8 Мб. Она же помещена и на прилагаемом к журналу диске.

Программный комплекс MVS распространяется в двух вариантах: в виде стандартной версии и свободно распространяемой, экспериментальной версии для некоммерческого использования. Стандартные версии содержат только проверенные решения, в то время как экспериментальная может содержать компоненты, которые в дальнейшем не будут использоваться в стандартных версиях.

НЕОБХОДИМОЕ ОБОРУДОВАНИЕ

Для установки любой из версий Model Vision Studium на вашем компьютере необходимо иметь 15–20 MB свободной дисковой памяти. Для возможности работы с 3D-анимацией нужна библиотека OpenGL, и, если на вашем компьютере установлена любая Windows старше Windows-95, беспокоиться нечего – эта библиотека входит в состав операционной системы.



БАЗОВЫЕ ПОНЯТИЯ

Для того, чтобы вспомнить основы моделирования, мы рекомендуем вам прежде всего прочесть соответствующие главы рекомендованного учебника «Информатика».

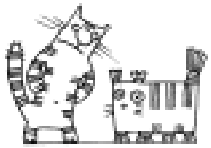
Это займет некоторое время, поэтому кратко напомним, зачем нам нужны модели окружающих нас объектов. И начать следует с исходных объектов, с которыми нам приходится иметь дело. Прежде всего, это объекты материальные, существующие вне нас. Материальные объекты могут в свою очередь быть природными (вулкан) и созданными руками человека (паровой котел). И те, и другие приходится моделировать по разным причинам:

– невозможности провести натурный эксперимент с исходными объектами (вряд ли мы когда-нибудь сможем «подвинуть» Землю, чтобы узнать, как после этого изменятся орбиты других планет или климат Земли);

– опасности работы с реальными объектами (военные, обучая солдат или моделируя их поведение в будущих реальных условиях, предпочитают использовать холостые патроны, но при этом все равно травмы на учениях случаются);

– различного рода преимущества от работы с моделью, а не с исходным объектом (скорость и дешевизна проведения эксперимента с моделью, возможность увидеть и измерить то, что на реальном объекте сделать затруднительно).

При моделировании природные объекты или явления, сложным образом взаимосвязанные между собой, вычленяются из их окружения и заменяются более простыми, но сохраняющими интересующие нас свойства. Собственно моделирование и заключается в умении выделить объекты и связи между ними, выбрать нужные свойства объекта и найти их заменитель, чтобы на основании проделанных с заменителем опытов можно было бы сделать достоверные выводы о поведении моделируемого объекта.



Реальный объект может быть заменен как реальным объектом – и тогда мы чаще всего говорим о макетировании – так и идеальным, в частности, математи-

ческой или компьютерной моделью. Математическая или любая другая абстрактная модель является той отправной точкой, с которой начинается построение компьютерной модели. Заметим, что компьютерная модель, если речь идет о выполняемой программе, не является идеальной (абстрактной) моделью, так как она выполняется на реальном компьютере, к которому может быть подключено реальное периферийное оборудование. Например, легко представить себе робота, управляемого встроенным компьютером.

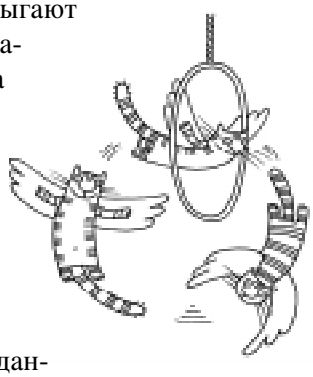
На наших уроках мы в основном будем говорить об идеальных моделях, создаваемых с помощью специальных языков моделирования. Языки моделирования – это специальные языки программирования, обычно графические, то есть имеющие графические аналоги основных своих синтаксических конструкций, представленные на экране дисплея конкретными образами, позволяющие создавать строгие (поддающиеся однозначному переводу на машинные языки) описания моделей различных объектов.

Среди различных языков моделирования выделим особо объектно-ориентированные языки моделирования, ключевыми понятиями которых является класс и экземпляр класса. Что такое класс, известно любому школьнику, даже никогда не изучавшему информатику. В нашем сознании классы занимают важнейшее место. Слова «кошка», «собака», «дом», «школа» – это имена всем известных классов, обобщающих, идеализирующих свойства конкретных Мурок, Шариков, домов, где мы живем, и школ, где учимся (экземпляров класса). Наше сознание прекрасно отличает один класс от другого не столько по названиям, сколько по их свойствам, в том числе, таким, как поведение (действие) или графический образ, ассоциирующийся с классом.

Объектно-ориентированные языки также оперируют с классами и их экземплярами, и они понятнее человеку, чем, например, процедурные языки, состоящие из чуть «облагороженных» машинных команд.

Одним из графических объектно-ориентированных языков моделирования является язык Model Vision Language (MVL), используемый в графической оболочке визуального моделирования Model Vision Studium [5].

Прежде чем разбираться с классами и экземплярами конкретного языка моделирования, попробуем чуть более формально определить интуитивное представление класса. В качестве моделируемого явления выберем цирковой номер, и будем наблюдать, как кошки под управлением дрессировщика прыгают через обруч. Создадим класс «Кот на арене», где кот характеризуется весом, цветом, «силой прыжка» и умеет прыгать по команде «Апп!» через обруч, расположенный на заданной высоте, а по команде «Домой» заканчивать выступление. Мы хотим создать компьютерную модель, которая позволяла бы нам выбирать конкретную кошку, подавать ей команду и наблюдать ее полет через обруч.



ПОДГОТОВКА К СОЗДАНИЮ КОМПЬЮТЕРНОЙ МОДЕЛИ

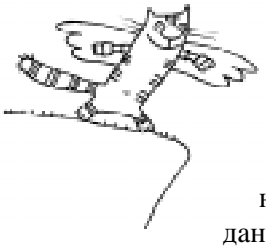
Этап 1. Выбор и описание основных переменных. Прежде всего создадим таблицу (таблица 1), где перечислим все свойства исходного объекта (моделируемого явления) и их представление на языке математики. Свойства объекта могут быть статическими (не зависящими от времени) и динамическими (зависящими от времени). Выберем имена переменных, которые будут характеризовать свойства

Таблица 1.

Название класса «Кот на арене»	
константы	
параметры	
вес	Значение по умолчанию: 8 (кг)
цвет	Значение по умолчанию: рыжий
переменные	
поведение	

выступающего кота – а именно его «вес» и «цвет». Очевидно, что переменную «вес» следует считать вещественной переменной. Переменную «цвет» – переменной перечислимого типа, с возможными значениями цвет={белый, черный, рыжий}.

Этап 2. Конструктивное описание поведения (методов). С точки зрения динамики полета, кота в первом приближении можно считать материальной точкой массы m, брошенной под углом к горизонту с заданной начальной скоростью. Предположим, что угол полета и начальную скорость каждый кот «выбирает сам», то есть при каждом новом полете пусть это будут некоторые случайные числа из заданного диапазона.



Таким образом, для описания полета нам понадобятся новые вспомогательные переменные – «угол_полета (alpha)», «координата_x (x)», «координата_y (y)», «горизонтальная_скорость (Vx)», «вертикальная скорость (Vy)», «начальная скорость (V0)». Переменные «высота» (h) и «расстояние» (L) указывают, на каком расстоянии от начала координат и на какой высоте расположена нижняя точка обруча. Вы в праве выбирать любые имена – «содержательные», или общепринятые математические, указанные в скобках. Переменную «команда» мы обсудим ниже.

Теперь становится понятным, почему мы некоторые переменные охарактеризовали как константы, а некоторые как параметры. Константы не будут менять своего значения никогда в наших экспериментах (цирковых представлениях), а параметры будут оставаться неизменными только на протяжении одного эксперимента (выступления конкретной кошки).

Мы оставили без пояснения графу *поведение*. Что означает карта поведения? Казалось бы, описать полет кота с помощью уравнений для тех, кто помнит школьную физику, не составляет труда. Но мы хотим подчеркнуть, что имеем дело с более сложным поведением, чем просто полет кота – а именно, полет начинается по команде «апп», заканчивается приземлением, после чего кот произ-

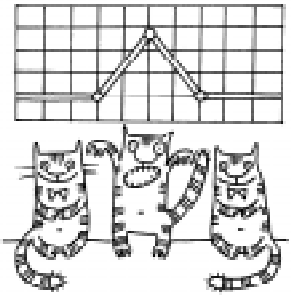


Таблица 2.

Название класса: «Кот на арене»	
константы	
g	Значение: 9.81 (м/сек^2)
x0	Значение: 0 (м)
y0	Значение: 0 (м)
параметры	
вес	Значение по умолчанию: 8 (кг)
высота (h)	Значение по умолчанию: 1(м)
цвет	Значение по умолчанию: рыжий
Расстояние (L)	Значение по умолчанию: 2 (м)
переменные	
команда	начальное значение: апп
x	начальное значение: x0 (м)
y	начальное значение: y0 (м)
Vx	начальное значение: 0(м/сек)
Vy	начальное значение: 0(м/сек)
V0	начальное значение: 0(м/сек)
alpha	начальное значение: 0(м/сек)
поведение	Карта поведения: Кот на арене.

вольно долго ожидает либо новой команды «апп» и повторяет полет, либо команды «домой», после чего представление заканчивается. Для этого и введена переменная «команда» перечислимого типа со значениями команда={нет_команды, апп, домой}.

Поведение кота на арене состоит из длительного ожидания команды, при этом никакие переменные не меняются, и длительного полета до момента приземления ($y \le 0$ И $Vy < 0$), когда переменные x, y, Vx, Vy меняют свои значения, подчинясь хорошо известным физическим законам:

$$\begin{aligned} x(t) &= x_0 + V_x \cdot t, \\ y(t) &= y_0 + V_y \cdot t, \\ V_x &= V_0 \cdot \cos(\alpha), \\ V_y &= V_0 \cdot \sin(\alpha) - g \cdot \frac{t}{2}. \end{aligned} \qquad (1)$$

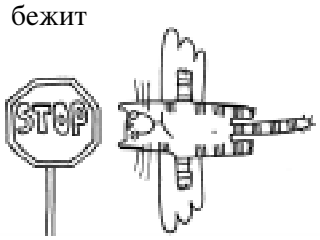
Поведение кота можно изобразить графически, с помощью карты поведения. Карта поведения – это рисунок, на котором изображаются длительные действия в виде прямоугольников с сглаженными краями, и стрелок, указывающих последовательность смен поведения (рисунок 1). Длительные действия (прямоугольники): «Полет», «Ожидание», «Конец (представления)» получают свои имена, а над стрелками пишутся условия окончания длительных действий : «Приземление», команда «Апп», команда «Домой».

Теперь можно каждому длительному действию сопоставить математический закон длительного действия в виде уравнений (таблица 3).

Таблица 3.

Название длительных действий	**Формулы или уравнения**
Полет	Формулы (1)
Ожидание	Null
Конец представления	Stop

Слово NULL воспринимается пакетом как указание на то, что данное длительное действие не связано с изменением значений переменных модели, и меняется только время. Как и в реальной жизни, время всегда бежит вперед, даже если мы ничего не делаем. Команда STOP говорит о том, что эксперимент завершен.



Следует также перечислить все условия окончания длительных действий и перечислить все подготовительные операции, необходимые для начала новых длительных действий (таблица 4).

ПЕРЕВОД ОПИСАНИЯ МОДЕЛИ НА ЯЗЫК MVL

Вызовите программу MVS (мы воспользовались версией MVS Free 3.2) и создайте новый проект с помощью команд основного меню – Проект/ Новый. Назовем его «Урок_1».

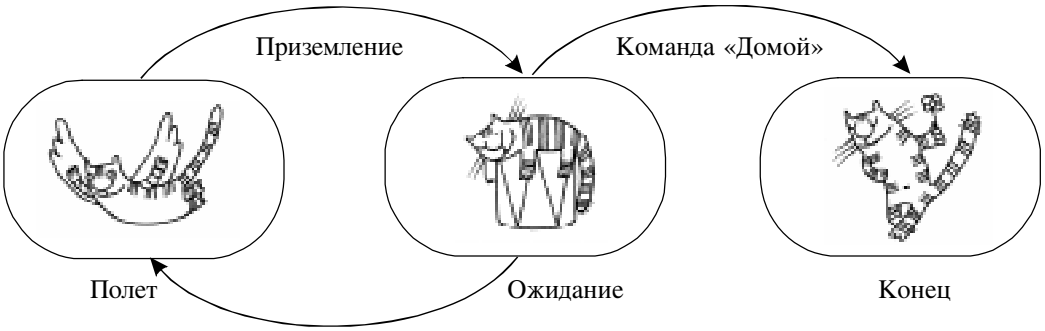
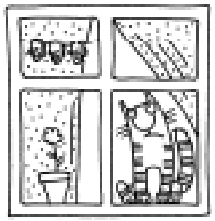


Рисунок 1. Карта поведения «Кот на арене».

Таблица 4.

Условия окончания действия	**Подготовительные операции к новому длительному состоянию**
Полет: y ≤ 0 И Vy < 0	Ожидание: Занять исходную позицию (x0,y0)
Ожидание: команда «Апп»	Полет: Вычислить новые значения V0, Vx0,Vy0 и alpha
Ожидание: команда «Домой»	Конец: нет



Перед вами откроется основное окно проекта с четырьмя дочерними окнами:

– окно «Проект», содержащее имя проекта и список разработанных классов,

– «Класс», в данном случае класс «Кот на арене»,

– «Уравнения»,

– и «Виртуальный стенд», в котором автоматически размещен один экземпляр класса «Кот на арене»_1, с параметрами, заданными по умолчанию.

Следуя руководству пользователя, которое вы найдете на диске, заполните все поля окна «Кот на арене», и вы получите вот такое описание класса (рисунок 2).

Для задания формул и уравнений в пакете предусмотрен специальный текстовый редактор (рисунок 3).

Карта поведения выглядит, как показано на рисунке 4, и на ней появился новый прямоугольник с именем Init, назначение которого мы не обсуждали. Ему не предписана ни система уравнений, ни условия окончания. Это означает, что мы сразу же попадем в длительное состояние «Ожидание». Остальная часть карты полностью совпадает с нашим рисунком из подготовительной части.

Осталось только объяснить назначение переменной LocalTime, появившейся

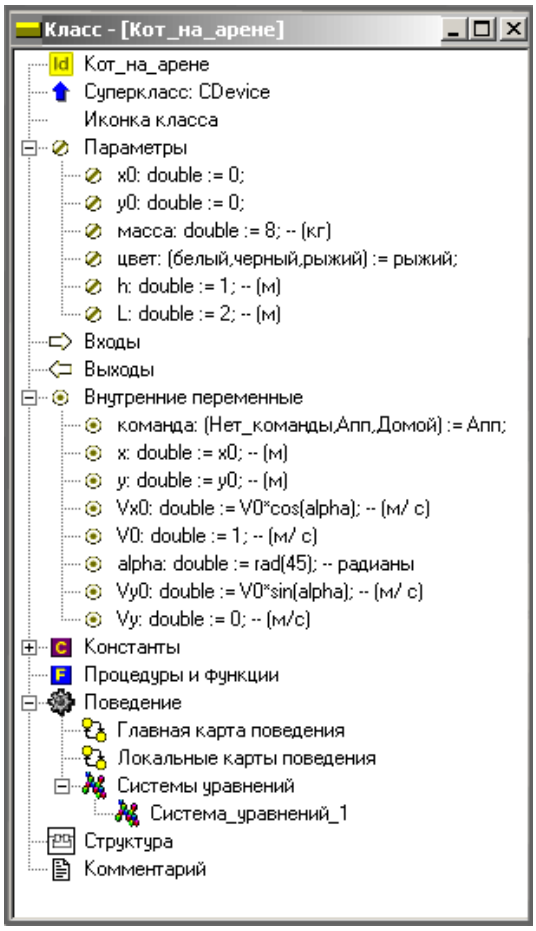


Рисунок 2. Класс «Кот на арене».

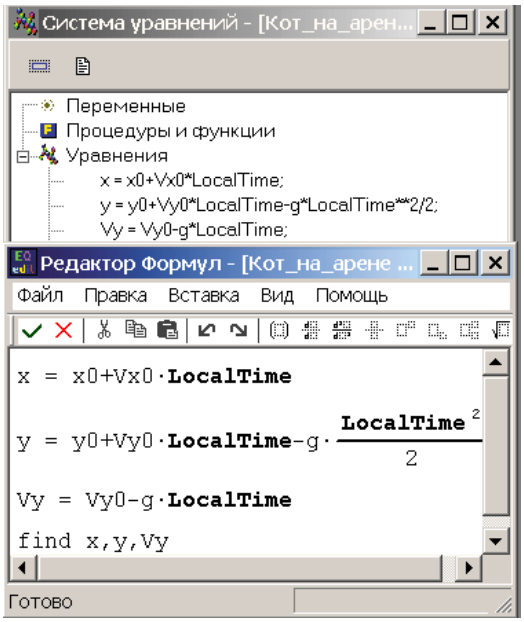


Рисунок 3. Система уравнений и графический редактор уравнений.

в уравнениях и нигде не описанной нами. Дело в том, что в пакете для глобального времени или модельного времени используется специальная предопределенная переменная Time, которая и отображается в специальном окне при исполнении модели. Как только нажмете кнопку «Старт» и начнете эксперимент, время начнет меняться непрерывно. С его помощью вы можете узнать, когда поступила первая команда «Апп», сколько длился любой прыжок каждого кота. Нам же для моделирования отдельного полета нужно время выдачи очередной команды «Апп», у нас уравнения полета на самом деле написаны не для глобального времени эксперимента, а для локального, начало отсчета совпадает с временем выдачи команды. Это и есть локальное время длительного состояния «Полет».

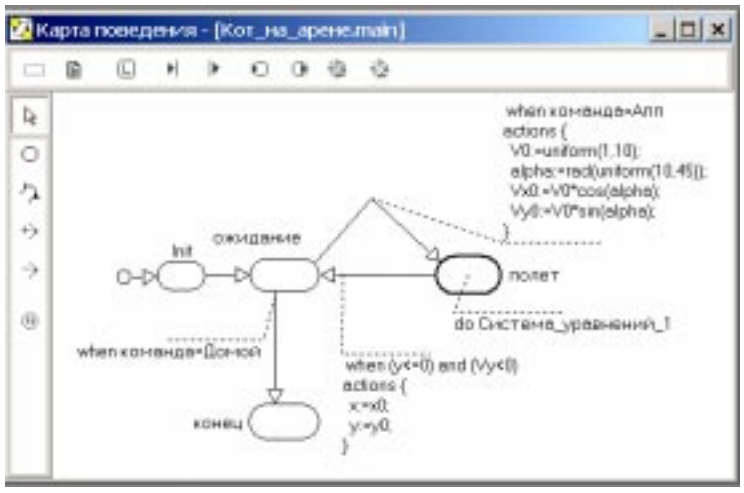


Рисунок 4. Карта поведения класса «Кот на арене».

Сравнив наши подготовительные действия и их запись на языке MVL, вы видите, что имеете дело с языком высокого уровня, синтаксические конструкции которого практически совпадают с нашим неформальным описанием модели. Мы привели эти описания еще и для того, чтобы показать, насколько современные языки далеко ушли от процедурных языков.

ПОДГОТОВКА МОДЕЛИ К ВЫПОЛНЕНИЮ

Прежде чем переходить к исследованию модели, необходимо составить хотя бы один тестовый пример, подтверждающий правильность нашей программы. Хорошо бы увидеть только один полет, с заданными значениями начальной скорости и углом, по умолчанию равными V0=1, alpha=45(градусов).

Для этого нам надо видоизменить программу. Воспользуемся преимуществами объектно-ориентированного подхода и создадим прямого потомка от класса «Кот на арене», назвав его «Тест» (см. каталог «Урок_ 1_Тест»).

У созданного потомка достаточно изменить только карту поведения (рисунок 5).

Здесь два отличия – закомментированы операторы, присваивающие значения начальной скорости и углу перед

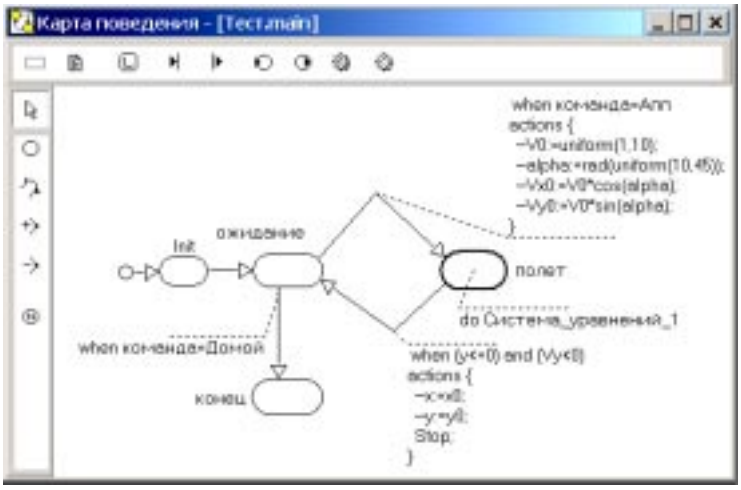


Рисунок 5. Тестирование модели.

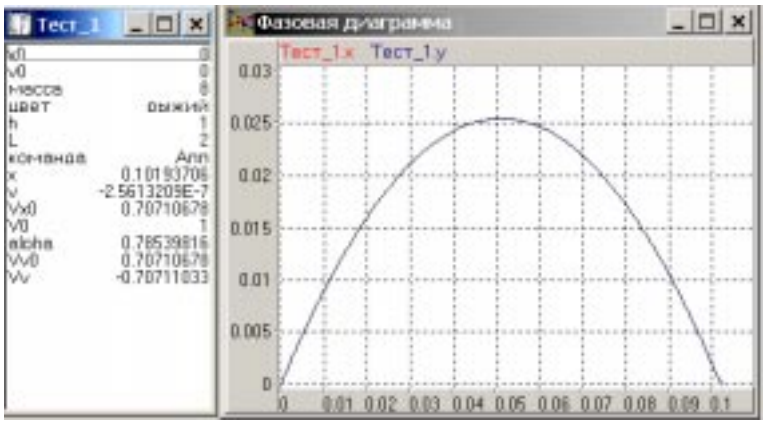


Рисунок 6. Результаты тестирования.

очередным прыжком и, таким образом, они принимают нужные нам значения «по умолчанию». Второе изменение касается окончания работы модели – теперь эксперимент заканчивается сразу же после первого прыжка.

Выполним сначала именно эту модель. Откомпилируем ее с помощью команды основного меню Модель/Пуск Редактора Моделей, либо нажав на кнопку, изображенную в виде зеленого треугольника. В результате откроется окно Испытательного стенда. Мы заранее подготовили два окна, где будут отображаться результаты (рисунок 6) – «Живая таблица» (слева) и «Временная диаграмма) (справа).

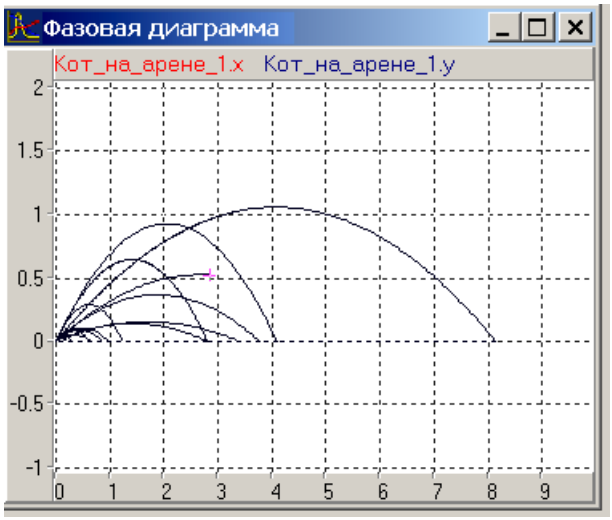
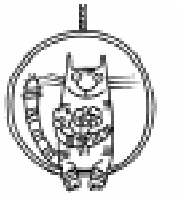


Рисунок 7. Без устали прыгающий кот.

Результаты эксперимента мы увидим на графике, где полет кота изображен в осях (x,y) – по горизонтальной оси отложена дальность полета, по вертикальной – высота. В таблице значения переменных соответствуют моменту касания земли.

Убедимся, что ответы правильные. Кот, как показывают вычисления, приземлится через t=0.14416040 секунд. Посмотрите на часы Испытательного стенда (расположенные в левом верхнем углу), и вы увидите, что время замерло на значении 0.14416072. При этом y=-2.57E-7, а Vy=0.70711033. Отрицательное значение высоты не должно вызывать удивления, так как условия окончания полета заданы в виде неравенств, которые вычислялись с некоторой, отличной от нуля точностью. Так что надо признать, что полет кота прошел успешно.



ВЫПОЛНЕНИЕ МОДЕЛИ

Теперь можно вернуться к основной модели, а именно открыть модель из папки Урок_1, откомпилировать ее и провести эксперименты на Испытательном стенде. Не забудьте только, что пока наш кот прыгает, не останавливаясь, так как значение переменной «команда» всегда равно «Апп». На панели управления Испытательным стендом есть кнопка, изображенная в виде двух вертикальных прямых, расположенная рядом с кнопкой «Старт». Эта кнопка и может остановить кота в любой момент, когда вы захотите. Результаты многократных прыжков изображены на рисунке 7. Как хорошо видно из рисунка, начальные скорости выбира-

ются нами неудачно, поэтому высота прыжков редко превышает 1 метр, где расположена нижняя кромка обруча.

УПРАВЛЕНИЕ ПРЫЖКАМИ

Наша модель пока еще не соответствуют предъявленным к ней требованиям, так как мы не можем подавать команды коту. В папке «Урок_1-_Управление» подготовлена еще одна модель, где на испытательном стенде вы найдете две конки «Апп» и «Домой». Нажимая их, вы имитируете подачу команд. Экспериментируя с этой моделью, не забывайте сначала нажимать кнопку «Старт» на испытательном стенде, а потом уже подавать команды. Обратите внимание, что в этой модели снова изменена карта поведения, и появились новые переменные К_Апп и К_Домой булевского типа. Это связано с тем, что кнопка данного типа может «работать» только с булевскими переменными.

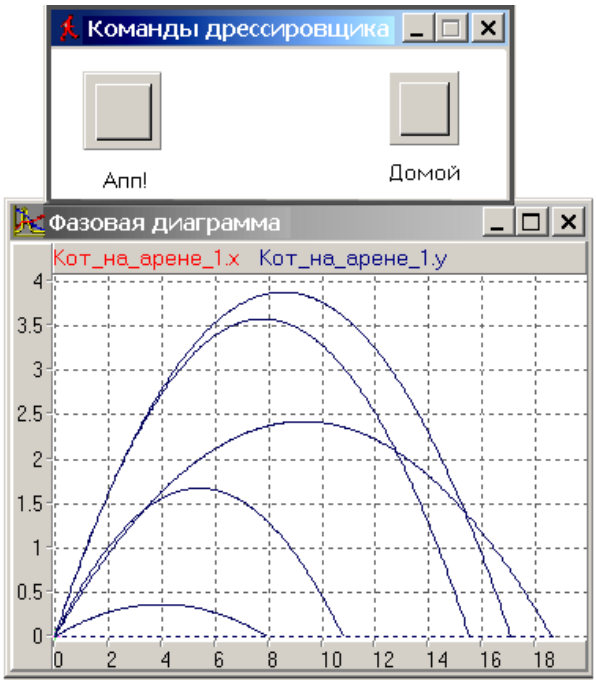
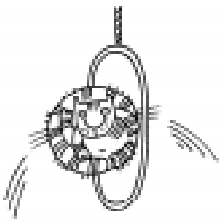


Рисунок 8. Панель управления, имитирующая команды дрессировщика.

ДИНАМИЧЕСКИЙ ОБРАЗ

И, наконец, добавим в модель трехмерную анимацию. В папке «Урок_1_3D» вы найдете еще одну модель. Хотя возможности трехмерной анимации и ограничены, однако, если вместо кота взять шар требуемого цвета, то наш цирковой номер будет выглядеть так, как показано на рисунке 9.



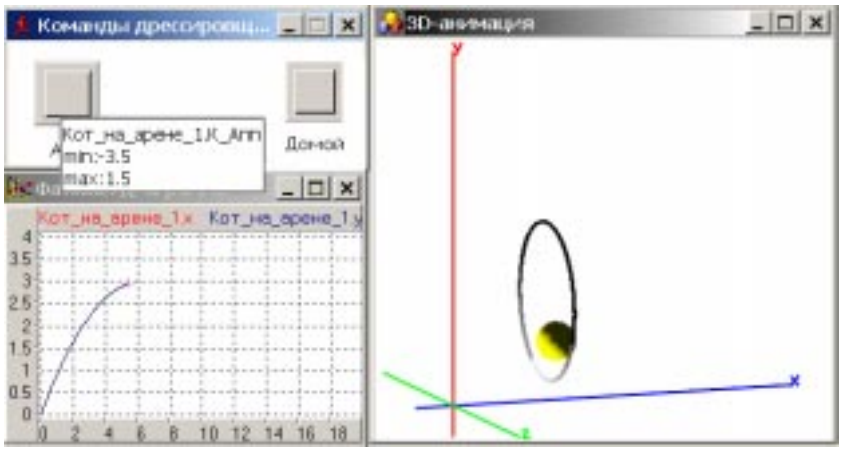
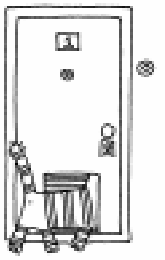


Рисунок 9. Полет кота-шара.

ВМЕСТО ЗАКЛЮЧЕНИЯ



В данной статье мы продемонстрировали все этапы создания виртуальной лаборатории и ее основные возможности. Мы надеемся, что, прочитав ее, вы сможете оценить, стоит ли вам учиться самим создавать модели с

помощью Model Vision Studium, или будет достаточно только использовать готовые модели, которые вы найдете в последующих номерах журнала.

ДОМАШНЕЕ ЗАДАНИЕ

Начните читать учебник «Информатика», рекомендованный вам, и попробуйте самостоятельно создать Толковый Словарь основных терминов. Приведите примеры моделей, которые кажутся вам наиболее интересными и наглядными, наиболее полно демонстрирующими возможности моделирования как инструмента познания окружающего нас мира.

Инсталлируйте программный комплекс MVS. Попробуйте выполнить модели из всех папок. Начните читать Руководство Пользователя.

ТОЛКОВЫЙ СЛОВАРЬ

Модель.

Физические и Идеальные модели.

Математическое и компьютерное моделирование.

Объект-модель.

Класс-экземпляр.

Колесов Юрий Борисович,
кафедра Распределенных Вычислений и Компьютерных Сетей Санкт-Петербургского Политехничекого Университета.

Сениченков Юрий Борисович,
кафедра Распределенных Вычислений и Компьютерных Сетей Санкт-Петербургского Политехничекого Университета.