Моим детям:
Сергею,
Веронике,
Димитрию

Предисловие

О чем эта книга

Основная цель этой книги — помочь читателю сделать свои первые шаги в объектно-ориентированном программировании (ООП). Я предлагаю это осуществить путем последовательного практического изучения двух языков программирования: Squeak и Java. Что такое Java, почти все знают. А вот что такое Squeak? Осмелюсь предположить, что большая часть читателей в лучшем случае лишь слышала об этом языке программирования. В России Squeak практически неизвестен. Возникает законный вопрос, чем же тогда вызван такой странный выбор автора? Я не буду анализировать причины почти полной безвестности, как Squeak, так и Smalltalk в нашем отечестве, скажу лишь о степени их известности за рубежом.

Два знаменательных события, произошедших недавно, возвращают нас к тому времени, когда словосочетание «объектно-ориентированный» было только что придумано. Сделал это Алан Кэй, работавший в 70-е годы XX столетия в исследовательском центре фирмы Xerox в Пало Альто. Там под его руководством был изготовлен прототип современного персонального компьютера с графическим интерфейсом «Динабук», там же им была предложена идея организации вычислительной среды на основе «клеток» — самостоятельных взаимодействующих объектов. Эта идея была реализована командой Алана в объектно-ориентированном языке программирования Smalltalk. Первое из событий, о которых идет речь, — присуждение Алану Кэю премии Тьюринга за 2003 год, которая считается своего рода Нобелевской премией в области вычислительной техники. Второе событие произошло 24 февраля 2004 года, когда была вручена почетная премия Чарльза Дрэпера, присуждаемая Национальной Академией Инженеров США. Одним из четырех лауреатов в этот день был Алан Кэй. «Эти четверо лауреатов составляли ядро замечательной группы инженерных умов, перевернувших наш взгляд на суть и назначение компьютеров», — сказал президент Академии Вильям Вулф.

Этих высоких наград Алан Кэй удостоился, в частности, и потому, что, во-первых, Smalltalk был первым языком программирования, в котором полностью, последовательно и очень выпукло была реализована концепция объектно-ориентированного программирования; во-вторых, Smalltalk как язык очень тесно связан со средой программирования, которая воплощает идею персональных вычислений. До сих пор Smalltalk служит источником идей, питающих развитие других языков программиро-

вания и связанных с ними технологий, поэтому Smalltalk остается вполне современным, несмотря на свой почтенный (с точки зрения технологического прогресса) возраст.

Squeak является одним из простейших, если не самым простым объектно-ориентированным языком программирования. Причем эта простота не делает его ущербным с точки зрения выразительных средств и реализованных технологий: в нем есть всё — от 3D-графики до встроенного веб-браузера. Именно поэтому во многих университетах США концепции ООП изучают на основе данного языка программирования. Опыт преподавания программирования показал также, что изучение Squeak или других версий Smalltalk воспитывает хороший стиль, расширяет сознание. Именно поэтому Smalltalk, особенно Squeak, прочно занимает вполне определенные позиции в нишах образовательных технологий и исследовательского программирования (см. http://www.squeakland.org/).

Возможны разные подходы к обучению программированию: вначале научить составлять алгоритмы, а затем учить принципам ООП; вероятно, можно поступить наоборот. В этой книге сделана попытка «убить двух зайцев»: поскольку изучение Squeak заставляет пользоваться объектами даже при решении простейших задач, обучаемый волей-неволей осваивает и ООП. Если обучаемый не испытывает отвращения к программированию, цель оказывается достигнутой. Об этом свидетельствует мой опыт преподавания соответствующих дисциплин: книга написана на основе курса лекций по основам программирования, который автор читал на математическом факультете Московского городского педагогического университета в 1998–2002 годах, а также спецкурса, прочитанного на факультете ВМиК МГУ осенью 2001 года.

И, наконец, хотелось бы сказать еще об одном аспекте этой книги. Обе среды, избранные в качестве инструментов обучения, — Squeak и Eclipse относятся к категории open source. В двух словах это — свободное распространение самого продукта, доступность кода, право вносить изменения в этот код, право продавать продукт, созданный на основе open source и некоторое другое (см. http://www.opensource.org/docs/definition.php). В каком-то смысле данная книга есть open source в действии.

Чем не является эта книга

Книга не является традиционным пособием по языкам программирования, она не может служить исчерпывающим руководством ни по Squeak, ни по Java. Книга также не является исчерпывающим руководством по составлению алгоритмов.

Как читать это пособие

Предполагается, что у вас на рабочем столе кроме этой книги будет компьютер, который вы будете использовать для экспериментов и выполнения упражнений. **Без компьютера читать эту книгу совершенно бессмысленно, многие вещи останутся просто непонятными.** Для изучения первой части книги необходимо установить на компьютере среду Squeak, для второй части — Java и среду Eclipse. Эти среды можно установить практически на

любой персональный компьютер (операционные системы: Windows-95, 98, 2000, XP, MAC OS, Linux и др.). Вам понадобится также мышь (в традиционном Smalltalk используется трехкнопочная мышь, но можно обойтись и обычной). Инструкции по установке излагаются по ходу дела, а сами среды (для платформы Win32) содержатся на прилагаемом компакт-диске. Если вы используете другую платформу, вам придется самостоятельно загрузить соответствующие среды из Интернета.

Хочу также обратить внимание читателя, что не менее, а подчас и более важной часть данного пособия являются тексты программ. Иногда они говорят больше, чем поясняющий текст.

В зависимости от вашего опыта и знаний можно предложить следующие возможные «маршруты» чтения пособия:

1) У вас нет никакого опыта программирования. В этом случае начинайте с начала и читайте книгу подряд, выполняя предложенные задания и упражнения.
2) У вас есть опыт программирования. Тогда можно начать с задачи 4 или 5, первой части, при необходимости обращаясь к приложениям за справкой.
3) Вас интересует только Java. В этом случае книгу лучше не покупать, так как, не освоив первую часть, вы не сможете освоить вторую.

Начальный уровень навыков

Предполагается, что вы обладаете навыками работы на персональном компьютере, которые позволят вам установить среды, содержащиеся на прилагаемом CD. Предполагается, что вы умеете работать со стандартным Windows-подобным интерфейсом и имеете некоторое представление о работе компьютера, т. е., например, слова «память» и «процессор» не вызывают у вас недоумения. Необходимо владеть английским на начальном уровне чтения технической документации, так как оба предлагаемых языка, как, впрочем, большинство языков программирования, основаны на английском. Это минимальные требования. Не требуется иметь каких-либо познаний в области программирования или уметь программировать, однако, если вы имеете опыт составления программ на любом языке программирования, то это не помешает.

Краткий обзор содержания

В первой части книги излагаются основные приемы и понятия, относящиеся к составлению алгоритмов, а также концепция и понятия объектно-ориентированного программирования. В качестве рабочего инструмента используется среда Squeak, которая является современной реализацией языка Smalltalk-80.

В этой части пособия принята следующая логика изложения: предлагается задача, затем приводится ее решение, по ходу которого вам, возможно, будет предложено самостоятельно проделать некоторые эксперименты. Затем следуют обсуждение и задания для самостоятельного выполнения. Раздел

«Обсуждение» обобщает опыт, полученный вами в процессе экспериментирования, он также содержит правила и определения.

Первую часть завершает сжатое изложение концепции ООП.

Во второй части читатель знакомится со средой Eclipse и Java. Здесь стиль изложения совершенно иной. Философия Java не несет в себе идеи персональных вычислений, интерактивности, и читателя уже не «ведут за руку», как в первой части. Для большинства задач первой части приведен код, демонстрирующий их решение на Java. Во второй части приведены также необходимые сведения по работе с библиотекой графического интерфейса SWT.

Состав CD-ROM

- Версия Squeak 3.2, совместимая с Windows 98/2000/NT/XP;
- Java SDK 1.4.2.04 для Windows 98/2000/NT/XP;
- документация по Java SDK 1.4.2.04;
- версия Eclipse 2.1.3. для Windows 98/2000/NT/XP;
- библиотека примеров Eclipse 2.1.3;
- файлы с примерами из книги.

Условные обозначения

— вопрос «по ходу дела».

— эксперимент «по ходу дела».

— упражнения или проекты для самостоятельного выполнения.

имя объекта сообщение — таким шрифтом набраны синтаксические правила.

`turtle go:100` — таким шрифтом набраны тексты программ первой части.

Благодарности

Автор приносит свои благодарности людям, которые так или иначе участвовали в работе над книгой, в особенности доценту МГПУ В. А. Кондратьевой, ассистировавшей автору в преподавании соответствующего курса; доценту МГПУ В. П. Моисееву, доценту СУНЦ при МГУ им. М. В. Ломоносова И. А. Фалиной, профессору МГУ В. К. Белошапке, замечания которых были учтены при подготовке рукописи; эксперту ООО «Люксофт» О. В. Морозу, который прочел рукопись и указал на некоторые ошибки.

Часть 1. Squeak

К российским читателям

Как могло случиться, что наряду с идеей структур данных и процедур, которая для многих до сих пор кажется вполне естественным способом конструирования и программирования компьютеров, несколько программистов решили представить компьютерные вычисления в совершенно другом, «объектно-ориентированном» виде? Наиболее простое объяснение заключается в том, что достаточно рано в истории компьютерных вычислений возникла потребность в освоении перспективных областей, подход к которым в рамках «нормального стиля программирования» был очень сложным. Большинство программистов любили создавать большие и сложные системы и, к сожалению, продолжают делать это до сих пор: и лишь единицы невзлюбили ненужную сложность настолько, что решили изобрести новый способ описания компьютерных вычислений.

Мысль о том, что логически целостный компьютер может быть построен на основе элементов и систем, взаимодействующих при помощи сообщений, была навеяна несколькими блистательными идеями 60-х годов, в особенности Скетчпадом (Sketchpad), Симулой (Simula), компьютером B5000, дизайном ARPAnet и результатами молекулярной биологии. Легко видеть, почему такой подход мог оказаться хорошим: если заставить это работать, то все компьютерные возможности по представлению отношений и процессов будут доступны на всех уровнях и для всех элементов, включая способность отвергать нежелательные сообщения. И, несмотря на то, что в то время еще не слишком много было известно о масштабировании компьютерных вычислений, понятие динамического объекта как элемента или системы означало, что как только возникнут серьезные идеи масштабирования, они могут быть реализованы достаточно просто.

Прямой предок Squeak — Smalltalk, разработанный в лаборатории Xerox PARC в 1970-х годах, — был первой завершенной и успешной попыткой построения компьютерной системы на основе идеи динамических объектов. Наряду с успехом этой идеи в деле становления идеи персональных вычислений, было интересно и тревожно наблюдать, как мало было воспринято из ее наиболее многообещающей части большинством программистов. Попросту говоря, похоже, что, несмотря на преимущества нового подхода, большинству компьютерных специалистов психологиче-

ски очень трудно освоить его после того, как какой-то подход уже ими освоен.

Squeak — это динамическая система, в которой применяется позднее связывание и которая предоставляет значительные возможности для метапрограммирования и, следовательно, для изменения основных представлений относительно описания процессов. Лучший способ начать работу с такой системой — постараться быть *идеалистичным* по отношению к тому, что вы делаете. Задавайте вопросы по поводу «внутренней структуры» желаемого нового артефакта, решайте задачи в терминах внутренней структуры, и затем используйте как программирование, так и метапрограммирование для того, чтобы сделать практические решения как можно более соответствующими внутренней структуре.

Хороший практический способ — воздерживаться от применения мощных библиотек и компонентов, разработанных экспертами, таких соблазнительных для разработки быстрого решения. Вместо того, чтобы применять их, попробуйте сделать набросок программы — и Squeak создан для этого, — для того чтобы глубже исследовать пространство дизайна. Несколько дополнительных дней, потраченных на это, позволят принимать более взвешенные решения по поводу того, как использовать эту библиотеку или расширить ее или вообще идти другим путем. Помните, что *вся* без исключения функциональность Squeak представляет собой то или иное расширение, созданное по той или иной причине. Механизм расширений полностью открыт и доступен, и поэтому все, что входит в Squeak, включено для удобства и может быть использовано по мере надобности.

Именно эта принципиальная интерактивность и расширяемость Squeak делают его столь популярным в сфере образования.

И, наконец, несмотря на то, что Squeak может показаться вполне современным, даже авангардистским по отношению к стандартам более традиционных языков, используемых сейчас, помните, что он представлял собой передний край развития программирования 30 лет назад. Но теперь он требует от нас воплощения лучших идей. Squeak обладает механизмами для качественного перерождения себя в плане воплощения этих идей, поэтому давайте попробуем найти, реализовать, и доказать их жизнеспособность.

С наилучшими пожеланиями,
Алан Кэй
Лос-Анджелес, Калифорния
Декабрь 2004

Smalltalk и Squeak: немного истории

Самым первым языком программирования, в зачатке содержавшим основные идеи объектно-ориентированного программирования, был Симула. Этот язык был разработан норвежскими учеными Оле-Йоханом Далом (Ole-Johan Dahl) и Кристеном Нигаардом (Kristen Nigaard) в начале 60-х годов XX столетия и предназначался для решения задач моделирования. Это было время расцвета кибернетики, время, в которое идея моделирования как нового мощного и универсального метода исследования владела умами многих ученых. Предполагалось, что при помощи моделирования можно разрешить многие проблемы, которые раньше считались неразрешимыми. Поэтому такого рода социальный заказ послужил стимулом к созданию нового языка программирования. Ранние версии Симулы содержали понятия активности и процесса, которые в Симуле-67 были переименованы в классы и объекты. Однако создатели Симулы не оценили всю мощь своего изобретения. Это суждено было сделать другому человеку.

Алан Кэй, разнообразно одаренный человек, изучал математику и молекулярную биологию в университете Колорадо, а позже обучался в университете Юта по программе «Электротехника» (Electrical Engineering). Там он познакомился с языком Симула. Базируясь на идеях этого языка и на своих познаниях в биологии, Кэй выдвинул концепцию «идеального компьютера», который подобно живому организму состоял бы из клеток, взаимодействующих между собой для достижения общей цели.

Осенью 1968 года Алан познакомился с Сеймуром Пейпертом в лаборатории искусственного интеллекта Массачусетского Технологического института. Там он впервые увидел детей, программирующих на ЛОГО. Это потрясло Алана и сильно повлияло на его взгляды на программирование и роль компьютеров в обществе. Как пишет сам Алан, именно в это время у него возникла идея персонального компьютера.

После защиты докторской диссертации Алан преподавал два года в Стэнфорде, это время он посвятил дальнейшей разработке идеи такого компьютера, который «дети могли бы носить повсюду с собой и использовать вместо бумаги». В это же время окончательно оформилась идея языка Smalltalk, основанного на метафоре автономных сущностей — клеток, взаимодействующих при помощи сообщений.

В 1972 году Алан Кэй приходит на работу в исследовательский центр фирмы Xerox в Пало Альто, Калифорния (Palo Alto Research Center, Xerox PARC). Он разрабатывает проект прототипа современных ноутбуков — Dynabook. К сожалению, технологии, которые позволили бы реализовать в полной мере этот проект, в то время еще не существовали. Руководство фирмы Xerox не выделило достаточных ресурсов на развитие идей Алана Кэя, и Dynabook остался лишь красивой идеей. В 1979 году PARC посещают основатели фирмы Apple Стив Джоббс (Steve Jobbs) и Джефф Раскин (Jeff Raskin). Они сразу оценили идею графического интерфейса, основанного на «окнах» и всплывающих меню, и гибкость языка Smalltalk. Все это было взято на вооружение и применено в операционной системе компьютеров Apple Macintosh.

Сейчас трудно представить себе персональный компьютер и его графический интерфейс иным, чем мы его видим, поэтому хочется процитировать фразу Алана Кэя, ставшую крылатой: «Лучший способ предсказать будущее — изобрести его».

Вы познакомитесь с современной реализацией Smalltalk-80 — Squeak. Наиболее популярные коммерческие версии Smalltalk в настоящее время доступны от фирм IBM (http://www-306.ibm.com/software/awdtools/smalltalk/) и Cincom (http://smalltalk.cincom.com/index.ssp). Основная область применения этих продуктов — сложные многокомпонентные системы, в том числе работающие в Интернете.

Основные приемы

Перед началом работы

Если на вашем компьютере установлена операционная система семейства Win32, то для того, чтобы начать работу со средой Squeak, достаточно скопировать каталог Squeak с прилагаемого CD на жесткий диск своего компьютера. Исполняемым файлом в этом каталоге является squeak.exe. Если на вашем компьютере установлена другая операционная система, то необходимо «скачать» соответствующие файлы с http://www.squeak.org/download/index.html

Задача 1 (объекты и сообщения)

Чему вы научитесь и что узнаете, изучая данный раздел

Вы получите первоначальное представление о том, что такое объекты и сообщения.

Дано

В нашем распоряжении имеется робот-черепашка, который живет в среде Squeak и способен рисовать на экране. Черепашка рисует, перемещаясь по экрану и оставляя на нем следы. Черепашкой можно управлять, посылая ей сообщения с просьбой выполнить то или иное действие. Например, получив сообщение `go:100`, черепашка перемещается из точки, где она в данный момент находится, на 100 экранных точек по прямой в том направлении, куда она в данный момент смотрит. Сообщение `turn:90` заставляет черепашку повернуться на 90° по часовой стрелке относительно своего текущего направления. После перемещения и поворота черепашка остается там, где она остановилась и смотрит туда, куда была повернута.

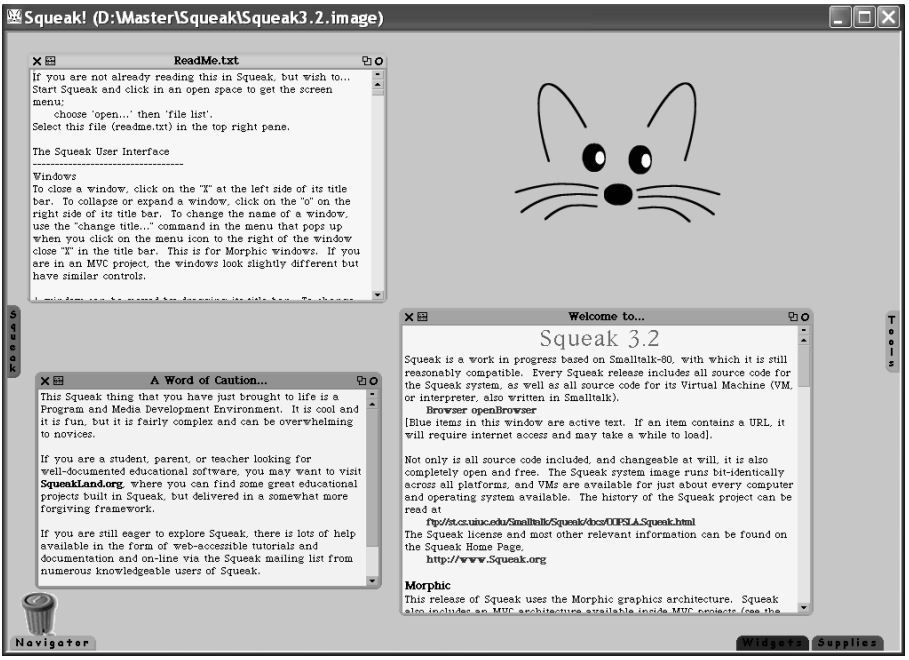
Требуется

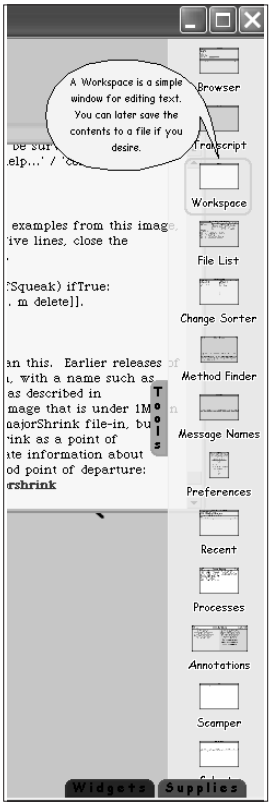
Нарисовать на экране квадрат со стороной 100 экранных точек. Цвет и толщина линии, расположение квадрата на экране не имеют значения.

Решение

Попробуйте самостоятельно написать последовательность сообщений, которая заставит черепашку нарисовать квадрат.

Написанная вами последовательность — ключ к решению задачи, **идея** ее решения, теперь нужно научиться подавать команды. Для этого загрузите среду Squeak и проделайте следующие манипуляции.





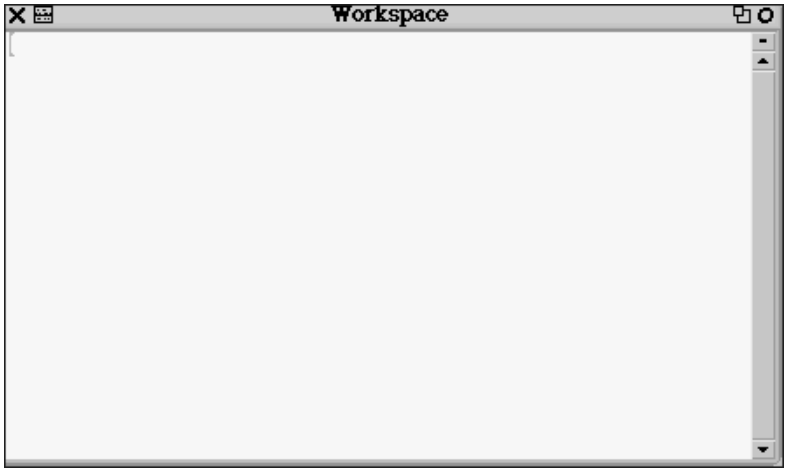
1. Запустите Squeak.

2. Откройте рабочее окно, которое вы будете использовать для разнообразных действий, речь о которых пойдет чуть ниже. Для этого выполните следующие операции.

   2.1. Щелкните мышью по закладке **Tools**.

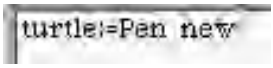
   2.2. Справа вы увидите палитру окон, из которой нужно в буквальном смысле «вытащить» окно **Workspace**.

Должно появиться вот такое окно:



3. В рабочем окне (**Workspace**) вы будете вводить сообщения черепашке и другим объектам и просить систему Squeak переслать эти сообщения адресатам. Чтобы было видно, что рисует черепашка, расположите рабочее окно в левом верхнем углу экрана.

   3.1. Вначале черепашку нужно создать. Наберите следующий текст:



   3.2. Подсветите (выделите) этот текст так же, как вы делаете это в обычном текстовом редакторе, и правой кнопкой мыши вызовите меню (каждое окно в Squeak имеет свое меню):

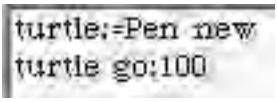
   3.3. В этом меню выберите **do it** (левой кнопкой) или нажмите на клавиатуре **Alt+d**. Пока что ничего видимого не произошло. Но Sque-

ak создал черепашку по имени `turtle`. Манипуляцию, состоящую из подсвечивания текста и выбора из меню **do it**, будем называть **выполнением** выражения.

3.4. Теперь наберите следующую строчку:

```
turtle go:100
```

и выполните только ее (т. е. должна быть подсвечена только эта строчка). Должен появиться вертикальный отрезок длиной 100



пикселей, нижний конец которого находится в центре экрана. Выполните таким же образом еще две строчки, после чего вам станет ясно, как решить задачу:

```
turtle turn:90
turtle go:100
```

Теперь вы можете самостоятельно довести решение задачи до конца. Если вы так не считаете, то ответьте на вопросы:

- В какой точке находится «новорожденная» черепашка?
- Куда она смотрит?

Независимо от того, выполнили вы задание или нет, проделайте следующие эксперименты.

- Закройте окно **Workspace**, щелкнув мышью по крестику в левом верхнем углу этого окна (все ненужные окна можно закрывать таким способом). Откройте новое рабочее окно. Не набирая первую строчку (`turtle:=Pen new`), попробуйте послать черепашке какое-нибудь сообщение (выполнить выражение `turtle go:100`).
- Попробуйте создать черепашку по имени `turtle`, но в дальнейшем называть ее иначе, например, `tortoise`.
- Попробуйте иначе писать сообщения, например, `GO:100` или `torn:90`.
- Попробуйте написать в окне какую-нибудь тарабарщину и выполнить ее.
- Поэкспериментируйте с другими сообщениями черепашке:

Сообщение	Действия черепашки
`up`	поднимает перо вверх (перестает рисовать)
`down`	опускает перо вниз
`north`	поворачивается на север (верх экрана)
`home`	становится в центр экрана
`color:n`	цвет рисуемой линии изменяется на цвет с номером *n*

Как сохранять результаты своей работы

Здесь пойдет речь о сохранении результатов вашей работы. Когда-нибудь вам наверняка захочется завершить сеанс работы с системой Squeak и выключить компьютер. При возобновлении работы с системой вам захочется опять увидеть то, что вы уже наработали; иными словами, вам потребуется как-то сохранять результаты вашей работы в Squeak. Squeak дает возможность сохранять «образ системы» — всё, вплоть до содержимого окон. Образ системы хранится в двух файлах: Squeak 3.2.image и Squeak 3.2.changes, расположенных в том же каталоге, что и сама система Squeak. Для сохранения образа системы:

1. Щелкните левой кнопкой мыши в любом свободном от окон месте экрана (но в пределах окна приложения Squeak). Должно появиться меню world.
2. Выберите в этом меню один из пунктов, относящихся к сохранению результатов работы или выходу из системы:
   - **save** — сохраняет образ системы (вплоть до содержимого рабочих окон) в стандартных файлах;
   - **save as** — сохраняет образ системы в файлах с названиями, заданными пользователем;
   - **save and quit** — сохраняет образ системы в стандартных файлах и осуществляет выход из системы.

Обсуждение

- Как вы увидите в дальнейшем, чтобы выполнить почти любое задание, нужно проявить изобретательность, решить головоломку, поэтому формулировки некоторых заданий могут напоминать формулировки математических задач. Вот и в нашей задаче заранее не вполне понятно, как заставить черепашку нарисовать квадрат. Но подумаем об иной формулировке задачи, которая больше соответствует тому, что вы создали. А создали вы запись — текст, состоящий из последовательности выражений. Такую запись в дальнейшем будем называть программой. Поэтому нашу задачу, можно сформулировать именно как задачу создания программы. Итак, мы будем различать простую постановку задачи, например, «нарисовать квадрат», и задачу создания программы — «создать (придумать, написать) программу для рисования квадрата».

- Роботов, подобных черепашке, т. е. таких, которые обитают в среде Squeak и понимают некоторый набор сообщений, будем называть **объектами**. Любой объект обладает:
  - поведением, которое характеризуется действиями, выполняемыми им в ответ на сообщения;
  - состоянием. Например, состояние черепашки характеризуется местоположением и направлением, а также другими параметрами, которыми мы пока не пользовались. Текущее состояние объекта есть результат выполнения объектом каких-либо действий. Вместе с тем результат выполнения действий объектом зависит от его состояния.

Обоснуйте два последних утверждения на примере поведения черепашки.

Как вы увидите в дальнейшем, объекты Squeak представляют собой своеобразные модели «настоящих» объектов, таких как числа, символы, строки текста и т. д.

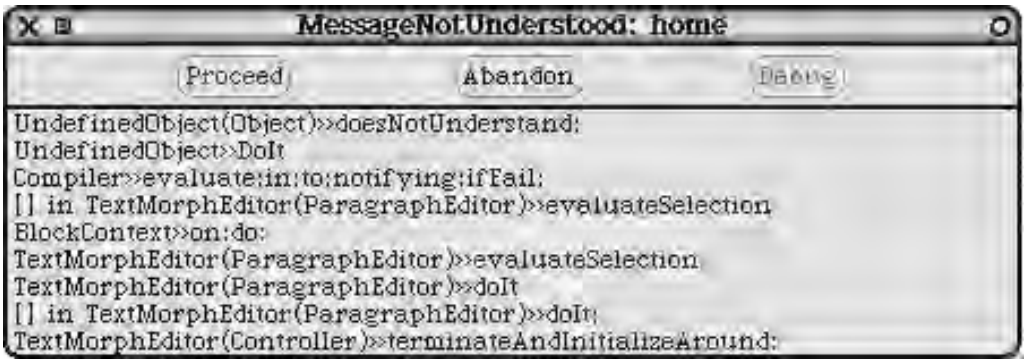
- Синтаксисом будем называть формальную запись, показывающую общий вид какой-либо языковой конструкции. Синтаксис **выражения** посылки сообщения таков:

  идентификатор сообщение.

  Смысл выражения состоит в посылке *сообщения* объекту, который представлен *идентификатором* — именем объекта. Выполняя выражение в рабочем окне, вы просите Squeak послать сообщение некоему объекту, т. е. дело обстоит так, как будто это вы посылаете сообщение объекту. В дальнейшем вы увидите, что объекты посылают сообщения друг другу, и это есть единственный способ взаимодействия объектов.

- Последовательность выражений можно выполнить за один раз, если в конце каждого выражения поставить точку и выделить весь текст программы.

- Объект может не понимать сообщение. В этом случае возникает такое окно:



[. . .]