Эндрю Ричардс

Modula-2 в производстве компьютерных игр

Andrew Richards (1994) «Modula-2 in the Games Industry» // The Modular Mag, No.2, p.34-36.

Эндрю Ричардс (1995) «Modula-2 в производстве компьютерных игр» // Технология программирования, Т.1, No.1, с.181-182.

Я смотрю на программирование не как на инженерную деятельность, а как на возможность проявить себя. Я готов потратить дни и недели на получение крошечного кода, который работает лишь слегка быстрее или расходует чуть меньше памяти, чем то, что написано кем-то до меня.

Я программирую почти исключительно на ассемблере, поэтому при переходе с одной платформы на другую требуется переписывать практически все. Мой последний проект состоял из 16.000 строк на ассемблере и 6 мегабайтов данных. При этом я дал слово, что все это будет работать в ROM объемом 1 Мб. Программа использует много процессов, работает с оборудованием напрямую и состоит только из одного исходного файла. Думаю, вы можете себе представить, каково отлаживать такую программу. Я пишу компьютерные игры для PC, Amiga и игровых консолей и в конце концов решил перейти с ассемблера на Modula-2.

Большинство тех, кто занимается программированием компьютерных игр, пишет их или исключительно на ассемблере или же на смеси ассемблера и C. Я не знаю никого, кто бы выбрал Modula-2, поэтому попробую разъяснить, почему же я так поступил. Очевидный ответ — это переносимость. На языке Modula-2 я могу определять специфические низкоуровневые системнозависимые модули с общим интерфейсом. Мой код может в дальнейшем импортировать эти модули, а потому будет полностью переносим между разными платформами. Однако то же самое я мог бы делать и на C. Практика показала, что для меня-то как раз переносить процедуры, написанные на языке ассемблера, довольно легко.

Так что наиболее важной причиной является скорость. Игра должна быть приятна в общении, хорошо выглядеть и звучать. А потому скорость — очень важный момент для тех, кто пишет игры. Особенно она важна в анимации — игры, которые могут перерисовывать экран с частотой смены кадров на дисплее (50 кадров в секунду в Европе и 60 — в Америке), будут выглядеть и восприниматься гораздо лучше, чем те, что не в силах поддерживать такую скорость. Экран обычно требует от 20 до 60 Кб видеопамяти. Видеошина, как правило, позволяет вам передавать по ней данные со скоростью 100Кб-2Мб в секунду, по 2-40 Кб на кадр. После принятия решения о написании кода и вычисления координат вы вдруг увидите, что со временами дела могут обстоять довольно скверно. Большинство программистов в этой ситуации прибегает к проверенной силе языка ассемблера. Это вполне бы подошло во времена 8-разрядных процессоров. Но нынешние 16- и 32-разрядные процессоры уже позволяют мне для нахождения хитроумных решений использовать Modula-2.

Итак, я обнаружил, что результаты действительно получаются весьма обнадеживающими. Для получения максимальной скорости процедуры рисования по-прежнему нужно писать на ассемблере, но использовать их лучше уже в Modula-2. Отличие здесь в том, что Modula-2 обладает способностью выражать сложнейшие алгоритмы с удивительной простотой. Например, если объект появляется на экране, его нужно нарисовать со всеми его деталькими. И если он затем перемещается, то придется перерисовать ту часть фона, которую он занимал и вдобавок нарисовать его на новом месте. Будем считать, что объект довольно сложный, поэтому я буду вынужден перерисовать целиком весь экран, что конечно же крайне расточительно. На языке Modula-2 с его великолепными структурами управления контролировать изменения такого кода очень легко.

Существует и другое важное преимущество Modula-2, которое ведет к повышению скорости. Если я пишу небольшой фрагмент, который считывает последовательно из файла кучу байтов, то работать он будет медленно, если я не создам буфер и не стану считывать через него. Если вы посмотрите на большинство библиотек C, отвечающих за чтение из файла, то вы увидите, что они также пользуются буфером. Далее они могут делать вызов к операционной системе, которая использует уже другой буфер. Если вы работаете с MS-DOS и SmartDrive, то между этими двумя буферами появится еще и третий. К чему все эти буфера? Ведь если бы я получил прямой доступ к буферу операционной системы, то я смог бы работать с ним, с операционной системой и с моим жестким диском. Вы можете увидеть такие «слоеные пирожки» почти в любой C-программе: один фрагмент кода опирается на другой, и каждый из них использует собственный буфер и механизм контроля доступа. В языке Modula-2 существуют только два уровня: низкоуровневый системнозависимый код и переносимый и надежный код с возможностью всех проверок. Вот основная причина того, что несмотря на накладные расходы, возникающие при работе с Modula-2, программы, которые на ней написаны, работают гораздо быстрее тех, что написаны на других языках.

Но главное, что мне нравится в Modula-2, — это ее способность помогать вам в ведении проекта. На C вы можете написать header-файлы и make-файлы и разделить весь свой код на несколько файлов с исходниками. В моем же компиляторе Logitech Modula-2 для перестроения всего проекта я использую только одну команду меню, даже если я только что добавил или удалил модуль (что для меня вполне обычное явление). Я могу даже использовать разные модели данных и кода для разных модулей — компилятор справится и с этим.

Такая возможность особенно ценна, когда я занимаюсь переносом кода, поскольку это означает, что я могу полностью сменить низкоуровневый модуль и в точности знаю, кто его использует и что нужно поправить. Так, например, на PC для рисования и смены спрайтов я буду использовать один лишь центральный процессор. Мой модуль `Sprites` в этом случае выглядит так, как показано на рисунке внизу.

Игровые консоли имеют аппаратные спрайты и ROM на картридже. Чтобы высветить объект, вы должны загрузить его в видео-RAM и затем поместить его в список спрайтов. Чтобы переместить спрайт, вы должны просто изменить этот список. Чтобы удалить

```
DEFINITION MODULE Sprites;
TYPE Sprite = POINTER TO SpriteDesc;
     SpriteDesc = RECORD
                    Width, Height: INTEGER;
                    Data: SYSTEM.ADDRESS;
                  END;

PROCEDURE Load (VAR S: Sprite;
                    SpriteNumber: INTEGER);
     (* загрузить спрайт из файла в память *)

PROCEDURE Free (S: Sprite);
     (* освободить память, выделенную под спрайт *)

PROCEDURE Put (S: Sprite; x,y: INTEGER);
     (* нарисовать спрайт на экране *)

PROCEDURE Clear (x, y, Width, Height: INTEGER);
     (* восстановить фон, ранее занятый спрайтом *)

PROCEDURE SetBackground (S: Sprite);
     (* выбрать, какой спрайт размером с экран использовать в качестве фона *)

 END Sprites.
```

```
DEFINITION MODULE Sprites;
TYPE SpriteData = RECORD
                    Address: CARDINAL;
                    Width, Height: INTEGER;
                  END;
     Sprite = INTEGER;
     (* индекс спрайта в общем списке *)

PROCEDURE Load (VAR Data: SpriteData;
                    SpriteNumber: INTEGER);
     (* загрузить спрайт из ROM-картриджа в видеопамять *)

PROCEDURE Free (VAR Data: SpriteData);
     (* освободить видеопамять, отведенную под спрайт *)

PROCEDURE Add (VAR S: Sprite;
               VAR Data: SpriteData;
                   x, y: INTEGER);
     (* добавить спрайт в список высвеченных спрайтов *)

PROCEDURE Remove (S: Sprite);
     (* удалить спрайт из списка высвеченных спрайтов *)

PROCEDURE Move (     S: Sprite;
                VAR Data: SpriteData;
                    x, y: INTEGER);
     (* переместить спрайт *)

PROCEDURE SetBackground (SpriteNumber: INTEGER);
     (* выбрать, какой спрайт размером с экран использовать в качестве фона *)

END Sprites.
```

спрайт, вы должны освободить выделенную ему видеопамять и удалить его из списка.

Взгляните на другое описание модуля `Sprites`.

Все это позволяет мне писать игры в том виде, который наиболее эффективен для целевой машины. На ассемблере или на C такая работа превратилась бы в сплошной кошмар. Преимущества Modula-2 здесь просто коллоссальные.

Я абсолютно уверен, что когда профессор Вирт создал Modula-2, он и не думал о таком повороте дел. Он задумывал язык, который был бы полезен для широкого круга задач, что лишний раз доказывает, насколько хороша Modula-2. Однако, поскольку я с ней работаю, я обнаружил множество неудобств. Все они исправлены в языке Oberon-2, но мне по-прежнему нужен беззнаковый тип и совершенно не нужна сборка мусора. Объектно-ориентированное программирование видится вполне подходящим для программирования игр, но к сожалению оно опирается на динамическое распределение памяти. И если играющий затратил целый час на то, чтобы дойти до конца очередного уровня, а в этот самый момент программе не хватит памяти, то мне не избежать лавины возмущенных телефонных звонков.

Я работаю с компилятором Logitech Modula-2, который порождает код для реального режима процессора 8086. Оптимизатор довольно хороший, но даже он не подходит для реализации низкоуровневых спрайтовых процедур. (Я бы сильно удивился, если бы он справился и с этой задачей.) Кроме того, отсутствие некоторых низкоуровневых процедур представляет вполне ощутимую проблему. Я не могу выполнять байт-ориентированные операции типа считывания байта из файла и изменения в этом байте знака. А мне приходится делать это довольно часто. К тому же в большой модели памяти проверка на переполнение стека не работает — программа просто рушится.

Отладчики — это конечно хорошо, но они либо требуют большого объема памяти, либо просто не могут работать с моими программами. Чтобы запустить отладчик в режиме 386 процессора (только в нем я и могу отлаживать свои жадные до памяти программы), я должен поправить CONFIG.SYS и AUTOEXEC.BAT и перезагрузиться. Затем я опять должен восстановить эти файлы, перезагрузиться и войти в редактор. А иногда происходят просто непонятные вещи, и я не знаю, как с ними быть. В какой-то момент посмертный отладчик нормально загружает сохраненный образ памяти, но показывает совсем не ту строку, на которой рухнула моя программа. Детально я с этим не разбирался, а попытка получить разъяснения от Logitech (теперь влившуюся в Symantec) окончилась безрезультатно.

Мой компилятор создает код и для Windows, но при этом он позволяет писать и такой код, который не допустим при работе под Windows. Было бы лучше, если бы он был все же построже. Мне доводилось писать некоторые фрагменты, которые великолепно работали на одном компьютере и не работали на другом. И я не удивился бы, если бы тоже самое происходило при работе с C, но от Modula-2 я жду большего.

Несмотря на все это я настоятельно рекомендую вам этот компилятор. В подавляющем большинстве случаев он работает безукоризненно. Он быстрый, прост в использовании и дает хороший код. Сейчас же я ищу подходящий компилятор для защищенного режима 386 процессора и для работы с языком ассемблера процессора 68000.

Крис Джонсон

Modula-2 в исследовательском центре NASA Lewis Research Center

Chris Johnson (1993) «Modula-2 at the NASA Lewis Research Center» // The Modular Mag, No.1, p.35.

Крис Джонсон (1995) «Modula-2 в исследовательском центре NASA Lewis Research Center» // Технология программирования, Т.1, No.1, с.182-183.

Когда открылась научная лаборатория по микрогравитационным материалам (Microgravity Materials Science Laboratory), ее оборудовали большим количеством систем съема данных, которые состояли из компьютеров Hewlett-Packard HP-85B. То были маленькие компьютеры с BASIC'ом, прошитым в ROM. Нас вполне устраивало оборудование для съема данных, но уровень компьютеров 85B нас устраивать уже не мог. А потому мы решили заменить их машинами семейства IBM AT. Их выбрали по той причине, что для AT имелось большое количество программного обеспечения, да и все наши задачи съема данных можно было просто решить путем использования платы с интерфейсом IEEE-488.

Смена компьютеров означала, естественно, пересмотр требований, которые мы предъявляли к программам на BASIC'е. В поле нашего зрения попал Pascal, и нам понравилась его структура. Затем мы увидели Modula-2 и решили, что именно этот язык удовлетворяет всем нашим требованиям. Его-то мы и выбрали для своей работы. В конечном итоге мы решили стандартизировать процесс съема данных и процесс управления в рамках языка Modula-2.

При этом мы не отказались полностью от других языков. Хотя они нам практически и не понадобились. Наша команда довольно мала, но с Modula-2 мы почувствовали себя сильнее.

Мы использовали язык для нескольких проектов и обнаружили, что он показывает себя великолепно, особенно когда приходится работать нескольким программистам над одним проектом. Мы склонны делить проекты по таланту и интересу — кому-то из нас нравится делать пользовательский интерфейс, другому — низкоуровневый аппаратный интерфейс. Модульная структура Modula-2 легко позволяет добиться желаемого. В тех проектах, которые мы разбивали подобным образом, у нас фактически не было проблем с интерфейсами между модулями, написанными разными программистами. Нашим первым заметным проектом был набор графических модулей, которые позволяли вывести обычные данные в формате (X,Y) на экран или на плоттер. Он велся таким образом, что базовые про-