

С/к Введение в современное программирование (v.5.5)

Физфак МГУ. 2006/7 уч. год.

Лекция 14

Организация интерфейса человек-компьютер = одна из важнейших задач программирования.

Концептуальная пропасть между мозгом и "камнем" **огромна**.

В частности, **GUI** = graphical user interface = графическое взаимодействие с пользователем = система мостов через эту пропасть.

80-90% мощности ПК бывает занята именно этим.

Homo Sapiens, семейство гоминид, отряд приматов: прекрасно развитое зрение (**Q** Почему?): бинокулярное, цветное, 1/3 головного мозга = "видеопроцессор".

Два следствия:

1) канал манипуляций ("сунуть в глаза клиенту/начальнику/научному руководителю/... красивую цветную картинку ... и чтобы кнопки нажимать, а там что-нибудь шевелилось ...")

2) мощное средство представления информации (связь "мозги<-->железо")
а) для анализа
б) для управления

ЯП и текст — хорошо, но не достаточно: глаза + 30% мозга, ими управляющая, могут воспринимать информацию гораздо быстрее в графическом виде.

Хотим видеть **картинку** — и в зависимости от этого **управлять** процессом.

Тенденция в сложных измерительно/управляющих системах: "цифровое" изображение (т.е. сгенерированное программой на дисплее) вместо "аналогового" (т.е. сгенерированное с помощью эл. лучей, магнитных полей и т.п.).

Пример — <http://www.inauka.ru/technology/article38383.html>: У истребителя Су-35 стрелочных приборов вообще не будет. Два больших дисплея + самая важная часть боевой и пилотажной информации будет передаваться на лобовое стекло шлема пилота.

Др. пример — система LabView (National Instruments): "виртуальный осциллограф" и т.п.

Набор "стандартн" средств.

Но как и с библиотеками процедур — никогда не достаточно. Относительно простые вещи — быстро. Но чуть нестандартные — затык.

Ср.: Для вычислений никакая библиотека процедур недостаточна — нужны средства для адаптации старых/построения новых.

Еще в большей степени это справедливо для "графических" задач.

Нужен набор средств, "погруженных" в мощный ЯП.

Именно это и предоставляет ББ.

ББ как "superglue"

Laser Lab, U. of Rochester: в данный момент идет разработка на ББ системы управления (самыми мощными в мире) лазерами.

BAE SYSTEMS (Eurofighter): **миллион** строк исходников под ББ (выч. + визуализация фазированных решеток: э.м. поля и т.п.)

В прошлом году несколько студентов делали такие "фасады" (каф. математики, акустики, физики атмосферы).

NB Система поддержки графики в .NET — **слабее**, чем в ББ — слабее в "поколенческом" смысле (содрана с продуктов Borland).

На годы вперед ББ остается уникальным средством.

Еще раз:

две самых важных области применений ББ:

— большие вычисления "синтетич." характера

— "суперклей" для интеграции разнородных измерительно-управляющих систем за произвольно сложным GUI-фасадом.

Ключевое преимущество ББ над всеми специализир. системами (LabView, MatLab, Mathematica ...) — отсутствие фунд. препятствий для развития.

В специализир. системах

— гнилые ЯП (*исключения не известны*)

— из коммер. соображений ограничен доступ к внутренним API ("мы рады сделать, что нужно нашим клиентам, см. расценки...").

Гнилость ЯП = неадекватная поддержка "развития" произвольных структур данных и алгоритмов.

Внутренние интерфейсы ББ ("каркас") выставлены наружу намеренно: ББ

"заточен" под расширение простым добавлением модулей и целых подсистем -- и даже заменой "родных" реализаций на специализированные.

Мы посмотрим основы устройства каркаса ББ.

Для ориентации:

— F1 (User interaction, View construction);

— примеры в подсистеме Obj;

— полные исходники: прежде всего подсистема Form (диалоги).

Ниже: Общие замечания; Основы построения View.

Views — Вьюшки

```

90
91
92 Порт (Port) -- абстракция изображающего устройства -- экран, принтер.
93 Размеры в пикселях.
94 Фрейм (Frame) -- абстракция доступа к порту ("окошко").
95 Вьюшка (View) -- объект, умеющий себя рисовать. Конкретная
96 функциональность задается программистом. Рисование -- в абстрактных
97 универсальных единицах (иначе ...). Общее управление (буферы,
98 преобразования и т.п.) -- реализует и скрывает ББ.
99 В старых системах (C) -- страницы кода, чтобы только открыть окно.
100
101 "представление", "вид" -- слишком туманно; ВЬЮШКА
102
103 Модуль Views, тип View:
104
105 DEFINITION Views; (* упрощено *)
106
107     IMPORT Ports, Fonts, Stores, Files, Models, Converters;
108
109     TYPE
110         Frame = POINTER TO ABSTRACT RECORD (Ports.Frame)
111             rider-: Ports.Rider;
112             ...
113             (f: Ports.Frame) DrawLine (x0, y0, x1, y1, s: INTEGER; col: Ports.Color),
114 NEW;
115             (f: Ports.Frame) DrawOval ... DrawPath ... DrawRect ...
116             (f: Ports.Frame) DrawString (x, y: INTEGER; col: Ports.Color; IN s:
117             ARRAY OF CHAR; ...), NEW;
118             (f: Ports.Frame) Input (OUT x, y: INTEGER; OUT modifiers: SET; OUT
119             isDown: BOOLEAN), NEW;
120             ...
121     END;
122
123     View = POINTER TO ABSTRACT RECORD (Stores.Store)
124         context-: Models.Context;
125         ...
126         (v: View) HandleCtrlMsg (f: Frame; VAR msg: CtrlMessage; VAR focus:
127         View), NEW, EMPTY;
128         (v: View) HandlePropMsg- (VAR msg: PropMessage), NEW, EMPTY;
129         ...
130         (v: View) Restore (f: Frame; l, t, r, b: INTEGER), NEW, ABSTRACT;
131         ...
132     END;
133
134     Message = ABSTRACT RECORD
135         view-: View
136     END;
137
138     ...
139     PROCEDURE RegisterView (v: View; loc: Files.Locator; name: Files.Name); (*
140     "фасады" *)
141     PROCEDURE OldView (loc: Files.Locator; name: Files.Name): View;
142     ...
143     PROCEDURE OpenView (v: View);

```

```

140     PROCEDURE OpenAux (v: View; title: Title); (* для диалогов *)
141     ...
142     PROCEDURE Update (v: View; rebuild: BOOLEAN);
143     ...
144     PROCEDURE ReadView (VAR rd: Stores.Reader; OUT v: View); (* "фасады" *)
145     PROCEDURE WriteView (VAR wr: Stores.Writer; v: View);
146     ...
147     PROCEDURE InstallFrame (host: Frame; v: View; x, y, level: INTEGER; focus:
148     BOOLEAN);
149     ...
150     PROCEDURE Omnicast (VAR msg: ANYREC);
151     ...
152 END Views.
153
154 DEFINITION Ports;
155
156     IMPORT Fonts;
157
158     CONST
159         arrowCursor = 0;
160         graphicsCursor = 2;
161         ...
162         black = 0; ... blue ... green ... grey75 ... red ...
163         white = 16777215;
164         ...
165         inch = 914400;
166         mm = 36000;
167         ...
168     TYPE
169         ...
170         Port = POINTER TO ABSTRACT RECORD
171             unit-: INTEGER; (* размер пикселя в универсальн. единицах *)
172             (p: Port) NewRider (): Rider, NEW, ABSTRACT;
173             ...
174         END;
175
176         Rider = POINTER TO ABSTRACT RECORD
177             (rd: Rider) Base (): Port, NEW, ABSTRACT;
178             ...
179             (rd: Rider) DrawLine (x0, y0, x1, y1, s: INTEGER ... (* пиксели *))
180             ...
181         END;
182
183         Color = INTEGER;
184
185         Point = RECORD
186             x, y: INTEGER
187         END;
188
189     ...
190     PROCEDURE IsPrinterPort (p: Port): BOOLEAN;
191     PROCEDURE RGBColor (red, green, blue: INTEGER): Color; (* "фасад" *)
192
193 END Ports.

```

Пример простейшего рисования

```

192
193
194 MODULE Kurs2006Ex50ris;
195   IMPORT Log := StdLog, Math, In := FVTsysIn (* Epse21SysIn *),
196     Views, StdCmds, Ports;
197
198   CONST mm = Ports.mm;
199
200   TYPE
201     View = POINTER TO RECORD ( Views.View )
202       color: Ports.Color
203     END;
204
205   PROCEDURE ( v: View ) Restore ( f: Views.Frame; l, t, r, b: INTEGER );
206   VAR
207     BEGIN
208       f.DrawLine ( 0, 0, 77*mm, 55*mm, 0, Ports.RGBColor( 0, 255, 0 ) );
209       f.DrawOval ( 10*mm, 20*mm, 40*mm, 50*mm, -1, v.color );
210     END Restore;
211
212   PROCEDURE Do*;
213   VAR v: View;
214   BEGIN
215     NEW( v ); v.color := Ports.red;
216     Views.OpenView( v )
217   END Do;
218
219 END Kurs2006Ex50ris.
220
221 ! Kurs2006Ex50ris.Do
222
223 Frame = POINTER TO ABSTRACT RECORD ( Ports.Frame )
224   ( f: Ports.Frame ) DrawLine ( x0, y0, x1, y1, s: INTEGER; col: Ports.Color ), NEW;
225   ( f: Ports.Frame ) DrawOval ( l, t, r, b, s: INTEGER; col: Ports.Color ), NEW;
226   ( f: Ports.Frame ) DrawPath ( IN p: ARRAY OF Ports.Point; n, s: INTEGER; col: Ports.Color;
227   path: INTEGER ), NEW;
228   ( f: Ports.Frame ) DrawRect ( l, t, r, b, s: INTEGER; col: Ports.Color ), NEW;
229   ( f: Ports.Frame ) DrawString ( x, y: INTEGER; col: Ports.Color; IN s: ARRAY OF CHAR;
230   font: Fonts.Font ), NEW;
231   ( f: Ports.Frame ) CharIndex ( x, pos: INTEGER; IN s: ARRAY OF CHAR; font: Fonts.Font ):
232   INTEGER, NEW;
233   ( f: Ports.Frame ) CharPos ( x, index: INTEGER; IN s: ARRAY OF CHAR; font: Fonts.Font ):
234   INTEGER, NEW;
235   ( f: Ports.Frame ) DrawSSString etc. -- для non-Unicode ...
236   ( f: Ports.Frame ) Input ( OUT x, y: INTEGER; OUT modifiers: SET; OUT isDown:
237   BOOLEAN ), NEW;
238   ( f: Ports.Frame ) SetCursor ( cursor: INTEGER ), NEW;
239   ( f: Ports.Frame ) Scroll ( dx, dy: INTEGER ), NEW;
240 END;
```

Общие замечания о вьюшках в Блэкбоксе

Все, что мы видим в любом окне ББ — вьюшки = View's (потомки абстрактного типа-предка Views.View).
 — Каждое окно — т.наз. корневой/root View.
 — Рисунку, который мы нарисовали, соответствует вьюшка Kurs2006Ex50ris.View (расширение типа Views.View, определенное в модуле Kurs2006Ex50ris).
 — Текстовому документу соответствует "текстовая вьюшка" TextViews.View.
 — Диалогу соответствует вьюшка FormViews.View. Каждой кнопке и т.п. соответствует тоже некая вьюшка.
 — Командер **!** — тоже вьюшка, вставленная в текст.
 И т.д.

NB Различать два аспекта у текста:
 — текст как данные (цепочка литер, командеров и др. вьюшек);
 — текст как нечто видимое (Tools, Document Size..., Window width, затем менять размеры окна — распределение букв по строкам меняется, хотя сам текст как данные — нет).
 Итак,

вьюшка (View) — абстракция "видимости" для неких данных.

Каждая вьюшка (View) может (по решению ее автора):
 — содержать в себе данные, которые он изображает,
 — реализовать протокол (метод Restore и т.п.) взаимодействия с ББ, чтобы ББ мог заставить его в нужные моменты "рисоваться", узнать размер, и т.п.;
 — реализовать протокол для взаимодействия с "юзером" (= клавиша + мышка; например, командер при клике по нему определяет тот текст, в котором он вставлен, анализирует его, извлекает "команду", и отдает запрос Блэкбоксу на ее выполнение).

Последнее означает, что в общем случае **вьюшка — штука активная** (когда нажимаем на клавишу, изображение текста меняется).

"Реализовать протокол" = определить запланированные методы (=процедуры), которые выполняют заранее оговоренные функции, чтобы, например, сообщить (Блэкбоксу) свой размер, отреагировать на нажатие клавиши или щелчок мышкой, сообщить ББ, что содержимое изменилось, и надо обновить изображение.

Единственный обязательный метод — **Restore** (иначе как ББ будет знать, как ее рисовать).

Все остальное — по мере необходимости (и знаний).
 Если не реализовали реакцию на нажатие какой-то клавиши, значит, ничего не будет происходить при нажатии той клавиши.
 Если не реализовали размер, значит, ББ будет брать какой-то размер по умолчанию.

И т.д.

Вьюшки бывают **двух главных видов**:
 — простые (как в первом примере или как командеры), обычно достаточно средств модуля Views.
 Вариант для спец приложений: Controls.Control;
 — контейнеры (тип Containers.View): могут содержать в себе другие View (глубина иерархии неограничена). Примеры: тексты, диалоговые формы.

289 **Рисование картинок**

290 Будем делать *простые* вьюшки, т.е. не содержащие других внутри себя.

291 Система предоставляет модуль Views (с полной документацией Info -->
292 Documentation). Там есть TYPE Views.View (Stores.Store) -- абстрактный тип для
293 создания картинок. Возможности:

294 создать содержимое
295 спасти в / воссоздать из файла
296 положить в системный буфер для дальнейшего paste, как если бы он был cut
297 откуда-то
298 открыть в отдельном окошке
299 вставить в любой документ ББ (текстовый, диалоговую форму —
300 посредством Ctrl+C)
301 вставить в любой документ, поддерживающий OLE (сам BlackBox, MS Word и
302 т.п.)
303 скопировать (cut) из любого документа
304 обновить содержимое (например, чтобы отобразить результат вычисления
305 или чтобы добавит эксп. точку)
306 обновить (после обновления содержимого) картинки во всех открытых
307 окошках, где эта вьюшка видна **
308 получать и обрабатывать сигналы от мышки и клавиатуры
309 одновременно по-разному показывать одни и те же данные (как гистограмму,
310 как сглаженную функцию ...)
311 ОЧЕНЬ МОЩНО, и вся "кухня" скрыта.
312 Будем осваивать постепенно.

313
314 Любая вьюшка должна иметь тип, расширяющий Views.View, определенный с
315 единственным абстрактным методом:

```
316 Views.View = POINTER TO ABSTRACT RECORD ( Stores.Store )
317 ...
318 ( v: View ) Restore ( f: Views.Frame; l, t, r, b: INTEGER ), NEW, ABSTRACT;
319 ...
320 ...
321 END;
```

```
322
323 Views.Frame = POINTER TO ABSTRACT RECORD ( Ports.Frame )
324 ... (* пока неинтересно *)
325 END;
```

```
326
327 Ports.Frame = POINTER TO ABSTRACT RECORD
328 ... (* ЗДЕСЬ то, что нам будет нужно *)
329 END;
```

330
331 **NB** Port|Frame -- абстракции, скрывающие все детали устройства (экран,
332 принтер ...).

```
333
334 MODULE Kurs2006Line; (* простейшая картинка *)
335 IMPORT Views, Ports, StdCmds, Stores, Fonts;
```

```
336
337 CONST mm = Ports.mm; cm = 10*Ports.mm;
```

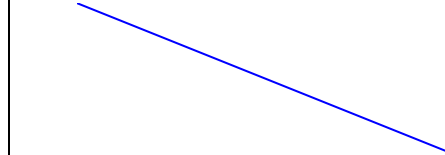
```
338
339 TYPE View* = POINTER TO RECORD ( Views.View ) END;
```

340

```
341 PROCEDURE ( v: View ) Restore* ( f: Views.Frame; l, t, r, b: INTEGER (*
342 просто игнорируем это *) );
343 BEGIN
344   f.DrawLine( 0, 0, 50 * mm, 20 * mm, 0, Ports.blue )
345   (* x0, y0, x1, y1, толщина, цвет *)
346 END Restore;
```

```
347
348 PROCEDURE New*;
349   VAR v: View;
350   BEGIN NEW( v ); Views.OpenView( v );
351 END New;
```

```
352
353 END Kurs2006Line.      ❶ Kurs2006Line.New
354 Вот:
```



355

356

357 Открывшаяся картинка -- тоже документ ББ, его можно спасти в файл, и снова
358 открывать.

359 Можно также вставить (paste) получившуюся картинку, скажем, в Word:
360 сделать Ctrl+Space чтобы отметить картинку целиком;
361 потянуть мышкой за любые квадратики по краям, чтобы подрегулировать
362 размер (потом научимся задавать размер сами);
363 сделать Ctrl+C;
364 переключиться на Word;
365 поставить курсор там, куда нужно вставить картинку;
366 сделать Ctrl+V;
367 Voila!

368 В Ворде можно дважды по ней кликнуть, и полоска меню станет ББ-овской:
369 можно применять все команды ББ'а, предусмотренные для картинки (пока мы
370 ничего не задали), и изменения будут отражаться прямо в Ворде.

371

372 ОДНАКО если стереть модуль Kurs2006Line (скомпилированный бинарник *.ocf),
373 то при следующем открытии документа вместо картинки будет крест — ни
374 Windows, ни BlackBox не будут знать, как интерпретировать этот объект.
375 Вся такая информация хранится в модуле Kurs2006Line.

376

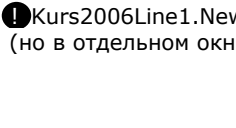
377 **NB** Вставка в Word аналогична сохранению в файл (т.наз. внутр. файл в
378 документах MS Office, поддерживающих вставные объекты по технологии OLE).
379 В дальнейшем поймем, что картинка сохраняется в файл и в Word только
380 потому, что у нашего объекта нет содержимого -- никаких полей, описывающих
381 картинку, мы не предусмотрели, поэтому когда ББ спасает эту вьюшку в файл,
382 он туда фактически только записывает имя объекта, а при восстановлении из
383 файла -- по этому имени восстанавливает объект (см. ранее пример с модулем
384 Meta. Картинка полностью статична, вся информация -- прямо в коде
385 процедуры. Мы этим еще займемся, но пока -- спасается и спасается.

386

387 **Размер вьюшки**

388 Линия рисовалась внутри графич. объекта размеров в текстовое поле целого
389 листа А4. Нехорошо — хотим сами задавать размер картинки.

390 Достаточно ввести одну процедуру — метод для View, — которую Блэббокс
391 будет в какой-то момент вызывать, а она будет ему сообщать про размер
392 картинки:

```
393
394 MODULE Kurs2006Line1; (* простейшая картинка *)
395   IMPORT Properties, Views, Ports, StdCmds, Stores, Fonts;
396   CONST mm = Ports.mm;
397
398   TYPE View* = POINTER TO RECORD ( Views.View ) END;
399
400   PROCEDURE ( v: View ) Restore* ( f: Views.Frame; l, t, r, b: INTEGER (*
401   игнорируем это *) );
402   BEGIN
403     f.DrawLine( 0, 0, 50 * mm, 20 * mm, 0, Ports.blue )
404   END Restore;
405
406   PROCEDURE ( v: View ) HandlePropMsg- ( VAR p: Properties.Message ); (*
407   Handle Properties Message = обработать сообщение о свойствах *)
408   BEGIN
409     WITH p: Properties.SizePref DO
410       p.w := 70 * mm; p.h := 10 * mm
411     ELSE (* ничего не делать,
412           т.к. наша вьюшка пока такая простая, что ничего больше не понимает
413           — а Блэббокс может много чего у нее спрашивать ... *)
414     END;
415   END HandlePropMsg;
416
417   PROCEDURE New*;
418   VAR v: View;
419   BEGIN NEW( v ); Views.OpenView( v );
420   END New;
421
422 END Kurs2006Line1.  !Kurs2006Line1.New
423 Вот такая будет картинка (но в отдельном окне):
```

Здесь край картинки.

424
425
426 **NB** Чтобы в открывшемся окошке получилась рамочка с 8ю квадратами,
427 кликнуть внутрь и нажать Ctrl+Space — это способ отметить целиком объект в
428 окне;
429 **NB** убрать рамочку — ESC.
430 У рамочки ширина и высота такие, как задано в SizePref (70x10 мм), и менять
431 мышкой не получается — раньше получалось.

432 Картинка спасается в файл (File --> Save As...) и правильно восстанавливается.
433 А также правильно копируется в другой документ. Это снова потому, что у нее
434 нет содержимого, которое нужно было бы копировать, а только тип — это
435 Блэббокс сам умеет сохранять/восстанавливать без подсказок (модуль Meta).

436 **Синтаксис** Присмотримся к сигнатуре ключевого здесь метода
437 **HandlePropMsg**. Раз раньше View работал без этого метода, значит, это метод не
438 абстрактный. А какой?
439 Кроме того, компилятор не потребовал поставить атрибут NEW. Значит, этот
440 метод был уже определен. Но Ctrl+D на Views.View не показывает его вообще.
441 Однако Ctrl+D показывает только клиентский интерфейс -- лишь часть полного
442 интерфейса.
443 А мы строим расширение для View, поэтому нужна другая часть интерфейса --
444 **Extension Interface** (Ctrl+Shift+D). А там увидим:

```
(v: View) HandlePropMsg- (VAR msg: Views.PropMessage), NEW, EMPTY;
```

446 **NB** Кстати, полный интерфейс = клиентский + "расширительный". Можно
447 посмотреть полный интерфейс весь сразу (Info, Interface...). Можно и настроить
448 ББ так, чтобы полный интерфейс показывался по Ctrl+D.

449
450 Заметим **implement-only** export (минус после имени процедуры): клиент,
451 написавший эту процедуру (т.е. мы), не сможет ее вызывать.

452
453 Наконец, в заголовке, кроме NEW, есть атрибут **EMPTY** = процедура "пустая",
454 т.е. ничего не делает, но, в отличие от ABSTRACT, конкретно определять ее не
455 нужно. EMPTY не может возвращать значения и иметь OUT-параметры
456 (естественно, т.к. пришлось бы договориться, во что ББ инициализирует OUT
457 параметры у пустых процедур -- а это потенциал для ошибок).

458
459 **Смысл** Здесь применяется некий стандартный прием-схема-pattern: способ
460 каркасу (в данном случае Блэббоксу) общаться с объектами ("передавать им
461 сообщения"), когда у них может быть много разных свойств.

462
463 Процедура (метод) HandlePropMsg -- это способ Блэббокса задавать вопрос
464 объекту -- в данном случае вьюшке v; при этом вопрос — это запись p,
465 передаваемая в качестве параметра.

466 Если v **понимает вопрос** — т.е. знает его конкретный тип, —
467 то он **отвечает** — т.е. заполняет поля своей информацией.

468 Если v не понимает вопроса — т.е. не может распознать конкретный тип p, то v
469 игнорирует вопрос (ELSE).

470 Тогда Блэббокс действует по своему разумению.

471
472 Вот что говорит документация (чтобы привыкнуть к жаргону ... целая наука):

473 **TYPE SizePref (Preference)**

474 The sender of this message proposes a size for the receiving view; the size
475 may be *Views.undefined*. The receiving view may override the proposal ...

476
477 Что делает ББ, когда пытается открыть вьюшку в окне:

478 -- проверяет, есть ли у данного типа конкретная реализация метода

479 HandlePropMsg

480 -- если есть, то вызывает *несколько раз*, передавая в качестве параметра

481 каждый раз разные Properties.Message

482 -- по полученной таким образом информации (в данном случае о размере)

483 принимает решения о том, какого размера открыть окошко

484 -- формирует соотв. область видеопамати, создает фрейм (объект типа

485 Ports.Frame), связанный с этой областью

486 -- вызывает у вьюшки метод Restore, передавая для рисования данный фрейм

487 -- выюшка вызывает процедуры-методы фрейма, которые пересчитывают все
 488 координаты в экранные координаты и осуществляют конкретное рисование
 489 (заполнение байтов, соотв. пикселям, числами, соответствующих цветов -- при
 490 этом эксплуатируются процедуры ОС)

491 **Кстати:** способ "передавать сообщения объекту" с помощью разных
 492 расширений некоего базового типа записей (в данном случае Properties.Message)
 493 был придуман и систематически использован Виртом и Гуткнехтом в Системе
 494 Оберон.
 495 Паттерн "команда" в терминологии Гамма и др.
 496 Такую схему организации еще называют "шина" -- software bus (**Q** что такое
 497 "шина"?).

498 **NB** Модуль, "задающий вопросы" (в данном случае Views и др.) может
 499 развиваться (т.е. мы можем "обучать" его задавать новые вопросы)
 500 **независимо от всех** уже написанных выюшек -- не трогая их никак и даже не
 501 перекомпилируя их модули.
 502 Вот вам пример "компонентно-ориентир. прог-я", когда программа составляется
 503 из независимо развивающихся частей, несмотря на их довольно сложное
 504 кооперативное взаимодействие.

505 **NB** Клиент может не забивать голову сложными вещами, если ему достаточно
 506 простых (как мы в первом примере не забивали голову тем, как задать размер
 507 выюшки).

508 Во втором семестре применим к работе с математическими функциями.

509 View с внутренним содержимым

510 Усложним задачу: добавим выюшке гибкости -- сделаем View зависящим от двух
 511 параметров x, y: INTEGER, задающих положение второго конца линии, а также
 512 изменим процедуру New:

```
513 MODULE Kurs2006Line2;
514   IMPORT Properties, Views, Ports, StdCmds, Stores, Fonts;
515   CONST mm = Ports.mm;
516   TYPE View* = POINTER TO RECORD ( Views.View ) x, y: INTEGER END;
517   PROCEDURE ( v: View ) Restore* ( f: Views.Frame; l, t, r, b: INTEGER );
518   BEGIN
519     f.DrawLine( 0, 0, v.x, v.y, 0 (* толщина *), Ports.blue (* цвет *) )
520   END Restore;
521   PROCEDURE ( v: View ) HandlePropMsg- ( VAR p: Properties.Message );
522   BEGIN
523     WITH p: Properties.SizePref DO
524       p.w := 70 * mm; p.h := 10 * mm
525     ELSE
526       END;
527   END HandlePropMsg;
528   PROCEDURE New* ( x, y: INTEGER );
529   VAR v: View;
530   BEGIN
531     NEW( v ); v.x := x * mm; v.y := y * mm;
```

```
532     Views.OpenView( v )
533   END New;
534
535 END Kurs2006Line2.
536
537 и при вызове !"Kurs2006Line2.New(55, 7)" получим некую картинку.
538
539 NB Картинка уже копируется пустым квадратом: содержимое не копируется,
540 т.к. мы не сумели пока рассказать ББ, что и как копировать. Поэтому такая
541 картинка существует фактически в единств. экземпляре, и ее невозможно
542 хранить в файле между ББ-сессиями.
543
544 Решим задачу: Менять картинку (например, положение конца линии).
545 Новый трюк: Views.Update( v, Views.keepFrames ) обновляет изображение
546 указанного v.
547
548 MODULE Kurs2006LineDialog;
549   IMPORT Properties, Views, Ports, In := FVTsysIn;
550
551   CONST mm = Ports.mm;
552
553   TYPE
554     View* = POINTER TO RECORD ( Views.View )
555       x, y: INTEGER
556     END;
557
558   VAR v*: View;
559
560   PROCEDURE ( v: View ) Restore* ( f: Views.Frame; l, t, r, b: INTEGER );
561   BEGIN
562     f.DrawLine( 0, 0, v.x * mm, v.y * mm, 0 (* толщина *), Ports.blue (*
563     цвет *) )
564   END Restore;
565
566   PROCEDURE ( v: View ) HandlePropMsg- ( VAR p: Properties.Message );
567   BEGIN
568     WITH p: Properties.SizePref DO
569       p.w := 30 * mm; p.h := 30 * mm
570     ELSE
571       END;
572   END HandlePropMsg;
573
574   PROCEDURE New*;
575   BEGIN NEW( v ); v.x := 0; v.y := 0; Views.OpenView( v )
576   END New;
577
578   PROCEDURE Update*;
579   VAR x, y: INTEGER;
580   BEGIN
581     In.Open; In.Int( x ); In.Int( y );
582     v.x := x; v.y := y;
583     Views.Update( v, Views.keepFrames )
584   END Update;
585
586 END Kurs2006LineDialog.
587
588 !Kurs2006LineDialog.New !Kurs2006LineDialog.Update 500 250!
```

580 Полезно нажать Ctrl+Space, чтобы видеть границы вьюшки.
 581 Теперь меняя x, y и вызывая Update -- меняем картинку.
 582
 583 **NB** Если будем менять *текст процедур* и перекомпилировать, то важно не
 584 забывать выгружать старую версию из памяти.
 585
 586 **Упр** Модифицировать модуль так, чтобы рисовать ломаную -- чтобы каждое
 587 нажатие Update добавляло новый сегмент к ломаной (от последней точки до
 588 точки со вновь указ. координатами). Добавляемые точки хранить в списке.
 589
 590 **Сохранение в файле содержимого картинки**
 591 При копировании картинки, скажем, в Log (или спасении/восстановлении из
 592 файла) получается пустой прямоугольник, хотя и правильного размера 70x10.
 593 Ясно, что "состояние" вьюшки никуда не спасается и не копируется.
 594
 595 Чтобы найти, как сохранять информацию о конкретном экземпляре View (x, y),
 596 нужно еще постараться: найти нужное = общая проблема любых сложных
 597 каркасов.
 598
 599 На самом деле "мама всех вьюшек" -- тип Views.View является потомком типа
 600 Stores.Store (как сразу видно из интерфейса), а этот тип специально
 601 предназначен для обеспечения ввода-вывода сложных объектов. Там есть две
 602 нужные нам пустые (по документации) процедуры:
 603
 604 (s: Store) Externalize- (VAR wr: Stores.Writer), NEW, EMPTY;
 605 (s: Store) Internalize- (VAR rd: Stores.Reader), NEW, EMPTY;
 606
 607 **NB** Устройство интерфейса тут аналогично процедуре HandlePropMsg.
 608
 609 Если эти две процедуры переопределить для нашей вьюшки, то получающаяся
 610 картинка спасается в файл и восстанавливается правильно. Но копирование
 611 (скажем, в Log или Word) все равно не работает: получается пустой
 612 прямоугольник, хотя и правильного размера.
 613 Чтобы все работало и здесь, нужно переопределить еще одну процедуру,
 614 предусмотренную в определении Views.View в виде
 615
 616 (v: View) CopyFromSimpleView- (source: Views.View), NEW, EMPTY;
 617
 618 CopyFrom... работает подобно последовательности Externalize, Internalize с
 619 использованием временного файла (посмотреть KursFiles ..Temp()).
 620 Авторы ББ сделали это для эффективности, т.к. здесь выигрыш существенный,
 621 хотя, теоретически говоря, можно было бы обойтись и без этого.
 622 **Упр** Можно ли нам сделать универсальную реализацию процедуры
 623 CopyFromSimpleView, которая бы внутри себя создавала временный файл и
 624 просто писала туда/читала оттуда объект?
 625

```

626 MODULE Kurs2006Line3;
627   IMPORT Properties, Views, Ports, Stores, Fonts;
628
629   CONST mm = Ports.mm;
630
631   TYPE View* = POINTER TO RECORD ( Views.View ) x, y: INTEGER END;
632
633   PROCEDURE ( v: View ) Restore* ( f: Views.Frame; l, t, r, b: INTEGER );
634   BEGIN
635     f.DrawLine( 0, 0, v.x, v.y, 0, Ports.blue )
636   END Restore;
637
638   PROCEDURE ( v: View ) Externalize- ( VAR wr: Stores.Writer );
639   BEGIN wr.WriteInt( v.x ); wr.WriteInt( v.y );
640   END Externalize;
641
642   PROCEDURE ( v: View ) Internalize- ( VAR rd: Stores.Reader );
643   BEGIN rd.ReadInt( v.x ); rd.ReadInt( v.y )
644   END Internalize;
645
646   PROCEDURE ( v: View ) CopyFromSimpleView- ( source: Views.View );
647   BEGIN
648     WITH source: View DO
649       v.x := source.x; v.y := source.y
650     END
651   END CopyFromSimpleView;
652
653   PROCEDURE ( v: View ) HandlePropMsg- ( VAR p: Properties.Message );
654   BEGIN
655     WITH p: Properties.SizePref DO
656       p.w := 70*mm; p.h := 10*mm
657     ELSE
658       END;
659   END HandlePropMsg;
660
661   PROCEDURE New* ( x, y: INTEGER );
662   VAR v: View;
663   BEGIN
664     NEW( v ); v.x := x * mm; v.y := y * mm;
665     Views.OpenView( v )
666   END New;
667
668 END Kurs2006Line3.
669
670 ❶ "Kurs2006Line3.New(100,10)"
671

```

672

673 Получившуюся вьюшку можно
674 — "размножить" (напр., скопировать 5 раз в Log -- когда откроется окошко со
675 вьюшкой, сделать Ctrl+Space -- появится черная рамочка; нажать Ctrl+C; в
676 нужном месте другого документа назвать Ctrl+V);
677 — спасти в файл (посредством обычного виндусового диалога) и восстановить
678 из него (перезапустите Блэкбокс перед восстановлением для драматичности --
679 чтоб уж без обмана);
680 — скопировать в Ворд-документ.

681
682 **Что происходит:**

683 Когда ББ пытается выполнить команду "спасти в файл" (получаемую, скажем,
684 из меню), он смотрит на вьюшку, которую надо спасти (а во всех окошках у
685 нас те или иные вьюшки -- и ничего, кроме разных вьюшек), и вызывает для
686 нее команду wr.WriteStore (предварительно, конечно, этот самый wr:
687 Stores.Writer подсоединяется к нужному файлу).
688 wr.WriteStore определяет с помощью средств модуля Meta (примеры у нас уже
689 были) имя типа в виде цепочки литер и записывает ее в файл, а затем
690 вызывает v.Internalize(wr) для данной вьюшки. В этот момент вьюшка как бы
691 сама записывает свое содержимое в файл.

692 При восстановлении из файла ББ выполняет rd.ReadStore, при этом сначала
693 считывается имя вьюшки (в виде цепочки литер), с помощью Meta создается
694 "пустая" вьюшка (соотв. модуль подгружается, если в памяти его еще нет), и
695 вызывается ее Internalize. Вьюшка восстанавливает свое содержимое из файла.