# Лекция 1

## Визуално програмиране

Визуалното програмиране, включително и Web дизайна е специфична област от софтуерната индустрия, която заема все по-голям дял в областта на програмното осигуряване. В момента то се налага като основно средство за разработване на софтуерни продукти. Визуалното програмиране стана възможно благодарение на няколко важни тенденции в развитието на компютърната индустрия. От една страна, това е изключителният напредък на техническата обезпеченост на компютърните системи (hardware). Компютрите сега разполагат с такива възможности, за каквито и най-големите оптимисти едва ли са мечтали само преди десетина, петнадесет години. С огромната оперативна памет и скорост на работа, сегашните компютри позволяват разработването на сложни графични и мултимедийни приложения, които са в основата на съвременното разбиране за ефективен и дружелюбен софтуер. Програмистите и разработчиците от по-старото поколение, не можеха да си позволят такова разточителство по отношение на памет, бързодействие, а в някои случаи и на неефективност на алгоритми и програми. Ограничените компютърни ресурси в по-старите системи изискваха максимално използване на възможностите на алгоритмите, докато сега това не е водещо изискване.

От друга страна, разработката на операционните системи с графичен интерфейс, промени до голяма степен стила на програмиране. От стандартния процедурен стил се премина към програмиране с управление на събития. От създаването на големи монолитни програмни продукти изпълняващи голям обхват от операции и функции се премина към създаването на малки програмни компоненти, които могат да се използват многократно в различни софтуерни продукти. Така програмните приложения започнаха да се композират от отделни предварително разработени и добре тествани компоненти.

На трето място, голям тласък на визуалните технологии даде развитието на Internet и по-специално World Wide Web, за който някои казват че е Гутенберговата преса на нашето време. Технологията на Web е изградена изцяло на графичното представяне на информацията. При това, поради необходимостта от пренос на данните посредством Internet (където комуникацията не е много бърза), се налага разработването на ефективни и икономични начини за представяне и обработка на информацията.

### 1 Основни принципи на визуално програмиране

Визуалните езици се отнасят към програмните езици от четвърто поколение. Разработването на този тип програмни средства се явява продължение на две важни концепции в развитието на програмните езици. От една страна, това е развитието на обектно-ориентираното програмиране (ООП), което даде възможност за разработване на универсални програмни обекти, осигуряващи многократното им използване в различни програмни приложения. От друга страна, голям тласък за развитие на визуалният подход даде разработването на графичния потребителски интерфейс за операционните системи и големите функционални възможности на свързаният с тях стил на програмиране с управление на събитията. Съчетаването на тези две технологии, даде възможност за разработване на интегрирани програмни среди, които преобразиха работата по създаване на компютърни програми. От една тежка, в много случаи изтощителна рутинна работа, програмирането се превръща в дизайнерска дейност за създаване на ефективни и дружелюбни интерфейсни форми за приложенията. Разбира се и при визуалното програмиране не може да се мине без специфичната дейност на добрите програмисти, но те се освобождават от тежката рутинна работа по създаване на интерфейса на програмите и се оставя само разработването на специфичният за даденото приложение програмен код. Това е качествено нов подход при създаване на програмните приложения. Програмистът пише по-малко програмен код, но това не значи че не му е необходима практика на програмиране, напротив може би му трябва още повече.

Езиците за визуално програмиране предоставят големи възможности за управление на приложенията, тясно свързани с графичният интерфейс на Windows или други операционни среди. Основните характеристики на визуалните програмни среди се определят от спецификата на програмиране с управление на събитията и обектно-ориентирания подход.

### 1.1. Програмиране с управление на събития.

Програмирането с управление на събития се появи още преди появата на графичния потребителски интерфейс GUI (Graphical User Interface). Такъв стил на програмиране може да се осъществи по различен начин. След появата на мишката (посочващо устройство на компютъра) и възможностите, които тя предостави за комуникация с потребителя, програмирането с управление на събития стана всеобща необходимост и за потребители и за разработчици.

До появата на програмирането с управление на събития, процедурният стил на програмиране "отгоре-надолу", се смяташе за най-рационалният начин за писане на приложения. Проектирането на програми в този стил е много ефективно, когато се създават програми с голям обхват на възможни действия и за обработка на голямо количество информация. Програми от този тип изглеждат доста ясни, добре структурирани и лесно се документират. Но подходът "отгоре-надолу", често довежда до прекомерно сложни програми, когато трябва да се организира диалогов стил на управление на работа на програмните приложения. На потребителите в този случай се налага да знаят много повече за стила на програмиране отколкото им е необходимо.

Програмирането с управление на събития не е променило процедурният подход на програмиране – то го допълва със средства, които позволяват да се отдели потребителският интерфейс от специфичната обработка на данните. Визуалните програмни среди използват подхода за програмиране с управление на събития, и по този начин осигуряват възможност на програмиста да се съсредоточи върху логиката на работа на самата програма и да остави грижата за диалога с потребителя на програмната среда.

Концепцията за програмиране с управление на събития е взета от реалния живот. Ежедневно човек се сблъсква с процеси използващи принцип за управление на събития. Такива процеси се срещат навсякъде около нас:

- Сигналът при позвъняване на телефона е събитие, което изисква определени действия. Трябва да се вдигне слушалката и да се проведе разговор. Това се извършва като отговор на събитието позвъняване на телефона.
- Показанието на часовника, отбелязващ времето е знак, че е завършил учебният час и като отговор на това събитие студентите напускат аудиторията.
- Падането на температурата в помещението под $20^oC$ предизвиква включване на нагревателя на интелегентните отоплителни устройства. В този случай сигналът от термостата е събитие, което предизвиква определено действие (включване на отоплителен уред).

Както се разбира от примерите, събитие е такова действие, което протича в реално време и като отговор предизвиква други действия.

Основното при програмирането с управление на събития е идентифицирането на определени действия (събития) и генериране на други действия като отговор на тези събития. При работа в графична среда (Windows), произтичат множество събития. Не всички събития са с еднаква важност. Трябва да се обработват само тези сигнали, които имат значение за съответната програма. Възможно е например, да не е от значение натискане на бутона на мишката, когато показалецът се намира някъде в пространството на екрана, но да е важно програмата да реагира по определен начин, когато е натиснат определен клавиш от клавиатурата. Средата за управление на събития, позволява да се идентифицира натискането на клавиш и да се даде възможност на програмното приложение веднага да премине към изпълнение на логически обособена група операции, която да се свърже с това натискане.

Програмирането с управление на събитията изисква специфичен подход при проектиране и разработване на програмни приложения. Програмистът най-напред обмисля външният вид на програмата (екранните форми за осъществяване на диалога с потребителя) и последователността с който ще се осъществява диалога. След това се разглеждат всички събития, които могат да се случат в процеса на работа на програмата. Такива събития могат да бъдат: разположението на показалеца на мишката в определена област от екрана; натискане на бутона на мишката, позициониране върху определен ред от списък с елементи; натискане на клавиш от клавиатурата и т.н. След идентифициране на възможните събития, за всяко събитие, имащо значения за даденото приложение, се съставя отделна процедура обработваща събитието. Затова, програми създадени за работа в графична среда (Windows), представляват съвкупност от всички процедури, обработващи събитията, които са възможни в процеса на работата им.

### 1.2. Обектно-ориентирано програмиране

Обектно-ориентираното програмиране (ООП) има определени преимущества и дава на програмистите конкретни инструменти за създаване на мощни програмни продукти. То осигурява реални средства за опростяване на разработките и за многократно използване на компоненти в приложенията.

Обектно-ориентираното програмиране превъзхожда процедурното програмиране "отгоре-надолу" по възможностите за създаване на многократно използваем код и за моделиране на ситуации от реалния свят. Появата на Microsoft Windows позволи съчетаването на ООП и GUI и даде големи възможности за улесняване на програмирането в графични среди. ООП опростява решаването на сложните задачи при програмиране и позволява развитието на програмните езици.

На основата на по-голяма част от обикновените програмни езици бяха създадени зарширения за използване на обектно-ориентираните технологии. На основата на език С беше разработен С++, Object Pascal е създаден на основата на Pascal и други.

Въпреки, че ранните ООП системи осигуряваха обектно-ориентирани възможности, те не притежаваха способността за визуално представяне на обектите и управление на взаимодействията с външни събития. Независимо от добрата организация, беше необходимо значително количество програмен код, за да се накарат обектите да се държат както трябва. Обектно-ориентираните езици си оставаха трудни за изучаване и използване от "обикновените програмисти" и това задържаше развитието и широката им употреба.

За преодоляване на този недостатък Microsoft създаде Visual Basic като среда за визуално програмиране, основана на езика BASIC. Въпреки, че във Visual Basic отсъстват указатели и някои други възможности, в него са реализирани много съществени елементи на ООП. Visual Basic стана най-популярният инструмент за програмиране под Windows, демонстрирайки важността на визуалната страна в разработването на програмни продукти. Той обаче, не е истински обектно-ориентиран програмен език и затова възможностите му са ограничени.

С появата на Borland Delphi и Visual C++ стилът на програмиране се промени драстично. Вече не се налага да се избира между визуално програмиране без истински възможности на ООП (Visual Basic) и обектно-ориентирано програмиране без визуални средства. С използването на Object Pascal или C++ в среда на визуално програмиране с характерни за ООП разширения се реализира една революционна комбинация, даваща мощно средство за програмиране.

### 1.3 OLE, COM и ActiveX технологии

През последните години софтуерната индустрия преживява истинска революция. Изграждането и пакетирането на програмни продукти се измести от модела на обширните, монолитни пакети изпълним код към такъв, при който повечето програми се състоят от множество относително малки градивни елементи (модули). В повечето случаи тези

градивни елементи се наричат компоненти и могат да се употребяват при програмиране с различни езици и програмни среди. Тези възможности се реализират благодарение на няколко основни стандартни спецификации, възприети от разработчиците на програмни компоненти за визуални програмни среди. Основните спецификации, които осигуряват този подход са следните:

- **OLE (Object Linking and Embedding)** – Свързване и внедряване на обекти е метод (технология) за споделяне на данни между приложенията. За да се използва OLE, едно приложение трябва да бъде OLE сървър, а друго OLE контейнер.

OLE сървър е приложение, което може да създаде и редактира OLE обект. OLE контейнерът е приложение, което може да съдържа OLE обект. Следователно, OLE обектите са данни и код, споделени между две приложения. Примери за OLE обекти са документи, електронни таблици, картини и други.

- **COM (Component Object Model)** е спецификация на Microsoft за създаване и прилагане на компоненти, които могат да се използват многократно. Могат да се използват различни езици за написване на COM обекти. След създаването си COM обекта може да се използва дори и от програмни среди не включени в кръга на средите с които могат да се разработват такива обекти. COM обектите обикновено се записват в DLL (Dynamic Link Library). DLL обектите могат да имат разширение ".dll' или ".ocx".

COM обектът представлява модул двоичен код, който изпълнява специфични функции. Достъпът до методите на COM обекта се извършва чрез неговия интерфейс. От гледна точка на програмирането, COM обектите са инстанции на класове и много приличат на класовете в Object Pascal или C++. Те се свързват с програмите, които ги използват посредством интерфейс. Всеки интерфейс съдържа съвкупност от методи (функции). Достъпът до COM обектите се осъществява посредством техните методи. COM обектите се имплантират в сървър (динамично свързани библиотеки – DLL), които се обслужват обикновено от операционната система.

COM обектите произхождат от OLE (Object Linking and Embedding) технологията на Microsoft.

- **ActiveX** е сравнително нов термин за програмирането в графични среди. Обектите съдържащи се в ActiveX се наричат контроли и приличат на COM обектите. Основната разлика между контролите на ActiveX и COM обектите е, че контролата на ActiveX има интерфейс, който може да се използва по време на дизайна. Контролите на ActiveX имат код, който позволява разгръщането им върху Web страница или в развойна среда. ActiveX е подмножество на COM.

Контролите на ActiveX представляват набори от функции, които са пакетирани в обект на COM. Този обект е съвсем самостоятелен, но при все това той не притежава способности да бъде стартиран и изпълнен самостоятелно. Контролите на ActiveX могат да бъдат стартирани единствено в контейнер на ActiveX, например в програма на Visual C++, Visual Basic, Delphi, C++ Builder или друга визуална програма.

## 2. Визуално програмиране с Delphi

Delphi е един от най-забележителните продукти в компютърната индустрия. Разработен е от софтуерната компания Borland – един от най-добрите разработчици на програмни среди. Delphi е среда за разработка на програмни продукти, която използва много прогресивни идеи и концепции, заложени в графичният интерфейс на Windows. В основата на Delphi стои програмният език Object Pascal – обектен вариант на популярния програмен език Pascal. Аналогична програмна среда, Borland разработва и за работа с езика C++, която е известна с наименованието C++ Builder и Java с наименование Java Builder.Програмистите твърдят, че Delphi е направил програмирането удоволствие. Това се признава не само от програмистите работещи с Delphi, но и от тези работещи с визуалната програмна среда на Microsoft (Developer Studio) или други програмни инструменти. Тази програмна среда, свързва

визуалните и обектно-ориентирани принципи на програмиране с приятелска среда за разработка. Тя дава на програмистите множество готови компоненти, които могат да се използват многократно. Това позволява да се правят бързи разработки на сериозни, високоефективни приложения за Windows.

### 2.1 Основни характеристики на Delphi

Delphi притежава широк набор от възможности, които определят голямата популярност на този софтуерен продукт. Тук трябва да се отнесат възможностите за проектиране на форми и диалогови прозорци, използване на широко разпространени технологии за създаване на менюта и палитри с инструменти, подържка на всички формати на известните бази от данни и други. Основните характеристики на Delphi са:

- **Наличие на многократно използвани и разширяващи се компоненти**. Delphi осигурява по-голямата част от използваните във Windows компоненти с общо предназначение, като икони, знаци, менюта, бутони, текстови полета, каскадни списъци и дори диалогови прозорци. Например, в много приложения се срещат характерни обекти, като диалоговите прозорци “Open File”, “Save”, “Print” и други. Тези и много други компоненти са вградени в Delphi и могат да се използват от програмистите наготово. Програмната среда на Delphi, позволява тези компоненти да се приспособят към изискванията на разработваната задача. В Delphi има и предварително определени визуални и невизуални обекти, включително и създадени вече прозорци. С тяхна помощ може да се осигури достъп и визуализация на данни само с няколко натискания на бутона на мишката, без да се прибягва да написването и на един ред програмен код. Предоставеният от Delphi внушителен списък от обекти го поставя начело сред системите, които предлагат визуална среда на програмиране.

- **Поддръжка на стандарта VBX (Visual Basic Extensions)**. Delphi дава възможност за интеграция на огромното море от VBX – обекти, създадени от многочислените производители на този тип софтуер. Това става чрез предоставена от програмната среда палитра с компоненти, което облекчава достъпът до тях.

- **Шаблони за приложения и форми**. Delphi предоставя вградени шаблони за форми и приложения, които могат да се използват за бързо създаване на приложни програми. В системата са включени често използвани диалогови прозорци и форми.

- **Настройка на средата за програмиране**. Палитрата с компонентите, редакторът на програмен код, шаблоните за прозорци и приложения са примери за областта в която програмната среда Delphi може да се настройва според желанието и изискванията на потребителя.

- **Компилирани програми**. Някои визуални среди за програмиране извършват компилация само на част от програмата, в резултат на което се генерира P-код. След това се съединяват в едно генерираният P-код и интерпретатор на P–кода за да се получи изпълним модул, или пък генерираният P-код се изпълнява в специална среда. Delphi не използва нито интерпретатор, нито P – код, а създава наистина компилирана програма, готова за изпълнение. Затова създадените от нея програми са толкова бързи, колкото и написаните програми от езиците от трето поколение. Delphi е един от най-бързите инструменти за разработване на приложения, използващи бази данни. Простите Delphi програми могат да се инсталират като единствен изпълним модул, без допълнителни DLL библиотеки.

- **Широки възможности за работа с бази от данни**. В Delphi е вграден механизмът BDE (Borland Database Engine) за работа с бази от данни. BDE е внимателно обмислена система, която представлява междинен слой за достъп до всички популярни формати на съвременни бази от данни. Той се използва и в системите “клиент/сървър”, като осигурява достъп до продукти от типа на Sybase SQL Server, Microsoft SQL Server, Oracle и Borland InterBase. Всеобщо е мнението, че BDE превъзхожда значително продукта на Microsoft

ODBC, давайки предимство в производителността за сметка на по-тясната връзка с форматите на базите от данни.

**2.2 Основни понятия и термини**

Всяка област от знанието има определен набор от общоприети термини и понятия, които улесняват представянето и възприемането на специфичната материя. Тази терминология дава на участниците, възможност да контактуват помежду си и да разменят информация. За правилното възприемане на същността на програмната среда Delphi е необходимо уточняването на следните понятия:

***А. Среда (Environment).*** Този термин се използва в два различни контекста, включващи програмна среда и среда за разработване. В тях има малки различия. Терминът "програмна среда", се използва по-често, когато се има пред вид работа с команден или пакетно ориентиран компилатор. При него, средата е необходима не само в процеса на разработване, но и при изпълнението на програмата. Разработваната програма не представлява самостоятелно приложение, тъй като изисква наличие на програмна среда за да се изпълни. Средите за разработка се намират едно стъпало по-високо по отношение на предлагания завършен набор от инструменти в рамките на една програма. Разработеното с тях приложение може да работи без наличие на разработващата среда. Обикновено такова приложение представлява напълно компилиран и завършен модул който работи самостоятелно.

***Б. Изключение (Exception).*** Изключенията са непланирани грешки. Дори и в най-добрите програми, съществуват събития, които са извън контрола на програмиста. Потребителите правят грешки. Например, потребителят може да въведе погрешна стойност в някое поле за данни, или да отвори файл, който не се подържа от програмата. Програмата трябва да е подготвена за всевъзможни сценарии да се обработва даден тип грешка винаги, когато е възможно. Не може да се предвиди всяка стъпка на потребителите, но могат да се предвидят най-често срещаните грешки и да се предприемат съответните действия.
Обработката на изключенията е елегантна форма за отстраняване на грешки. Обикновено, тази технология се използва от съставителите на компоненти за ползване при визуалното програмиране. При писането на малки програми, не е необходимо писането на програмен код за обработка на грешките (той може да се окаже по-голям от самата програма).

***В. Обект.***Понятието "обект" се използва като взаимозаменяем с термина "компонент" в средата на Delphi. По-често този термин се използва като компонент или група компоненти използвани в терминологията на визуалното програмиране. При разработката на приложения, в термина компонент, акцентът се поставя върху инструменталният аспект (разглежда се като инструмент за обработка на данни).
Компонентите се включват в програмата с помощта на палитрата на компоненти, която се предоставя от програмната среда. По време на изпълнение на програмата, компонентът е по-близо по смисъл до обект от терминологията на обектно-ориентираното програмиране, отколкото до "компонент" от визуалното програмиране.
Понятието "обект" претърпя известно развитие в сравнение с времето, когато се използваше като инстанция (екземпляр) от даден клас в обектно-ориентираното програмиране. Сферата на приложение на това понятие се разшири при въвеждането на визуалният принцип на програмиране. Поради тази причина, сега много софтуерни разработчици поставят етикета обектно-ориентирано приложение, въпреки, че то е само програмиране с управление на събития или се основава на обекти.
Разглеждан като инструмент за разработване на приложения в Delphi, обектът е елемент с който може да се извършват някакви действия. В Delphi и другите визуални програмни среди

това са бутони, знаци, прозорци за текстови съобщения, списъци, менюта и други. Следователно, "обект" е доста широко понятие и не е нужно да се описва подробно – достатъчно е да се каже, че в него се включват елементите от които се изгражда програмата. Подробностите за всеки конкретен обект могат да се уточняват в зависимост от контекста в който се използва. За средата на Delphi, под обект може да се разбира прост елемент (рамка, бутон, текстово поле) или сложна форма с множество елементи. Автономен модул, който получава съобщения също може да се разглежда като обект.

Визуалното програмиране добавя ново измерение при създаването на програмни продукти, като дава възможност за **изобразяване на обектите върху екрана преди изпълнението на програмите (в процеса на проектиране на интерфейса**). Без визуалното програмиране процесът на възпроизвеждане на даден обект изисква написването на големи фрагменти програмен код, който създава и настройва обекта на "място". Кодираните обекти и компоненти могат в този случай да се видят само при стартиране (изпълнение) на програмата. При такъв подход, за да могат обектите да се настроят да изглеждат и се държат според изискванията се изисква уморителен процес на многократно поправяне на програмният код.

Благодарение на средствата за визуална разработка, може да се работи с обектите, като се държат пред погледа на програмиста и на практика се получава резултат още в процеса на проектирането. Способността обектът да се вижда такъв, какъвто изглежда по време на изпълнение на програмата, премахва необходимостта от множество ръчни операции, характерни за работа без визуални средства, независимо дали е обектно-ориентирана или не. След като обектът е поставен във формата на средата за визуално програмиране, всичките му атрибути (размер, положение на екрана, свойства и т.н.) се записват като програмен код, съответстващ на обекта, автоматично от програмната среда.

Разполагането на обектите в Delphi, е свързано с по-тесни отношения между обектите и реалният програмен код. Обектите се разполагат във формата и съответстващият за тях програмен код се записва в изходния файл с текста на съответният програмен модул. Този код се компилира и осигурява по-голяма производителност в сравнение с визуалните среди, при които информацията се интерпретира по време на изпълнение на програмата.

Компонентите в Delphi включват **визуални и невизуални** типове. Визуалните типове се появяват по един и същ начин във формите, както по време на проектирането, така и при изпълнение на приложението. Невизуалните компоненти не са видими по време на изпълнение на програмите. Голяма част от компонентите за работа с бази от данни са невизуални компоненти, защото те не предизвикват някакъв видим ефект във формата, но имат важни функции за осъществяване на връзка с данните или организират управлението им.

### *Г. Свойство*

Освен способността да реагират на събития , обектите имат и свойства. Наред с различните характеристики, свойствата включват и такива признаци, като цвят, височина, ширина, положение върху екрана и други. Те указват не само как изглежда обектът, но и определят поведението му посредством някои невизуални характеристики.

Свойствата се отнасят към обекта, така както локалните променливи се отнасят към подпрограмата (процедурата). Локалните променливи принадлежат на процедурата и се използват в процеса на изпълнението й. По подобен начин свойствата са непосредствено свързани с обекта – като атрибути, които описват подробно структурата му.

Изменението на локалните променливи вътре в процедурата влияе на хода на нейното изпълнение. По подобен начин изменението на свойствата на обекта влияе на самият обект. Свойствата принадлежат на обекта, но за разлика от локалните променливи те не могат да бъдат "видими" за външни обекти. Преглеждането им дава възможност да се изследва съдържанието и специфичните особености на обекта разглеждан отвън.

Може да се влияе на поведението на обекта по време на изпълнение на програмата или в стадия на проектирането, като изменяте свойствата му. В даденият по-горе пример на обекта Human може да се промени адреса, като се присвои нова стойност на свойството Address.

### *Д. Събитие*

Програмите на Delphi използват методите за управление на събитията за организиране на взаимодействието между програмното приложение и потребителя. Обкръжението включва действията на потребителя и действията на самата операционна система. Изпълнението на голяма част от кода, който се пише в Delphi се предизвиква от събития, генерирани от потребителя или от системата. В термините на Delphi процедурата, която се изпълнява при възникване на събитие, се нарича обработчик на събитие (event handler).

За да се изясни динамиката на обработването на събитията в програмната среда Delphi, можем да се разгледат функциите в една фирма с нейните служители. Всеки служител има длъжност и изброени служебни задължения, които определят уменията му, нивото на отговорност и ролята му във всекидневната дейност на компанията. Компанията взаимодейства с външният свят чрез последователност от събития – например телефонни разговори, писма, факсове, командировки, срещи и т.н. Всяко събитие се обработва от съответният служител. Например, ако събитието е позвъняване по телефона на определен служител, то той е отговорен за вдигане на телефонната слушалка и провеждане на разговора – той е обработчик на събитието.

Аналогично на горният пример, Delphi отговаря на съобщения – сигнали от мишката, клавиатурата и системата, привеждайки в действие съответните логически фрагменти – процедури. Ако процедурата не е свързана с дадено събитие, то при възникването му тя го игнорира.

Жизненият цикъл на събитие, възникнало при натискане на показалеца на мишката върху бутон с наименование *btnOk* например, изглежда така:

1. Възниква събитие – натискане на бутон от потребителя.
2. Обектът *btnOk* идентифицира натискането като събитие, което влиза във функционалните му задължения и трябва да бъде обработено.
3. Delphi търси процедура с име, съответстващо на името на задействания обект (*btnOk* ) и името на събитието (*Click* – натискане с бутон на мишката). В случая ще бъде изпълнена процедурата *btnOkClick ()*.

Програмирането с управление на събития изисква от програмната среда да генерира подходяща структура, която постоянно да следи за възникване на събития по време на изпълнение на програмата. Това може да бъде организирано чрез програмен код, представен например с помощта на известните в езика Pascal оператори **Repeat … Until** и **Case** както е показано в примера.

```
{Пример – организация на следене на
възникнали събития}
Repeat
   GetNextEvent (NewEvent) ;
   Case NewEvent of
                Event1:  HandleEvent1() ;
                Event2:  HandleEvent2() ;
                Event3:  HandleEvent3() ;
                EndApplication:
                        Begin
                                HandleExit() ;
                                Break;
                        End;
                Else  IgnoreEvent();
        End ;
Until False;
```

Този фрагмент представлява пример за програма с организация на управление на събитията с помощта на цикъл. След получаване на събитието операторът **Case** проверява включено ли е то в списъка на събитията обработвани от приложението.

Ако това не е така, събитието се игнорира. Процедурите HandleEvent1, HandleEvent2, HandleEvent3 се явяват обработчици на събития (означени с етикети Event1, Event2, Event3).

Голяма част от кода, който трябва да се

пише в Delphi, представлява процедури за обработка на събития. Delphi освобождава програмистите от явното създаване на циклите за управление на събитията (блокът Repeat.... Until). Системата сама се грижи за извикването на процедурите за обработка на събитията посредством така наречената обвивка за работа с процедурите (event-driven framework).

### *Е. Обекти и събития*

От изложеното до тук става ясно, че събитията възникват при действия от страна на потребителя или системата. С помощта на Delphi може лесно да се свърже събитие с програмен код, който да се използва за управление на потребителски интерфейс (UI).
Като правило всички обекти могат да реагират на събития. Въпреки че някои прости обекти, като бутоните и текстовите полета, са способни да реагират само на малък брой събития, съществува възможност да се избере някое от следните действия:

-**Да се игнорира събитието**. Поведението на обектите по подразбиране (при разполагането им във формата) е да игнорира всички събития. Програмистът решава, кои от събитията трябва да се обработят и кои да се игнорират.

-**Да се прихване събитието**. За целта събитието трябва да се свърже с обработчик на събитие (процедура), след което да се настрои, модифицира или измени поведението на обекта, зададено по подразбиране.Изменение на поведението на обекта може да се изрази в промяна на вида на показалеца на мишката при попадение в областта на обекта или в извеждане на съобщение за потребителя за предстоящо активиране на дадена функция при натискане на бутона на мишката.Разглежданите варианти не се реализират от програмната среда за визуално програмиране. Потребителят сам трябва да реши какви обработчици на събития трябва да създаде и как да модифицира съответните обекти в процеса на работа. За целта, той трябва да има много точна представа за това какво се иска от програмата във всеки момент от изпълнението й. Преди да се започне работа по реализацията на програмата трябва да се уточни какъв брой форми ще са необходими и какви елементи ще съдържа всяка форма.