УДК 004.42

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ГРАФИЧЕСКИХ РЕШЕНИЙ ПРИ РАЗРАБОТКЕ ПОЛЬЗОВАТЕЛЬСКИХ ПРИЛОЖЕНИЙ В ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ СРЕДАХ ПРОГРАММИРОВАНИЯ

Суин И.А., Козлов С.В.

Смоленский государственный университет, Смоленск, e-mail: suin224@yandex.ru

В статье затронута проблема организации работы с объектами графического типа при разработке пользовательских приложений. Автором рассмотрены особенности взаимодействия графических объектов при программировании приложений на форме. Обсуждаются возможности программной реализации математических подходов средствами среды объектно-ориентированного программирования. На примере игрового приложения DX-Ball проанализированы программные аспекты реализации взаимодействия графических элементов на форме при организации движения в среде Lazarus. На основании проведенного анализа автором предложено корректное решение реализации физики взаимодействия шарика с остальными графическими объектами на форме. А именно предложена программная реализация изменения положения шарика при его соударении с блоками и границами формы как графическими элементами. В завершении статьи охарактеризованы иные проблемы, которые возникают при работе с графикой на форме при разработке пользовательских приложений в объектно-ориентированных средах программирования.

Ключевые слова: информатика, программирование, информационно-коммуникационные технологии, графика, пользовательское приложение, объектно-ориентированное программирование, алгоритм

FEATURES OF REALIZATION OF GRAPHIC SOLUTIONS IN THE DEVELOPMENT OF USER APPLICATIONS IN OBJECT-ORIENTED PROGRAMMING ENVIRONMENTS

Suin I.A., Kozlov S.V.

Smolensk state University, Smolensk, e-mail: suin224@yandex.ru

The article touches a problem of work organization with image objects while developing custom applications. Features of graphic objects interaction during form application programming has considered. Software implementation capabilities with mathematical approaches using methods object-oriented programming. Program aspects of graphic elements interaction realization during organization of the movement environment were analyzed for form application at Lazarus based on game application DX-Ball. Proper solution of interaction physics implementation with other graphical objects, especially ball's position changing implementation during collisions with blocks or form's border, was suggested based on in-app module analysis. There's description of potential problems with graphics during form application's design with object-oriented integrated development environment.

Keywords: informatics, programming, information and communication technologies, graphic, user application, object-oriented programming, algorithm

В настоящее время современные объектно-ориентированные среды программирования предоставляют пользователю широкий выбор инструментов по разработке проектов различного типа. Программист может создавать как стандартные консольные приложения, проекты на форме, так и достаточно узконаправленные приложения, например, текстовое приложение FCPUnit или программы InstantFPC. В тоже время в независимости от типа проекта каждый из них должен отвечать ряду базовых

принципов. Одним из таких важнейших принципов является принцип наглядности, в соответствии с которым информация представленная в проекте должна легко зрительно восприниматься любым пользователем. Особенно это важно в программных продуктах социальной направленности [8] и обучающих программных средствах [9]. Высокая степень зрительного восприятия достигается в программных продуктах за счет графических элементов интерфейса и диалоговых компонентов программных продуктов.

Как известно в объектно-ориентированных средах программирования очень большое внимание уделяется работе с графическими объектами, а также процессу взаимодействия с ними. Большинство современных приложений представляют собой проекты на форме, на которой могут быть одновременно размещены графические объекты различного класса [4]. Например, программист может помещать нужные ему изображения в такие объекты как:

- TBitBtn – кнопка, на которую можно нанести изображение;

- TImage – объект, предназначенный непосредственно для помещения изображения на форму;

- задний фон формы при помощи функции self.canvas.draw (рис. 1).

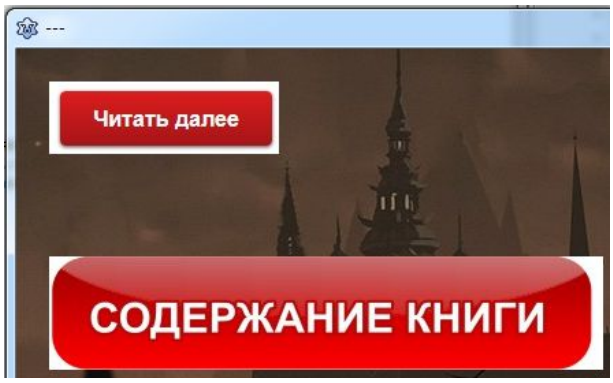

Рис. 1. Графические элементы в среде Lazarus

Подобного рода объекты существуют в каждой современной объектно-ориентированной среде программирования. Это позволяет разрабатывать приложения с качественным графическим интерфейсом. Для этого можно использовать как более простые, так и более мощные программные среды. От сред программирования MS Visual Studio и Embarcadero Studio до бесплатно распространяемой среды разработок программного обеспечения Lazarus.

В тоже время, несмотря на возможности используемой объектно-ориентированной среды программирования, наличие большого количества объектов на форме может негативно сказаться на производительности приложения. В качестве примера можно привести игровое приложение DX-Ball. Цель компьютерной игры состоит в том, чтобы, управляя движением шарика на поле, разбить им все имеющиеся блоки (рис. 2).

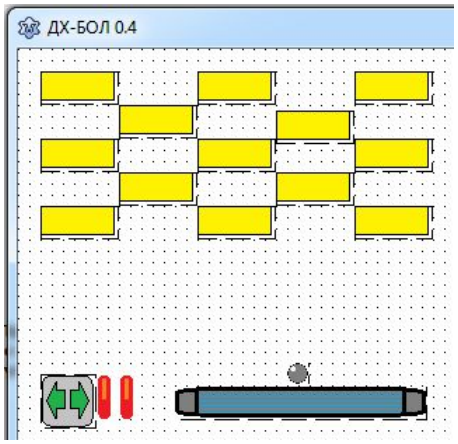

Рис. 2. Схематические изображение DX-Ball

Если размещать каждый блок на форме, как отдельный объект класса Image, то это обернется не только большой нагрузкой на ресурсы компьютера, но и необходимостью прописывать внутри программы избыточно большое количество повторяющегося программного кода. В связи с этим предложим один из вариантов решения данной проблемы, которая возникла при реализации проекта DX-Ball в объектно-ориентированной среде разработок Lazarus.

В начале производится расчет количества блоков, которое может быть помещено на имеющуюся форму по горизонтали и вертикали. В примере на рисунке 3 число блоков по горизонтали равно восемнадцати, а по вертикали равно восьми. Таким образом, с помощью рядов блоков построчно задают изображение (рис. 3).

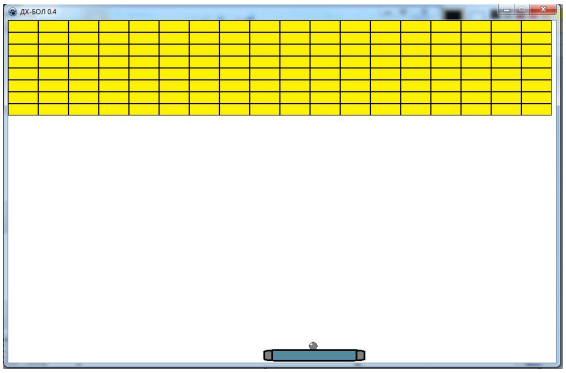

Рис. 3. Число блоков на экране

На основе данной информации создается двумерный массив, размером восемнадцать на восемь. В дальнейшем эта матрица и будет являться «блоками» на экране. По умолчанию все элементы данной матрицы являются единицы.

При запуске программы соответствующая процедура последовательно «рисует» на форме восемнадцать блоков, начиная с левого верхнего угла. Далее программа переходит на следующий ряд и вызывает процедуру снова. Так повторяется восемь раз. Таким образом, ни один блок не является графическим объектом, по сути это просто задний фон формы.

Возникает логичный вопрос: «Каким образом происходит взаимодействие графического объекта «шарик» с блоками?» Именно с этой целью и был создан двумерный массив восемнадцать на восемь, исходно заполненный одними единицами. Каждый раз когда объект шарика попадает в ту часть формы, в которой могут находиться блоки (расстояние равное восемь умноженное на число рядов блоков пикселей от верха формы) программа начинает определять, на месте какого элемента начальной матрицы сейчас находится шарик. Если на соответствующей позиции в матрице стоит единица, значит, в данный момент происходит процесс столкновения шарика с блоком.

Такое стечение обстоятельств «провоцирует» выполнение следующей части программы. Траектория шарика изменяется, соответствующий элемент матрицы заменяется нулевым значением. На место, где раньше был блок, помещается новое изображение блока соответствующего размера, одного цвета с цветом заднего фона формы. Также общее число оставшихся блоков, которое для удобства игрока всегда отображается на небольшой соседней форме, уменьшается на единицу. Игра заканчивается, когда это число достигнет нуля. Таким образом, существующий объект (шарик) взаимодействует не с подобными себе объектами, а с элементами двумерной матрицы, в прямом взаимодействии с которой находятся изображения на заднем фоне формы.

Приведем алгоритмическое решение описанных действий средствами объектно-ориентированной программной среды Lazarus (рис. 4).

```
475  procedure TForm1.TimerSHARTimer(Sender: TObject);  // движение шара

     if SHAr.Top<(j*image.Height) Then       // проверка на столкновение с блоком
       begin
       sbl:=(SHAR.top div image.Height)+1;   //определение позиции блока
       lbl:=(SHAR.Left div image.Width)+1;
480         if t[lbl,sbl] <> 0 then           // варианты действий
             begin
              case t[lbl,sbl] of
                 1:begin         // обычый блок - удаляем
                   t[lbl,sbl]:=0;
485                if nas<>1 then
                    begin
                    dy:=-dy;
                    SHAR.Top:=SHAR.Top + dy;
                    end;
490                img:=TPicture.Create;
                   img.LoadFromFile('img\s0.bmp');
                   self.canvas.draw((lbl-1)*image.width,(sbl-1)*image.Height,img.Graphic);

                    if maf=1 then
495                 mak:=1;

                    ost:=ost - 1; // всего блоков стало меньше
                    form4.OSTb.Caption:=inttostr (ost); // переносим на счет

500                if ost=0 then  // ПОБЕДА
```

Рис. 4. Часть кода процедуры столкновения шарика с блоком

Безусловно представленное решение рассматриваемой проблемы обладает, как и всякое другое решение, определенными достоинствами, так и недостатками. Главным достоинством программы можно считать упрощенную процедуру написания программного

кода. Также к плюсам можно отнести сведение к минимуму количества графических элементов на форме.

К недостаткам, характерным для многих приложений подобного рода, прежде всего, относится некорректное взаимодействие шарика с блоками. Дело в том, что шарик «не видит», с какой частью блока он сталкивается. По законам логики и физики – в случае столкновения с верхней или нижней гранью шарик должен отскакивать в противоположную сторону по горизонтали. В случае же столкновения с одной из боковых сторон, он должен отскакивать в противоположную сторону по вертикали. В данной игре шарик «распознает» сам факт столкновения и всегда отскакивает по горизонтали.

Другой проблемой является процесс обновления блоков. Как уже было описано ранее – все блоки являются не более чем набором цветных пикселей на форме. Из-за этого, когда шарик «проходит» по картинке блока, но не разрушает его, он стирает часть этого блока (рис. 5).

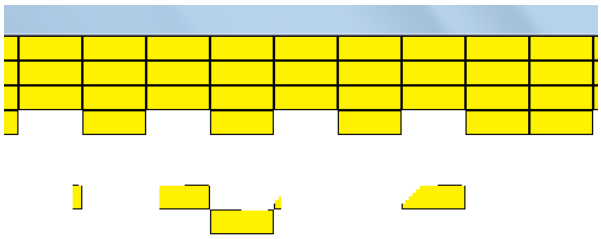

Рис. 5. Пример частичного «стирания» блоков

Последнюю проблему частично удалось решить написанием процедуры, которая каждые несколько секунд перерисовывает не разрушенные блоки (рис. 6).

```
 97 procedure TForm1.OBNOVATimer(Sender: TObject);
  . var
  .   k,ver,lev,her:integer;
100 begin
  .   her:=1;
  .   k:=1;
  .   While her<=j do
  .    begin
105       case t[k,her] of
  .           1:begin
  .             FreeAndNil(image);
  .             image:=TPicture.Create;
  .             image.LoadFromFile('img\s1.bmp');
110           ver:=image.Height*(her-1);
  .             lev:=image.Width*(k-1);
  .             self.Canvas.Draw(lev,ver,image.Graphic);
  .             end;
```

Рис. 6. Процедура перерисовки не разрушенных блоков

Все перечисленные недостатки могут быть математически, а, следовательно, и программно решены. Так, например, возможно применить в решении методы

функционального анализа [3], инварианты теории графов [2] или методы теории экспертных систем [7]. Использование того или иного метода предопределяет общая цель разработки программного средства и требования, которые предъявляются к нему заказчиком. Они, определяют направления для дальнейшего совершенствования программы. Подчеркнем, что программное решение можно осуществить не только средствами объектно-ориентированных сред программирования, но в специализированных системах моделирования, например, UNITY3D [1] или Advanced Tester [5]. Также возможно использование средств сервис-ориентированных систем [6]. Решение, приведенное в данной статье, демонстрирует наличие различных способов взаимодействия с графическими элементами посредством программного кода в объектно-ориентированных средах программирования. В завершение заметим, что для анализа ситуации с графическими объектами на форме проекта можно применить и иные методологии математического моделирования и универсальных программных решений.

Список литературы

1. Ковалева И. В., Баженов Р. И. Разработка двухмерной игры в системе трехмерного моделирования UNITY3D // Перспективные информационные технологии (ПИТ 2017): сборник трудов международной научно-технической конференции. – 2017. – С. 790-792.

2. Козлов С. В. Интерпретация инвариантов теории графов в контексте применения соответствия Галуа при создании и сопровождении информационных систем // International Journal of Open Information Technologies. – 2016. – Т. 4. № 7. – С. 38-44.

3. Козлов С. В. Применение методов функционального анализа при формировании оптимальных стратегий обучения школьников // Международный журнал экспериментального образования. – 2016. – № 3-2. – С. 182-185.

4. Козлов С. В. Система индивидуального тестирования «Комплекс измерения обученности» // Системы компьютерной математики и их приложения. – Смоленск: СмолГУ, 2007. – С. 223-225.

5. Козлов С.В. Функциональные назначения и возможности информационно-образовательного ресурса «ADVANCED TESTER» // Горизонты науки. – 2011. – № 2. – С. 9.

6. Максимова Н. А. Разработка приложений на основе сервис-ориентированных систем // NovaInfo.Ru. – 2016. – Т. 3. № 46. – С. 41-44.

7. Размахнина А. Н., Баженов Р. И. О применении экспертных систем в различных областях // Постулат. – 2017. – № 1 (15). – С. 38.

8. Сенчилов В. В. Возможности программного обеспечения при дистанционном обучении математике детей с особыми образовательными потребностями / Сенчилов В. В., Быков А. А., Киселева О. М., Тимофеева Н. М. // Евразийское Научное Объединение. – 2017. – Т. 2. № 8 (30). – С. 111-112.

9. Тимофеева Н. М., Киселева О. М. О применении программных средств в процессе обучения // Системы компьютерной математики и их приложения. – Смоленск: Изд-во СГПУ, 2005. – С. 233-235.