МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительной математики и информатики**

**КУРСОВАЯ РАБОТА**

**ДИНАМИЧЕСКОЕ ИЗОБРАЖЕНИЕ. ДВИЖЕНИЕ ШАРИКА С ЗАТУХАЮЩЕЙ АМПЛИТУДОЙ**

Работу выполнил Матунов А.А.

 (подпись, дата)

Факультет Математики и компьютерных наук курс 2 2 Направление 02.03.01 Математика и компьютерные науки

Научный руководитель

преподаватель К.Ю. Дуриндина

 (подпись, дата)

Нормоконтролер

преподаватель А.А. Цыбенко

 (подпись, дата)

Краснодар 2018

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc514251415)

[1 Теоретическая часть 4](#_Toc514251416)

[1.1 Из истории PascalABC.NET 4](#_Toc514251417)

[1.2 Графический модуль PascalABC.NET 6](#_Toc514251418)

[2 Разработка алгоритма 20](#_Toc514251419)

[3 Тестовые примеры 23](#_Toc514251420)

[Заключение 27](#_Toc514251421)

[Список использованных источников 28](#_Toc514251422)

ВВЕДЕНИЕ

Была поставлена задача: программно реализовать физическую модель движения шарика по дуговой траектории с постепенным «затуханием».

Цель:составить алгоритм для реализации программного кода, решающего поставленную задачу, и разработать сопроводительную документацию к нему.

Задание включает в себя следующие виды работ:

1) пошаговая разработка алгоритма и его описание;

2) описание алгоритма на языке программирования;

3) отладка программы;

4) демонстрация возможностей программы;

5) составление документации.

В качестве языка программирования был выбран язык PascalABC.NET.

## 1 Теоретическая часть

## 1.1 Из истории PascalABC.NET

PascalABC.NET – это язык Delphi (Object Pascal) для платформы Microsoft.NET, содержащий все основные элементы современных языков программирования: модули, классы, перегрузку операции, интерфейсы, исключения, обобщенные классы, сборку мусора, а также некоторые средства параллельности, в том числе директивы OpenMP. Система PascalABC.NET включает в себя также простую интегрированную среду, ориентированную на эффективное обучение программированию.

Язык Паскаль был разработан Никлаусом Виртом в 1970 г. как язык со строгой типизацией и интуитивно понятным синтаксисом. В 80–е годы наиболее известной реализацией стал компилятор Turbo Pascal фирмы Borland, в 90–е ему на смену пришла среда программирования Delphi, которая стала одной из лучших сред для быстрого создания приложений под Windows. Delphi ввела в язык Паскаль ряд удачных объектно–ориентированных расширений, обновленный язык получил название Object Pascal. Из альтернативных реализаций Object Pascal следует отметить много – платформенный open source компилятор Free Pascal.

Основная причина создания PascalABC.NET – необходимость в современной простой и мощной бесплатной среде для языка Object Pascal, опирающейся на передовую платформу программирования Microsoft.NET. Платформа .NET обеспечивает язык PascalABC.NET огромным количеством стандартных библиотек и позволяет легко сочетать его с другими .NET–языками: C#, Visual Basic.NET, управляемый C++, Delphi.NET, Delphi Prism и другие Платформа .NET предоставляет также такие языковые средства, как единый механизм обработки исключений, единый механизм управления памятью в виде сборки мусора, а также возможность свободного использования классов, наследования, полиморфизма и интерфейсов между модулями, написанными на разных .NET–языках.

Язык PascalABC.NET близок к реализации Delphi (Object Pascal). В нем отсутствует ряд специфических языковых конструкций Delphi, таких как открытые массивы, записи с вариантами, Variant–тип и тому подобное, некоторые конструкции изменены (не требуется писать overload, иной синтаксис перегрузки операций, вместо процедурных переменных используются делегаты .NET, интерфейсы реализованы в стиле C# и так далее). Кроме этого, добавлен ряд возможностей: можно описывать переменные внутри блока, методы в записях, подпрограммы с переменным числом параметров, имеются операции +=, –=, \*=, /=, в секции uses можно подключать пространства имен .NET, методы можно описывать непосредственно в теле класса или записи, поля классов можно инициализировать при описании, множества set могут быть на базе произвольных типов, введены операторы foreach и lock, переменные циклов for и foreach можно описывать непосредственно в заголовке цикла, имеются обобщенные классы (generics) и другие.

Наиболее близким по идеологии к PascalABC.NET является язык Object Pascal системы CodeGear DelphiPrism. Однако он сильно изменен в сторону .NET: нет глобальных описаний, все описания помещаются в класс, содержащий классовый метод Main, отсутствует ряд стандартных подпрограмм языка Паскаль. Кроме того, система Delphi Prism – платная (за исключением консольного компилятора) и не содержит собственной оболочки (встраивается в Visual Studio).

Интегрированная среда PascalABC.NET обеспечивает подсветку синтаксиса, подсказку по коду (подсказка по точке, подсказка параметров подпрограмм, всплывающая подсказка по коду), форматирование текста программы по запросу, переход к определению и реализации имени, элементы рефакторинга.

Внутренние представления PascalABC.NET позволяют создавать компиляторы других языков программирования и встраивать их в среду с помощью специальных плагинов.

## 1.2 Графический модуль PascalABC.NET

Возможности модуля растровой графики GraphABC практически совпадают с графическими возможностями Borland Delphi. Процедуры и функции рисования и установки параметров рисования аналогичны методам и свойствам класса TCanvas в Delphi. К примеру, вместо свойства Canvas.Brush.Color используется пара: процедура SetBrushColor(color) и функция BrushColor.

Модуль GraphABC представляет собой простую графическую библиотеку и предназначен для создания несобытийных графических и анимационных программ в процедурном и частично в объектном стиле. Рисование осуществляется в специальном графическом окне, возможность рисования в нескольких окнах отсутствует. Кроме этого, в модуле GraphABC определены простейшие события мыши и клавиатуры, позволяющие создавать элементарные событийные приложения. Основная сфера использования модуля GraphABC – обучение.

Модуль GraphABC основан на графической библиотеке GDI+, но запоминает текущие перо, кисть и шрифт, что позволяет не передавать их в качестве параметров при вызове графических примитивов. К свойствам пера, кисти и шрифта можно получать доступ как в процедурном, так и в объектном стиле. К примеру, для доступа к цвету текущего пера используется процедура SetPenColor(c) и функция PenColor, а также свойство Pen.Color.

В модуле GraphABC можно управлять самим графическим окном и компонентом GraphABCControl, на котором осуществляется рисование. По умолчанию компонент GraphABCControl занимает всю клиентскую часть графического окна, однако, на графическое окно можно добавить элементы управления, уменьшив область, занимаемую графическим компонентом (к примеру, так сделано в модулях Robot и Drawman).

Для работы с рисунками используется класс Picture, позволяющий рисовать на себе те же графические примитивы, что и на экране.

Режим блокировки рисования на экране (LockDrawing) позволяет осуществлять прорисовку лишь во внеэкранном буфере, после чего с помощью метода Redraw восстанавливать все графическое окно. Данный метод используется для ускорения анимации и создания анимации без мерцания.

Модуль GraphABC содержит константы, типы, процедуры, функции и классы для рисования в графическом окне. Онᴎ подразделяются на следующие группы:

1. графические примитивы;
2. действия с цветом;
3. действия с пером;
4. действия с кистью;
5. действия со шрифтом;
6. действия с рисунками;
7. действия с графическим окном;
8. задание режимов вывода;

Наиболее используемые графические процедуры и функции.

Управление экраном.

SetWindowWidth(w) – устанавливает ширину графического окна;

SetWindowHeight(h) – устанавливает высоту графического окна.

Графические примитивы.

Точка SetPixel(x,y,color) – закрашивает один пиксел с координатами (*x*,*y*) цветом color. К примеру, SetPixel(300,200,clred).

Линии Line(x1,y1,x2,y2) – рисует отрезок с началом в точке (*x*1,*y*1) и концом в точке (*x*2,*y*2). К примеру, line(100,50,500,250).

Процедура MoveTo(x,y) – передвигает невидимое перо к точке с координатами (*x*,*y*). Эта процедура обычно работает в паре с процедурой LineTo(x,y).

Процедура LineTo(x,y) – рисует отрезок от текущего положения пера до точки (*x*,*y*). Координаты пера при этом также становятся равными (*x*,*y*).

Rectangle(x1,y1,x2,y2) – рисует прямоугольник, заданный координатами противоположных вершин (*x*1,*y*1) и (*x*2,*y*2). К примеру, Rectangle(50,50,200,200).

Процедура Polygon(А, n) – строит ломаную линию по *n* точкам, координаты которых заданы в массиве А элементов типа Point.

Процедура Polyline(А, n) – строит замкнутую ломаную линию по *n* точкам, координаты которых заданы в массиве А элементов типа Point.

Окружность Circle(x,y,r) – рисует окружность с центром в точке (x,y) и радиусом r. К примеру, Circle(500,200,100).

Дуга окружности Arc(x,y,r,a1,a2) – рисует дугу окружности с центром в точке (*x*,*y*) и радиусом *r*, заключенной между двумя лучами, образующими углы *a*1 и *a*2 с осью *OX* (*a*1 и *a*2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки). К примеру, Arc(300,250,150,45,135).

Некоторые используемые цвета.

Таблица 1 – Палитра цветов

|  |  |  |  |
| --- | --- | --- | --- |
| Код | Название | Код | Название |
| clblack | черный | clGreen | зеленый |
| clPurple | фиолетовый | clBrown | коричневый |
| clWhite | белый | clBlue | синий |
| clMaroon | темно–красный | clSkyBlue | голубой |
| clRed | красный | clYellow | желтый |
| clNavy | темно–синий | clCream | кремовый |
| clAqua | бирюзовый | clFuchsia | сиреневый |
| clOlive | оливковый | clTeal | сине–зеленый |
| clSilver | серебряный | clLime | ярко–зеленый |
| clMoneyGreen | цвет зеленых денег | clLtGray | светло–серый |
| clDkGray | темно–серый | clMedGray | серый |

Random(16777215) – случайный цвет из всей палитры цветов Паскаля

Функция RGB(r,g,b) возвращает целое значение, являющееся кодом цвета͵ который содержит r красную, g зеленую и b синюю составляющие (r, g и b – целые в диапазоне от 0 до 255, причем, 0 соответствует минимальной интенсивности, 255 – максимальной). К примеру, RGB(Random(255), Random(255), Random(255)), где функция Random(х) возвращает случайное целое в диапазоне от 0 до (*x*–1).

Процедура цвет линии SetPenColor(color) устанавливает цвет пера, задаваемый параметром color. К примеру, setpencolor(clred).

Процедура заливка цветом FloodFill(x,y,color) заливает область одного цвета цветом color, начиная с точки (x,y). К примеру, Rectangle(50,50,200,200); FloodFill(100,100,clBlue)

Шрифты.

Процедура procedure SetFontName(name: string) – устанавливает наименование шрифта.

Процедура procedure SetFontStyle(style: integer) – устанавливает стиль шрифта.

Процедура procedure SetFontSize(sizesz: integer) – устанавливает размер шрифта в пунктах.

Процедура procedure SetFontColor(color: integer) – устанавливает цвет шрифта.

Процедура procedure TextOut(x, y, s) – выводит строку s в позицию (x,y). Точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s).

К примеру:

SetFontName('Arial');

SetFontStyle(fsBoldItalic);

SetFontSize(40);

SetFontColor(RGB (Random(255), Random(255), Random(255))).

Расширенный перечень графических процедур и функций.

Графические примитивы.

procedure SetPixel(x,y,color: integer) – закрашивает один пиксел с координатами (x,y) цветом color.

function GetPixel(x,y: integer) – возвращает текущее значение цвета для пиксела с координатами (x,y).

procedure MoveTo(x,y: integer) – передвигает невидимое перо к точке с координатами (x,y), эта функция работает в паре с функцией LineTo(x,y).

procedure LineTo(x,y: integer) – рисует отрезок от текущего положения пера до точки (x,y), координаты пера при этом также становятся равными (x,y).

procedure Line(x1,y1,x2,y2: integer) – рисует отрезок с началом в точке (x1,y1) и концом в точке (x2,y2).

procedure Circle(x,y,r: integer) – рисует окружность с центром в точке (x,y) и радиусом r.

procedure Ellipse(x1,y1,x2,y2: integer) – рисует эллипс, заданный своим описанным прямоугольником с координатами противоположных вершин (x1,y1) и (x2,y2).

procedure Rectangle(x1,y1,x2,y2: integer) – рисует прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2).

procedure RoundRect(x1,y1,x2,y2,w,h: integer) – рисует прямоугольник со скругленными краями, (x1,y1) и (x2,y2) задают пару противоположных вершин, а *w* и *h* – ширину и высоту эллипса, используемого для скругления краев.

procedure Arc(x,y,r,a1,a2: integer) – рисует дугу окружности с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки).

procedure Pie(x,y,r,a1,a2: integer) – рисует сектор окружности, ограниченный дугой (параметры процедуры имеют тот же смысл, что и в процедуре Arc).

procedure Chord(x,y,r,a1,a2: integer) – рисует фигуру, ограниченную дугой окружности и отрезком, соединяющим ее концы (параметры процедуры имеют тот же смысл, что и в процедуре Arc).

procedure TextOut(x,y: integer; s: string) – выводит строку s в позицию (x,y) (точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки s).

procedure FloodFill(x,y,color: integer) – заливает область одного цвета цветом color, начиная с точки (x,y).

procedure FillRect(x1,y1,x2,y2: integer) – заливает прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2), цветом текущей кисти.

procedure Polygon(var a; n: integer) – строит ломаную по n точкам, координаты которых заданы в массиве a элементов типа Point.

procedure Polyline(var a; n: integer) – строит замкнутую ломаную по n точкам, координаты которых заданы в массиве a элементов типа Point.

Цветовые константы и функции для работы с цветом

Модуль GraphABC содержит константы и функции для работы с цветами. Тип ColorType, описывающий цвет, определен следующим образом:

type ColorType=integer – стандартные цвета задаются символическими константами.

Таблица 2 – Символические константы

|  |  |  |  |
| --- | --- | --- | --- |
| Код | Цвет | Код | Цвет |
| clBlack | черный | clPurple | фиолетовый |
| clMaroon | темно–красный | clRed | красный |
| clGreen | зелёный | clBrown | коричневый |
| clSkyBlue | голубой | clYellow | желтый |
| clAqua | бирюзовый | clOlive | оливковый |
| clTeal | сине–зелёный | clGray | темно–серый |
| clMoneyGreen | цвет зеленых денег | clLtGray | светло–серый |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| clMedGray | серый | clSilver | серебряный |
| clFuchsia | сиреневый | clWhite | белый |
| clLime | ярко–зеленый | clNavy | темно–синий |
| clDkGray | темно–серый | clBlue | синий |
| clCream | кремовый |  |  |

Для работы с цветами используются следующие функции.

function RGB(r,g,b: integer): ColorType – возвращает целое значение, являющееся кодом цвета͵ который содержит красную, зеленую и синюю составляющие с интенсивностями r, g и b соответственно (r, g и b – целые в диапазоне от 0 до 255, причем, 0 соответствует минимальной интенсивности, 255 – максимальной).

function GetRed(color: ColorType): integer – выделяет красную составляющую из цвета color (целое в диапазоне от 0 до 255).

function GetGreen(color: ColorType): integer – выделяет зеленую составляющую из цвета color (целое в диапазоне от 0 до 255).

function GetBlue(color: ColorType): integer – выделяет синюю составляющую из цвета color (целое в диапазоне от 0 до 255).

Действия с пером.

function PenX: integer;

function PenY: integer – возвращают текущие координаты пера.

procedure SetPenColor(color: integer); – устанавливает цвет пера, задаваемый параметром color.

function PenColor: integer – возвращает текущий цвет пера.

procedure SetPenWidth(w: integer) – устанавливает ширину пера, равную w пикселам.

function PenWidth: integer – возвращает текущую ширину пера.

procedure SetPenStyle(ps: integer) – устанавливает стиль пера, задаваемый параметром ps.

function PenStyle: integer – возвращает текущий стиль пера.

Стили пера задаются следующими именованными константами:

1. psSolid – сплошная линия (установлено по умолчанию);
2. psDash – штриховая линия;
3. psDot – пунктирная линия;
4. psDashDot – штрихпунктирная линия;
5. psDashDotDot – линия, чередующая штрих и два пунктира;
6. psClear – отсутствие линии.

procedure SetPenMode(m: integer) – устанавливает режим пера, задаваемый параметром *m*.

function PenMode: integer – возвращает текущий режим пера. Режим пера определяет, как цвет пера взаимодействует с цветом поверхности.

Режимы пера задаются следующими именованными константами:

1. pmCopy – обычный режим, при рисовании цвет поверхности заменяется цветом пера;
2. pmNot – режим инвертирования; при рисовании цвет поверхности инвертируется (становится негативным), а цвет пера при этом игнорируется.

Действия с кистью.

procedure SetBrushColor(color: integer) – устанавливает цвет кисти, задаваемый параметром color.

function BrushColor: integer – возвращает текущий цвет кисти.

procedure SetBrushPicture(fname: string) – устанавливает в качестве образца для закраски кистью образец, хранящийся в файле fname, при этом текущий цвет кисти при закраске игнорируется.

procedure ClearBrushPicture – очищает рисунок–образец, выбранный для кисти.

procedure SetBrushStyle(bs: integer) – устанавливает стиль кисти, задаваемый параметром bs.

function BrushStyle: integer – возвращает текущий стиль кисти.

Стили кисти задаются следующими именованными константами:

1. bsSolid;
2. bsCross;
3. bsClear;
4. bsDiagCross;
5. bsHorizontal;
6. bsBDiagonal;
7. bsVertical;
8. bsFDiagonal.

Действия со шрифтом.

procedure SetFontColor(color: integer) – устанавливает цвет шрифта.

function FontColor: integer – возвращает текущий цвет шрифта.

procedure SetFontSize(sz: integer) – устанавливает размер шрифта в пунктах.

function FontSize: integer – возвращает текущий размер шрифта в пунктах.

procedure SetFontName(name: string) – устанавливает наименование шрифта.

function FontName: string – возвращает текущее наименование шрифта.

По умолчанию установлен шрифт, имеющий наименование MS Sans Serif. Наиболее распространенные шрифты – ϶ᴛᴏ Times, Arial и Courier New. Наименование шрифта можно набирать без учета регистра.

procedure SetFontStyle(fs: integer) – устанавливает стиль шрифта.

function FontStyle: integer – возвращает текущий стиль шрифта.

Стили шрифта задаются следующими именованными константами:

1. fsNormal – обычный;
2. fsBold – жирный;
3. fsItalic – наклонный;
4. fsBoldItalic – жирный наклонный;
5. fsUnderline – подчеркнутый;
6. fsBoldUnderline – жирный подчеркнутый;
7. fsItalicUnderline – наклонный подчеркнутый;
8. fsBoldItalicUnderline – жирный наклонный подчеркнутый.

function TextWidth(s: string): integer – возвращает ширину строки s в пикселях при текущих настройках шрифта.

function TextHeight(s: string): integer – возвращает высоту строки s в пикселях при текущих настройках шрифта.

Действия с графическим окном.

procedure ClearWindow – очищает графическое окно белым цветом.

procedure ClearWindow(c: ColorType) – очищает графическое окно цветом c.

function WindowWidth: integer – возвращает ширину графического окна.

function WindowHeight: integer – возвращает высоту графического окна.

function WindowLeft: integer – возвращает отступ графического окна от левого края экрана.

function WindowTop: integer – возвращает отступ графического окна от верхнего края экрана.

function WindowCaption: string – возвращает заголовок графического окна.

procedure SetWindowWidth(w: integer) – устанавливает ширину графического окна.

procedure SetWindowHeight(h: integer) – устанавливает высоту графического окна.

procedure SetWindowLeft(l: integer) – устанавливает отступ графического окна от левого края экрана.

procedure SetWindowTop(t: integer) – устанавливает отступ графического окна от верхнего края экрана.

procedure SetWindowSize(w,h: integer) – устанавливает ширину и высоту графического окна.

procedure SetWindowPos(l,t: integer) – устанавливает отступ графического окна от левого и верхнего края экрана.

procedure SetWindowCaption(s: string) – устанавливает заголовок графического окна.

procedure SetWindowTitle(s: string) – устанавливает заголовок графического окна. Синоним SetWindowCaption.

procedure SaveWindow(fname: string) – сохраняет содержимое графического окна в файл с именем fname.

procedure LoadWindow(fname: string) – выводит в графическое окно рисунок из файла с именем fname. Файл ищется вначале в текущем каталоге, а затем в каталоге PascalABC\Media\Images.

procedure FillWindow(fname: string) – заполняет графическое окно мозаикой из рисунка, содержащегося в файле с именем fname.

procedure FillWindow(n: integer) – заполняет графическое окно мозаикой из рисунка с описателем n.

procedure CloseWindow – закрывает графическое окно.

function ScreenWidth: integer – возвращает ширину экрана.

function ScreenHeight: integer – возвращает высоту экрана.

procedure CenterWindow – центрирует графическое окно по центру экрана.

procedure MaximizeWindow – максимизирует графическое окно на экране.

procedure NormalizeWindow – восстанавливает положение графического окна на экране.

Все размеры устанавливаются и возвращаются в пикселах.

Задание режимов вывода

procedure SetDrawingSurface(n: integer) – устанавливает в качестве канвы для рисования рисунок с описателем n. В результате весь графический вывод осуществляется не на экран, а на рисунок; настройки кисти, пера и шрифта также реализуются для рисунка.

procedure SetDrawingSurface(p: Picture) – устанавливает в качестве канвы для рисования рисунок с описателем n. В результате весь графический вывод осуществляется не на экран, а на рисунок; настройки кисти, пера и шрифта также реализуются для рисунка.

procedure RestoreDrawingSurface – устанавливает в качестве канвы для рисования графическое окно.

procedure Redraw – осуществляет перерисовку окна вывода при заблокированном выводе в графическое окно.

procedure LockDrawing – блокирует вывод в графическое окно, осуществляя рисование только во внеэкранном буфере. Для перерисовки графического окна требуется вызвать процедуру Redraw. В случае если графический вывод перенаправлен в рисунок вызовом процедуры SetDrawingSurface, то не оказывает никакого воздействия на вывод.

procedure UnlockDrawing – снимает блокировку вывода в графическое окно.

procedure LockScreenBuffer – блокирует вывод во внеэкранный буфер графического окна. После вызова этой процедуры рисование незначительно ускоряется, однако, изображение графического окна перестает восстанавливаться.

procedure UnlockScreenBuffer – снимает блокировку вывода во внеэкранный буфер графического окна.

function DrawingIsLocked: boolean – возвращает True, если вывод в графическое окно заблокирован, и False в противном случае.

procedure SetRedrawProc(procedure RedrawProc) – устанавливает пользовательскую процедуру для перерисовки содержимого графического окна, вызываемую автоматически в тот момент, когда требуется его перерисовка. Сегодня используется в модуле ABCObjects для автоматической перерисовки всех графических объектов и фона.

Блокировка вывода в графическое окно с последующим вызовом Redraw используется для простейшего создания анимации без мерцания.

## 2 Разработка алгоритма

В процессе исследования было выделено несколько подходов к созданию динамических изображений:

1. Стереть изображение полностью, затем перерисовать его заново, но уже с необходимыми изменениями, которые и будут создавать динамический эффект.
2. Закрасить ненужные элементы цветом фона, перерисовывая их в другом месте, что даст иллюзию их перемещения.

В программе используется первый способ рисования динамического изображения.

Для удобства разобьем код на несколько основных этапов:

1. Рисование фигуры «шарик».
2. «Перекатывание» шарика по дуге с угасанием.

Опишем более подробно этапы программы:

1. Рисование фигуры «шарик».

С помощью встроенной процедуры построения окружности рисуем фигуру – «шарик». Заметим, что в виду 2D реализации под шариком будем понимать проецированное изображение шарика на плоскость.

1. «Перекатывание» шарика по дуге с угасанием.

Используя процедуру очищения графического окна, стираем изображения и перерисовываем, создавая динамический эффект.

Инициируем графический модуль:

Uses GraphAbc.

Обозначаем переменные:

var

i, x, y, f, k: integer;

u, j: real;

Задаем размеры графического окна:

Window.Init(0, 0, 500, 500, clbLUE);//задаём параметры экрана

 Выполним рисование дуги окружности, по которой движется модель шарика:

 Arc(250, 250, 122, –180, 0);//задаём центр окружности и точки, ограничивающие дугу.

 Задаём угол начального положения шарика на дуге:

u := 3 \* pi / 2;

Опишем цикл обработки анимационной модели:

while k <> 15 do//количество полных колебаний

Обозначим шаг затухания, для обеспечения соответствующего физического эффекта:

j := j + 0.211;//шаг затухания колебания шарика

Опишем условие движения шарика по «часовой стрелке»:

while u < (5 \* pi / 2) – j do//движение шарика по часовой стрелке

Опишем «движение» шарика по часовой стрелке:

begin

 setbrushcolor(clRED);

 circle(250 – round(100 \* sin(u)), 250 + round(100 \* cos(u)), 10);

 LockDrawing;

 SLEEP(120);

 SetBrushColor(clwhite);

 Rectangle(0, 0, 500, 500);

 SetBrushColor(clBlack);

 Arc(250, 250, 110, –180, 0);

 unLockDrawing;

 u := u + 0.126;

 end;

 Выполним проверку угла, чтобы не «вылететь» за пределы дуги, и условие движения «против часовой стрелки»:

if u >= ((5 \* pi / 2) – j ) then

 while u > (3 \* pi / 2) + j do//движение против часовой стрелки

Опишем «движение» шарика против часовой стрелки:

begin

 setbrushcolor(clRed);

 circle(250 – round(100 \* sin(u)), 250 + round(100 \* cos(u)), 10);

 LockDrawing;

 SLEEP(120);

 SetBrushColor(clwhite);

 Rectangle(0, 0, 500, 500);

 SetBrushColor(clBlack);

 Arc(250, 250, 110, –180, 0);

 unLockDrawing;

 u := u – 0.126;

 end;

Введём изменение параметра цикла:

inc(k);

end;

Опишем поведение шарика после завершения «колебательного цикла»:

setbrushcolor(clRED);

circle(250 – round(100 \* sin(2 \* pi)), 250 + round(100 \* cos(2 \* pi)), 10);

end.

## 3 Тестовые примеры

Пример правильного выполнения алгоритма.

 Шаг первый: рисование шарика.

Рисунок 1 – Рисование шарика

Шарик перекатывается первый раз.

Рисунок 2 – Первое колебание шарика

Начало затухания колебательного движения.

Рисунок 3 – Затухание движения шарика

После выполнения ряда последовательных прохождений цикла движения, шарик почти останавливается, имитируя тем самым естественное колебательное движение.

Рисунок 4 – Продолжение затухание движения шарика

В конечном итоге шарик останавливается, располагаясь ровно по центру дуги.

Рисунок 5 – Остановка движения шарика

В процессе отладки программы ошибок не было выявлено. Программа работает верно.

## ЗАКЛЮЧЕНИЕ

Графические средства PascalABC недостаточно широки. В графическом режиме можно выводить большое количество объектов и изображений, в том числе изменяя их так, чтобы создавался динамический эффект.

В процессе исследования был проведен анализ поставленной задачи и разработан алгоритм решения задачи, описанный на языке программирования PascalABC.NET.

Написанный программный код демонстрирует компьютерную реализацию физической модели движения шарика по дуговой траектории с постепенным «затуханием».

Возможно и дальнейшее усовершенствование программы:

1. запускать и останавливать движение шарика по нажатию кнопки;
2. задавать курсором начальное положение шарика на дуге. Затем начинать движение.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интернет–ресурс: http://Pascalabc.net
2. Абрамян М.Э. Практикум по программированию на языке Паскаль: Массивы, строки, файлы, рекурсия, указатели –– Ростов–на–Дону: ООО «ЦВВР», 2004. –– 187 с. –– ISBN 5–94153–001–3
3. Михалкович, С.С. Pascal ABC & Programming Taskbook ME / С.С. Михалкович, М.Э. Абрамян. [Электронный ресурс]. – Режим доступа – [http://sunschool.math.rsu.ru/](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fsunschool.math.rsu.ru%2F)
4. Основы программирования: С. Окулов – Москва, Бином. Лаборатория знаний, 2010
5. Долинер, Л.И. Основы программирования в среде PascalABC.NET: учебное пособие / Л.И. Долинер; Министерство образования и науки Российской Федерации, Уральский федеральный университет имени первого Президента России Б. Н. Ельцина; науч. ред. Г.А. Матвеева. - Екатеринбург: Издательство Уральского университета, 2014. - 129 с.: ил. - Библиогр. в кн. - ISBN 978-5-7996-1260-3