МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»



# СИСТЕМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ

***МЕТОДИЧЕСКИЕ УКАЗАНИЯ***

по выполнению курсовой работы
для студентов специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)»

Витебск
2018

УДК 004.92

Составители:

Н.Л. Надежная, А.К. Гниденко, А.С. Соколова, Е.Б. Дунина

Рекомендовано к изданию редакционно-издательским советом УО «ВГТУ», протокол № 9 от 30.11.2018.

**Системы компьютерной графики**: методические указания по выполнению курсовой работы / сост. Н. Л. Надежная [и др.]. – Витебск : УО «ВГТУ», 2018. – 37 с.

Методические указания являются руководством по выполнению курсовой работы по дисциплине «Системы компьютерной графики», определяют общие требования, предъявляемые к курсовой работе, освещают последовательность ее подготовки, требования к структуре, содержанию и оформлению как самой работы, так и научно-справочного аппарата и приложений.

**УДК 004.92**

# СОДЕРЖАНИЕ

# 1 ОБЩАЯ ХАРАКТЕРИСТИКА КУРСОВОЙ РАБОТЫ

В соответствии с учебным планом студенты 1 курса дневной и заочной форм обучения специальности 1-40 05 01-01 «Информационные системы и технологии (в проектировании и производстве)» выполняют курсовую работу по дисциплине «Системы компьютерной графики» на тему **«Разработка приложения для создания и преобразования графических объектов»**.

Целью курсовой работы является углубление теоретических и практических знаний по рассматриваемой теме и, как следствие, рост знаний студента; формирование умений использования литературных источников, выработка навыков изложения изученного материала, развитие мышления, приобретение студентами навыков написания научных работ в соответствии с избранной специальностью, проверка усвоения учебного материала по дисциплине в целом.

***Исходные данные к проекту:*** *написать графическое приложение, эмулирующее движение базовой фигуры таким образом, чтобы ее центр описывал заданную траекторию. При движении вдоль траектории фигура одновременно вращается вокруг центральной точки, а также меняет цвет. Скорость движения фигуры вдоль траектории и скорость ее вращения задаются пользователем.*

Программу необходимо реализовать в виде проекта в выбранной среде разработки. Программа должна содержать главное окно. В рабочей области окна при запуске программы следует предусмотреть отображение базовой фигуры в исходном состоянии. Необходимо реализовать возможность масштабирования фигуры с сохранением пропорций и способ возврата к исходному масштабу. При изменении масштаба фигура должна быть перерисована в рабочей области окна. Запуск и останов режима эмуляции движения, а также все параметры данного режима задаются пользователем при помощи графических элементов интерфейса, например, меню или панели инструментов.

Руководитель оказывает студенту помощь при подготовке работы, консультирует его по всем возникающим вопросам. При этом работа выполняется самостоятельно студентом.

## 1.1 Рекомендации к содержанию

Расчетно-пояснительная записка и графический материал оформляются в соответствии с требованиями действующей редакции ГОСТ 7.32-2001 «Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления».

Структурными элементами пояснительной записки (ПЗ) к курсовой работе являются:

титульный лист;

задание на курсовую работу;

содержание;

введение;

основная часть;

заключение;

список использованных источников;

приложения.

Пример содержания расчетно-пояснительной записки (перечень вопросов, подлежащих разработке):

Введение

1. Анализ исходного графического объекта и методов преобразования фигур

1.1 Математическое описание операций преобразования плоских фигур

1.2 Выбор и обоснование языка программирования и среды разработки

1.3 Задание базовой фигуры

2. Разработка алгоритма работы программы

3. Описание основных компонентов программы и последовательности разработки

4. Исходный код программы

5. Проверка корректности работы программы в различных режимах

Заключение

Список использованных источников

Приложение А

Приложение Б

*Титульный лист* является первой страницей курсовой работы. Он не нумеруется, но включается в общую нумерацию страниц работы. Форма титульного листа курсовой работы представлена в приложении А.

*Задание* на курсовую работу оформляется на типовом бланке, подписывается студентом и руководителем. Задание подшивается после титульного листа, не нумеруется, но включается в общую нумерацию.

Во *введении* обосновывается актуальность темы, приводится краткая характеристика основных аспектов изучаемой проблемы, формулируются цель и задачи проектирования. Пример приведен в приложении Б.

*Основная часть* расчетно-пояснительной записки должна дать полное представление о выполненной работе и соответствовать заданию на проектирование. В ее состав включаются материалы, которые поясняют результаты работы и методику ее выполнения. Основная часть расчетно-пояснительной записки делится на разделы и подразделы, названия которых не должны дублировать название темы курсовой работы. Заголовки должны быть лаконичными и соответствовать содержанию работы.

В *заключении*, в соответствии с целью и конкретными задачами, обозначенными во введении, формулируются основные результаты и выводы.

*Список использованных источников* содержит библиографическое описание источников, использованных для проектирования. На все источники в тексте расчетно-пояснительной записки должны быть даны ссылки. Не допускается включать в список источники, ссылки на которые в расчетно-пояснительной записке отсутствуют.

В *приложения* могут включаться материалы иллюстративного, вспомогательного характера, необходимые проектные документы, листинги программных продуктов и материалы, размещение которых в основной части нецелесообразно.

Выполненная в полном объеме и допущенная к защите курсовая работа предоставляется на защиту в папке-скоросшивателе или в переплетенном виде.



### 1.2 Оформление пояснительной записки

Объем курсовой работы 25–35 страниц текста. Пояснительная записка выполняется на стандартных листах бумаги (формат А4) с рамками и кодом выполняемой работы:

*УО «ВГТУ» КР.0ХХ 1-40 05 01-01 ПЗ,*

где ХХ – номер студента по списку группы.

Надписи в рамке оформляются в соответствии с ГОСТ 2.104-2006 «Единая система конструкторской документации. Основные надписи». Шрифт надписей ISOCPEUR обычный, курсив, черный. Размер шрифта: для кода работы и названия раздела – 14 пт, наименование организации – 11 пт, фамилия и инициалы разработчика и проверяющего – 9 пт. На первой странице раздела – рамка с большим штампом, на остальных – рамка с маленьким штампом.

Требования к объему работы должны быть соблюдены. При значительном количестве информации по избранной теме умение уложиться в предлагаемый объем помогает приобрести и развить навыки выделения главного, обобщения информации, формулирования собственных выводов.

Требования для текста:

– шрифт –Times New Roman, 14 пт, обычный, нормальный, черный;

– межстрочный интервал – 1,5;

– красная строка – 1,25 см;

– отступы – 0;

– расстояние от рамки до границ текста слева и справа не менее 3 мм, сверху и снизу – не менее 10 мм;

– нумерация страниц – сквозная в рамке;

– заголовки 1-го уровня (разделов) – шрифт – Times New Roman, 14 пт, все прописные, жирный, нормальный, черный, выравнивание по ширине с красной строкой 1,25 см;

– заголовки 2-го уровня (подразделов) – шрифт – Times New Roman, 14 пт, как в предложениях, обычный, нормальный, черный, выравнивание по ширине с красной строкой 1,25 см;

– заголовки всех уровней записываются без точки в конце, не подчеркивая. Переносы слов в заголовках не допускаются. Если заголовок состоит из двух предложений, их разделяют точкой.

Рисунки подписываются снизу посередине. Перед рисунком и после его подписи вставляется пустая строка. Нумеруются рисунки в пределах раздела. Номер состоит из номера раздела и порядкового номера рисунка в разделе, разделенных точкой. Номер записывается после слова «Рисунок», после номера ставится тире и с прописной буквы записывается название рисунка.

Формулы выравниваются посредине и нумеруются по правому краю с помощью табуляции, причем номер записывается в круглых скобках. Нумерация формул в пределах раздела. Номер формулы состоит из номера раздела и порядкового номера формулы в разделе, разделенных точкой. От текста строка с формулой отделяется пустыми строками. Ссылки в тексте на порядковые номера формул дают в скобках, например, «... в формуле (2.3)…».

# 2 АНАЛИЗ ИСХОДНОГО ГРАФИЧЕСКОГО ОБЪЕКТА И МЕТОДОВ ПРЕОБРАЗОВАНИЯ ФИГУР

## 2.1 Математическое описание операций преобразования плоских фигур

Точки на *xy*-плоскости можно перенести в новые позиции путем добавления к координатам этих точек констант переноса. Для каждой точки *P(x, y)*, которая перемещается в новую точку *P′(x’, y’)*, сдвигаясь на *Dx* единиц параллельно оси *x* и на *Dy* единиц параллельно оси *y*, можно написать уравнения:

$$x' = x + Dx\,,\ y' = y + Dy\,.$$

Определяя векторы-строки

$$P = [x\ y]\,,\ P' = [x'\ y']\,,\ T = [Dx\ Dy]\,,$$

эти уравнения можно переписать в векторной форме:

$$[x'\ y'] = [x\ y] + [Dx\ Dy],$$
$$P' = P + T. \qquad (2.1)$$

Объект можно перенести, применяя вышевыведенное уравнение к каждой его точке. Все точки, принадлежащие отрезку, можно так же перенести путем перемещения одних лишь крайних точек отрезка и последующего вычерчивания нового отрезка между получившимися в результате точками. Это справедливо также для масштабирования (растяжения) и поворота.

Точки можно промасштабировать (растянуть) в *Sx* раз вдоль оси *x*: и в *Sy* раз вдоль оси *y*, получив в результате новые точки:

$$x' = x \cdot Sx\ ,\ y' = y \cdot Sy\ .$$

Определяя *S* как $\begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$, можно записать в матричной форме

$$[x'\ y'] = [x\ y]\begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix},$$

или

$$P' = P \cdot S\ . \qquad (2.2)$$

Отметим, что масштабирование производится относительно начала координат. Если масштабные множители больше 1, то объект увеличивается и отдаляется от начала координат. Однородное масштабирование, для которого *Sx* = *Sy*, не влияет на пропорции.

Точки могут быть повернуты на угол $\theta$ относительно начала координат. Математически поворот определяется следующими уравнениями:

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta. \end{cases}$$

В матричной форме имеем

$$[x'\ y'] = [x\ y]\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

или

$$P' = P \cdot R\ , \qquad (2.3)$$

где через *R* обозначена матрица поворота. Как и в случае масштабирования, поворот производится относительно начала координат.

Преобразования переноса, масштабирования и поворота в матричной форме записываются в виде (2.1), (2.2), (2.3). К сожалению, перенос реализуется отдельно (с помощью сложения) от масштабирования и поворота (с помощью умножения). Представим их таким способом, чтобы все эти три элементарных преобразования можно было легко объединять вместе.

Если мы выразим точки в однородных координатах, то все три преобразования можно реализовать с помощью умножений. Однородные координаты – система координат, используемая в проективной геометрии, подобно тому, как декартовы координаты используются в евклидовой геометрии. Однородные координаты обладают тем свойством, что определяемый ими объект не меняется при умножении всех координат на одно и то же ненулевое число. Из-за этого количество координат, необходимое для представления точек, всегда на одну больше, чем размерность пространства, в котором эти координаты используются. Например, для представления точки на прямой в одномерном пространстве необходимы две координаты и три координаты для представления точки на плоскости в двумерном пространстве.

Если для точки задано ее представление в однородных координатах *P(X, Y, W),* то можно найти ее двумерные декартовы координаты как $x = X / W$ и $y = Y / W \ \ (W \neq 0)$. Пусть $W = 1$, тогда операция деления не требуется.

Точки теперь описываются трехэлементными вектор-строками, поэтому матрицы преобразований, на которые умножается вектор точки, чтобы получить другой вектор точки, должны иметь размер 3x3. Уравнения переноса записываются в виде матрицы преобразования однородных координат следующим образом

$$[x'\ y'\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix}$$

$$P' = P \cdot T(Dx, Dy), \qquad (2.4)$$

где

$$T(Dx, Dy) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix}.$$

Если точку *P* перенести в точку *P′* на расстояние *(Dx1, Dy1),* а затем в *P″* на расстояние *(Dx2 Dy2),* то мы получим суммарный перенос на расстояние *(Dx1+Dx2, Dy1+Dy2).*

Уравнения масштабирования в матричной форме записываются в виде

$$[x'\ y'\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Определяя

$$S(Sx, Sy) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

получим

$$P' = P \cdot S(Sx, Sy). \qquad (2.5)$$

Очевидно, что последовательные переносы являются аддитивными, а последовательные масштабирования будут мультипликативными. Если заданы

$$P' = P \cdot S(Sx_1, Sy_1),$$
$$P'' = P' \cdot S(Sx_2, Sy_2),$$

то получим:

$$P'' = \left(P \cdot S(Sx_1, Sy_1)\right) \cdot S(Sx_2, Sy_2) = P \cdot \left(S(Sx_1, Sy_1) \cdot S(Sx_2, Sy_2)\right).$$

Результатом произведения матриц $S(Sx_1, Sy_1) \cdot S(Sx_2, Sy_2)$ будет следующая матрица

$$\begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Sx_1 \cdot Sx_2 & 0 & 0 \\ 0 & Sy_1 \cdot Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

И, наконец, уравнения поворота можно представить в виде:

$$[x'\ y'\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Полагая

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

имеем

$$P' = P \cdot R(\theta). \qquad (2.6)$$

Основное преимущество объединенных преобразований состоит в том, что к точке более эффективно применять одно результирующее преобразование, чем ряд преобразований друг за другом.

Рассмотрим поворот объекта относительно некоторой произвольной точки $P_i$. Поскольку нам известно, лишь как поворачивать вокруг начала координат, разобьем исходную проблему на три более легкие задачи. Таким образом, чтобы произвести поворот относительно точки $P_i$, необходимо выполнить последовательно три элементарных преобразования:

1) перенос, при котором точка $P_i$ перемещается в начало координат;
2) поворот;
3) перенос, при котором точка из начала координат возвращается в первоначальное положение $P_i$.

Первый перенос производится на $(-x_1, -y_1)$, в то время как последующий – на $(x_1, y_1)$ – является обратным ему. Результат существенно отличается от того, который получился бы, если бы применялся один только поворот.

Результирующее преобразование имеет вид

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ x_1(1-\cos\theta) + y_1\sin\theta & y_1(1-\cos\theta) - x_1\sin\theta & 1 \end{bmatrix}.$$

## 2.2 Выбор и обоснование языка программирования и среды разработки

Курсовая работа носит учебно-исследовательский характер и должна быть направлена на более глубокое изучение выбранной темы, чем она раскрывается в учебной литературе; содержать элементы научного исследования; иметь четкое построение и логическую последовательность в изложении материала.

Студентам предоставляется право самостоятельного выбора языка программирования для решения поставленной задачи.

Рассмотрим пример изложения в расчетно-пояснительной записке выбора и обоснование языка программирования C#.

Пример

Для реализации курсовой работы был выбран язык C#, который является языком высокого уровня и позволяет быстро и эффективно создавать приложения.

Язык программирования C# – это очередная ступень бесконечной эволюции языков программирования. Его создание вызвано процессом усовершенствования и адаптации, который определял разработку компьютерных языков в течение последних лет. Подобно всем успешным языкам, которые увидели свет раньше, C# опирается на прошлые достижения постоянно развивающегося искусства программирования.

C# – элегантный объектно-ориентированный язык, предназначенный для разработки разнообразных безопасных и мощных приложений, выполняемых в среде .NET Framework. С помощью языка C# можно создавать обычные приложения Windows, XML-веб-службы, распределенные компоненты, приложения «клиент-сервер», приложения баз данных и т. д. Visual C# предоставляет развитый редактор кода, конструкторы с удобным пользовательским интерфейсом, встроенный отладчик и множество других средств, упрощающих разработку приложений на базе языка C# и .NET Framework.

Как объектно-ориентированный язык, C# поддерживает понятия инкапсуляции, наследования и полиморфизма. Все переменные и методы, включая метод Main – точку входа приложения – инкапсулируются в определения классов. Класс может наследовать непосредственно из одного родительского класса, но может реализовывать любое число интерфейсов.

C# создавался как язык компонентного программирования, и в этом одно из главных достоинств языка, направленное на возможность повторного использования созданных компонентов.

Microsoft Visual Studio 2015 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.

Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных.

Вы можете создавать:

– приложения и игры, которые выполняются не только на платформе Windows, но и на Android и iOS;

– веб-сайты и веб-службы на основе ASP.NET, JQuery, AngularJS и других популярных платформ;

– приложения для самых разных платформ и устройств, включая, но не ограничиваясь: Office, Sharepoint, Hololens, Kinect и «Интернета вещей»; игры и графические приложения для разных устройств Windows, включая Xbox, с поддержкой DirectX.



## 2.3 Задание базовой фигуры

Требуется отрисовать в левом нижнем углу pictureBox базовую фигуру, изображенную на рисунке 2.1. Исходный графический объект представляет собой правильный шестиугольник, в который вписан правильный треугольник. Расстояние между противоположными вершинами шестиугольника равно *a*.

Базовая фигура задается с помощью математических формул, которые описывают данные объекты, и методов задания объектов языка программирования C#.

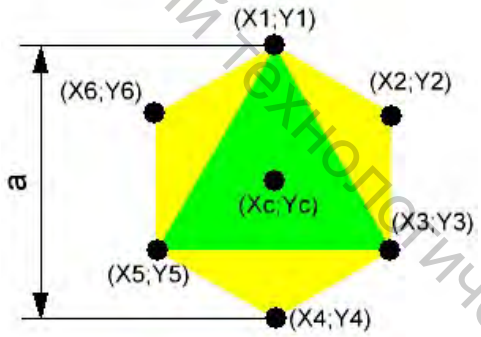



Рисунок 2.1 – Исходный графический объект

Для этого обозначим точки фигуры и выразим их координаты через координаты центральной точки $(x_c, y_c)$ и определяющий размер фигуры *a*:

$$x_1 = x_c;\qquad y_1 = y_c - a/2;$$
$$x_2 = x_c + (a/2)\cos 30^\circ;\qquad y_2 = y_c - (a/2)\sin 30^\circ;$$
$$x_3 = x_c + (a/2)\cos 30^\circ;\qquad y_3 = y_c + (a/2)\sin 30^\circ;$$
$$x_4 = x_c;\qquad y_4 = y_c + a/2;$$
$$x_5 = x_c - (a/2)\cos 30^\circ;\qquad y_5 = y_c + (a/2)\sin 30^\circ;$$
$$x_6 = x_c - (a/2)\cos 30^\circ;\qquad y_6 = y_c - (a/2)\sin 30^\circ.$$

Отображение графики обеспечивает компонент PictureBox. Графическая поверхность компонента PictureBox представляет собой объект Graphics,

методы которого и обеспечивают вывод графики. Чтобы создать графическую поверхность компонента pictureBox1, необходимо написать следующий код, обязательно добавив пространство имен System.Drawing:

```
Graphics g = pictureBox1.CreateGraphics();
```

Графическая поверхность состоит из отдельных точек – пикселов. Положение точки на графической поверхности характеризуется горизонтальной (*X*) и вертикальной (*Y*) координатами. Координаты точек отсчитываются от левого верхнего угла и возрастают слева направо (координата *X*) и сверху вниз (координата *Y*). Левая верхняя точка графической поверхности имеет координаты (0, 0).

Для формы **PictureBox** в свойстве **BorderStyle** установлено **FixedSingle** (неизменяемая одинарная рамка), то есть в процессе выполнения программы с помощью курсора мы не сможем менять размер формы.

Для отрисовки шестиугольника и треугольника используется полигон с заливкой:

```
g.FillPolygon(br1, p1);//отрисовка шестиугольника
g.FillPolygon(br2, p2);//отрисовка треугольника
```

где p1, p2 – массив точек для шестиугольника и треугольника соответственно,

```
Brush br1= new SolidBrush(Color.FromArgb(255,255,0)); – желтая кисть,
Brush br2= new SolidBrush(Color.FromArgb(0,255,0)); / – зеленая кисть.
```

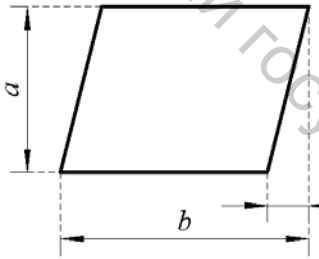
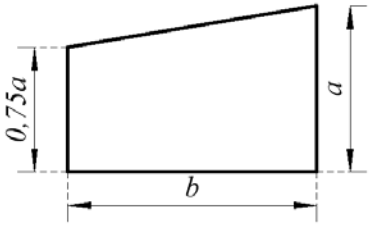
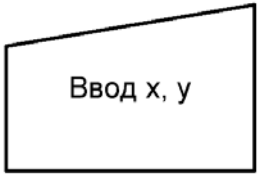
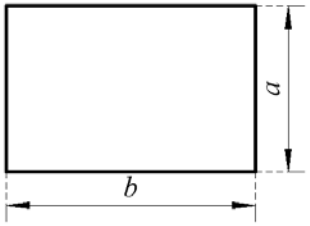
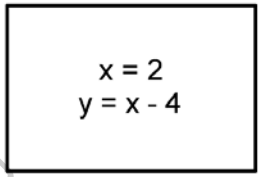
Цвет задается в модели aRGB: альфа (прозрачность), красный, зеленый, синий (от 0 до 255).

# 3 РАЗРАБОТКА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ

АЛГОРИТМ – это последовательность команд, ведущих к какой-либо цели. Это строго определенная процедура, гарантирующая получение результата за конечное число шагов. Это правило, указывающее действия, в результате цепочки которых происходит переход от исходных данных к искомому результату. Указанная цепочка действий называется алгоритмическим процессом, а каждое отдельное действие – его шагом. Блок-схемы алгоритмов – это графическое описание алгоритмов как последовательности действий.

Блок схемы алгоритмов выполняются в соответствии с ГОСТ 19.701-90 «Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». Перечень, наименование, обозначение и размеры некоторых наиболее распространенных символов и отображаемых ими функций приведены в таблице 3.1.

Таблица 3.1 – Описание символов на блок-схемах программ

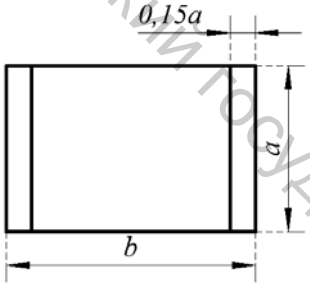
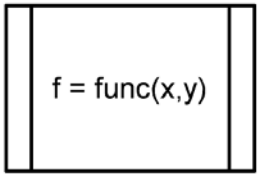
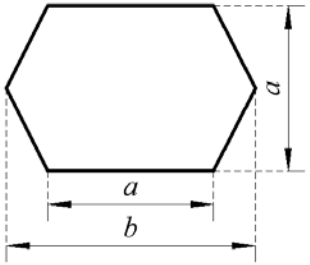
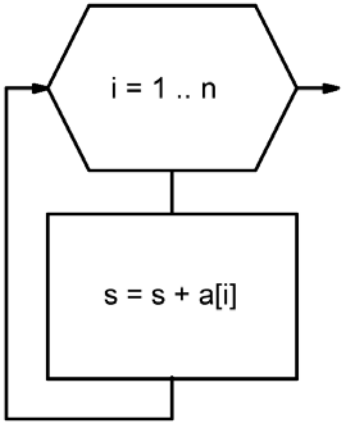
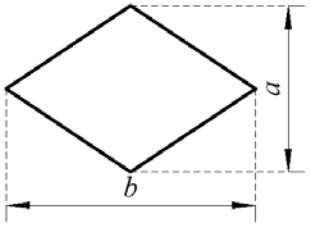
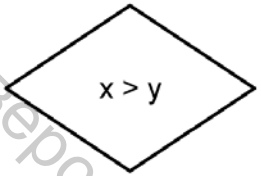
| Наименование | Обозначение и размеры в *мм* | Функция | Пример |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Данные |  | Символ отображает данные, носитель данных не определен.<br>В ГОСТ определено множество символов ввода/вывода, например вывод на магнитные ленты, дисплеи и т.п. Если источник данных не принципиален, обычно используется данный символ. |  |
| Ручной ввод |  | Символ отображает данные, вводимые вручную во время обработки с устройств любого типа (клавиатура, переключатели, кнопки, световое перо, полоски со штриховым кодом). |  |
| Процесс |  | Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).<br>В данном блоке обычно размещают одно или несколько операций присваивания, не требующих вызова внешних функций. |  |





15

Продолжение таблицы 3.1

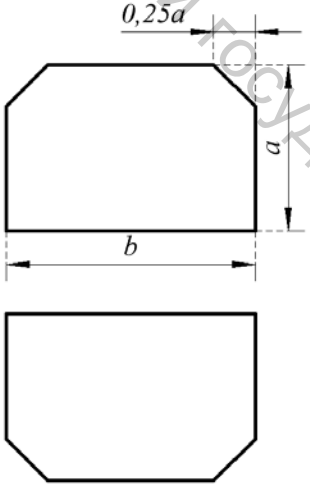
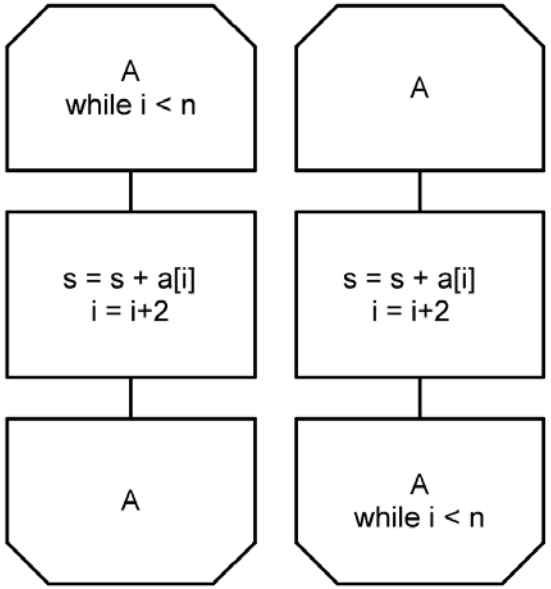
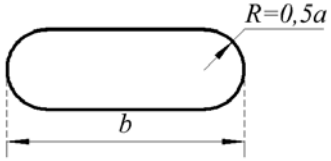
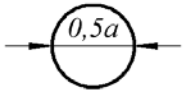
| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Предопределенный процесс |  | Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле). |  |
| Подготовка |  | Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы). Используется обычно для задания циклов со счетчиком. |  |
| Решение |  | Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. |  |



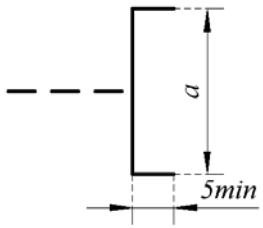
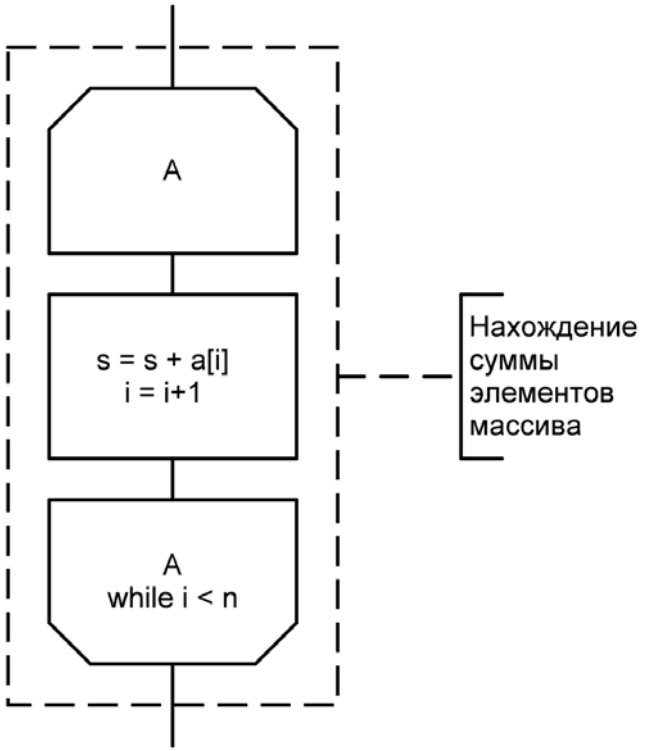


16

Продолжение таблицы 3.1

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Граница цикла |  | Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т. д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие.<br>Используется для описания циклов с пред- и постусловием. |  |
| Терминатор |  | Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных). |  |
| Соединитель |  | Разрыв линий потока |  |

Окончание таблицы 3.1

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Комментарий |  | Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев или примечаний должен быть помещен около ограничивающей фигуры. |  |

Размер *a* должен выбираться из ряда 10, 15, 20 *мм*. Допускается увеличивать размер *a* на число, кратное 5. Размер *b* равен 1,5 *a*. Допускается устанавливать *b* равным 2 *a*.

Правила выполнения блок-схем:

1. Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий.

2. Символы должны быть, по возможности, одного размера. Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация.

3. Минимальное количество текста, необходимого для понимания функции данного символа, следует помещать внутри данного символа. Если объем текста, помещаемого внутри символа, превышает его размеры, следует использовать символ комментария.

4. В схемах может использоваться идентификатор символов. Это связанный с данным символом идентификатор, который определяет символ для использования в справочных целях в других элементах документации (например, в листинге программы). Идентификатор символа должен располагаться слева над символом.

5. Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным.

В случаях, когда необходимо внести большую ясность в схему (например, при соединениях), на линиях используются стрелки.

Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.

6. В схемах следует избегать пересечения линий.

7. Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.

8. При необходимости линии в схемах следует разрывать для избежания излишних пересечений или слишком длинных линий, а также если схема состоит из нескольких страниц.

9. Несколько выходов из символа следует показывать несколькими линиями от данного символа к другим символам либо одной линией от данного символа, которая затем разветвляется в соответствующее число линий.

Каждый выход из символа должен сопровождаться соответствующими значениями условий, чтобы показать логический путь, который он представляет, с тем, чтобы эти условия и соответствующие ссылки были идентифицированы.

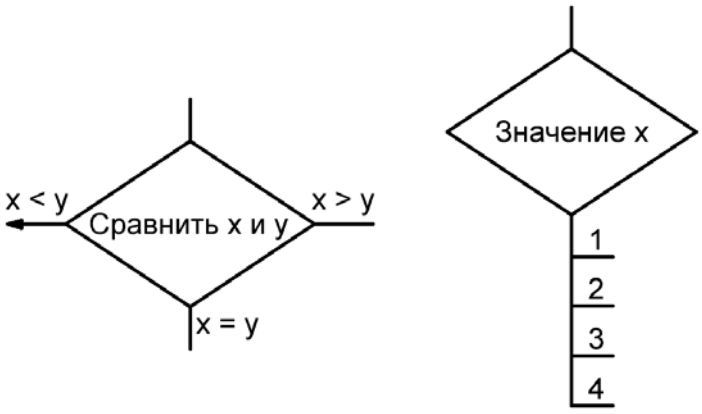



Рисунок 3.1 – Примеры нескольких выходов из символа на блок-схеме

Основные этапы построения фигуры:

1. Вычисление координат точек фигуры. Реализация отрисовки фигуры на основе проведенных вычислений.

2. Реализация движения фигуры по дуге из заданной начальной точки с условием возможности изменения направления движения и вращения при достижении заданной точки.

3. Реализация вращения фигуры относительно ее центра.

4. Реализация последовательного изменения цвета заштрихованной и незаштрихованной областей базовой фигуры.

5. Реализация возможности изменения направления движения и вращения при касании о границу рабочей области.

6. Реализация возможности изменения масштаба, скорости движения и вращения, а также остановки движения и вращения, сброса всех параметров до первоначальных базовой фигуры при помощи графических элементов интерфейса на панели инструментов.

На рисунках 3.2 и 3.3 приведен основной алгоритм и алгоритм *движения Фигуры*.

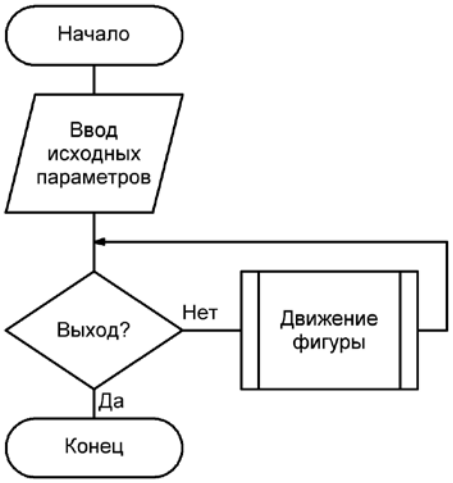



Рисунок 3.2 – Основной алгоритм

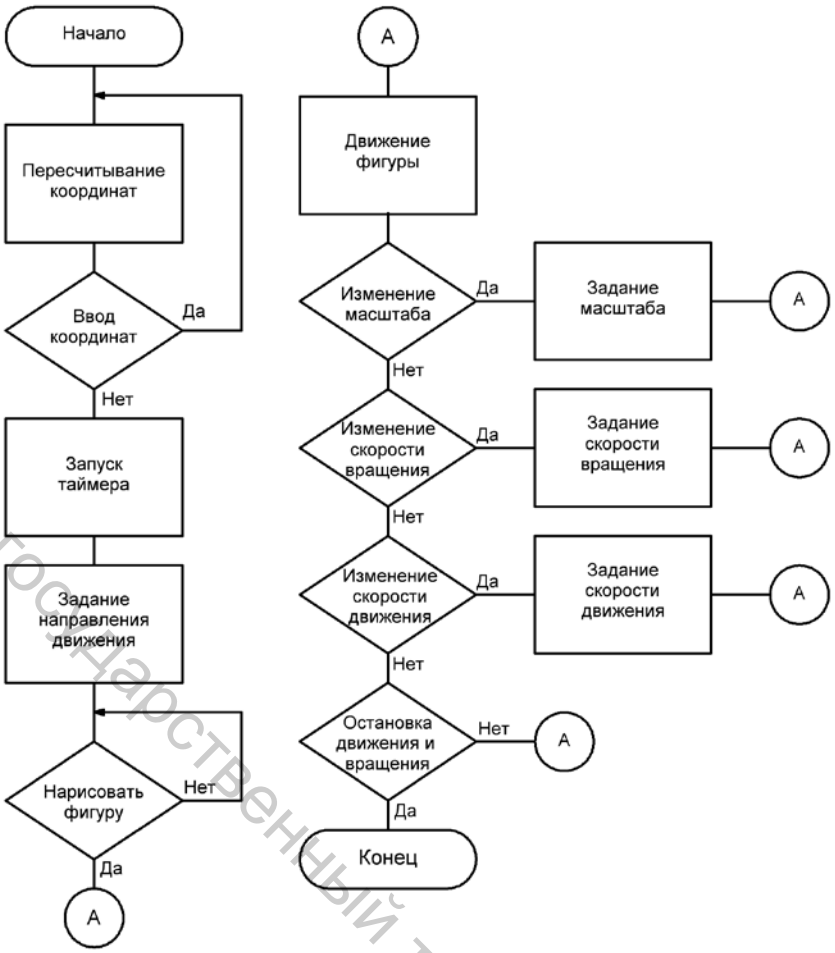



Рисунок 3.3 – Алгоритм движения фигуры

# 4 ОПИСАНИЕ ОСНОВНЫХ КОМПОНЕНТОВ ПРОГРАММЫ И ПОСЛЕДОВАТЕЛЬНОСТИ РАЗРАБОТКИ

Реализуемое графическое приложение оснащено следующими компонентами:

1. **PictureBox.** Компонент PictureBox – это область рисования. Компонент визуальный и используется для создания на форме некоторой области для рисования, то есть назначение этого компонента – дать простое окно для рисования изображений. Основных свойств объект не имеет. Использование этого компонента позволяет нам осуществить прорисовку фигуры в различных положениях. Его размеры 595; 500 пикселей.

2. **ButtonClick.** Компонент Button (кнопка) – командная кнопка, с помощью которой пользователь может вызывать выполнение какого-либо действия. Событие ButtonClick выполняет определённую команду, прописанную в конструкторе, при нажатии на кнопку.

3. **Timer.** Компонент Timer (таймер) служит для инициирования какого-либо события через определенные интервалы (промежутки) времени. Данный

компонент не является визуальным, поэтому при проектировании его значок можно поместить в любое место формы. Использование таймеров помогает нам осуществить движение фигуры, вращение, изменение цвета областей фигуры и её отскок при касании границ рабочей области.

4. **TrackBar.** Компонент TrackBar представляет собой ползунок, с помощью которого можно изменять размер объекта, скорость и др. В данном примере используется для увеличения фигуры, увеличения скорости движения и вращения. Ограничено такое изменение минимальным и максимальным значением. Использование TrackBar помогает взаимодействовать с фигурой и всячески её преображать.

5. **Label.** Компонент Label предназначен для показа текста на форме нашей программы, который не будет меняться в течение работы программы. Текст надписи компонента Label можно изменить. Он нужен для того, чтобы было проще понимать, какой элемент за что отвечает. В данном приложении выполняет роль пояснительных компонентов.

6. **TextBox.** Компонент TextBox предназначен для ввода данных на форме нашей программы, которые могут меняться в течение работы программы. В данной программе TextBox нужен для того, чтобы можно было вводить начальные координаты базовой фигуры во время запуска программы.

Основные компоненты программы: PictureBox, Timer(timer1), Button(btnDraw, btnStart, btnStop, btnDefaultA), Label(label1-4, lblXcYc), TrackBar(trbFigRotationRate, trbFigSpeed, trbFigSize):

timer1 – позволяет запускать определенные действия по истечению некоторого периода времени;

btnDraw – кнопка для отрисовки фигуры;

btnStart – кнопка запуска таймера;

btnStop – кнопка остановки таймера и возврата фигуры в начальное положение;

btnDefaultA – кнопка, устанавливающая размер фигуры по умолчанию;

label1-4 – отображение текста для пояснения;

lblXcYc – отображение координат центра тяжести фигуры;

trbFigRotRate – ползунок изменения скорости вращения;

trbFigSize – ползунок изменения размера;

trbFigSpeed – ползунок изменения скорости движения.

Чтобы учесть изменение цвета фигуры при движении вдоль траектории и одновременном вращении вокруг центральной точки, был проведен анализ, как должны изменяться цветовые координаты R, G, B в процессе движения. Мы должны учесть, что шестиугольник в начале траектории был желтого цвета (R=255; G=255; B=0), в конце – синего (R=0; G=0; B=255). Таким образом, координаты R и G при движении фигуры в прямом направлении убывают от 255 до 0, а координата B возрастает от 0 до 255. В промежуточных точках цветовые координаты линейно зависят от $x_c$ (рис. 4.1).

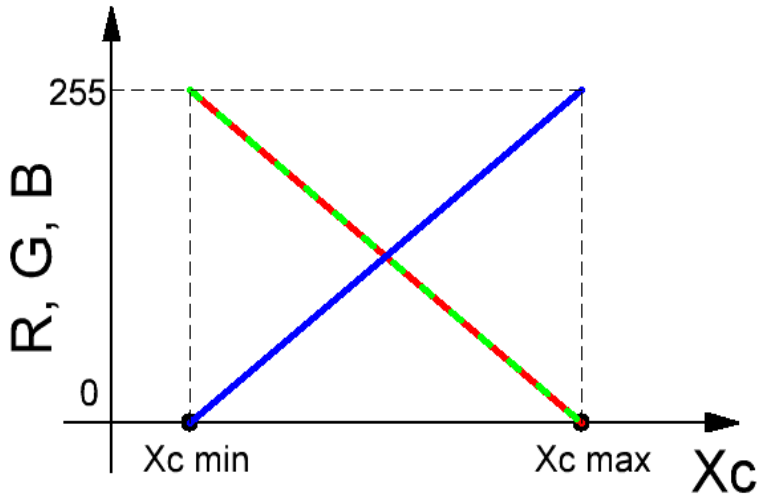



Рисунок 4.1 – Изменение цветовых координат R, G, B

Уравнения зависимости компонент R, G, B от $x_c$ получим как уравнение прямой по двум точкам:

$$\frac{x_c - x_{c\min}}{x_{c\max} - x_{c\min}} = \frac{R - 255}{0 - 255},$$
$$\frac{x_c - x_{c\min}}{x_{c\max} - x_{c\min}} = \frac{G - 255}{0 - 255},$$
$$\frac{x_c - x_{c\min}}{x_{c\max} - x_{c\min}} = \frac{B - 0}{255 - 0}.$$

Из этих уравнений необходимо выразить $R, G, B$, а $x_{c\min}$ и $x_{c\max}$ определяются шириной pictureBox и размером фигуры $a$. Этот процесс в коде программы реализован в двух методах: byte colorInc() – для возрастающих цветовых координат и byte colorDec() – для убывающих цветовых координат. Аналогичные рассуждения проводятся при учете изменения цвета треугольника от зеленого цвета (R=0; G=255; B=0) в начале траектории, до пурпурного (R=255; G=0; B=255) в конце.

# 5 ИСХОДНЫЙ КОД ПРОГРАММЫ

В процессе разработки программы вначале производились вычисления для задания базовой фигуры, затем был выбран наиболее подходящий для практической реализации целей и задач язык программирования C#. Был разработан алгоритм работы программы и написан код программы:

```
public partial class Form1 : Form
    { // объявляем переменные
        const int default_a = 120;
        int dir = 1; //направление движения по молчанию вперед
        double dx = 0.1; //приращение по x при движении
        double xc, yc; //координаты центра
        int a; //размер фигуры и координаты центра
        int v = 1; //скорость движения
        Point[] p1 = new Point[6]; //массив точек для шестиугольника
        Point[] p2 = new Point[3]; //массив точек для треугольника
        double[] x = new double[6];
        double[] y = new double[6];
        double fi; // угол, на который фигура поворачивается при
обновлении таймера
        double c_fi = 0; // угол, на который фигура повернулась
        int rotation_rate = 1; // скорость вращения

        public Form1()
        {
            InitializeComponent();
        }

        // при загрузке формы
        private void Form1_Load(object sender, EventArgs e)
        {
            a = default_a; // задаем размер фигуры
            // Вычисляем координаты центра, чтобы фигура находилась в
левом нижнем углу
            // pictureBox1.Width и pictureBox1.Height - ширина и высота
pictureBox
            xc = Convert.ToInt32((a / 2.0) * Math.Cos(Math.PI / 6));
            yc = Convert.ToInt32(pictureBox1.Height - a / 2.0);
            fi = 3 * dir * Math.PI / 180;
            // начальное положение ползунков
            trbFigRotRate.Value = rotation_rate;
            trbFigSize.Value = a;
            trbFigSpeed.Value = v;
        }

        // расчет координат x точек при вращении
        private double[] x_rot(double[] x0, double[] y0)
        {
            double[] x1 = new double[x.Length];
            for (int i = 0; i < x.Length; i++)
            {
                x1[i] = (x0[i]) * Math.Cos(c_fi) - (y0[i]) *
Math.Sin(c_fi);
            }
            return x1;
        }
```

```
        // Расчет координат y точек при вращении
        private double[] y_rot(double[] x0, double[] y0)
        {
            double[] y1 = new double[x.Length];
            for (int i = 0; i < x.Length; i++)
            {
                y1[i] = (y0[i]) * Math.Cos(c_fi) + (x0[i]) *
 Math.Sin(c_fi);
            }
            return y1;
        }

        // Расчет возрастающих цветовых координат
        private byte colorInc()
        {
            try
            {
                double col = 255 * (xc - (a / 2) * Math.Cos(Math.PI / 6))
/
                    (pictureBox1.Width - a * Math.Cos(Math.PI / 6));
                return Convert.ToByte(col);
            }
            catch { return 0; }
        }

        // Расчет убывающих цветовых координат
        private byte colorDec()
        {
            try
            {
                double col = -255 * ((xc - (a / 2) * Math.Cos(Math.PI /
6)) /
                                     (pictureBox1.Width - a *
Math.Cos(Math.PI / 6))) + 255;
                return Convert.ToByte(col);
            }
            catch { return 0; }
        }

        // Метод рисования фигуры
        private void ris()
        {
            Graphics g = pictureBox1.CreateGraphics(); // Создание
графического объекта
            Brush br1 = new SolidBrush(Color.FromArgb(colorDec(),
colorDec(), colorInc())); // кисть шестиугольника
            Brush br2 = new SolidBrush(Color.FromArgb(colorInc(),
colorDec(), colorInc())); //кисть треугольника
            g.Clear(SystemColors.Control); // стирание
```

```
            //рассчет координат точек фигуры относительно центра
            x[0] = Convert.ToInt32(0);
            y[0] = Convert.ToInt32(-a / 2.0);
            x[1] = Convert.ToInt32((a / 2.0) * Math.Cos(Math.PI / 6));
            y[1] = Convert.ToInt32(-(a / 2.0) * Math.Sin(Math.PI / 6));
            x[2] = Convert.ToInt32((a / 2.0) * Math.Cos(Math.PI / 6));
            y[2] = Convert.ToInt32((a / 2.0) * Math.Sin(Math.PI / 6));
            x[3] = Convert.ToInt32(0);
            y[3] = Convert.ToInt32(a / 2.0);
            x[4] = Convert.ToInt32(-(a / 2.0) * Math.Cos(Math.PI / 6));
            y[4] = Convert.ToInt32((a / 2.0) * Math.Sin(Math.PI / 6));
            x[5] = Convert.ToInt32(-(a / 2.0) * Math.Cos(Math.PI / 6));
            y[5] = Convert.ToInt32(-(a / 2.0) * Math.Sin(Math.PI / 6));

            //поворот фигуры
            double[] temp = new double[6];
            temp = x; //копируем массив x во временную переменную
            x = x_rot(temp, y);
            y = y_rot(temp, y);

            //заполняем массив точек шестиугольника
            p1[0] = new Point((int)(x[0] + xc), (int)(y[0] + yc));
            p1[1] = new Point((int)(x[1] + xc), (int)(y[1] + yc));
            p1[2] = new Point((int)(x[2] + xc), (int)(y[2] + yc));
            p1[3] = new Point((int)(x[3] + xc), (int)(y[3] + yc));
            p1[4] = new Point((int)(x[4] + xc), (int)(y[4] + yc));
            p1[5] = new Point((int)(x[5] + xc), (int)(y[5] + yc));
            //заполняем массив точек треугольника
            p2[0] = new Point((int)(x[0] + xc), (int)(y[0] + yc));
            p2[1] = new Point((int)(x[2] + xc), (int)(y[2] + yc));
            p2[2] = new Point((int)(x[4] + xc), (int)(y[4] + yc));

            g.FillPolygon(br1, p1);//отрисовка шестиугольника
            g.FillPolygon(br2, p2);//отрисовка треугольника

            lblXcYc.Text = String.Format("X: {0:0.00}, Y: {1:0.00}", xc,
yc); // вывод координат центра тяжести точки
        }

        // При нажатии кнопки "Нарисовать фигуру"
        private void btnDraw_Click(object sender, EventArgs e)
        {
            ris();
        }

        // При нажатии кнопки "Движение фигуры"
        private void btnStart_Click(object sender, EventArgs e)
        {
            timer1.Start();
```

```
        }

        // При нажатии кнопки "Остановить фигуру"
        private void btnStop_Click(object sender, EventArgs e)
        {
            timer1.Stop();
            xc = Convert.ToInt32((a / 2.0) * Math.Cos(Math.PI / 6));
            yc = Convert.ToInt32(pictureBox1.Height - a / 2.0);
            c_fi = 0;
            ris();
        }

        // При нажатии кнопки "Размер по умолчанию"
        private void btnDefaultA_Click(object sender, EventArgs e)
        {
            a = default_a;
            if (!timer1.Enabled) //проверка включенности таймера(true-
таймер выключен)
            {
                xc = Convert.ToInt32((a / 2.0) * Math.Cos(Math.PI / 6));
                yc = Convert.ToInt32(pictureBox1.Height - a / 2.0);
                ris();
            }
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (x.Max() + xc > pictureBox1.Width)
            {
                xc = pictureBox1.Width - x.Max();//если фигура заходит за
границу,
                //уменьшаем координату центра
                dir = -1;//меняем направление движения
            }

            //проверяем касание левой границы
            if ((x.Min() + xc) < 0)
            {
                xc = -x.Min();
                dir = 1;
            }

            c_fi += fi * rotation_rate;

            ris(); //перерисовываем фигуру

            dx = v * dir; //приращение координаты зависит от направления
и скорости
            fi = 1 * dir * Math.PI / (180); //угол меняет направление
            xc = xc + dx;
```

```
            yc = pictureBox1.Height / 2 + ((pictureBox1.Height - a) / 2)
* Math.Sin(0.03 * (xc - a / 2)); //движение по синусоиде
        }

        private void trbFigRotRate_Scroll(object sender, EventArgs e)
        { // Функция для изменения скорости вращения
            rotation_rate = trbFigRotRate.Value;
        }

        private void trbFigSpeed_Scroll(object sender, EventArgs e)
        { // Функция для изменения скорости без изменения направления
движения
            v = trbFigSpeed.Value;
        }

        private void trbFigSize_Scroll(object sender, EventArgs e)
        { // Функция для изменения размера фигуры
            a = trbFigSize.Value;
            if (!timer1.Enabled) //проверка включенности таймера(true-
таймер выключен)
            {
                xc = Convert.ToInt32((a / 2.0) * Math.Cos(Math.PI / 6));
                yc = Convert.ToInt32(pictureBox1.Height - a / 2.0);
            }
            ris();
        }
    }
```

# 6 ПРОВЕРКА КОРРЕКТНОСТИ РАБОТЫ ПРОГРАММЫ В РАЗЛИЧНЫХ РЕЖИМАХ

При тестировании проверяется реализация функциональных требований, то есть возможность программного продукта выполнять те функции, которые были описаны в задании курсовой работы. Функциональное ручное тестирование является наиболее распространенным видом тестирования, применяемым повсеместно. Панель инструментов окна содержит:

- кнопку для задания фигуры по умолчанию;
- кнопку нарисовать фигуру;
- кнопку, задающую движение фигуры;
- кнопку остановки фигуры.

А также имеется возможность изменять размер, скорость движения фигуры и скорость вращения фигуры вокруг центральной точки.

На рисунках 6.1 и 6.2 изображён процесс отрисовки фигуры в начале движения, имеющий размер по умолчанию и с учетом масштабирования.

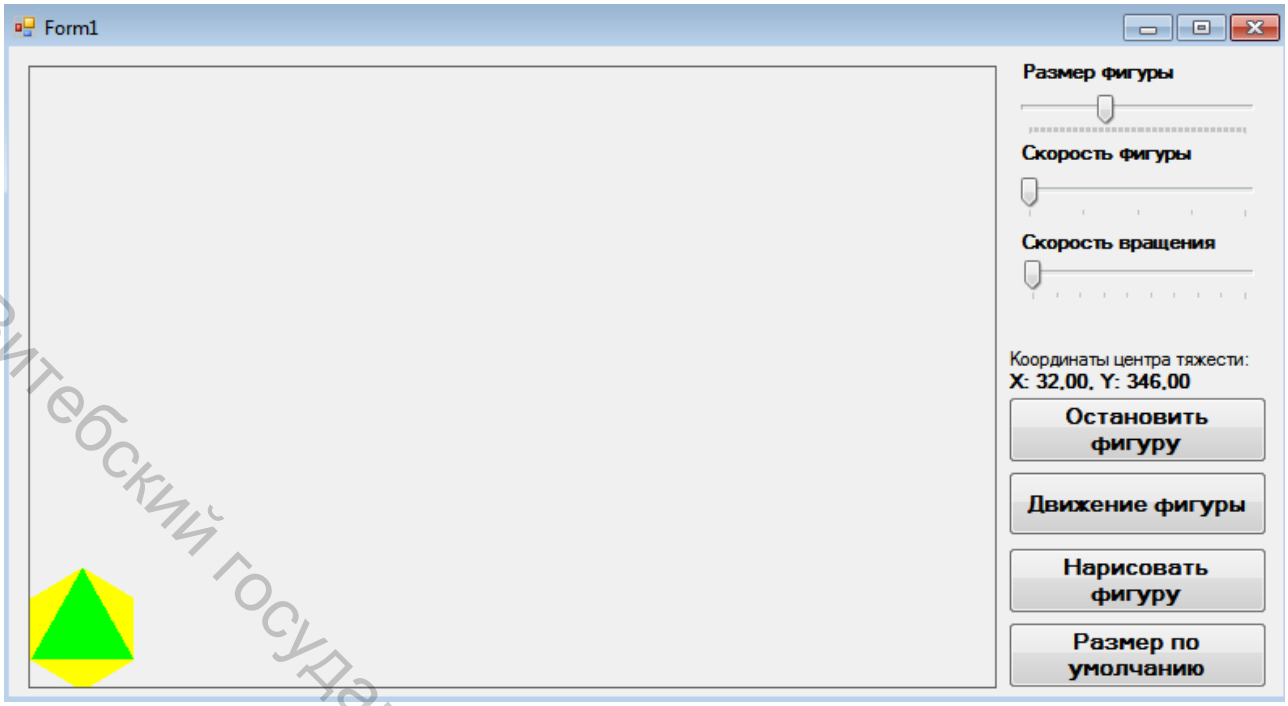



Рисунок 6.1 – Рабочее окно программы с начальным положением фигуры


Рисунок 6.2 – Рабочее окно программы с увеличенным масштабом фигуры

На рисунках 6.3 и 6.4 демонстрируется работа программы с изменяющимся цветом фигуры в произвольной точке и в конце траектории.

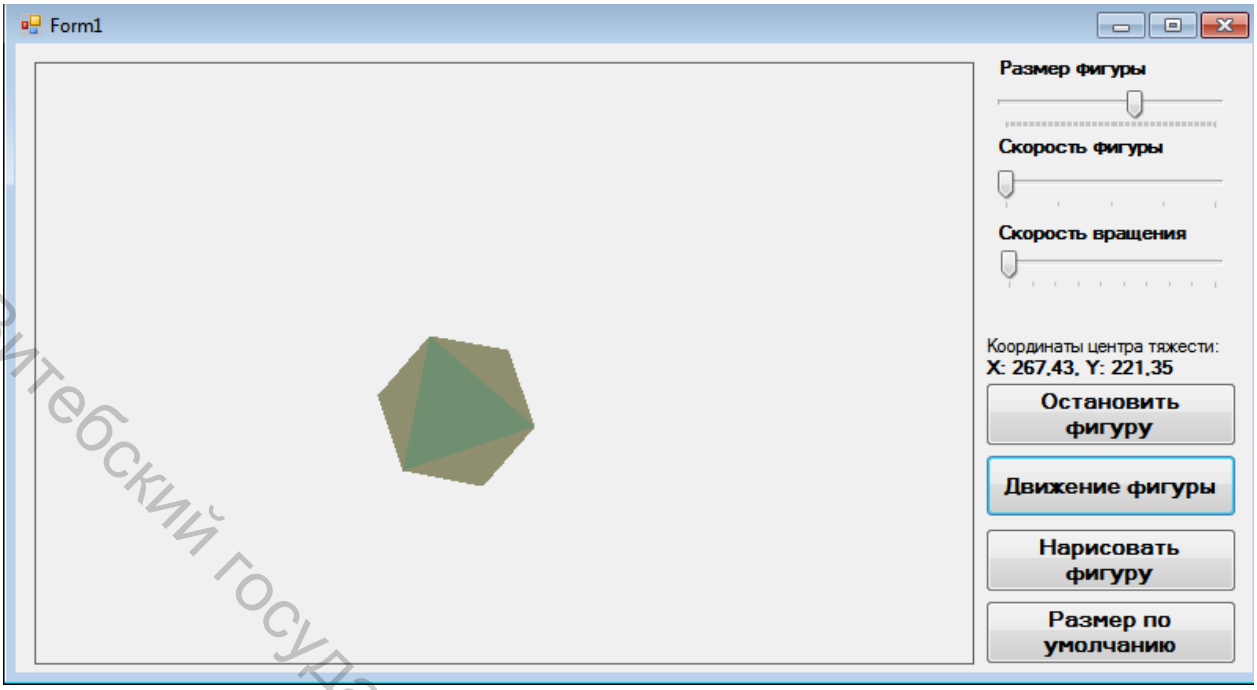



Рисунок 6.3 – Рабочее окно программы с изменяющимся цветом фигуры при движении

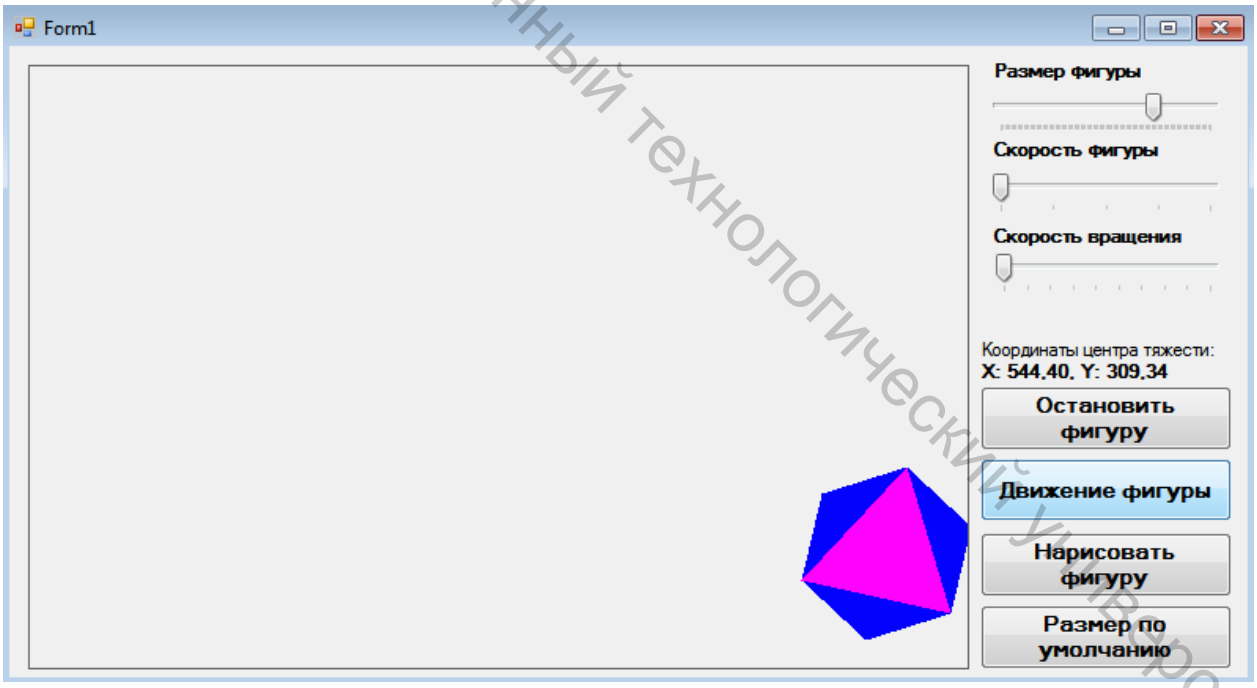



Рисунок 6.4 – Рабочее окно программы с изменённым цветом фигуры в конце траектории

При тестировании программы выполнена проверка возможности изменения скорости движения фигуры и скорости вращения данной фигуры вокруг центральной точки. При тестировании программы неточности не обнаружены, программа работает корректно в разных режимах.

# 7 ОБЩИЕ РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ ЗАКЛЮЧЕНИЯ

Заключение – важный структурный элемент курсовой работы, в котором подводятся итоги исследования. В нем необходимо системно и последовательно изложить наиболее важные положения (выводы), предложения по работе.

Выводы и предложения, сформулированные в главах и параграфах, содержательно повторяются в **заключении** как структурной части курсовой работы. Общий вывод в конце **заключения** – ответ на поставленную цель.

Выводы и предложения должны быть ясными, четкими, однозначными, непротиворечивыми, изложены с использованием научного стиля.

Объем заключения, как правило, составляет около одной-двух страниц. Пример заключения к курсовой работе приведен в приложении В.

# ЛИТЕРАТУРА

1. Викторова, Т. С. Системы компьютерной графики / Т. С. Викторова, С. Д. Парфенов. – Вязьма : филиал ФГБОУ ВПО «МГИУ» в г. Вязьме, 2012. – 165 с.

2. Постнов, К. В. Компьютерная графика / К. В. Постнов. – Москва : МГСУ , 2009 – 247 с.

3. Миронов, Д. Ф. Компьютерная графика в дизайне / Д. Ф . Миронов. – СПб : БХВ-Петербург, 2008. – 341 с.

4. Яне, Б. Цифровая обработка изображений / Б. Яне. – М.: Техносфера, 2007. – 584 с.

5. Миано, Дж. Форматы и алгоритмы сжатия изображений в действии / Дж. Миано. – Москва: Триумф, 2003. – 336 с.

6. Домасев, М. В. Цвет, управление цветом, цветовые расчеты и измерения / М. В. Домасев, С. П. Гнатюк. – Санкт-Петербург: Питер, 2009. – 224 с.

7. Павловскач, Т.А. C/C++. Процедурное и объектно-ориентированное программирование: учебник для студентов высших учебных заведений, обучающихся по направлению подготовки дипломированных специалистов «Информатика и вычислительная техника» / Т. А. Павловская. – Санкт-Петербург: Питер, 2015. – 495 с.

# ПРИЛОЖЕНИЕ А
# ОБРАЗЕЦ ОФОРМЛЕНИЯ ТИТУЛЬНОГО ЛИСТА КУРСОВОЙ РАБОТЫ

Министерство образования Республики Беларусь
Учреждение образования
«Витебский государственный технологический университет»
Кафедра «Информационные системы и автоматизация производства»



КУРСОВАЯ РАБОТА
по дисциплине «Системы компьютерной графики»
на тему «Разработка приложения для создания и преобразования графических объектов»

*Выполнил:*
студент факультета_____
_____
группы_____ _____ _____
*подпись* *Ф.И.О.*

*Проверил:*
Руководитель_____
*должность*
_____
_____
*ученая ст епень, ученое звание* *Ф.И.О.*

_____
отметка о допуске к защите
«____» _____20___г._____
*дат а* *подпись руководит еля*

Витебск, 20___

# ПРИЛОЖЕНИЕ Б
# ОБРАЗЕЦ ОФОРМЛЕНИЯ ВВЕДЕНИЯ К КУРСОВОЙ РАБОТЕ

## ВВЕДЕНИЕ

Данная работа подразумевает собой создание графического приложения, которое эмулирует движение базовой фигуры таким образом, чтобы ее центр двигался по синусоиде. При движении вдоль траектории фигура одновременно вращается вокруг центральной точки, а также меняет цвет. Скорость движения фигуры вдоль траектории, а также скорость вращения вокруг центральной точки должна задаваться пользователем. Необходимо предусмотреть возможность масштабирования фигуры без изменения пропорций и возможностью возврата к исходному размеру.

Запуск и остановка режима эмуляции движения, а также все параметры данного режима задаются пользователем при помощи графических элементов интерфейса, а именно панели инструментов.

Для реализации проекта будет использоваться язык программирования высокого уровня C#, с применением среды разработки Visual Studio 2015.

Среда Microsoft Visual C# поддерживает технологию так называемой быстрой разработки (ее также часто называют «компонентной»). В ее основе лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть рутины, оставляя программисту работу по созданию диалоговых окон (визуальное проектирование) и функций обработки событий (событийное программирование).

Основоположником всех компонентов является класс Component



| | | | | | УО «ВГТУ» КР.025 1-40 05 01-01 ПЗ | | |
|---|---|---|---|---|---|---|---|
| Изм | Лист | № докум. | Подпис | Дат | | | |
| Разраб. | | Иванов А.Ю. | | | Введение | Лит. | Лист 34 / Листов 37 |
| Провер. | | Ф.И.О | | | | | |
| Реценз. | | | | | | УО «ВГТУ» каф. ИСАП гр. Ит-4 | |
| Н. Контр. | | | | | | | |
| Утверд. | | | | | | | |

(пространство имен System.ComponentModel). Этот класс умеет взаимодействовать со средой разработчика и позволяет инспектору объектов отображать свойства и события компонентов. Все видимые компоненты – это наследники класса Control (пространство имен System.Windows.Forms), который, в свою очередь, является прямым наследником класса Component.

Технология Microsoft .NET основана на идее универсального программного кода, который может быть выполнен любым компьютером, вне зависимости от используемой операционной системы. Универсальность программного кода обеспечивается за счет предварительной (выполняемой на этапе разработки) компиляции исходной программы в универсальный промежуточный код (CIL-код, Common Intermediate Language), который во время запуска программы транслируется в выполняемый.

Исходя из цели курсовой работы, необходимо решить следующие задачи:

1) проанализировать исходный объект;
2) выбрать язык программирования;
3) разработать алгоритм работы программы;
4) написать код программы;
5) произвести тестирование программы.

# ПРИЛОЖЕНИЕ В
# ОБРАЗЕЦ ОФОРМЛЕНИЯ ЗАКЛЮЧЕНИЯ К КУРСОВОЙ РАБОТЕ

## ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было разработано графическое приложение, которое непосредственно показывает возможности создания и преобразования графических объектов с помощью языка программирования C#.

Данное приложение эмулирует движение базовой фигуры таким образом, чтобы ее центр описывал синусоидальную траекторию. При движении вдоль траектории фигура одновременно вращается вокруг центральной точки, а так же меняет цвет. Скорость движения фигуры вдоль траектории и скорость вращения вокруг центральной точки задается пользователем. Реализована возможность масштабирования фигуры без изменения пропорций и предусмотрен способ возврата к исходному размеру.

Запуск и остановка режима эмуляции движения, а так же все параметры данного режима задаются пользователем при помощи графических элементов интерфейса, а именно панели инструментов.

При тестировании программы неточности не обнаружены, программа работает корректно в разных режимах.

В итоге были решены все поставленные задачи:

1) проанализирован исходный объект;
2) выбран язык программирования C#;
3) разработан алгоритм работы программы;
4) написан код программы;
5) произведено тестирование программы.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | УО «ВГТУ» КР.025 1-40 05 01-01 ПЗ | | |
| Изм. | Лист | № докум. | Подпис | Дат | | | |
| Разраб. | | Иванов А.Ю. | | | Заключение | Лит. | Лист 36 / Листов 37 |
| Провер. | | Ф.И.О | | | | | |
| Реценз. | | | | | | УО «ВГТУ» каф. ИСАП гр. Ит-4 | |
| Н. Контр. | | | | | | | |
| Утверд. | | | | | | | |

Учебное издание

# СИСТЕМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ

Методические указания по выполнению курсовой работы

Составители:
Надежная Наталья Леонидовна
Гниденко Андрей Константинович
Соколова Анна Сергеевна
Дунина Елена Брониславовна

Редактор *Т.А. Осипова*
Корректор *Т.А. Осипова*
Компьютерная верстка *Н.Л. Надежная*

---

Подписано к печати 11.12.2018. Формат 60x90 $^{1}/_{16}$. Усл. печ. листов 2,3.
Уч.-изд. листов 2,4. Тираж 30 экз. Заказ № 360.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.