**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ**

государственное профессиональное образовательное учреждение   
Ярославской области

Рыбинский полиграфический колледж

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| Класс визуального компонента-игры «Arcanoid» | |
|  | |
| по дисциплине | Объектно-ориентированное программирование |
|  | |

Пояснительная записка

|  |
| --- |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы | 3-ИС-2 |  |  |  | Пахомов И А |
|  | *(Код учебной группы)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Руководитель | преподаватель |  |  |  | Смирнов В Б |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *( И.О.Фамилия)* |
| Нормоконтроль | преподаватель |  |  |  | Смирнов В Б |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |

г. Рыбинск

2022

содержание

Введение 3

1 Исследовательский раздел 5

2 Конструкторский раздел 6

2.1 Математическая модель 6

2.2 Разработка структуры класса 7

2.3 Разработка графического интерфейса компонента 7

2.4 Описание входных и выходных данных класса 7

2.5 Разработка алгоритмов класса 8

3 Технологический раздел 8

3.1 Отладка программного кода класса 9

3.2 Инструкция по установке компонента в среду разработки 9

3.3 Инструкция по эксплуатации компонента 11

4 Раздел охраны труда 12

Заключение 12

{Заполняй сразу угловой штамп внизу страницы.}

Введение

На сегодняшний день вычислительная техника тесно связана практически со всеми людьми на планете. Вычислительная техника, упрощает жизнь в десятки раз. С её помощью человек может развлечься, найти буквально все, что хочет в глобальной сети Интернет. В данный момент, человек не может представить жизнь без всех удобств, что дает вычислительная техника.

Вычислительная техника, не всегда была такой удобной и простой для понимания. Все начиналось с машинного кода, на основе которого работают языки программирования, способствующие работе современных программ и приложений. Упомянутые выше языки программирования служат для упрощения программного кода. Первый язык программирования “Ассемблер” – низкоуровневый, создан по принципу «Одна инструкция – одна строка». Низкоуровневые языки, ориентированы на конкретный тип процессора и учитывают его особенности. Их появление, действительно облегчило написание кода, так как вместо единиц и нулей, можно было писать программу командами, состоящую из символов, приближенных к разговорному языку.

Наиболее важной проблемой, с которыми столкнулись разработчики языков второго поколения, стала задача убедить клиентов в том, что созданный компилятором код выполняется достаточно хорошо, чтобы оправдать отказ от программирования на ассемблере. Скептицизм по поводу возможности создания эффективных программ с помощью автоматических компиляторов был довольно распространён, поэтому разработчикам таких систем должны были продемонстрировать, что они действительно могут генерировать почти такой же эффективный код, как и при ручном кодировании, причём практически для любой исходной задачи. Но сложность разработки в нём больших программных комплексов, привела к появлению языков третьего поколения — языков высокого уровня. Такие языки были машинно-независимы, для каждого такого языка был компилятор. В связи с ростом сложности программ и развитием программных средств, возникла необходимость увеличить производительность труда программистов, что привело к разработке структурного программирования. Следующим достижением были [процедуры](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [функции](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), то есть, если есть задача, которая выполняется несколько раз, то её можно объявить как функцию или как процедуру и в выполнении программы просто вызывать её. Программный код становится меньше. Следующим достижением было объединение разнородных данных, которые используются в программе в связке, в структуры. С развитием технологий родилось объектно-ориентированное программирование. При использовании структур данных в программе вырабатываются и соответствующие им функции для работы с ними. Это привело к мысли их объединить и использовать совместно, так появились классы. Теперь программирование можно было разбить на классы и тестировать не всю программу, состоящую из 10 000 строк кода, а разбить программу на 100 классов, и тестировать каждый класс. Объектно-ориентированное программирование сочетает лучшие принципы структурного программирования с новыми концепциями [инкапсуляции](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [полиморфизма подтипов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D0%BE%D0%B2) и [наследования](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

В ходе работы над курсовым проектом требуется разработать компонент-игру «Arcanoid» для одной из современных сред разработки программного обеспечения. Планируется, что компонент-игра будет реализовывать базовую версию игры «Arcanoid», будет выполнять все необходимое для полноценного игрового процесса. Актуальность проекта, прежде всего, заключается в возможности использовать его результаты в прикладных программах. Программистам не потребуется писать программный код с нуля, они смогут посмотреть где и как использовался метод или функция, достаточно понять программный код и взять себе необходимое.

Как таковая игра будет полезна для детей дошкольного и младшего школьного возраста, она позволит им потратить некоторое время, также поможет развить логику, ловкость рук и реакцию у детей помладше. Для начинающих программистов данный проект будет полезен тем, что позволит взглянуть на возможности, которые им предстоят в будущем, это заинтересует их обучатся программированию.

1 Исследовательский раздел

Для реализации проекта существуют две наиболее подходящие среды программирования - Borland Delphi 7 и Visual Studio 2019. Рассмотрим Borland Delphi 7. Среда не новая, легкая в понимании, практичная, есть возможность использовать только один язык программирования и к сожалению, среда устарела.

Visual Studio 2019 – это новая среда программирования с удобным интерфейсом и возможностью писать код на многих языках программирования, также у этой среды имеются ошибки, которые не имеют четкого решения. (тут тавтология(имеются - имеют). Убрать или оставить ?) Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). В отличии от других сред Visual Studio более удобен и практичен, с этой средой намного приятнее работать. У этой среды есть минус – не все ошибки, как упомянуто выше имеют четкое решение.

Сравним среды программирования, начиная с Borland Delphi7. Плюсы среды: работает на старых компьютерах, занимает не много места на диске, работает на Microsoft Windows 2000 и старше. Система предсказуема, то есть имеет решения на все ошибки.

Минусы среды Borland Delphi7 : при компилировании имеет достаточно большой размер .exe фалов на диске, писать код можно только на одном языке (Delphi). В целом среда неплохая, для своих задач будет хорошо справляться.

Плюсы Visual Studio 2019: совершенно новая система, удобный интерфейс и дизайн, много полезных дополнений, можно писать на многих языках к примеру на C#, C++, F#. Среда постоянно обновляется.

Минусы Visual Studio 2019: ошибки, на которые так и не найдены решения, не бесплатна, занимает большой размер на диске.

Для реализации компонента-игры «Arcanoid», выбрана среда программирования Visual Studio 2019. Среда имеет максимум преимуществ и минимум недостатков.

В качестве родительского класса в проекте будет выбран класс Control. Класс отвечает за визуальное представление в форме. OnPaint - один из методов класса Control, отвечает за отрисовку в компоненте. Также есть метод Invalidate, который перерисовывает элемент управления. Проект будет соответствовать требованиям графического интерфейса операционных систем Microsoft Windows.

В глобальной сети «Интернет» есть аналоги игры «Arcanoid». Рассмотрим «Arcanoid базовый» с сайта «https://play.arkanoid.online/». Игрок контролирует небольшую платформу-ракетку, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз. Удар шарика по кирпичу приводит к разрушению кирпича. После того как все кирпичи на данном уровне уничтожены, происходит переход на следующий уровень, с новым набором кирпичей.

Есть и аналоги с некоторыми разнообразиями: определённые кирпичи нужно ударять несколько раз, иногда появляются летающие враги, от которых отталкивается шарик, удар по некоторым кирпичам приводит к выпадению из них капсул-призов — приз активируется, если поймать такую капсулу ракеткой. У таких аналогов есть один недостаток – они все требуют доступ в Интернет.

Проект будет насыщен всей базовой логикой: управление платформой с помощью манипулятора типа «мышь» и клавиатуры, генерация кирпичиков, движение шарика по правильной траектории, счёт и количество жизней. Также будет реализована адаптивность компонента. Отрисовка игрового поля, будет происходить с помощью метода OnPaint.

Основным процессом для реализации является движение шарика по правильной траектории. Написать логику для правильной траектории – довольно сложно, не всем хочется смотреть, как шарик летит, только под углом 45 градусов.

2 Конструкторский раздел

2.1 Математическая модель

За движение шарика будет отвечать метод, в котором будут происходить проверки выхода шарика за пределы поля, отскок шарика от платформы, отскок от кирпичика, отскок от стенки.

Скорость полета шарика будет зависеть от интервала обновления таймера. Чем больше интервал, тем медленнее летит шарик, сама скорость будет изменяемой, благодаря интервалу таймера. Угол отражения шарика будет зависеть от угла его падения при столкновении с платформой, кирпичиком или стенкой. Если объект стоит на месте, то угол падения равен углу отражения, если же объект будет двигаться, то угол падения увеличивается. После каждого обновления, будет вызываться метод проверки столкновении шарика. Игра окончена в том случае, если пользователь не поймал мяч платформой.

Изначально платформа с шариком будут рисоваться в центре игрового поля. Блоки рисуются в верхней части игрового поля, с каждым пятым сломанным блоком, будет добавляться новая линия блоков, все блоки будут цвета, который укажет пользователь. Движение платформы будет происходить по координате X, сама платформа будет иметь цвет, который укажет пользователь. За движение шарика будут отвечать координаты X и Y, цвет шарика задаётся пользователем.

{В этом подразделе было бы неплохо описать, как будет описываться траектория полета шарика, и как она будет меняться при ударе о движущуюся платформу! Приведи зависимость или хотя бы как-то опиши ее!}

Так я вроде во втором абзаце написал это (Угол отражения шарика будет зависеть от угла его падения при столкновении с платформой, кирпичиком или стенкой. Если объект стоит на месте, то угол падения равен углу отражения, если же объект будет двигаться, то угол падения увеличивается.)

2.2 Разработка структуры компонента

Для описание метода проектирования будет использоваться декомпозиция, начиная с “черной сферы”, заканчивая “вторым этапом” декомпозиции. Черная сфера – это система, в которой доступны лишь входные и выходные параметры. Модель «Черная сфера» изображена на рисунке 1.1

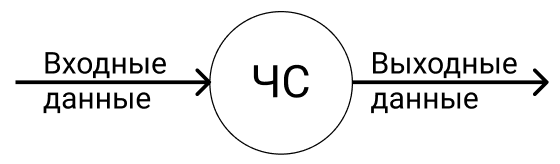


Рисунок 1.1 – Модель “Черная сфера”

В компонент-игре будут присутствовать такие входные параметры: размеры, цвета, управляющее воздействие пользователя. На выходе будем получать: изображение на экране, счёт. Размеры будут редактироваться с помощью манипулятора типа “мышь”, цвета задаются пользователям, управляющее воздействие пользователя будет осуществляться с помощью клавиатуры и манипулятора типа “мышь”. Счёт и изображение будет выводится на экран. Класс в виде черной сферы изображен на рисунке 1.2.

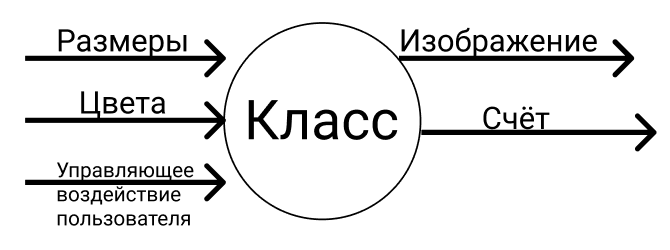


Рисунок 1.2 – Представление класса в виде модели «Черная сфера»

Декомпозиция – операция мышления, состоящая в разделении целого на части. Декомпозиция поможет нам правильно понять суть того или иного свойства или метода.

Получим первый этап декомпозиции с основными внутренними структурами данных, необходимые для хранения процессов и информации компонента. От изменения размеров компонента будут зависеть игровые элементы. Изменения цветов будут вызывать перерисовку компонента и вывод его на экран с указанным цветом. Нажатие клавиш будет обрабатываться событием, которое вызывается при нажатии определенной клавиши, тем самым меня направление платформы по оси X. Событие отвечающие за изменение счёта, будет выводить его на экран. Первый этап декомпозиции представлен на рисунке 1.3

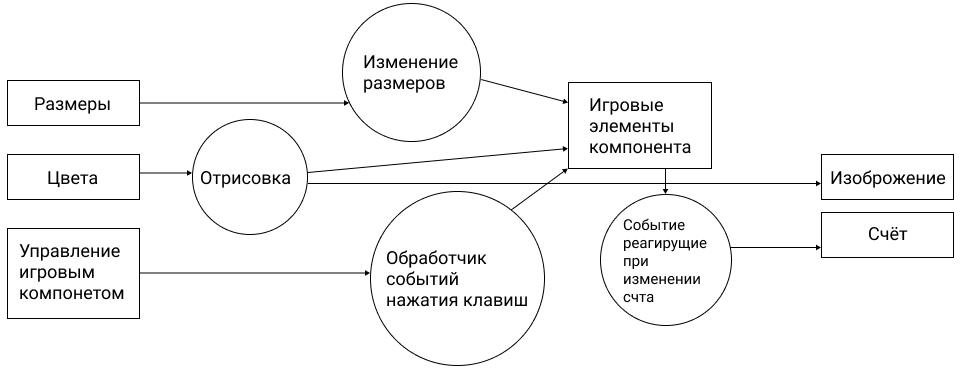


Рисунок 1.3 - Первый этап декомпозиции

В первом этапе декомпозиции … (нужно код доделать окончательно, чтобы начать делать второй этап).

2.3 Разработка графического интерфейса компонента

Интерфейсы современных пользовательских программ имеют технологию WIMP – Windows, Icons, Mouse, Pop-up. Получается так, что основными элементами графических интерфейсов являются окна, пиктограммы, компоненты ввода-вывода и мышь. Окно – это ограниченная рамкой область экрана, которая может менять размеры и местоположение в пределах экрана. Окна можно разделить на пять категорий: основные, дочерние, окна диалога, информационные, окна меню. Пиктограмма – это небольшое окно с графическим изображением. Различают следующие пиктограммы: программные пиктограммы, пиктограммы дочерних окон, пиктограммы панели инструментов, пиктограммы объектов. Прямое манипулирование объектами- это возможность замены команды воздействия на некоторый объект физическим действием в интерфейсе, осуществляемым с помощью мыши. При этом любая область экрана рассматривается как адресат, который может быть активизирован при подведении курсора и нажатии клавиши мыши. По реакции на воздействие различают следующие типы адресатов: указание и выбор, буксировка и «резиновая нить» (перенос объекта и его границ), экранные кнопки. Для реализации диалогов, управляемых пользователем, применяют меню различных видов. Для реализации диалогов, управляемых системой, обычно используют диалоговые окна.

Развитие графических интерфейсов Windows имеет большую историю. Windows 1.0 - особенностью этой операционной системы было наличие окон, которые можно было расположить друг около друга и возможность управления манипулятором типа “мышь”. В следующей версии операционной системы Windows (Windows 2.0) окна могли перекрывать друг друга, для отображения интерфейса использовались 16 цветов, также появилось функции “свернуть” и “развернуть”. В Windows 3.0 пользователь мог любоваться 256-ю цветами, что однозначно делало работу более приятной, хотя основные элементы интерфейса остались без существенных изменений. Windows 95-98-98SE-2000-ME-XP-Vista-7 закрепляют успех и позиции Microsoft на рынке операционных систем. В этом ряду стоит выделить, наверное, Windows Vista – в плане развития графического интерфейса она не выделяется из общего ряда, но прохладно принята общественностью. Windows 8 – пионер операционных систем с оконным интерфейсом, «тайлами» и возможностью управления с помощью мыши, клавиатуры и сенсорного экрана (пальцами).

Визуальный компонент должен иметь концепцию графических образов и набор приемов управления в соответствии с теми, что используются в операционной системе Windows. Это облегчит действия конечного пользователя и сэкономит время. Также будет обеспечена возможность использования манипулятора типа “мышь” для управления параметрами и настройки компонента.

Визуальный компонент будет выглядеть в виде поля, где высота больше, чем ширина в два раза. Также на поле будут присутствовать кирпичики, занимающее от высоты поля. Платформа, будет располагаться на оси X - по центру, а по оси Y - от высоты игрового поля, также на самой платформе будет шарик, который в ходе игры будет двигаться сбивая кирпичики. Компонент будет адаптивным, то есть будет иметь возможно подстраиваться под размеры окна и также будет иметь публичные свойства для изменения цвета некоторых элементов.

2.4 Описание входных и выходных данных компонента

В подразделе 2.2 “Разработка структуры компонента” была получена структурная схема компонента. Для удобства все свойства и события класса сведены в табличный вид. Таблица свойств представлена в таблице 1.1

Таблица 1.1 – Свойства класса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя | Тип данных | Тип доступа | Доступ на чтение | Доступ на запись |
| Score | Int | чтение/запись | Используется специально разработанный механизм, который вернет значение переменной score | Используется специально разработанный механизм обновления значения переменной score |
| MapColor | Color | чтение/запись | Используется специально разработанный механизм, который вернет значение переменной \_mapColor | Используется специально разработанный механизм обновления значения переменной \_mapColor, обновление картинки на экране |
| BallColor | Color | чтение/запись | Используется специально разработанный механизм, который вернет значение переменной \_ballColor | Используется специально разработанный механизм обновления значения переменной \_ballColor, обновление картинки на экране |
| PlatformColor | Color | чтение/запись | Используется специально разработанный механизм, который вернет значение переменной \_platformColor | Используется специально разработанный механизм обновления значения переменной \_platformColor, обновление картинки на экране |
| BlockColor | Color | чтение/запись | Используется специально разработанный механизм, который вернет значение переменной \_blockColor | Используется специально разработанный механизм обновления значения переменной \_blockColor, обновление картинки на экране |

Таблица событий представлена в таблице 1.2.

Таблица 1.2

|  |  |  |  |
| --- | --- | --- | --- |
| Имя события | Тип данных | Тип доступа | Метод доступа на чтение |
| RecScore | EventHandler | чтение | Событие происходящие при изменении счёта. Отвечает за изменение счёта на экране |

2.5 Разработка алгоритмов компонента

В подразделе 2.2 “Разработка структуры компонента” была получена структурную схему компонента. Из структурной схемы следует разработать алгоритмы всех методов в блок-схемы. (Для начала сделай 2-3 блок-схемы к уже известным подпрограммам и описание к ним и пришли мне на проверку в следующий раз!!! Обычно тут бывает много ошибок, я тебе все их перечислю. А потом уже будешь делать в полном объеме!)

XToCol – метод преобразующий координаты шарика в колонку массива. Входные параметры x и dir типа Int, выходные параметры отсутствуют. Блок-схема метода приведена на рисунке 1.4

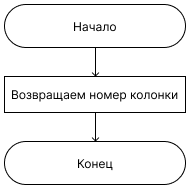


Рисунок 1.4

YToRow – метод преобразующий координаты шарика в столбец массива. Входные параметры y и dir типа Int, выходные параметры отсутствуют. Блок-схема метода приведена на рисунке 1.5

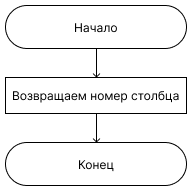


Рисунок 1.5

InitFunction – метод инициализирующий массив игрового поля. Входные и выходные параметры отсутствуют. Блок-схема метода приведена на рисунке 1.6

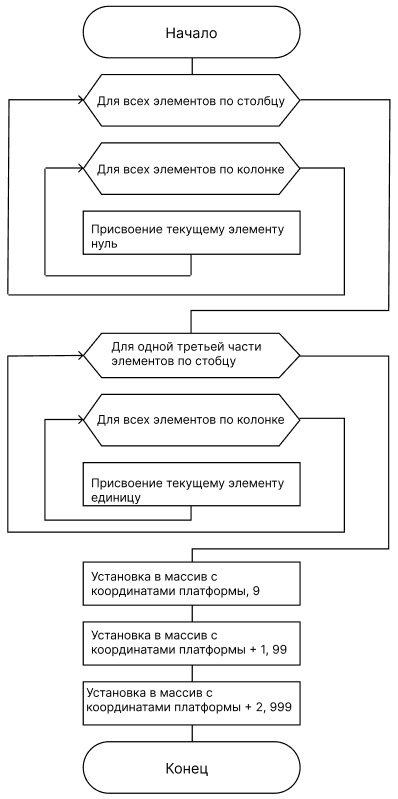


Рисунок 1.6

3 Технологический раздел

3.1 ОТЛАДКА КОМПОНЕНТА

3.2 Инструкция по установке компонента

Установка компонента в проект происходит следующим образом: в обозревателе решений правой кнопкой мыши кликаем на ветку “Ссылки”. В выпадающем меню выбираем “Добавить ссылку”. Вышеупомянутые действия показаны на рисунке 3.1

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Рисунок 3.1

После выполнения вышеописанных действий на экране появляется окно “Менеджер ссылок”. В данном окне нам необходимо выбрать последний раздел “Обзор” и соответственно нажать на кнопку “Обзор”. Все описанные действия представлены на рисунке 3.2

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Рисунок 3.2

В появившемся окне необходимо выбрать файловый пакет с компонентом и выбрать его. Рисунок 3.3

Изображение выглядит как текст, монитор, снимок экрана, черный

Автоматически созданное описание

Рисунок 3.3

После выбора файлового пакета, необходимо в панели элементов создать новую вкладку, и вызвать на ней контекстное меню, после выбрать команду “выбрать элементы”. Рисунок 3.4

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Рисунок 3.4

В появившемся окне необходимо нажать кнопку “Обзор” в правом нижнем углу окна. Рисунок 3.5

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.5

В окне выбора файлов необходимо выбрать файловый пакет с компонентом, тот же, что и на рисунке 3.3. Рисунок 3.6

Изображение выглядит как текст, монитор, снимок экрана, внутренний

Автоматически созданное описание

Рисунок 3.6

В панели элементов появился компонент, перетаскиваем его на форму. Результат показан на рисунке 3.7

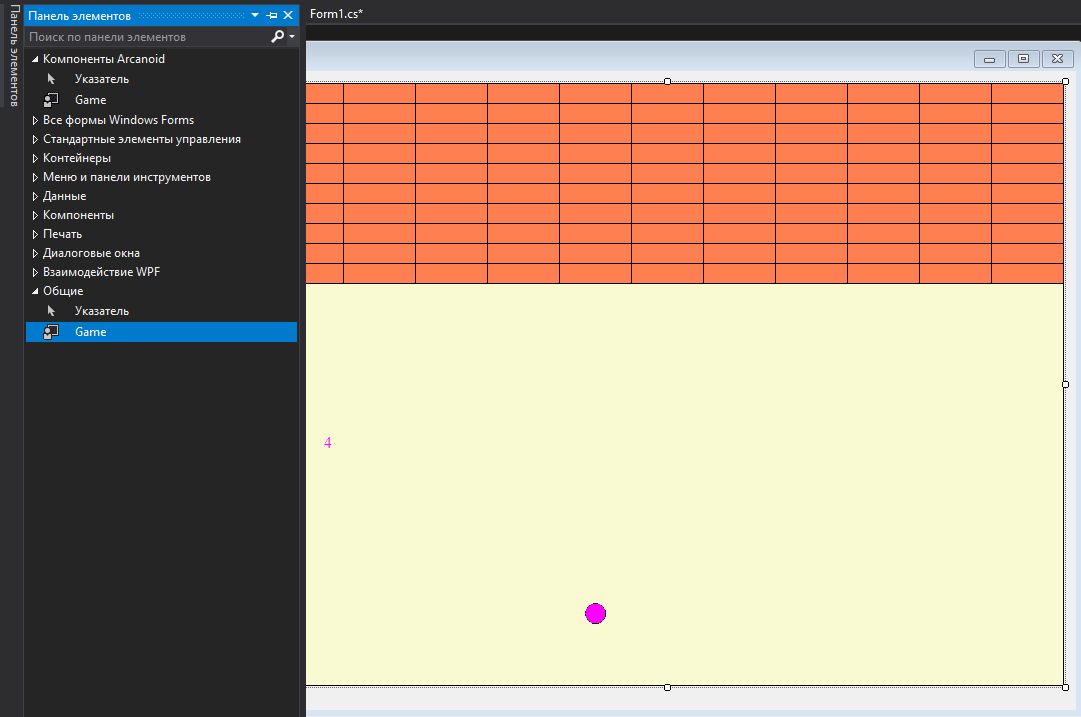


Рисунок 3.7

3.3 Инструкция по эксплуатации компонента

4 Раздел охраны труда

Заключение

Список источников

Приложение А