Федеральное государственное бюджетное образовательное учреждение высшего образования

«Костромской государственный университет»

(КГУ)

Институт автоматизированных систем и технологий

Кафедра информационных систем и технологий

Пояснительная записка

по дисциплине «Технология программирования» на тему:

Игра «Арканоид»

Выполнили: Манукянц А.Э.

Омеличева Н.А

Группа: 15-ИС-2Б

Проверил и утвердил: Киприна Л.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017г.

Кострома

2017

Оглавление

[Назначение и область применения программы 3](#_Toc484734739)

[Технические характеристики 3](#_Toc484734740)

[Пользовательские методы 3](#_Toc484734741)

[Описание и методы организации данных 4](#_Toc484734742)

[Описание и обоснование выбора технических средств 4](#_Toc484734743)

[Алгоритм выполнения программы 5](#_Toc484734744)

[Диаграмма классов 5](#_Toc484734745)

# Назначение и область применения программы

Игра «Арканоид» носит развлекательный характер. Целью данной работы является ознакомление и углубленное изучение технологий программирования.

# Технические характеристики

Для работы приложения необходима программная платформа .NET Framework 4.0

# Пользовательские методы

Полный текст кода программы представлен в приложении.

В проекте присутствует форма GameForm, которая отвечает за игровой процесс.

В форме GameForm реализованы следующие методы:

1. createLevel – метод вызывается в начале игры, располагает блоки на форме согласно матрице уровня.
2. GameForm – конструктор формы, помимо инициализации формы и вызова метода createlevel(), создаёт поток сервера или клиента в зависимости от данного булевого значения isItServ
3. startGame - метод запускается, если происходит событие Click для label «start game». Обнуляет статистику (жизни и баллы), создает основные объекты игры (блоки, нижнюю и верхнюю ракетки и мяч) и запускает таймеры движения и баллов.
4. createBlocks – создает объекты блоков и добавляет их на форму согласно схеме уровня.
5. createNewBall – метод создает новый объект мяча и устанавливает его на середину ведущей ракетки.
6. endgame – останавливает таймеры движения и баллов, определяет победителя.
7. movementTimer\_Tick – метод, в котором происходит основной процесс игры, определяется передвижения мяча и проверяется его столкновения. Так же начисляются баллы за уничтожение блоков и выводятся на форму.
8. scoreTimer\_Tick – метод, вычитающий каждую секунду 1 балл.
9. GameForm\_KeyDown – событие на нажатие пробела (запуск мяча с ведущей ракетки) и Esc (выход из игры).
10. Restart\_Click – метод запускается, если происходит событие Click на label «restart». Происходит перезапуск игры, удаляются объекты с формы, обнуляется статистика, запускается метод startGame.

В классе **Ball** реализованы следующие методы:

Ball – конструктор, устанавливающий размер объекта и загружающий картинку мяча.

move – перемещение мяча.

flipX – отражение мяча по оси Х.

flipY – отражение мяча по оси Y.

collision - метод для обработки столкновений мяча с блоками, ракетками и границами игрового поля.

fromTheRacket – обработка удара мяча от ракетки.

В классе **Racket** реализован следующие методы:

Racket – конструктор, устанавливающий размер объекта, определяющий положение ракетки на форме и загружающий картинку ракетки.

upgradeSize – метод, который запускается, если был уничтожен бонусный блок. Увеличивает размер ракетки.

В классе **Block** реализован следующий метод:

Block – конструктор, определяющий тип, размер объекта и загружающий картинку блока.

В классе **ArkaSocket** реализованы следующие методы:

1. ServerRun – метод для создание сокета типа Listener и запуск его в бесконечном цикле на порте 11000. Метод принимает данные(координаты ракетки клиента) и записывает их в clientCursor. После каждого запроса клиента, метод отправляет ответ – всю об игре, происходящей на серверном приложении.
2. ClientRun – метод для создания сокета типа Sender и его запуска. Сокет пытается соединиться с сервером. После соединения метод делает запросы, отсылая координаты курсора компьютера и принимает ответ от сервера, записывает его в переменные для дальнейшего их использования в форме GameForm. При неудачном соединении или его обрыве, вызывается метод ServerRun.

# Описание и методы организации данных

Все картинки, используемые в игре, хранятся в папке Resources.

# Описание и обоснование выбора технических средств

Технические средства были выбраны на основании требований изложенных в техническом задании.

# Алгоритм выполнения программы

1. Игрок нажимает на кнопку: «start game» или клавишу Esc для выхода из игры.
2. Если игрок нажимает кнопку «start game», то перед ним появляется поле для игры, блоки и ракетка противника. Справа отображается статистика игры. Слева режим игры и подсказки.
3. Игрок-сервер запускает мяч с помощью клавиши пробел. Управление ракеткой осуществляется с помощью мыши. Мяч уничтожает блок при столкновении, при этом начисляются баллы или присваиваются бонусы ведущей ракетке.
4. Игра продолжается до тех пор, пока у одного из игроков не закончатся жизни (побеждает противоположный игрок) или все блоки не будут уничтожены (побеждает игрок с большим количеством баллов).
5. Чтобы начать уровень заново, игрок должен нажать на кнопку «restart».
6. Для выхода из игры игрок нажимает на клавишу Esc.

# Тестирование

1. **Тестирование класса ArkaSocket**
   1. **Тестирование метода ClientRun**

Код метода:

private void createBlocks()

{

blocks = new List<Block>();

(1) for (int i = 0; i < levels[currentLevel].GetLength(0); i++)

{

(2) for (int j = 0; j < levels[currentLevel].GetLength(1); j++)

{

char blockType = levels[currentLevel][i, j];

(3) if (blockType != ' ')

{

Block block = new Block(blockType);

block.Location = new Point(j + j \* block.Size.Width + (this.Width/2 - 408), i + i \* block.Size.Height + this.Height/2 - 200);

blocks.Add(block);

Controls.Add(block);

}

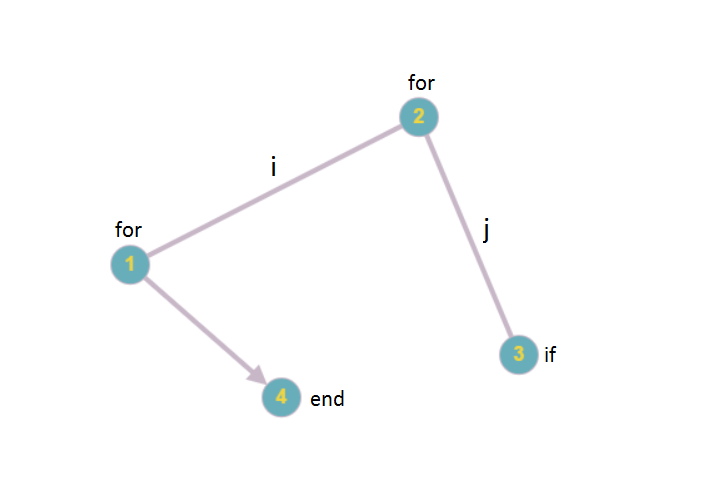
}

}

(4)

}

Граф управления:



На для работы метод использует внутренний параметр levels[currentLevel]. Формирует

List<Block> и строит фигуру из блоков, которая отображается на форме.

Тестовые варианты:

**1 Тест:**

levels[currentLevel] = new char[17, 17] {

{' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ','S',' ',' ',' ',' ',' ','S',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ','3',' ',' ',' ',' ',' ','3',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ','3',' ',' ',' ','3',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ','3',' ',' ',' ','3',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ','B','2','2','B','2','2','B',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ','2','1','1','1','1','1','2',' ',' ',' ',' ',' '},

{' ',' ',' ',' ','2','1','3','1','1','1','3','1','2',' ',' ',' ',' '},

{' ',' ',' ',' ','B','1','S','1','1','1','S','1','B',' ',' ',' ',' '},

{' ',' ',' ','2','4','1','1','1','1','1','1','1','4','2',' ',' ',' '},

{' ',' ',' ','2','4','1','1','1','1','1','1','1','4','2',' ',' ',' '},

{' ',' ',' ','2',' ','4','1','1','1','1','1','4',' ','2',' ',' ',' '},

{' ',' ',' ','2',' ','4','4','4','4','4','4','4',' ','2',' ',' ',' '},

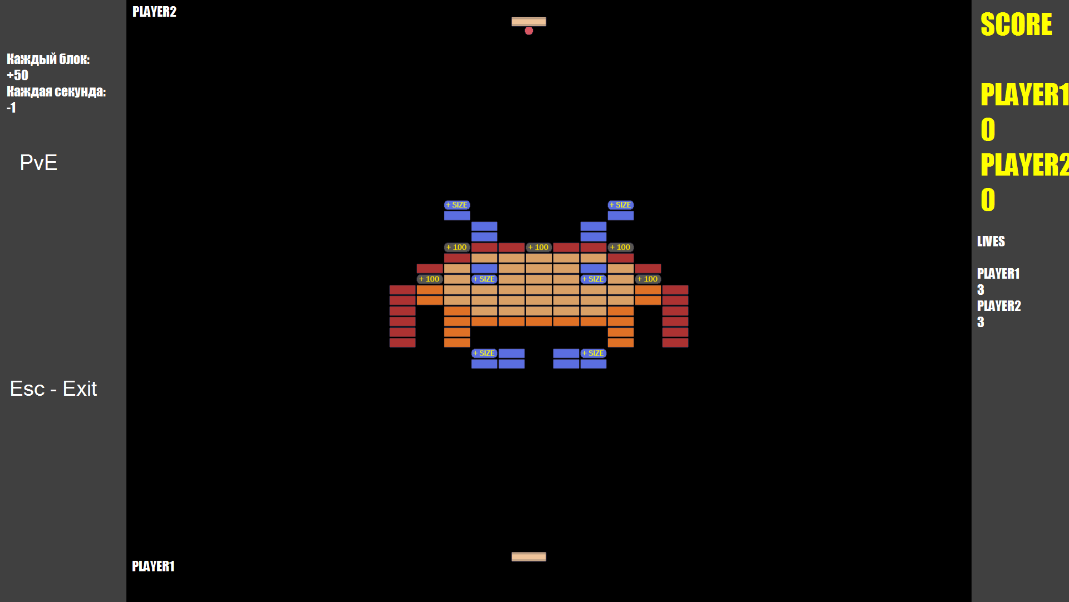
{' ',' ',' ','2',' ','4',' ',' ',' ',' ',' ','4',' ','2',' ',' ',' '},

{' ',' ',' ','2',' ','4',' ',' ',' ',' ',' ','4',' ','2',' ',' ',' '},

{' ',' ',' ',' ',' ',' ','S','3',' ','3','S',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ','3','3',' ','3','3',' ',' ',' ',' ',' ',' '},

}



**2 Тест:**

levels[currentLevel] = new char[10, 10] {

{' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},

{' ',' ',' ',' ',' ','S',' ',' ',' ',' '},

{' ',' ',' ',' ',' ','3',' ',' ',' ',' '},

{' ',' ',' ',' ',' ',' ','3',' ',' ',' '},

{' ',' ',' ',' ',' ',' ','3',' ',' ',' '},

{' ',' ',' ',' ',' ','B','2','2','B','2'},

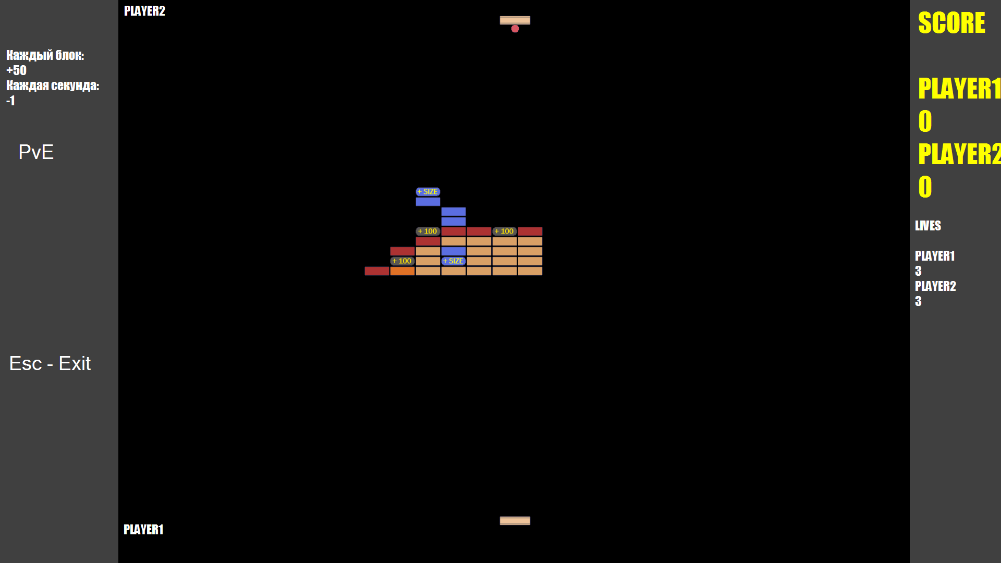
{' ',' ',' ',' ',' ','2','1','1','1','1'},

{' ',' ',' ',' ','2','1','3','1','1','1'},

{' ',' ',' ',' ','B','1','S','1','1','1'},

{' ',' ',' ','2','4','1','1','1','1','1'},

}



**3 Тест:**

* + 1. **Тестирование белого ящика**

# 

# Диаграмма классов

