Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«Ивановский государственный энергетический университет  
имени В. И. Ленина»

Кафедра ПОКС

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту по курсу**

**«Объектно-ориентированное программирование»**

**Тема:**

**«Разработка игры "ГАИшник v2.0"»**

Выполнил: студент группы 2-41хх

Ткачёв Н. В.

Проверил: к.т.н. Алыкова А. Л.

Иваново 2019

**План пояснительной записки**

[Постановка задачи 4](#_Toc10582457)

[Общее описание задачи 4](#_Toc10582458)

[1. Техническое задание 4](#_Toc10582459)

[1.1. Название программного продукта 4](#_Toc10582460)

[1.2. Основание для разработки 4](#_Toc10582461)

[1.3. Требования к системе в целом 5](#_Toc10582462)

[1.3.1. Диаграмма вариантов использования 5](#_Toc10582463)

[1.3.2. Требования к функциональным характеристикам 5](#_Toc10582464)

[1.3.3 Требование к информационному обеспечению 6](#_Toc10582465)

[1.3.4. Требования к лингвистическому обеспечению 6](#_Toc10582466)

[1.3.5. Требования к программному обеспечению 6](#_Toc10582467)

[1.3.6. Требования к техническому обеспечению 6](#_Toc10582468)

[2. Объектно-ориентированный анализ предметной области 7](#_Toc10582469)

[3. Объектно-ориентированное проектирование 7](#_Toc10582470)

[3.1. Разработка статической объектной модели 7](#_Toc10582471)

[3.1.1. Диаграмма классов 8](#_Toc10582472)

[3.2. Разработка динамической модели 8](#_Toc10582473)

[3.2.1. Диаграмма последовательности 9](#_Toc10582474)

[4. Реализация объектов 10](#_Toc10582475)

[4.1. class Form1: Form 10](#_Toc10582476)

[Раздел private 10](#_Toc10582477)

[Раздел public 10](#_Toc10582478)

[4.2. class Form2: Form 11](#_Toc10582479)

[Раздел private 11](#_Toc10582480)

[Раздел public 11](#_Toc10582481)

[4.3. class Form3: Form 11](#_Toc10582482)

[Раздел private 11](#_Toc10582483)

[4.4. class Travel 12](#_Toc10582484)

[Раздел public 12](#_Toc10582485)

[Раздел protected 12](#_Toc10582486)

[4.5. class СarP: Travel 12](#_Toc10582487)

[Раздел private 12](#_Toc10582488)

[Раздел public 12](#_Toc10582489)

[Раздел protected 12](#_Toc10582490)

[4.6. class CarC: CarP 13](#_Toc10582491)

[Раздел public 13](#_Toc10582492)

[Раздел protected 13](#_Toc10582493)

[4.7. class GAI: Travel 13](#_Toc10582494)

[Раздел private 13](#_Toc10582495)

[Раздел public 13](#_Toc10582496)

[4.8. class Svetofor 14](#_Toc10582497)

[Раздел private 14](#_Toc10582498)

[Раздел public 14](#_Toc10582499)

[4.9. class CarEventArgs: EventArgs 14](#_Toc10582500)

[Раздел public 14](#_Toc10582501)

[5. Руководство пользователя 15](#_Toc10582502)

[5.1. Рекомендация по установке 15](#_Toc10582503)

[5.2. Инструкция по использованию 15](#_Toc10582504)

[Приложение. Код программ 18](#_Toc10582505)

[Class Travel 18](#_Toc10582506)

[Class GAI: Travel 18](#_Toc10582507)

[Class CarP: Travel 21](#_Toc10582508)

[Class CarC: CarP 25](#_Toc10582509)

[Class Svetofor 25](#_Toc10582510)

[Class CarEventArgs: EventArgs 26](#_Toc10582511)

[Class Form1: Form 26](#_Toc10582512)

[Class Form2: Form 31](#_Toc10582513)

[Class Form3: Form 32](#_Toc10582514)

# Постановка задачи

1. Игра «ГАИшник v2.0»
2. Начало разработки 04.02.2019, окончание – 20.05.2019.

# Общее описание задачи

Задачей этого курсового проекта является написание компьютерной игры «ГИАшник v2.0».

По жанру "ГАИшник v2.0" — 2D симулятор, в котором игроку предстоит побыть в роли сотрудника ГИБДД. Главная цель игры – достичь звания «Полковник».

Игрок может двигаться как по дороге, так и по тротуарам. Так же он имеет возможность останавливать машины, тем самым зарабатывая монетки и звание. Для того чтобы достичь определённого звания, игроку нужно остановить конкретное число машин.

В игре 2 типа проезжающего транспорта: служебный и обычный.

Игра является бесконечной и заканчивается только в том случае, когда транспорт собьёт пользователя 3 раза.

# 1. Техническое задание

## 1.1. Название программного продукта

Игра "ГАИшник v2.0" реализована по принципу известной игры "Честный Гаишник". Название игры включает в себя версию игры и профессию персонажа, за которого игроку предстоит играть.

## 1.2. Основание для разработки

Игра разрабатывается с целью выполнения курсового проекта по дисциплине «Объектно-ориентированное программирование».

## 1.3. Требования к системе в целом

### 1.3.1. Диаграмма вариантов использования

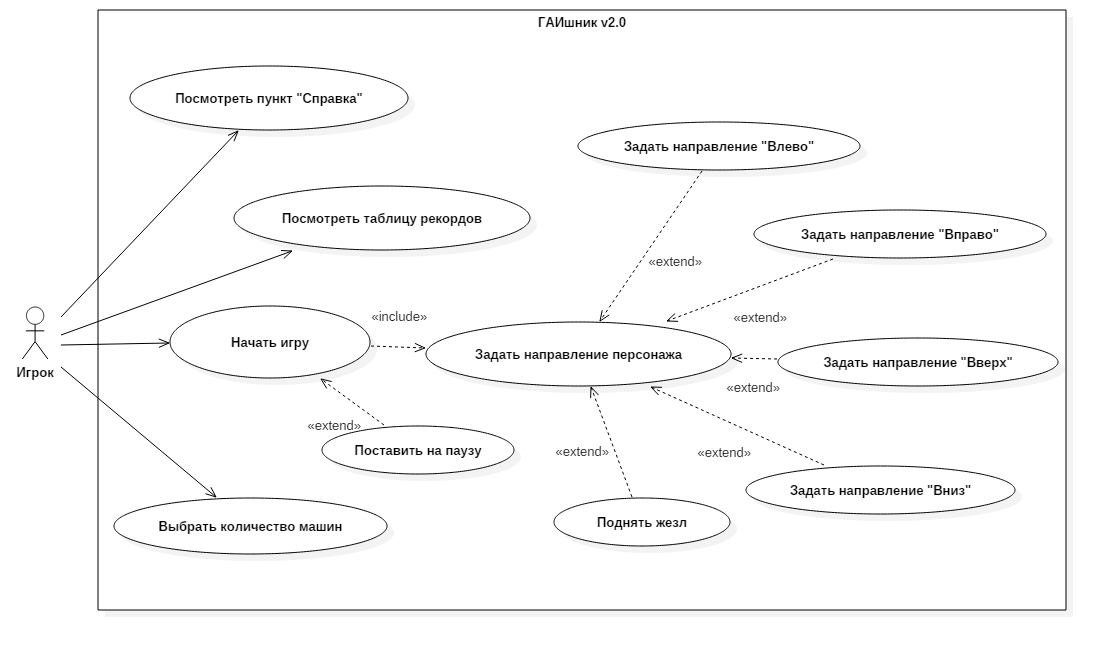


Рисунок 1 Диаграмма вариантов использования

### 1.3.2. Требования к функциональным характеристикам

1. Игра должна иметь обычное окно Windows;
2. Перед запуском игры, игрок попадает в главное меню;
3. В меню должны присутствовать 4 кнопки: «Новая игра», «Справка», «Выход» и «Scores»;
4. По нажатию кнопки «Scores» должна выводиться таблица рекордов;
5. У игрока должна быть возможность перед запуском ознакомится с правилами игры, управлением и разработчиком;
6. Должна присутствовать возможность перед началом игры задать количество машин или сгенерировать их случайным образом;
7. Игра должна использовать графический движок 2D;
8. На экране должны присутствовать: игрок, транспорт, 4-х полосная дорога, светофор, перекресток, монетки, жизни и прогресс звания игрока;
9. Изначально игрок имеет 3 жизни, ноль монеток и звание «Офицер»;
10. Должна присутствовать возможность передвигать игрового персонажа по вертикали, горизонтали и доставать жезл;
11. Перемещение игрового персонажа по игровому полю осуществляется с помощью клавиатуры;
12. Должна быть возможность остановить машину в момент вытянутого жезла;
13. Во время вытянутого жезла, игровой персонаж не может двигаться;
14. На перекрестке транспорт должен случайным образом выбирать направление;
15. Светофор должен иметь только 2 цвета: «Зеленый» и «Красный»;
16. При нажатии клавиши "Space" все движущиеся объекты останавливаются, и игрок может передохнуть. Чтобы выйти из режима паузы достаточно снова нажать клавишу "Space";
17. При столкновении с транспортом игрок теряет жизнь;
18. Если у игрока закончились жизни, то игра заканчивается;
19. При завершении игры, игроку предоставляется возможность сохранить свой рекорд;
20. После сохранения рекорда, игрок переходит в главное меню.

### 1.3.3 Требование к информационному обеспечению

При сохранении рекордов, в папке с исполнительным файлом появляются текстовый файл «rec.txt», который хранит таблицу рекордов. При нажатии на кнопку «Scores» происходит считывание информации из файла.

### 1.3.4. Требования к лингвистическому обеспечению

Интерфейс игры написан на русском языке. В данной версии многоязыковая поддержка не поддерживается.

### 1.3.5. Требования к программному обеспечению

Система работает под ОС Windows 7, Windows 8, Windows 10. Для работы программы на компьютере должен быть установлен ".NET Framework".

### 1.3.6. Требования к техническому обеспечению

Для работы программы необходимо 10 Мб свободного места на жестком диске и 100 Мб оперативной памяти (рекомендуется больше).

Для игры необходима клавиатура, для выбора между пунктами меню необходима компьютерная мышь.

# 2. Объектно-ориентированный анализ предметной области

Все элементы, которые мы видим, находятся на полотне (игровом поле). Примем их за объекты. Игрок управляет **ГАИшником**. При нажатии клавиш: W, D, S, A игровой персонаж начинает движение вверх, вправо, вниз и влево соответственно, при нажатии клавиши F игровой персонаж достает жезл, чтобы останавливать машины.

На поле стоит **четыре светофора**, которые имеют только 2 сигнала, каждый светофор регулирует движение определенного направления.

**Транспортное средство** двигается в определенном направление. На перекрестке случайным образом выбирают новое направление.

Если игрок останавливает машину, то ему начисляется 10 монеток.Если игрок попадает под машину, то он теряет 1 жизнь. Когда жизни заканчиваются, игровой персонаж умирает, игра заканчивается со счётом, равным количеству монеток.

Объекты взаимодействуют посредством механизма событий.

# 3. Объектно-ориентированное проектирование

## 3.1. Разработка статической объектной модели

За полотно примем основной класс Windows Forms, который называется Form1. Форма будет содержать в себе некоторые статические объекты WF, такие как: фоновое изображение и некоторые другие визуальные составляющие, не требующие их вынесения в отдельный класс. Для задания параметров игры и сохранения рекордов потребуются 2 дополнительных формы: Form2 и Form3.

В игре есть несколько сущностей, таких как CarP и GAI. Для этих сущностей используется базовый класс Travel. В конструкторе базового класса объектов параметрами передаются координаты, направление и изображение.

Объект CarC унаследован от класса Travel. Для реализации событийного управления создадим отдельные классы событий.

### 3.1.1. Диаграмма классов

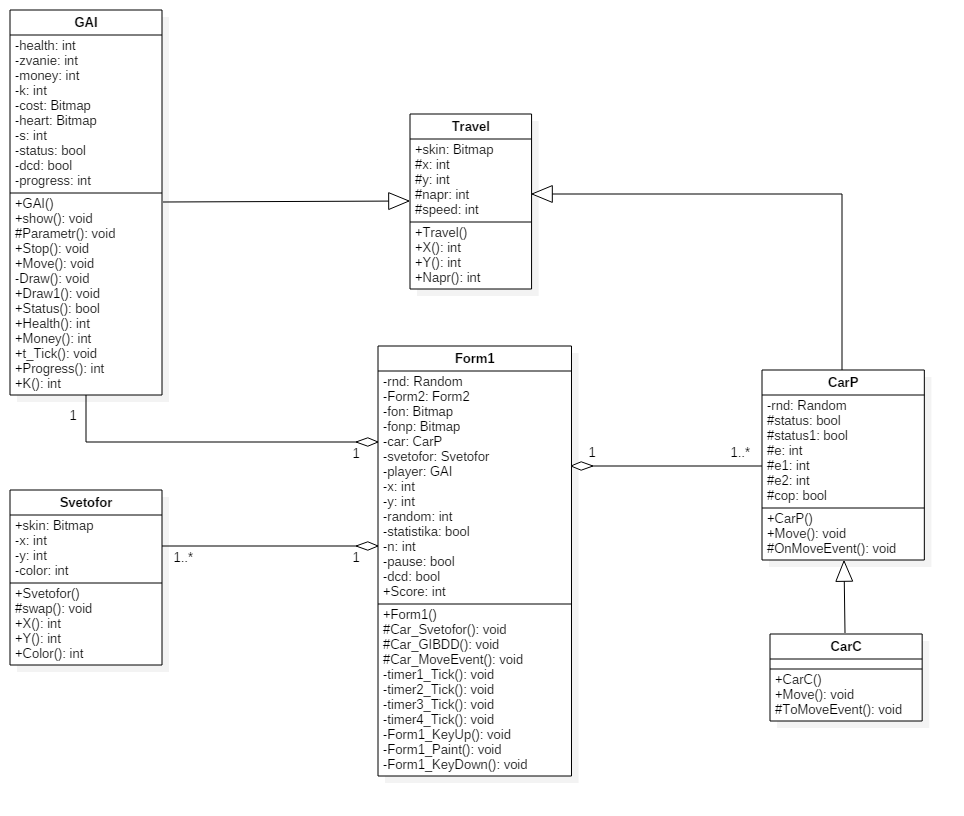


Рисунок 2 Диаграмма классов

## 3.2. Разработка динамической модели

Встроенный в WF обработчик нажатия клавиш посылает форме событие нажатия клавиши, которое обрабатывается определенным образом. При нажатии клавиш A, W, D, S «влево», «вверх», «вправо», «вниз» игровой персонаж передвигается в соответствующую сторону (движение ограниченно границами окна).

При нажатии клавиши «F» игровой персонаж достает жезл, чтобы останавливать машины.

При нажатии клавиши «Space» игра ставиться на паузу. При повторном нажатии на эту же клавишу игра снимается с паузы.

Смена сигнала светофора происходит за счёт таймера.

Передвижение машин происходит за счёт таймера и random-элемента, который инициализируется в форме. При движении машина генерирует сообщение и отправляет его на форму, где происходит проверка на столкновения и сигнал светофора.

При пересечении объекта игрового персонажа с объектом «машина», у него отнимается одна жизнь. Как только у персонажа станет ноль жизней, игра закончиться.

### 3.2.1. Диаграмма последовательности

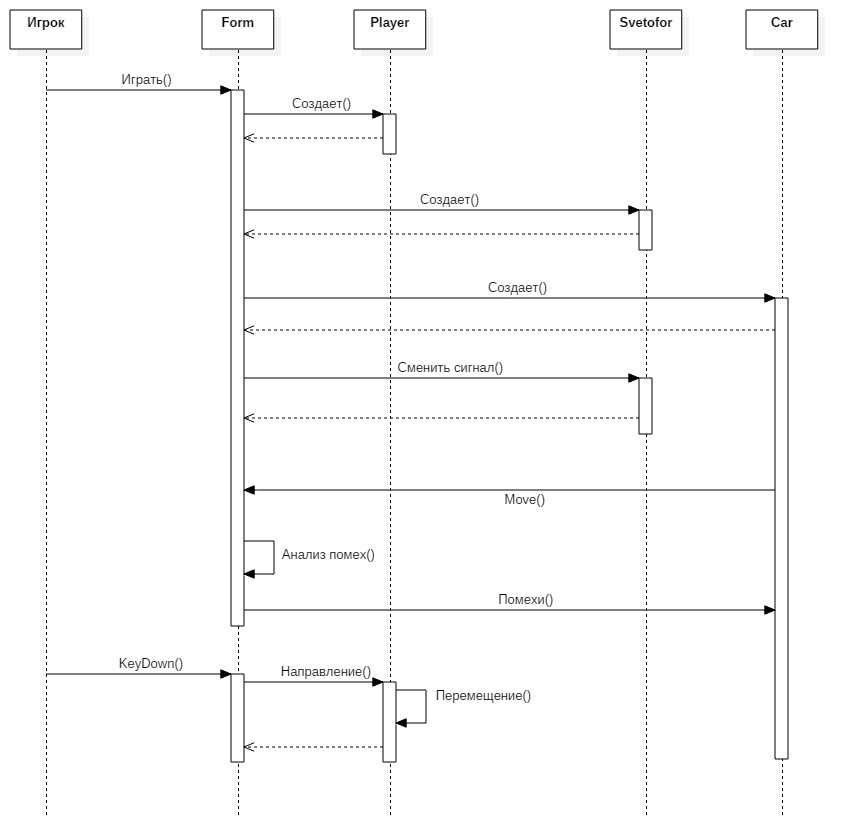


Рисунок 4 Диаграмма последовательностей

# 4. Реализация объектов

## 4.1. class Form1: Form

Класс является агрегатом, в нем создаются все объекты и идет отрисовка. Этот класс является основным в программе, именно через него осуществляется общение всех объектов.

### Раздел private

static Random rnd = new Random(); // Переменная для генерации случайных значений

private Form2 form2 = new Form2();

private Bitmap fon; //Игровая карта

private Bitmap fonp; //Затемнение карты

private CarP[] car; //Объект машина

private Svetofor[] svetofor; //Объект светофор

private GAI player; //Объект игрок

private int x, y, random; //Параметры для случайного появления машин

private Boolean statistika = true; //Начисление монеток за машину только 1 раз

private int n; //Кол-во машин

private Boolean pause = false; //Флаг паузы

private bool dcd = true; //Анимация паузы

void Car\_Svetofor(object sender, CarEventArgs ev) //Обработчик события «проверка светофора»

void Car\_GIBDD(object sender, CarEventArgs ev) //Обработчик события «проверка на ГАИшника»

void Car\_MoveEvent(object sender, CarEventArgs ev) //Обработчик события «проверка на столкновение»

private void timer1\_Tick(object sender, EventArgs e) //Таймер для движения машин

private void timer2\_Tick(object sender, EventArgs e) //Таймер для смены сигнала светофора

private void timer3\_Tick(object sender, EventArgs e) //Таймер для запрета движения, во время поднятия жезла

private void timer4\_Tick(object sender, EventArgs e) //Таймер анимации

private void Form1\_KeyUp(object sender, KeyEventArgs e) //Замена изображения игрока при отжимания клавиши

private void Form1\_Paint(object sender, PaintEventArgs e) //Обрисовка графики

private void Form1\_KeyDown(object sender, KeyEventArgs e) //Функция, для управления игровым персонажем

### Раздел public

static public int Score; //Очки игрока  
public delegate void FormEvent(int r); //Делегат, служащий для передачи данных светофору и игровому персонажу

public static event FormEvent Svetofor; //Событие на переключение светофора

public static event FormEvent GIBDD; //Событие игровому персонажу

## 4.2. class Form2: Form

Класс предназначен для задания параметров игры, а также вывода таблицы рекордов. После выставления параметров, идет их передача в Form1.

### Раздел private

FileInfo file = new FileInfo("rec.txt"); //Файл с таблицей рекордов

private void button1\_Click(object sender, EventArgs e) //Кнопка старта

private void button2\_Click(object sender, EventArgs e) //Кнопка справки

private void button3\_Click(object sender, EventArgs e) //Кнопка выхода

private void button4\_Click(object sender, EventArgs e) //Кнопка, чтобы посметь таблицу рекордов

private void button5\_Click(object sender, EventArgs e) //Кнопка выхода в меню

private void trackBar1\_Scroll(object sender, EventArgs e) //Ползунок, который задает количество машин

private void TableScores() //Функция, отображающая таблицу рекордов

### Раздел public

static public int n; //Количество машин

public int N() //Возвращает количество машин

## 4.3. class Form3: Form

Класс предназначен для записи в таблицу рекордов.

### Раздел private

private Form1 form1 = new Form1();

private int scores; //Очки игрока

FileInfo file = new FileInfo("rec.txt"); //Файл с таблицей рекордов

private void button1\_Click(object sender, EventArgs e) //Кнопка, чтобы записать рекорд и вернуться в главное меню

private void update\_file() //Функция, записывающая рекорд игрока в файл

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e) //Функция проверяющая никнейм

## 4.4. class Travel

Сегмент, основа машин и игрового персонажа.

### Раздел public

public Bitmap skin; //Изображение объекта

public int X() //Возвращает координату x

public int Y() //Возвращает координату y

public int Napr() //Возвращает направление движения

### Раздел protected

protected int x,y; //Координаты

protected int napr; //Направление движения

protected int speed; //Скорость движения

## 4.5. class СarP: Travel

Спецтранспорт, который управляется таймером, направление задается случайным образом, имеет преимущество над транспортом.

### Раздел private

static Random rnd = new Random(); //Переменная для генерации случайных значений

### Раздел public

public delegate void CarEvent(object sender, CarEventArgs ev); //Делегат, для связи с формой

public static event CarEvent Svetofor; //Событие проверки светофора

public static event CarEvent GIBDD; //Событие проверка игрового персонажа

public CarP(int x, int y, int napr, int speed) : base(x, y, napr, speed) //Конструктор, задает начальные параметры

public virtual void Move(Form1 f1) //Функция для движения машин

### Раздел protected

protected Boolean status, status1; //status-проверка выполнен ли поворот; status1-выбрана ли полоса для поворота

protected int e; //Следящие направление машины

protected int e1, e2; //Границы поворота

protected Boolean cop; //Статус машины true–спец false–обычный

protected void OnMoveEvent(object sender, CarEventArgs ev) //Функция передающая сообщение

## 4.6. class CarC: CarP

Транспорт, который управляется таймером, направление задается случайным образом.

### Раздел public

public static event CarEvent MoveEvent; //Событие на столкновение транспорта

public CarC(int x, int y, int napr, int speed) : base(x, y, napr, speed) //Конструктор, задает начальные параметры

public override void Move(Form1 f1) //Функция для движения машин

### Раздел protected

protected void ToMoveEvent(object sender, CarEventArgs ev) //Функция передающая сообщение

## 4.7. class GAI: Travel

Игровой персонаж, которым управляет игрок, задавая ему направление.

### Раздел private

private int health; //Здоровье

private int zvanie; //Должность

private int money; //Монетки

private int k = 0; //Количество остановленных машин

private Bitmap cost, heart; //Изображение машинок и жизней

private int s = 30; //Размер изображения жизней

private bool status; //true - жезл поднят false - жезл опущен

private bool dcd = true; //Анимация движения

private int progress = 10; //Заполнение шкалы звания

void Parametr(int r) //Функция изменяющая здоровье, кол-во монеток и звание

private void Draw() //Функция меняющая изображение игрового персонажа

### Раздел public

public GAI(int x, int y, int napr, int speed) : base(x, y, napr, speed) //Конструктор

public void show(Graphics g, Form1 f1) //Отображение игрового персонажа и его параметров

public void Stop() //Изменение изображения персонажа при поднятом жезле и опущенном

public void Move(int key) //Функция передвижения игрового персонажа

public void Draw1() //Изменение изображения персонажа после отпуская жезла

public bool Status() //Возвращает статус персонажа

public int Health() //Возвращает количество жизней

public int Money() //Возвращает количество монеток

public void t\_Tick() //Функция для анимации движения

public int Progress() //Возвращает значение для заполнения шкалы звания

public int K() //Возвращает количество остановленных машин

## 4.8. class Svetofor

Светофор, который имеет 2 сигнала: «Запрещающий движение» и «Разрешающий движение». По таймеру меняет сигналы.

### Раздел private

private int x, y; //Координаты светофора

private int color; //Цвет сигнала

void swap(int r) //Меняет сигнал светофора

### Раздел public

public int X() //Возвращает координату x

public int Y() //Возвращает координату y

public int Color() //Возвращает цвет сигнала

## 4.9. class CarEventArgs: EventArgs

Класс сообщения, которое несет в себе информацию, позволяющую понять, есть ли на пути движения машины препятствие или нет. Используется при движении машины.

### Раздел public

public int x, y; //Координаты машины

public int napr; //Направление движения

public bool danger; //Помеха true-есть false-нет

public int xnext, ynext; //Следующие координаты машины

public Boolean cop; //Статус машины true–спец false–обычный

# 5. Руководство пользователя

## 5.1. Рекомендация по установке

Для работы программы достаточно скопировать в отдельную папку .exe исполнительный файл игры и запустить его. При сохранении рекорда в папке с исполнительным файлом появится файл рекордов.

## 5.2. Инструкция по использованию

Для запуска игры откройте исполнительный файл «ГАИшник v2.0».



Рисунок 5 Главное меню игры

Для начала игры нажмите на кнопку «Играть».

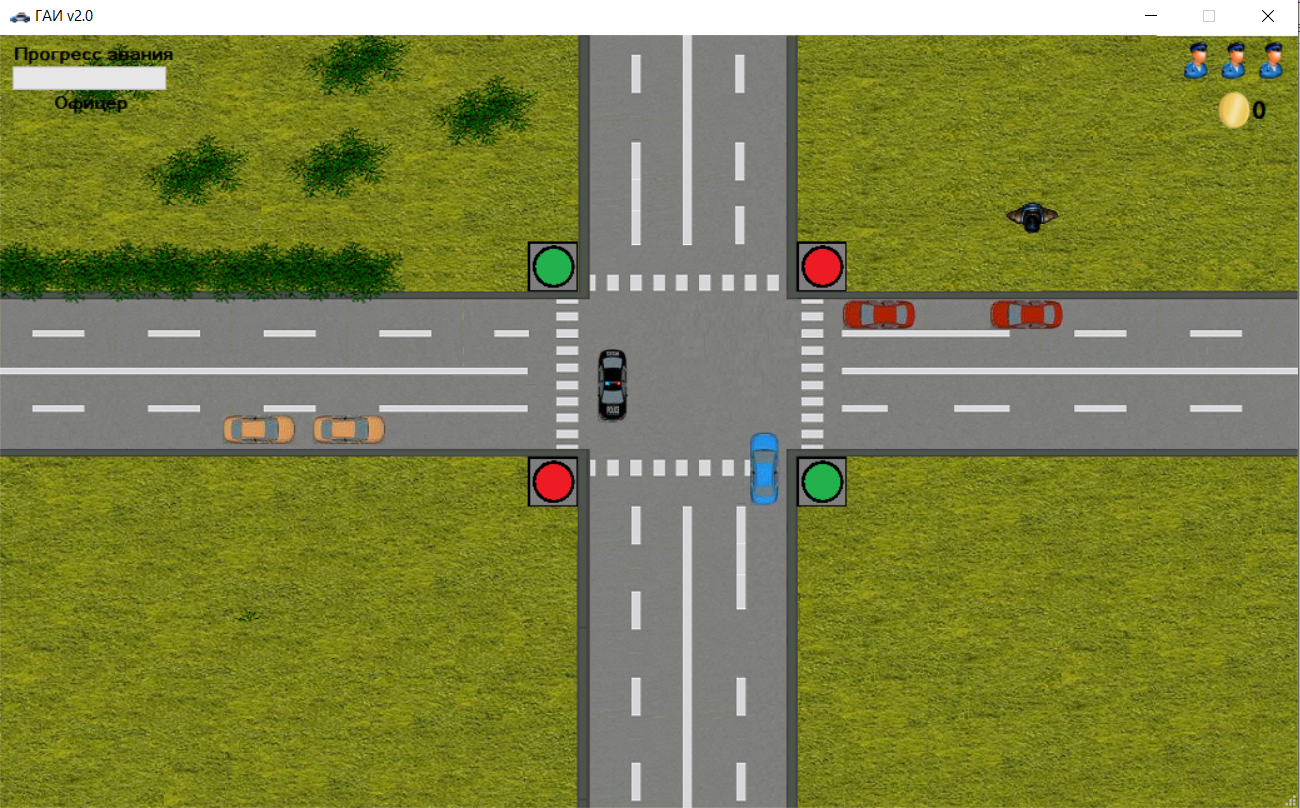


Рисунок 6 Карта игры

Чтобы посмотреть список рекордов нажмите кнопку «Scores». Чтобы вернуться обратно в главное меню, нажмите кнопку «Меню».

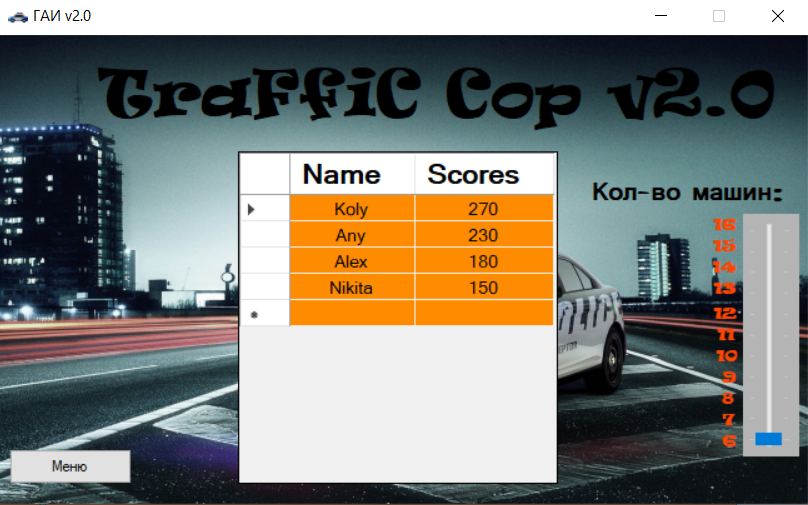


Рисунок 7 Таблица рекордов

Для того, чтобы изменить количество машин, мышкой потяните за ползунок.

Чтобы узнать правила игры и управление, нажмите на кнопку «Справка».

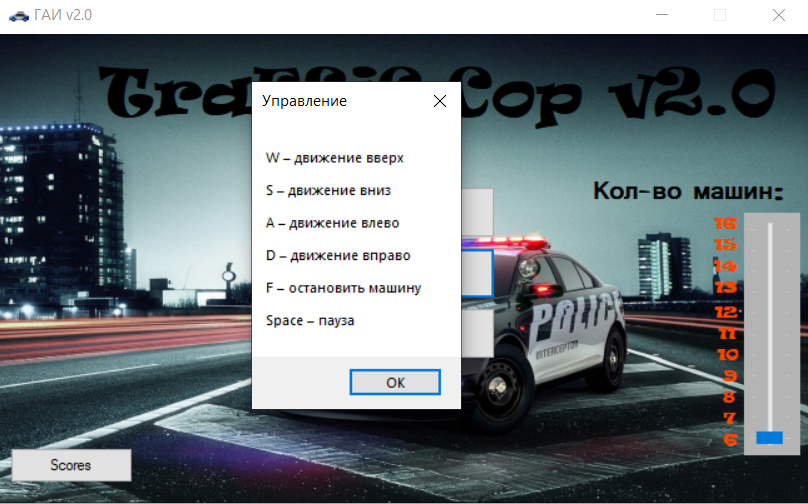


Рисунок 8 Окно справки

Для перемещения игрового персонажа используйте клавиши W, A, S, D. Чтобы остановить машину, нажмите на клавишу F. Нажмите «Space», чтобы сделать паузу. Машины не будут двигаться и светофоры менять сигналы, игровой персонаж также не сможет двигаться. Чтобы снять паузу снова нажмите клавишу «Space». Вы должны останавливать машины, но помните за что вас может сбить машина!

После того как у вас закончатся жизни, вам будет предложено ввести своё имя для сохранения вашего рекорда.



Рисунок 9 Окно по окончанию игры

Чтобы выйти в главное меню, нажмите на кнопку «Продолжить».

# Приложение. Код программ

## Class Travel

public class Travel

{

public Bitmap skin; //Изображение объекта

protected int x,y; //Координаты

protected int napr; //Направление

protected int speed; //Скорость

public Travel(int x, int y, int napr, int speed)

{

this.x = x;

this.y = y;

this.napr = napr;

this.speed = speed;

}

public int X()

{

return x;

}

public int Y()

{

return y;

}

public int Napr()

{

return napr;

}

}

## Class GAI: Travel

public class GAI : Travel

{

private int health; //Здоровье

private int zvanie; //Должность

private int money; //Монетки

private int k = 0; //Кол-во остановленных машин

private Bitmap cost, heart; //Изображение монеток и жизней

private int s = 30; //Размер изображения жизней

private bool status; //true - жезл поднят false - жезл опущен

private bool dcd = true;

private int progress = 10; //Заполнение шкалы званий

public GAI(int x, int y, int napr, int speed) : base(x, y, napr, speed)

{

Form1.GIBDD += new Form1.FormEvent(Parametr); //Изменение параметров игрока

health = 3;

zvanie = 1;

money = 0;

status = false;

cost = new Bitmap(Properties.Resources.Монетки);

heart = new Bitmap(Properties.Resources.жизнь);

skin = new Bitmap(Properties.Resources.DownPlayer\_s);

}

public void show(Graphics g, Form1 f1)

{

g.DrawImage(cost, new Rectangle(f1.ClientRectangle.Right - 65, 45, 30, 30));

switch (health)

{

case 1: g.DrawImage(heart, new Rectangle(f1.ClientRectangle.Right - 35, 5, s, s)); break;

case 2: g.DrawImage(heart, new Rectangle(f1.ClientRectangle.Right - 35, 5, s, s)); g.DrawImage(heart, new Rectangle(f1.ClientRectangle.Right - 65, 5, s, s)); break;

case 3: g.DrawImage(heart, new Rectangle(f1.ClientRectangle.Right - 35, 5, s, s)); g.DrawImage(heart, new Rectangle(f1.ClientRectangle.Right - 65, 5, s, s)); g.DrawImage(heart, new Rectangle(f1.ClientRectangle.Right - 95, 5, s, s)); break;

}

g.DrawImage(skin, new Rectangle(x, y, 50, 50));

}

void Parametr(int r)

{

switch (r)

{

case 1: money += 10; k++; break;

case 2: x = 800; y = 120; health--; break;

}

switch (k)

{

case 0: progress = 10; break; //Офицер

case 10: progress = 10; break; //Мл.Сержант

case 20: progress = 10; break; //Сержант

case 30: progress = 10; break; //Ст.Сержант

case 40: progress = 10; break; //Прапорщик

case 50: progress = 10; break; //Ст.Прапорщик

case 60: progress = 10; break; //Мл.Лейтенант

case 75: progress = 6; break; //Лейтенант

case 90: progress = 6; break; //Ст.Лейтенант

case 105: progress = 6; break; //Капитан

case 120: progress = 6; break; //Майор

case 135: progress = 6; break; //Подполковник

case 155: progress = 5; break; //Полковник

}

}

public void Stop()

{

if (status)

{

status = false;

switch (napr)

{

case 1: skin = new Bitmap(Properties.Resources.UpPlayer\_s); break;

case 2: skin = new Bitmap(Properties.Resources.DownPlayer\_s); break;

case 3: skin = new Bitmap(Properties.Resources.LeftPlayer\_s); break;

case 4: skin = new Bitmap(Properties.Resources.RigthPlayer\_s); break;

}

}

else

{

status = true;

switch (napr)

{

case 1: skin = new Bitmap(Properties.Resources.UpPlayer\_stick); break;

case 2: skin = new Bitmap(Properties.Resources.DownPlayer\_stick); break;

case 3: skin = new Bitmap(Properties.Resources.LeftPlayer\_stick); break;

case 4: skin = new Bitmap(Properties.Resources.RigthPlaye\_stick); break;

}

}

}

public void Move(int key)

{

if (!status)

{

switch (key)

{

case 1: y -= speed; napr = 1; Draw(); break; //Вверх

case 2: y += speed; napr = 2; Draw(); break; //Вниз

case 3: x -= speed; napr = 3; Draw(); break; //Влево

case 4: x += speed; napr = 4; Draw(); break; //Вправо

}

}

}

private void Draw()

{

if (dcd)

{

switch (napr)

{

case 1: skin = new Bitmap(Properties.Resources.UpPlayer\_go1); break;

case 2: skin = new Bitmap(Properties.Resources.DownPlayer\_go1); break;

case 3: skin = new Bitmap(Properties.Resources.LeftPlayer\_go1); break;

case 4: skin = new Bitmap(Properties.Resources.RigthPlaye\_go1); break;

}

}

else

{

switch (napr)

{

case 1: skin = new Bitmap(Properties.Resources.UpPlayer\_go2); break;

case 2: skin = new Bitmap(Properties.Resources.DownPlayer\_go2); break;

case 3: skin = new Bitmap(Properties.Resources.LeftPlayer\_go2); break;

case 4: skin = new Bitmap(Properties.Resources.RigthPlaye\_go2); break;

}

}

}

public void Draw1()

{

switch (napr)

{

case 1: skin = new Bitmap(Properties.Resources.UpPlayer\_s); break;

case 2: skin = new Bitmap(Properties.Resources.DownPlayer\_s); break;

case 3: skin = new Bitmap(Properties.Resources.LeftPlayer\_s); break;

case 4: skin = new Bitmap(Properties.Resources.RigthPlayer\_s); break;

}

}

public bool Status()

{

return status;

}

public int Health()

{

return health;

}

public int Money()

{

return money;

}

public void t\_Tick()

{

dcd = !dcd;

}

public int Progress()

{

return progress;

}

public int K()

{

return k;

}

}

## Class CarP: Travel

public class CarP : Travel

{

public delegate void CarEvent(object sender, CarEventArgs ev); // делегат

public static event CarEvent Svetofor; // событие проверка светофора

public static event CarEvent GIBDD; // событие проверка на гаишника

static Random rnd = new Random();

protected Boolean status, status1;

protected int e; //Следущее направление машины

protected int e1, e2; //Границы поворота

protected Boolean cop; //Статус машины true–спец false–обычный

public CarP(int x, int y, int napr, int speed) : base(x, y, napr, speed)

{

switch (napr)

{

case 0: skin = new Bitmap(Properties.Resources.Police); break;

case 1: skin = new Bitmap(Properties.Resources.Police2); break;

case 2: skin = new Bitmap(Properties.Resources.Police1); break;

case 3: skin = new Bitmap(Properties.Resources.Police3); break;

}

cop = true;

}

public virtual void Move(Form1 f1)

{

CarEventArgs ev = new CarEventArgs(); //создаем сообщение

ev.napr = napr;

ev.x = x;

ev.y = y;

ev.cop = cop;

switch (napr)

{

case 0: { ev.xnext = x - speed; ev.ynext = y; } break;

case 1: { ev.xnext = x + speed + 70; ev.ynext = y + 23; } break;

case 2: { ev.xnext = x + 23; ev.ynext = y - speed; } break;

case 3: { ev.xnext = x; ev.ynext = y + speed + 70; } break;

}

OnMoveEvent(this, ev); // вызываем обработчик события

if (!ev.danger)

{

switch (napr)

{

case 0: //Влево

if (x <= f1.ClientRectangle.Right + 100 && x >= f1.ClientRectangle.Left - 200)

{

x -= speed;

if (status1 == false && x >= f1.ClientRectangle.Right - 150 && x <= f1.ClientRectangle.Right - 100)

{

e = rnd.Next(2);

status1 = true;

switch (e)

{

case 0: e1 = 0; e2 = 0; break;

case 1: y = 212; e1 = 560; e2 = 600; break;

case 2: y = 242; e1 = 478; e2 = 500; break;

}

}

else

{

if (x >= e1 && x <= e2 && status == false)

switch (e)

{

case 0:

napr = 0; status = true; break; //Прямо

case 1:

napr = 2; x = 600; status = true; skin.RotateFlip(RotateFlipType.Rotate90FlipX); break; //Направо

case 2:

napr = 3; x = 478; status = true; skin.RotateFlip(RotateFlipType.Rotate270FlipX); break; //Налево

}

}

}

else

{

e1 = -300;

e2 = -300;

status = false;

status1 = false;

y = 304;

x = -90;

skin.RotateFlip(RotateFlipType.Rotate180FlipY);

napr = 1;

}

break;

case 1: //Вправо

if (x <= f1.ClientRectangle.Right + 100 && x >= f1.ClientRectangle.Left - 200)

{

x += speed;

if (status1 == false && x >= f1.ClientRectangle.Left + 200 && x <= f1.ClientRectangle.Left + 250)

{

e = rnd.Next(2);

status1 = true;

switch (e)

{

case 0: e1 = 0; e2 = 0; break;

case 1: y = 304; e1 = 478; e2 = 520; break;

case 2: y = 272; e1 = 580; e2 = 600; break;

}

}

else

{

if (x >= e1 && x <= e2 && status == false)

switch (e)

{

case 0:

napr = 1; status = true; break; //Прямо

case 1:

napr = 3; x = 478; status = true; skin.RotateFlip(RotateFlipType.Rotate90FlipX); break; //Направо

case 2:

napr = 2; x = 600; status = true; skin.RotateFlip(RotateFlipType.Rotate270FlipX); break; //Налево

}

}

}

else

{

e1 = -300;

e2 = -300;

status1 = false;

status = false;

y = 212;

x = f1.ClientRectangle.Right + 90;

skin.RotateFlip(RotateFlipType.Rotate180FlipY);

napr = 0;

}

break;

case 2: //Вверх

if (y <= f1.ClientRectangle.Bottom + 100 && y >= f1.ClientRectangle.Top - 200)

{

y -= speed;

if (status1 == false && y >= f1.ClientRectangle.Bottom - 150 && y <= f1.ClientRectangle.Bottom - 100)

{

e = rnd.Next(2);

status1 = true;

switch (e)

{

case 0: e1 = 0; e2 = 0; break;

case 1: x = 600; e1 = 272; e2 = 304; break;

case 2: x = 560; e1 = 212; e2 = 222; break;

}

}

else

{

if (y >= e1 && y <= e2 && status == false)

switch (e)

{

case 0:

napr = 2; status = true; break; //Прямо

case 1:

napr = 1; y = 304; status = true; skin.RotateFlip(RotateFlipType.Rotate90FlipY); break; //Направо

case 2:

napr = 0; y = 212; status = true; skin.RotateFlip(RotateFlipType.Rotate270FlipY); break; //Налево

}

}

}

else

{

e1 = -300;

e2 = -300;

status1 = false;

status = false;

y = -90;

x = 478;

skin.RotateFlip(RotateFlipType.Rotate180FlipX);

napr = 3;

}

break;

case 3: //Вниз

if (y <= f1.ClientRectangle.Bottom + 100 && y >= f1.ClientRectangle.Top - 200)

{

y += speed;

if (status1 == false && y >= f1.ClientRectangle.Top + 10 && y <= f1.ClientRectangle.Top + 150)

{

e = rnd.Next(2);

status1 = true;

switch (e)

{

case 0: e1 = 0; e2 = 0; break;

case 1: x = 478; e1 = 212; e2 = 242; break;

case 2: x = 520; e1 = 294; e2 = 304; break;

}

}

else

{

if (y >= e1 && y <= e2 && status == false)

switch (e)

{

case 0:

napr = 3; status = true; break; //Прямо

case 1:

napr = 0; y = 212; status = true; skin.RotateFlip(RotateFlipType.Rotate90FlipY); break; //Направо

case 2:

napr = 1; y = 304; status = true; skin.RotateFlip(RotateFlipType.Rotate270FlipY); break; //Налево

}

}

}

else

{

e1 = -300;

e2 = -300;

status1 = false;

status = false;

y = f1.ClientRectangle.Bottom + 90;

x = 600;

skin.RotateFlip(RotateFlipType.Rotate180FlipX);

napr = 2;

}

break;

}

}

}

protected void OnMoveEvent(object sender, CarEventArgs ev)

{

if (Svetofor != null)

Svetofor(sender, ev);

if (GIBDD != null)

GIBDD(sender, ev);

}

## Class CarC: CarP

public class CarC : CarP

{

public static event CarEvent MoveEvent; // событие

public CarC(int x, int y, int napr, int speed) : base(x, y, napr, speed)

{

switch (napr)

{

case 0: skin = new Bitmap(Properties.Resources.Красная1\_влево\_); break;

case 1: skin = new Bitmap(Properties.Resources.Оранжевая1\_вправо\_); break;

case 2: skin = new Bitmap(Properties.Resources.Синия1\_вверх\_); break;

case 3: skin = new Bitmap(Properties.Resources.Police3); break;

}

cop = false;

}

public override void Move(Form1 f1)

{

CarEventArgs ev = new CarEventArgs(); //создаем сообщение

ev.napr = napr;

ev.x = x;

ev.y = y;

switch (napr)

{

case 0: { ev.xnext = x - speed; ev.ynext = y; } break;

case 1: { ev.xnext = x + speed + 70; ev.ynext = y; } break;

case 2: { ev.xnext = x; ev.ynext = y - speed; } break;

case 3: { ev.xnext = x; ev.ynext = y + speed + 70; } break;

}

ToMoveEvent(this, ev); // вызываем обработчик события

if (!ev.danger)

base.Move(f1);

}

protected void ToMoveEvent(object sender, CarEventArgs ev)

{

if (MoveEvent != null)

MoveEvent(sender, ev);

}

}

## Class Svetofor

public class Svetofor

{

public Bitmap skin;

private int x, y; //Координаты

private int color; //Цвет светофора( 0-красный, 1-зеленый )

public Svetofor(int x, int y, int color)

{

Form1.Svetofor += new Form1.FormEvent(swap); //Смена сигнала

this.x = x;

this.y = y;

this.color = color;

switch (color)

{

case 0: skin = new Bitmap(Properties.Resources.Red); break;

case 1: skin = new Bitmap(Properties.Resources.Green); break;

}

}

void swap(int r)

{

color++;

color %= 2;

switch (color)

{

case 0: skin = new Bitmap(Properties.Resources.Red); break;

case 1: skin = new Bitmap(Properties.Resources.Green); break;

}

}

public int X()

{

return x;

}

public int Y()

{

return y;

}

public int Color()

{

return color;

}

}

## Class CarEventArgs: EventArgs

public class CarEventArgs : EventArgs //класс сообщения

{

public int x, y; //Координаты машины

public int napr; //Направление движения

public bool danger; //Есть помеха

public int xnext, ynext; //Следующие координаты машины

public Boolean cop;

}

## Class Form1: Form

public partial class Form1 : Form

{

public delegate void FormEvent(int r); // делегат

public static event FormEvent Svetofor; // событие на переключение светофора

public static event FormEvent GIBDD; // событие гаишнику

static Random rnd = new Random();

private Form2 form2 = new Form2();

private Bitmap fon; //Игровая карта

private Bitmap fonp; //Затемнение карты

private CarP[] car; //Объект машина

private Svetofor[] svetofor; //Объект светофор

private GAI player; //Объект игрок

private int x, y, random; //Параметры для рандомного появления машин

private Boolean statistika = true;

private int n; //Кол-во машин

private Boolean pause = false; //Поставить игру на паузу

private bool dcd = true; //Анимация паузы

static public int Score; //Очки

public Form1()

{

InitializeComponent();

SetStyle(ControlStyles.OptimizedDoubleBuffer | ControlStyles.AllPaintingInWmPaint | ControlStyles.UserPaint, true);

UpdateStyles();

CarP.Svetofor += new CarP.CarEvent(Car\_Svetofor); //Проверка светофоров

CarP.GIBDD += new CarP.CarEvent(Car\_GIBDD); //Проверка на гаишника

CarC.MoveEvent += new CarC.CarEvent(Car\_MoveEvent); //Проверка машин

fon = Properties.Resources.Фон; //Игровая карта

fonp = Properties.Resources.Pause2; //Пауза

n = form2.N(); //Кол-во машин

car = new CarP[n];

for (int i = 0; i < n; i++)

{

switch (i % 4)

{

case 0:

switch (rnd.Next(1))

{

case 0: y = 212; break;

case 1: y = 242; break;

}

car[i] = new CarC(ClientRectangle.Right, y, 0, rnd.Next(3, 5)); break; //Машина едет влево

case 1:

switch (rnd.Next(1))

{

case 0: y = 304; break;

case 1: y = 272; break;

}

car[i] = new CarC(ClientRectangle.Left - 150, y, 1, rnd.Next(3, 5)); break; //Машина едет вправо

case 2:

switch (rnd.Next(1))

{

case 0: x = 600; break;

case 1: x = 560; break;

}

car[i] = new CarC(x, ClientRectangle.Bottom, 2, rnd.Next(3, 5)); break; //Машина едет вверх

case 3:

switch (rnd.Next(1))

{

case 0: x = 478; break;

case 1: x = 520; break;

}

car[i] = new CarC(x, ClientRectangle.Top - 150, 3, rnd.Next(3, 5)); break; //Машина едет вниз

case 4:

switch (random = rnd.Next(4))

{

case 0: x = ClientRectangle.Right; y = 212; break;

case 1: x = ClientRectangle.Left - 150; y = 304; break;

case 2: x = 600; y = ClientRectangle.Bottom; break;

case 3: x = 478; y = ClientRectangle.Top - 150; break;

}

car[i] = new CarP(x, y, random, rnd.Next(4, 7)); break; //Спец.транспорт

}

}

svetofor = new Svetofor[4];

svetofor[0] = new Svetofor(637, 165, 0); //Светофор для т/с приблежающегося слева

svetofor[1] = new Svetofor(422, 337, 0); //Светофор для т/с приблежающегося справа

svetofor[2] = new Svetofor(637, 337, 1); //Светофор для т/с приблежающегося сверху

svetofor[3] = new Svetofor(422, 165, 1); //Светофор для т/с приблежающегося снизу

player = new GAI(800, 120, 3, 3);

}

void Car\_Svetofor(object sender, CarEventArgs ev) // обработчик события «cветофор»

{

int d0 = 0;

int r = 40;

switch (ev.napr) // проверяем, близко ли светофор

{

case 0: d0 = ev.x - svetofor[0].X(); break;

case 1: d0 = svetofor[1].X() - ev.x; break;

case 2: d0 = ev.y - svetofor[2].Y(); break;

case 3: d0 = svetofor[3].Y() - ev.y; break;

}

switch (ev.napr)

{

case 0:

if (d0 < r && d0 > 0 && svetofor[0].Color() == 0)

ev.danger = true;

else ev.danger = false;

break;

case 1:

if (d0 < r && d0 > 0 && svetofor[1].Color() == 0)

ev.danger = true;

else ev.danger = false;

break;

case 2:

if (d0 < r && d0 > 0 && svetofor[2].Color() == 0)

ev.danger = true;

else ev.danger = false;

break;

case 3:

if (d0 < r && d0 > 0 && svetofor[3].Color() == 0)

ev.danger = true;

else ev.danger = false;

break;

}

}

void Car\_GIBDD(object sender, CarEventArgs ev)

{

if (player.Status() && !ev.cop && ev.xnext >= player.X() && ev.xnext <= player.X() + 60 && ev.ynext >= player.Y() && ev.ynext <= player.Y() + 60)

{

ev.danger = true;

if (statistika)

{

GIBDD(1);

progressBar1.Value += player.Progress();

if (progressBar1.Value == 100 || (progressBar1.Value == 90 && player.Progress() == 6))

{

progressBar1.Value = 0;

}

statistika = false;

}

}

else

{

if (player.Status() && ev.cop && ev.xnext >= player.X() && ev.xnext <= player.X() + 60 && ev.ynext >= player.Y() && ev.ynext <= player.Y() + 60)

{

GIBDD(2);

}

if (!player.Status() && ev.x >= player.X() && ev.x <= player.X() + 40 && ev.y >= player.Y() && ev.y <= player.Y() + 40)

{

GIBDD(2);

}

}

}

void Car\_MoveEvent(object sender, CarEventArgs ev)// обработчик события «запрос автомобиля»

{

for (int i = 0; i < n; i++)

{

if (ev.xnext >= car[i].X() && ev.xnext <= car[i].X() + 70 && ev.ynext >= car[i].Y() && ev.ynext <= car[i].Y() + 70)

{

ev.danger = true;

break;

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (!pause)

{

switch (player.K())

{

case 0: label3.Text = "Офицер"; break;

case 10: label3.Text = "Мл.Сержант"; break;

case 20: label3.Text = "Сержант"; break;

case 30: label3.Text = "Ст.Сержант"; break;

case 40: label3.Text = "Прапорщик"; break;

case 50: label3.Text = "Ст.Прапорщик"; break;

case 60: label3.Text = "Мл.Лейтенант"; break;

case 75: label3.Text = "Лейтенант"; break;

case 90: label3.Text = "Ст.Лейтенант"; break;

case 105: label3.Text = "Капитан"; break;

case 120: label3.Text = "Майор"; break;

case 135: label3.Text = "Подполковник"; break;

case 155: label3.Text = "Полковник"; break;

}

label1.Text = Convert.ToString(player.Money());

if (player.Health() == 0)

{

Score = player.Money();

timer1.Enabled = false;

Hide();

Form3 Over = new Form3();

Over.ShowDialog();

Close();

}

for (int i = 0; i < n; i++)

car[i].Move(this);

}

Refresh();

} //Таймер для движения машин

private void timer2\_Tick(object sender, EventArgs e)

{

if (!pause)

{

Svetofor(0);

}

} //Таймер для смены сигнала светофора

private void timer3\_Tick(object sender, EventArgs e)

{

timer3.Enabled = false;

player.Stop();

statistika = true;

}

private void timer4\_Tick(object sender, EventArgs e)

{

player.t\_Tick();

if (pause)

{

dcd = !dcd;

if (dcd)

fonp = Properties.Resources.Pause1;

else

fonp = Properties.Resources.Pause2;

}

} //Таймер анимации

private void Form1\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyData != Keys.F)

player.Draw1();

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

g.DrawImage(fon, new Rectangle(0, 0, 1280, 720));

player.show(g, this);

for (int i = 0; i < n; i++)

g.DrawImage(car[i].skin, car[i].X(), car[i].Y());

for (int i = 0; i < 4; i++)

g.DrawImage(svetofor[i].skin, new Rectangle(svetofor[i].X(), svetofor[i].Y(), 40, 40));

if(pause)

g.DrawImage(fonp, new Rectangle(-125, -10, 1280, 720));

} //Отрисовка графики

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (!pause)

{

switch (e.KeyData)

{

case Keys.W: player.Move(1); break;

case Keys.S: player.Move(2); break;

case Keys.A: player.Move(3); break;

case Keys.D: player.Move(4); break;

case Keys.F:

if (!player.Status())

{

timer3.Enabled = true;

player.Stop();

}

break;

}

}

if (e.KeyValue == 32)

{

if (!pause)

pause = true;

else pause = false;

}

}

}

## Class Form2: Form

public partial class Form2 : Form

{

static public int n; //Кол-во машин

FileInfo file = new FileInfo("rec.txt");

public Form2()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

n = trackBar1.Value;

Hide();

Form1 Играть = new Form1();

Играть.ShowDialog();

Close();

} //Кнопка играть

private void button2\_Click(object sender, EventArgs e) //Кнопка справка

{

DialogResult res = MessageBox.Show("Игра была разработана студентом ИГЭУ группы 2 - 41хх Ткачёвым Н.В. в качестве курсового проекта по курсу Объектно-ориентированное программирование", "Об авторе", MessageBoxButtons.OK);

if (res == DialogResult.OK)

{

MessageBox.Show("По жанру TraFfiC Cop v1.0 – симулятор ГАИшника, в котором игрок управляет полицейским, который ходит по дороге и останавливает нарушителей. Цель игры состоит в получение звания 'Полковника'. Для достижения цели нужно останавливить порядка 100 нарушителей. Чем выше звание, тем более легче становится тормозить машины.", "Правила игры");

if (res == DialogResult.OK)

{

MessageBox.Show("W – движение вверх\n\nS – движение вниз\n\nA – движение влево\n\nD – движение вправо\n\nF – остановить машину\n\nSpace – пауза", "Управление");

}

}

}

private void button3\_Click(object sender, EventArgs e)

{

Close();

} //Кнопка выход

private void trackBar1\_Scroll(object sender, EventArgs e)

{

n = trackBar1.Value;

}

public int N()

{

return n;

}

private void button4\_Click(object sender, EventArgs e) //Кнопка score

{

button4.Visible = false;

button5.Visible = true;

TableScores();

}

private void TableScores()

{

dataGridView1.Visible = true;

if (file.Exists)

{

StreamReader sr = new StreamReader(file.Name, System.Text.Encoding.Default);

DataSet ds = new DataSet();

ds.Tables.Add("Score");

ds.Tables[0].Columns.Add("Name");

ds.Tables[0].Columns.Add("Scores");

string row = sr.ReadLine();

while (row != null)

{

string[] rvalue = System.Text.RegularExpressions.Regex.Split(row, " ");

ds.Tables[0].Rows.Add(rvalue);

row = sr.ReadLine();

}

dataGridView1.DataSource = ds.Tables[0];

}

else

{

file.Create();

}

}

private void button5\_Click(object sender, EventArgs e) //Кнпока меню

{

button4.Visible = true;

button5.Visible = false;

dataGridView1.Visible = false;

}

}

## Class Form3: Form

public partial class Form3 : Form

{

private Form1 form1 = new Form1();

private int scores;

FileInfo file = new FileInfo("rec.txt");

public Form3()

{

InitializeComponent();

scores = Form1.Score;

label3.Text = Convert.ToString(scores);

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "")

MessageBox.Show("Введите имя", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.None);

else

{

update\_file();

Hide();

Form2 Menu = new Form2();

Menu.ShowDialog();

Close();

}

}

private void update\_file()

{

if (file.Exists)

{

StreamWriter sw = new StreamWriter(file.FullName, true, System.Text.Encoding.Default);

sw.WriteLine(textBox1.Text + " " + scores.ToString());

sw.Close();

}

else

{

file.Create();

StreamWriter sw = new StreamWriter(file.FullName, false, System.Text.Encoding.Default);

sw.WriteLine(textBox1.Text + " " + scores.ToString());

sw.Close();

}

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (int)Keys.Space)

e.KeyChar = '\0';

}

}