ФГБОУ ВО «Ивановский государственный энергетический университет имени В. И. Ленина»
Факультет информатики и вычислительной техники
Кафедра программного обеспечения компьютерных систем

Курсовая работа

по дисциплине «Объектно-ориентированное программирование»

на тему: Игра «Легушка»

Выполнил:
студент группы 2-41 Иванов Д.С. _____ _____
(подпись) (дата)

Руководитель:
доцент каф. ПОКС Алыкова А.Л. _____ _____
(подпись) (дата)

Иваново 2022

Постановка задачи

Игра “Легушка” предназначена для развлечения. Программа должна иметь управление с помощью клавиатуры/мыши и иметь сохранение прогресса.

Общее описание задачи

“Легушка” позаимствовала основной геймплей из игры “Crossy road” с некоторыми нововведениями.

Сценарий игры: главный герой (лягушка) появляется посреди леса. Главная цель – набрать максимальное количество очков путём прохождения вперёд по карте, преодолевая различные препятствия:

1. Машины и грузовики, едущие по дорогам с различной скоростью, которые могут задавить лягушку.
2. Поезда, мчащиеся по железным дорогам и тоже давящие лягушку.
3. Вода. Игра не основана на реальных событиях, поэтому лягушка боится воды. Преодолеть её можно, прыгая по кувшинкам или движущимся брёвнам.

Техническое задание

Полное название программного продукта “Легушка”.

Основание для разработки - «Учебный план по программе бакалавриата. Направление 09.03.04 - Программная инженерия (Разработка программно-информационных систем)».

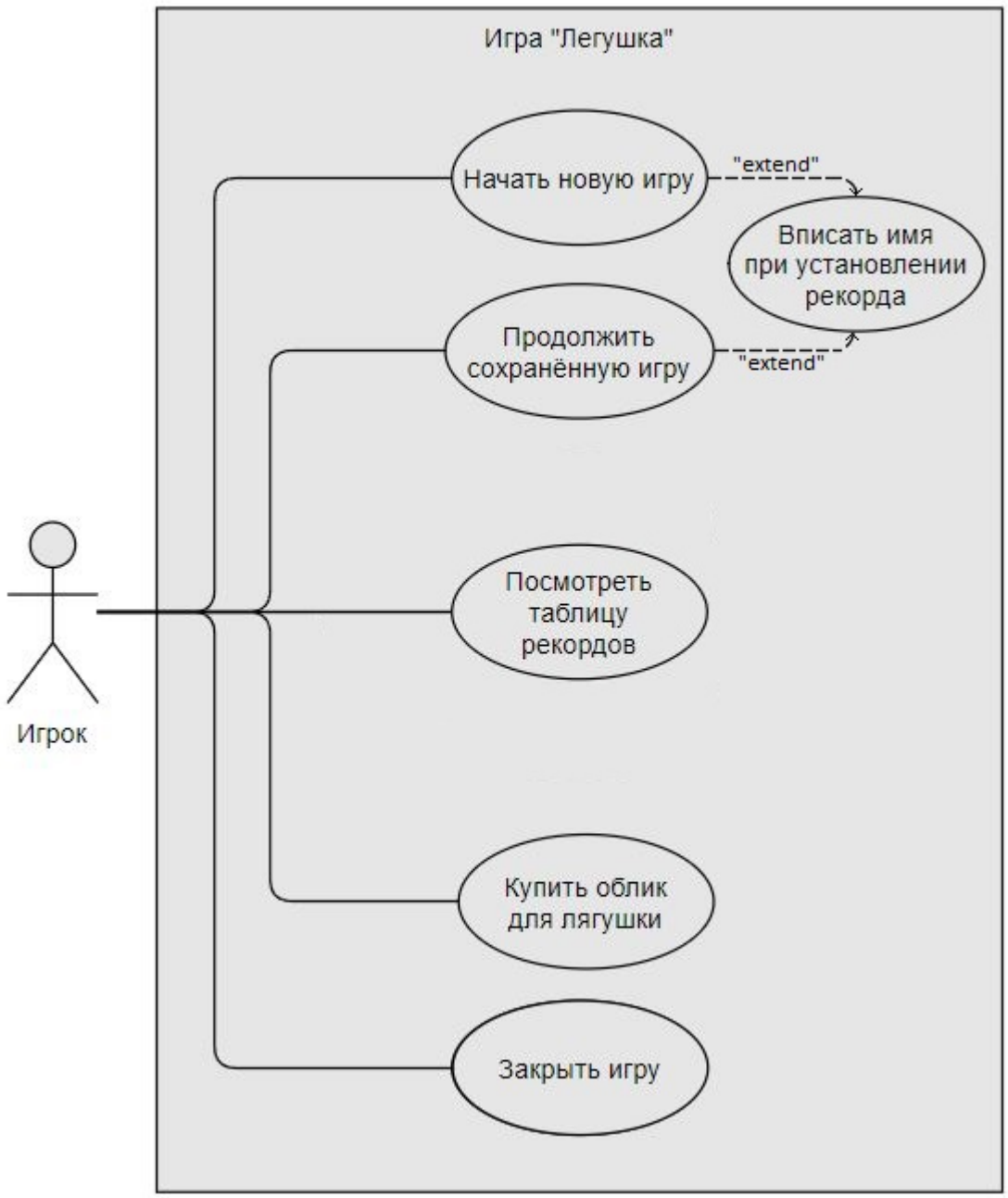

Рис. 1 Диаграмма вариантов использования

Требования к функциональным характеристикам

1. Управление лягушкой должно происходить при помощи клавиатуры
2. Навигация по меню должна происходить при помощи мыши
3. Игра должна иметь сохранение как глобального, так и локального прогрессов
4. Лягушка должна умирать при соприкосновении с водой/машиной/грузовиком/поездом
5. Лягушка не может сходиться в сторону, где стоит дерево
6. Должна быть возможность начать новую игру, не заканчивая старую

7. При установке рекорда должно появляться окно ввода имени игрока

Требования к информационному обеспечению

Для работы игры используются сохранённые изображения, которые будут загружаться при запуске для отображения объектов.

Требования к лингвистическому обеспечению

Игра и документация только на русском языке. Многоязыковая поддержка отсутствует.

Требования к программному обеспечению

Наличие установленной .NET 6.0

Требования к техническому обеспечению

Для игры необходимо 1 Мбайт свободного дискового пространства и 200 Мбайт свободной оперативной памяти. Для управления необходимы клавиатура и мышь.

Этапы разработки

1. Реализация абстрактного класса блоков
2. Реализация классов каждого типа блоков
3. Реализация абстрактного класса предметов окружения
4. Реализация классов каждого типа предметов окружения
5. Реализация абстрактного класса движущихся предметов окружения
6. Реализация классов каждого типа движущихся предметов окружения
7. Реализация класса карты
8. Реализация класса лягушки
9. Реализация класса игры
10. Реализация взаимодействия объектов окружения между собой
11. Реализация взаимодействия лягушки с объектами окружения
12. Реализация меню
13. Реализация таблицы лидеров
14. Реализация магазина
15. Реализация информационной справки
16. Реализация сохранения и загрузки игрового прогресса

Объектно-ориентированный анализ предметной области

Игра разработана с помощью объектно-ориентированного подхода. В игре существует карта, состоящая из клеток. В клетках расположены различные блоки. Также есть второй слой клеток, состоящий из объектов окружения.

Независимо от клеток существуют движущиеся объекты и лягушка. На данной основе выделим следующие классы:

1. Абстрактный класс блок
2. Земля
3. Дорога
4. Железная дорога
5. Вода
6. Абстрактный класс предмет окружения
7. Дерево
8. Кувшинка
9. Монетка
10. Абстрактный класс движущийся предмет окружения
11. Машина
12. Грузовик
13. Поезд
14. Бревно
15. Лягушка

Данные классы являются основными в нашей игре.

Объектно-ориентированное программирование

Статическая модель

Block – абстрактный класс, от которого наследуются земля, дорога, железная дорога и вода. Он хранит свои координаты и изображение для отрисовки, метод для обновления координат, виртуальный метод для отрисовки и 3 абстрактных метода для коллизии с блоком, обновления параметров блока после десериализации и возврате данных о том, какие движущиеся предметы окружения могут находиться на этом блоке.

Earth, Road, Train Road и Water – унаследованные от Block классы, каждый по своему реализующий его абстрактные методы и имеющие некоторые дополнительные методы и параметры.

Item – абстрактный класс предмета окружения, от которого наследуются дерево, кувшинка и монетка. Идентичен по параметром и методам с классом блока, отсутствует лишь абстрактный метод для возврата данных о движущихся предметах окружения.

Tree, Lily, Coin – унаследованные от Item классы, реализующие его абстрактные методы. Монетка также имеет дополнительные параметры и методы для анимации.

Moving Item – абстрактный класс, унаследованные от Item. Дополнительно имеет такие параметры, как: скорость, направление, ширина и объект класса Rectangle для упрощения проверки столкновений с другими объектами. Также имеет несколько геттеров и переопределяет некоторые методы родительского класса.

Car, Truck, Train, Log – унаследованные от Moving Item классы, реализующие его абстрактные методы. Также у поезда есть задержка перед появлением, чтобы дать некоторое время игроку убежать от своей гибели.

Frog – класс лягушки. Хранит в себе координаты, направление, картинки для отрисовки, таймер для анимации и переменную о том, жива ли она. Также хранит методы для восстановления после десериализации, обновления координат при начале новой игры, отрисовки и обработки событий.

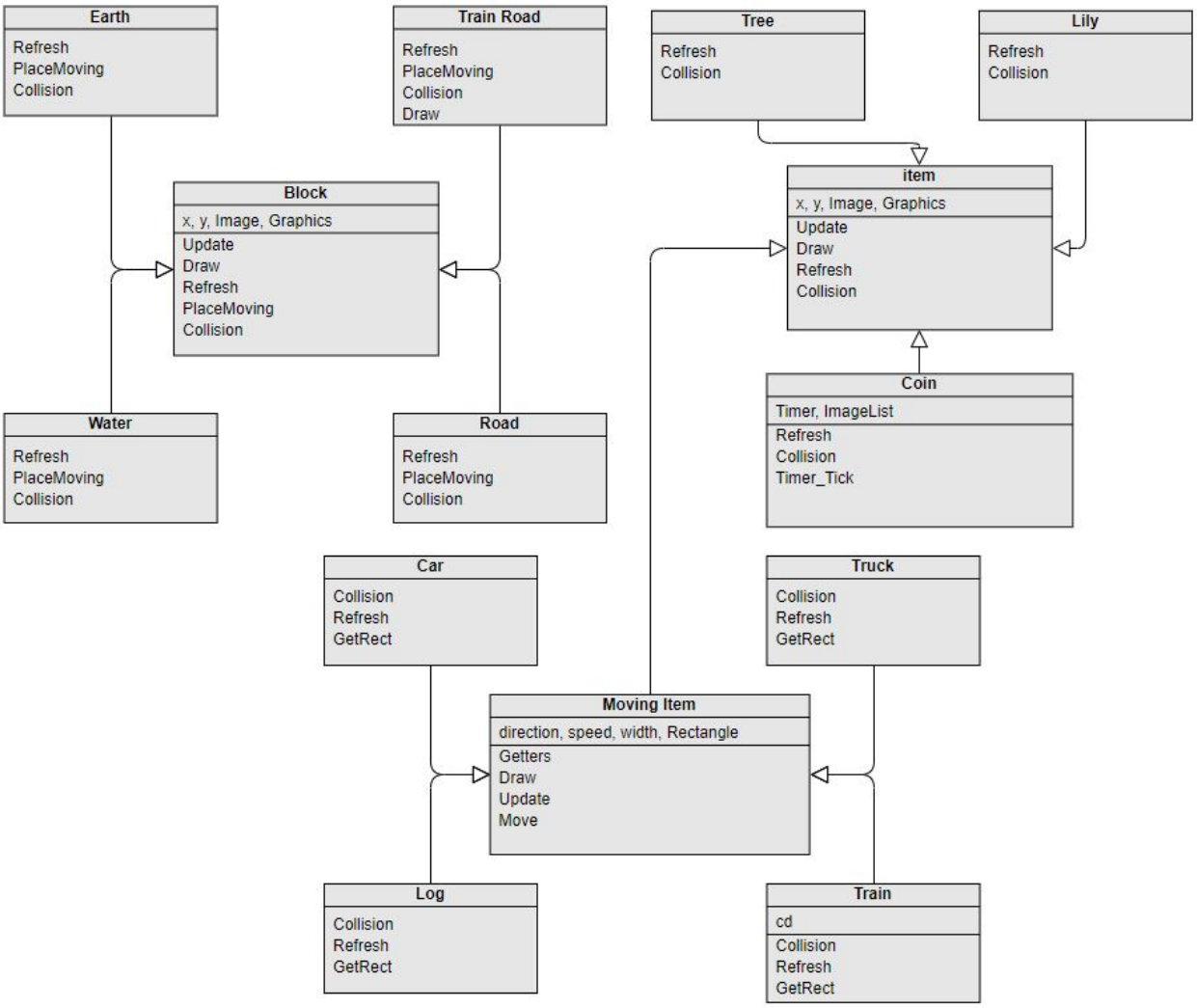

Рис. 2 Диаграмма классов. Наследование

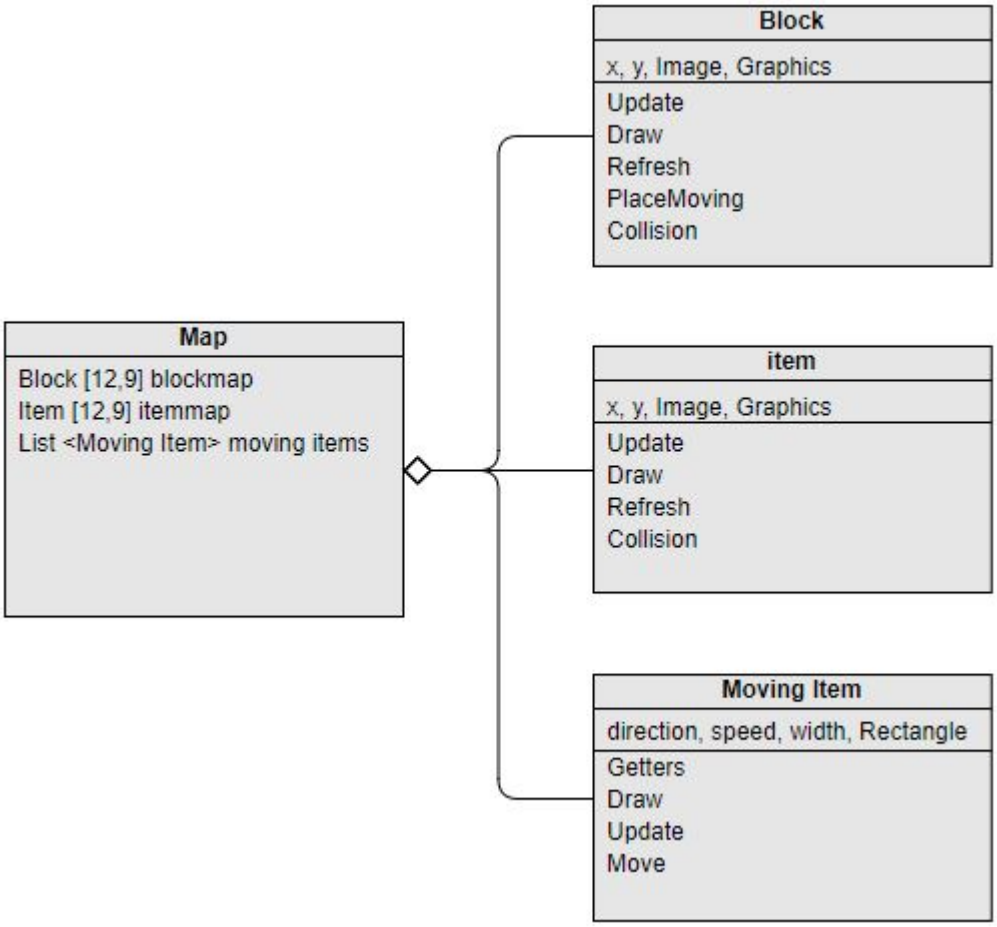

Рис. 3 Диаграмма классов. Агрегация

Динамическая модель

Рассмотрим динамическую модель программы на примере хода лягушки в другую клетку.

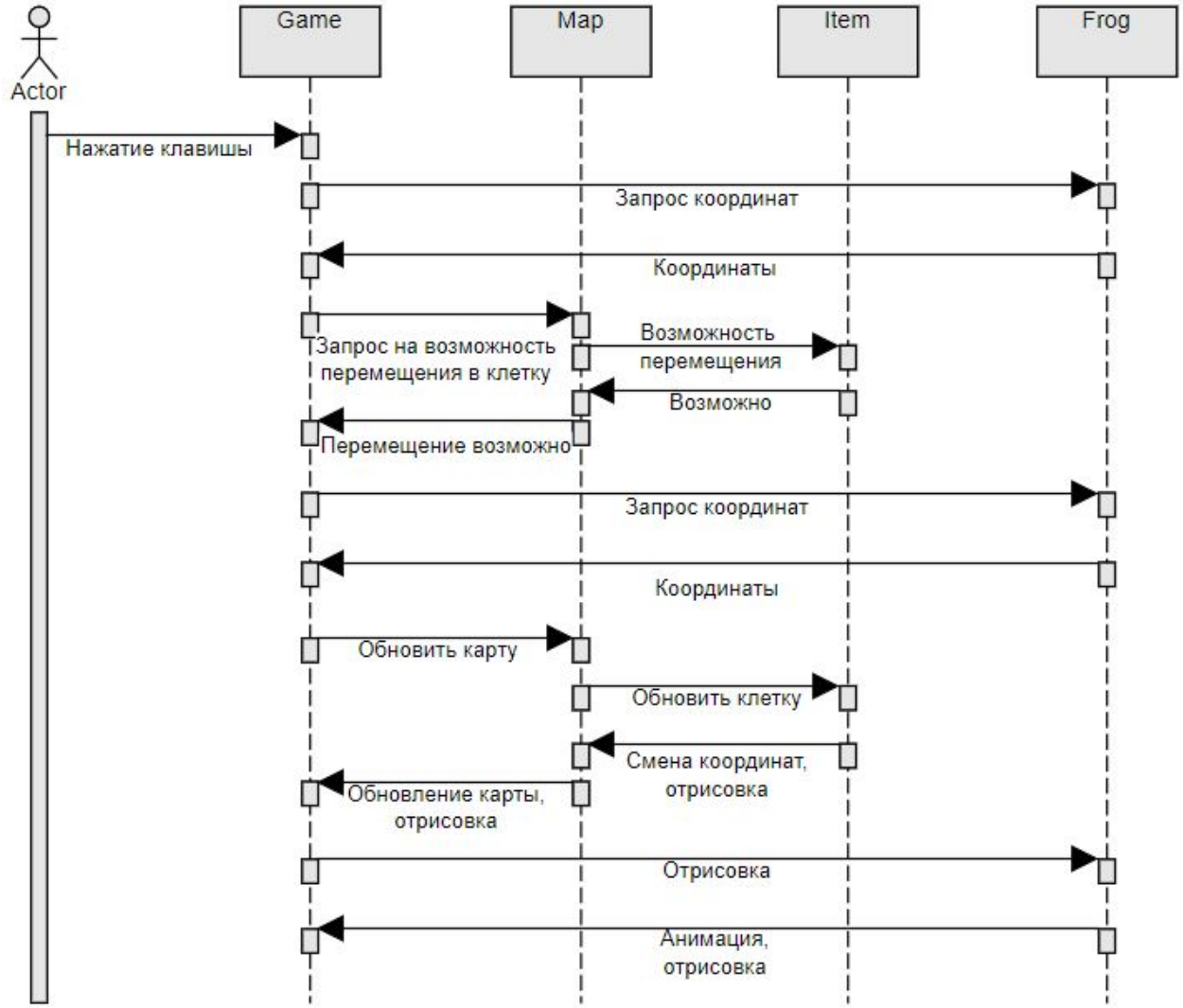

Рис. 4 Диаграмма последовательности. Ход лягушки

Реализация классов

Block

Public

public Block(int x, int y, Image img, Bitmap screen) // Конструктор блока

public void Update() // Обновление расположения блока при движении лягушки вверх

public virtual void Draw() // Отрисовка блока

public abstract void Refresh(Image img, Bitmap screen); // Обновление несериализуемых компонентов класса после десериализации

public abstract int PlaceMoving(ref bool direction) // Функция, возвращающая тип движущегося предмета, которого можно поставить на данный блок

public abstract void Collision(Message msg) // Функция коллизии с другими объектами

Protected

protected int x, y; // Координаты блока
protected int cd; // Время, через которое можно будет поставить следующий движущийся предмет на блок
protected Image img; // Картинка для отрисовки
protected Graphics graphics; // Графикс для отрисовки

Earth

Public

public Earth(int x, int y, Image img, Bitmap screen) : base(x, y, img, screen) // Конструктор
// Переопределённые функции родительского класса
public override void Collision(Message msg)
public override void Refresh(Image img, Bitmap screen)
public override int PlaceMoving(ref bool direction)

Road

Public

public Road(int x, int y, Image img, Bitmap screen, bool direction) : base(x, y, img, screen) // Конструктор
// Переопределённые функции родительского класса
public override void Collision(Message msg)
public override void Refresh(Image img, Bitmap screen)
public override int PlaceMoving(ref bool direction)

Water

Public

public Water(int x, int y, Image img, Bitmap screen, bool direction) : base(x, y, img, screen) // Конструктор
// Переопределённые функции родительского класса
public override void Collision(Message msg)
public override void Refresh(Image img, Bitmap screen)
public override int PlaceMoving(ref bool direction)

Protected

bool direction; // Направление бревна
Random r; // Рандом для направления бревна

Train road

Public

public Train_road(int x, int y, Image img, Bitmap screen, bool direction, Image redroad) : base(x, y, img, screen) // Конструктор
// Переопределённые функции родительского класса
public override void Collision(Message msg)
public override void Refresh(Image img, Bitmap screen)
public override int PlaceMoving(ref bool direction)
public override void Draw()

Protected

bool direction; // Направление поезда

Image redroad; // Картинка дорожки, сигнализирующей о приближающемся поезде

int waitfortrain = 0; // Задержка приближающегося поезда

Random r; // Рандом для направления поезда

Item

Public

public Item(int x, int y, Image img, Bitmap screen) // Конструктор

virtual public void Update() // Обновление расположения предмета окружения при движении лягушки вверх

virtual public void Draw() // Отрисовка предмета окружения

public abstract void Collision(Message msg) // Функция коллизии с другими объектами

public abstract void Refresh(Image img, Bitmap screen) // Обновление несериализуемых компонентов класса после десериализации

Protected

protected int x, y; // Координаты предмета окружения

protected Image img; // Картинка для отрисовки

protected Graphics graphics; // Графикс для отрисовки

Tree

Public

public Tree(int x, int y, Image img, Bitmap screen) : base(x, y, img, screen) // Конструтор

// Переопределённые функции родительского класса

public override void Collision(Message msg)

public override void Refresh(Image img, Bitmap screen)

Lily

Public

public Lily(int x, int y, Image img, Bitmap screen) : base(x, y, img, screen) // Конструтор

// Переопределённые функции родительского класса

public override void Collision(Message msg)

public override void Refresh(Image img, Bitmap screen)

Coin

Public

public Coin(int x, int y, Image img, Bitmap screen, ImageList imgl) : base(x, y, img, screen) // Конструктор

protected void t_tick(object sender, EventArgs eArgs) // Тик таймера

// Переопределённые функции родительского класса

public override void Refresh(Image img, Bitmap screen)

public override void Collision(Message msg)

Protected

Timer t; // Таймер для анимации

ImageList imgl; // Массив картинок для анимации

int index = 0; // Индекс текущей картинки

Moving Item

Public

public Moving_Item(int x, int y, Image img, Bitmap screen, bool direction, int speed, int width) : base(x, y, img, screen) // Конструктор

public int GetSpeed() // Возвращает скорость объекта

public void ChangeSpeed(int speed) // Меняет скорость объекта

public int GetY() // Возвращает y координату

public int GetX() // Возвращает x координату

public abstract Rectangle GetRect() // Возвращает прямоугольник для коллизии

override public void Draw() // Отрисовка блока

public override void Update() // Обновление расположения объекта при движении лягушки вверх

virtual public void Move() // Движение объекта

Protected

protected bool direction; // Направление

protected int speed; // Скорость

protected int width; // Ширина

protected Rectangle rect; // Прямоугольник для упрощения выявления столкновений с другими объектами

Car

Public

public Car(int x, int y, Image img, Bitmap screen, bool direction, int speed, int width) : base(x, y, img, screen, direction, speed, width) // Конструктор

// Переопределённые функции родительского класса

public override void Collision(Message msg)

public override void Refresh(Image img, Bitmap screen)

public override Rectangle GetRect()

Log

Public

public Log(int x, int y, Image img, Bitmap screen, bool direction, int speed, int width) : base(x, y, img, screen, direction, speed, width) // Конструктор

// Переопределённые функции родительского класса

public override void Collision(Message msg)

public override void Refresh(Image img, Bitmap screen)

public override Rectangle GetRect()

Train

Public

public Train(int x, int y, Image img, Bitmap screen, bool direction, int speed, int width) : base(x, y, img, screen, direction, speed, width) // Конструктор

// Переопределённые функции родительского класса

public override void Collision(Message msg)

public override void Refresh(Image img, Bitmap screen)

public override void Move()

public override Rectangle GetRect()

Protected

int cd = 120; // Задержка перед движением

Truck

Public

public Truck(int x, int y, Image img, Bitmap screen, bool direction, int speed, int width) : base(x, y, img, screen, direction, speed, width) // Конструктор

// Переопределённые функции родительского класса

public override void Collision(Message msg)

public override void Refresh(Image img, Bitmap screen)

public override Rectangle GetRect()

Frog

Public

public Frog(int x, int y, ImageList imgl, Bitmap screen) // Конструктор

public void Refresh(ImageList imgl, Bitmap screen) // Обновление несериализуемых компонентов класса после десериализации

protected void t_tick(object sender, EventArgs eArgs) // Тик таймера

public void Round() // Округление координат лягушки для использования, как индексов массива

public void Shrink(ImageList imgl) // Обновление до стартового состояния

public void Direction() // Смена направления

public void Event(Message msg) // Обработчик событий

public void GetCoordinates(Message msg) // Геттер координат

public void Draw() // Отрисовка

Protected

protected int x, y; // Координаты

protected int direction; // Направление

protected ImageList imgl; // Массив картинок для анимации

protected Image img; // Картинка для отрисовки

protected Graphics graphics; // Графикс для отрисовка

protected int startx, starty; // Стартовая позиция лягушки

protected bool dead; // Мертва ли лягушка

Timer t; // Таймер для анимации

Map

Public

public Map(Form1 form, Bitmap screen) // Конструктор

public void Refresh(Form1 form, Bitmap screen) // Обновление несериализуемых компонентов класса после десериализации

public void Shrink() // Обновление карты в начальное положение

public void Collision(Message msg, ref int coins) // Проверка на столкновения между объектами на карте

public void DrawMoving() // Отрисовка движущихся объектов

public void Draw() // Отрисовка блоков и предметов окружения

public void Update() // Обновление карты при движении лягушки вверх

public void Move() // Движение движущихся объектов

public void CheckForMove(ref Message msg) // Проверка может ли лягушка идти в определённом направлении

Protected

protected Block[,] blockmap; // Массив блоков

protected Item[,] itemmap; // Массив предметов окружения

protected List<Moving_Item> moving_items; // Список движущихся объектов

protected List<int[]> templates; // Список шаблонов

Form1 form; // Ссылка на форму

Bitmap screen; // Ссылка на картинку для отрисовки

int[,] startmap; // Стартовое положение карты

// Буферные переменные

Random rnd;

protected bool direction;

protected int car_type;

Rectangle r;

Message msg;

int index;

// Картинки блоков, предметов окружения и движущихся объектов

protected void GetTemplate() // Наугад берущийся шаблон для обновления карты

Menu

Public

public MenuMessage msg; // Сообщение для контроля ситуации

public Form1 form; // Ссылка на форму

// Картинки для отрисовки

public LeaderBoard leaders; // Таблица лидеров

public Shop shop; // Магизин

public Menu(MenuMessage msg, Form1 form, string[] filebuf) // Конструктор

public string[] GetLeaders() // Возвращает таблицу лидеров

public string[] GetCosts() // Возвращает массив цен на лягушек в магазине

public void Draw() // Отрисовка меню

public void SetFocus() // Установка фокуса на невидимый объект формы
public void Hide() // Скрытие всех кнопок

Protected

// Обработчики нажатий кнопок
// Кнопки

LeaderBoard

Public

public LeaderBoard(Form1 form, string[] filebuf) // Конструктор
public bool Check(int score) // Проверяет является ли кол-во очков рекордом
public void DeleteBase() // Обнуляет таблицу рекордов
public string[] GetInfo() // Возвращает массив имён и кол-ва очков
public void Add(string name, int score) // Добавляет в таблицу рекорд
public void Draw() // Отрисовка

Protected

protected string[] names; // Массив имён лидеров
protected int[] scores; // Массив кол-ва очков
protected Graphics g; // Графикс для отрисовки

Shop

Public

public Shop(MenuMessage msg, Form1 form, string[] costs) // Конструктор
public string[] GetCosts() // Возвращает цены на лягушек
public void DeleteBase() // Возвращает цены и делает всех лягушек не купленными
public void Middle() // Выставляет индексы левой, средней и правой лягушек
public void DrawButtons() // Отрисовка кнопок
public void Draw() // Отрисовка
public void Hide() // Скрывает все кнопки

Protected

List<Image> images2; // Массив картинок для боковых лягушек
List<Image> images4; // Массив картинок для лягушек посередине
// Кнопки
int l, m, r; // Индексы левой, средней и правой лягушек
Form1 form; // Ссылка на форму
MenuMessage msg; // Ссылка на сообщение
Graphics g; // Графикс для отрисовки
int[] costs; // Массив цен
bool[] bought; // Массив, определяющий, куплена лягушка или нет
Image back; // Задний фон
Bitmap screen; // Картинка для отрисовки
Graphics gscreen; // Графикс для отрисовки на картинку
// Обработчики нажатий кнопок

Game

Public

public Game(Form1 form) // Конструктор

Protected

protected Message msg; // Сообщение для управления игрой
protected Map map; // Карта
protected Frog frog; // Лягушка
protected Form1 form; // Ссылка на форму
protected Bitmap screen; // Картинка для отрисовки на форму
protected bool dead; // Мертва лягушка или нет
protected int score; // Текущий счёт
protected Timer t, t2; // Таймеры t для постоянных вызовов метода Play, t2 для задержки между нажатиями клавиш
protected Menu menu; // Меню
protected MenuMessage menumsg; // Сообщение для управления меню
protected int coordinate; // y координата лягушки
protected TextBox input; // Окно ввода имени при установке рекорда
protected List<ImageList> img; // Массив массивов картинок лягушки
protected Graphics gscreen, gform; // Графикс формы и картинки для отрисовки на форму
protected Image plateimage; // Задний фон для инпута
protected string[] filebuf; // Массив для считывания и записи в файл информации о рекордах, кол-ве монет и тд
private void Form_FormClosed(object sender, FormClosedEventArgs e) // Функция закрытия формы
protected void Serialization() // Сериализация
protected void DeSerialization() // Десериализация
protected void t_tick(object sender, EventArgs eArgs) // Тик таймера
protected void t2_tick(object sender, EventArgs eArgs) // Тик таймера
protected void KeyDown(object sender, KeyEventArgs e) // Обработчик нажатия клавиши
protected void Draw() // Отрисовка в зависимости от того где находится пользователь (в меню/игре/магизине...)
protected void Collision() // Проверка на столкновения
protected void Shrink() // Обновление игры до стартового состояния
protected void Play() // Весь игровой процесс
protected void Event(string str) // Обработчик событий

Руководство пользователя

Для корректной работы программы необходимы следующие файлы:
Папки: img.
Файлы: data.txt, Легушка.deps.json, Легушка.dll, Легушка.exe, Легушка.pdb, Легушка.runtimeconfig.dev.json, Легушка.runtimeconfig.json

Инструкция по использованию

Для запуска игры запустите Легушка.exe. Перед вами появится меню игры. Кнопка “Новая игра” начинает новую игру, “Лидеры” показывает таблицу рекордов, “Магазин” открывает магазин для кастомизации персонажа, “Инфо” показывает краткое описание и цели игры, “Выход” закрывает программу.

Во время игры управление осуществляется при помощи клавиш WASD, выход в меню при помощи клавиши Escape, возвращение из меню в игру происходит по нажатию этой же клавиши. При поражении начать новую игру можно не выходя в меню, нажав Space. В остальных пунктах программы управление осуществляется с помощью мыши.

Рис. 5 Меню игры

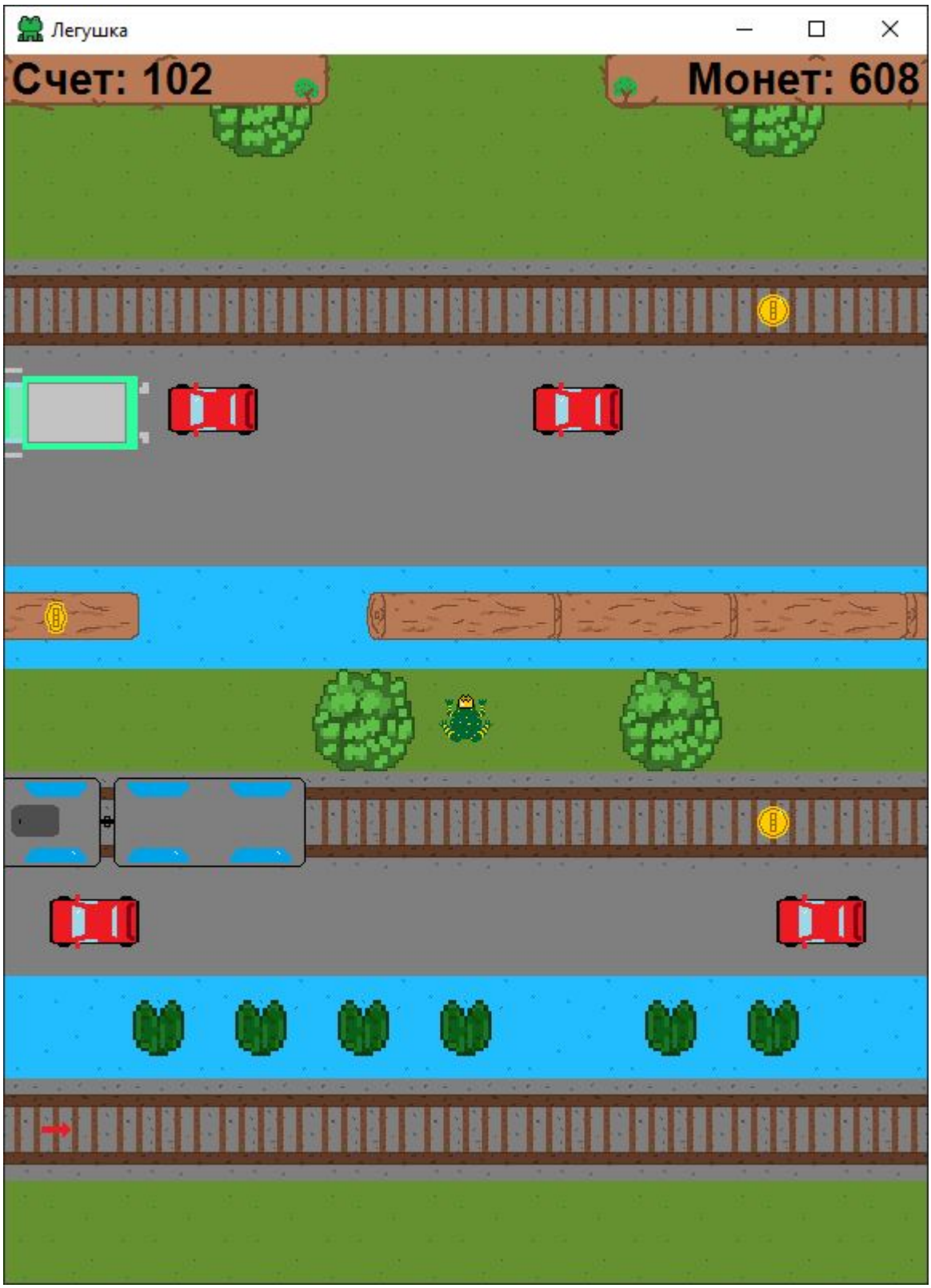

Рис. 6 Игра

Рис. 7 Таблица рекордов

Рис. 8 Внутриигровой магазин

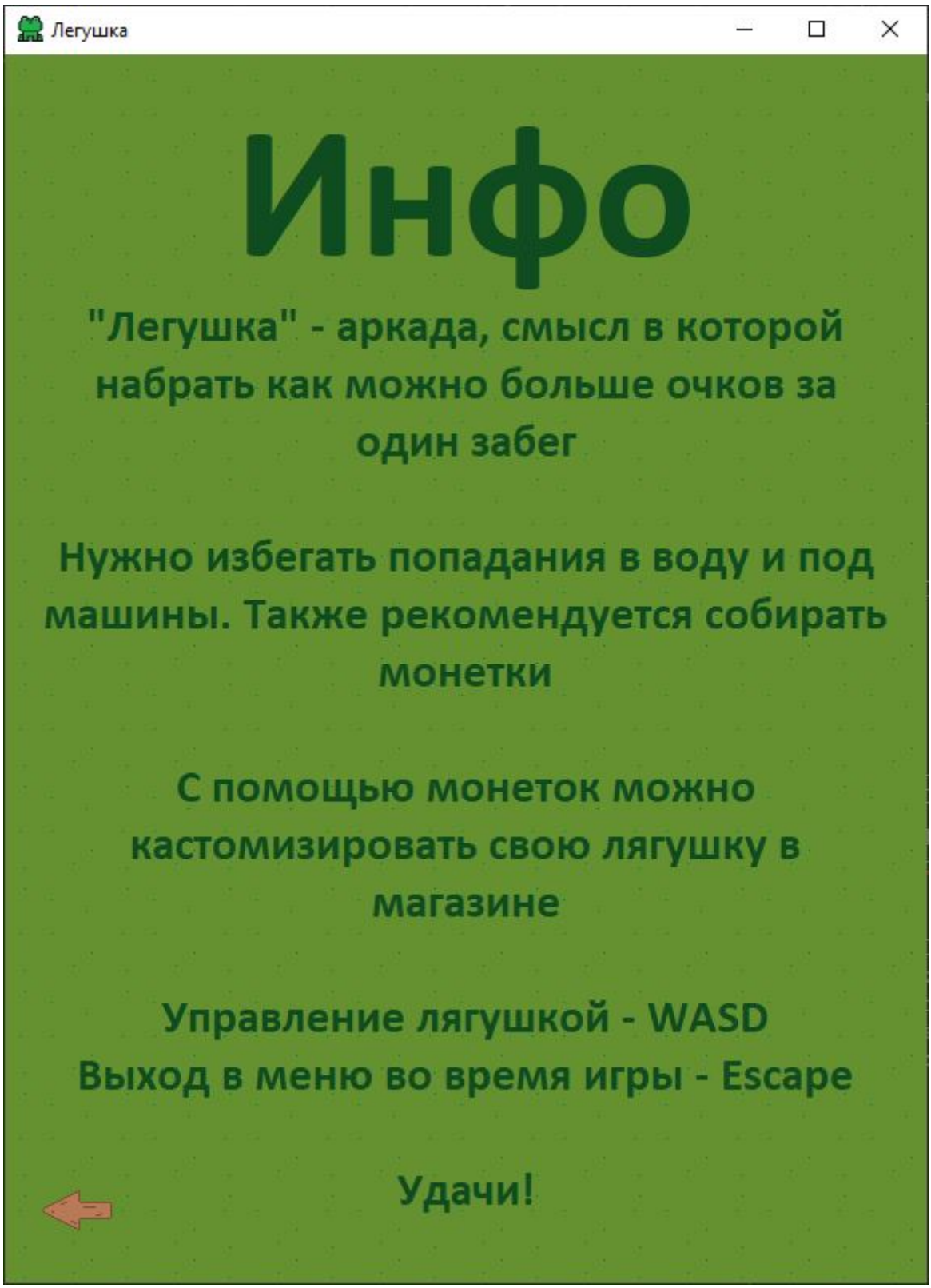

Рис. 9 Краткая информация об игре

Приложение. Код программы

Block.cs

```
using System;
using System.Drawing;

namespace Легушка
{
   [Serializable]
   abstract class Block
   {
      protected int x, y;
      protected int cd;
      [NonSerialized]
      protected Image img;
      [NonSerialized]
      protected Graphics graphics;
      public Block(int x, int y, Image img, Bitmap screen)
      {
         cd = 0;
         this.x = x;
         this.y = y;
         this.img = img;
         graphics = Graphics.FromImage(screen);
      }

      public void Update() { y++; }

      public virtual void Draw()
      {
         graphics.DrawImage(img, new Point(x * 64, y * 64));
      }

      public abstract void Refresh(Image img, Bitmap screen);
      public abstract int PlaceMoving(ref bool direction);
      public abstract void Collision(Message msg);
   }
}
```

Earth.cs

```
using System;
using System.Drawing;

namespace Легушка
{
```

```
        [Serializable]
        class Earth : Block
        {
            public Earth(int x, int y, Image img, Bitmap screen) : base(x, y, img,
screen) { }
            public override void Collision(Message msg)
            {
                msg.key = "E";
            }

            public override void Refresh(Image img, Bitmap screen)
            {
                this.img = img;
                graphics = Graphics.FromImage(screen);
            }

            public override int PlaceMoving(ref bool direction)
            {
                return 0;
            }
        }
    }
```

Road.cs

```
    using System;
    using System.Drawing;

    namespace Легушка
    {
        [Serializable]
        class Road : Block
        {
            bool direction;
            [NonSerialized]
            Random r;
            public Road(int x, int y, Image img, Bitmap screen, bool direction) :
base(x, y, img, screen)
            {
                r = new Random();
                this.direction = direction;
            }

            public override void Refresh(Image img, Bitmap screen)
            {
                this.img = img;
```

```
            graphics = Graphics.FromImage(screen);
            r = new Random();
        }

        public override void Collision(Message msg)
        {
            msg.key = "R";
        }

        public override int PlaceMoving(ref bool direction)
        {
            if (cd > 0) cd--;
            else if (r.Next(100) < 2)
            {
                direction = this.direction;
                cd = 64;
                if (r.Next(100) > 30) return 2;
                return 3;
            }
            return 0;
        }
    }
}
```

Train road.cs

```
using System;
using System.Drawing;

namespace Легушка
{
    [Serializable]
    class Train_road : Block
    {
        bool direction;
        [NonSerialized]
        Image redroad;
        int waitfortrain = 0;
        [NonSerialized]
        Random r;
        public Train_road(int x, int y, Image img, Bitmap screen, bool direction,
Image redroad) : base(x, y, img, screen)
        {
            r = new Random();
            this.redroad = redroad;
            this.direction = direction;
```

```
        }
        public override void Collision(Message msg)
        {
            msg.key = "T";
        }

        public override void Refresh(Image img, Bitmap screen)
        {
            this.img = img;
            graphics = Graphics.FromImage(screen);
            r = new Random();
            redroad = Properties.Resources.redtrainroad;
        }

        public override void Draw()
        {
            if (waitfortrain > 0)
            {
                waitfortrain--;
                if (direction == false)
                {
                    redroad.RotateFlip(RotateFlipType.Rotate180FlipNone);
                    graphics.DrawImage(redroad, new Point(x * 64, y * 64));
                    redroad.RotateFlip(RotateFlipType.Rotate180FlipNone);
                }
                else graphics.DrawImage(redroad, new Point(x * 64, y * 64));
            }
            else graphics.DrawImage(img, new Point(x * 64, y * 64));
        }

        public override int PlaceMoving(ref bool direction)
        {
            if (cd > 0) cd--;
            else if (r.Next(100) < 1)
            {
                direction = this.direction;
                cd = 256;
                waitfortrain = 120;
                return 4;
            }
            return 0;
        }
    }
}
```

Water.cs

```
using System;
using System.Drawing;

namespace Легушка
{
   [Serializable]
   class Water : Block
   {
      bool direction;
      [NonSerialized]
      Random r;
      public Water(int x, int y, Image img, Bitmap screen, bool direction) :
base(x, y, img, screen)
      {
         r = new Random();
         this.direction = direction;
      }
      public override void Collision(Message msg)
      {
         msg.key = "DIE";
      }
      public override void Refresh(Image img, Bitmap screen)
      {
         this.img = img;
         graphics = Graphics.FromImage(screen);
         r = new Random();
      }
      public override int PlaceMoving(ref bool direction)
      {
         if (cd > 0) cd--;
         else if (r.Next(100) < 5)
         {
            direction = this.direction;
            cd = 64;
            return 1;
         }
         return 0;
      }
   }
}
```

Item.cs

```
using System;
using System.Drawing;
```

```
namespace Легушка
{
    [Serializable]
    abstract class Item
    {
        protected int x, y;
        [NonSerialized]
        protected Image img;
        [NonSerialized]
        protected Graphics graphics;
        public Item(int x, int y, Image img, Bitmap screen)
        {
            this.x = x;
            this.y = y;
            this.img = img;
            graphics = Graphics.FromImage(screen);
        }

        virtual public void Update() { y++; }

        virtual public void Draw()
        {
            graphics.DrawImage(img, new Point(x * 64, y * 64));
        }

        public abstract void Collision(Message msg);
        public abstract void Refresh(Image img, Bitmap screen);
    }
}
```

Lily.cs

```
using System;
using System.Drawing;

namespace Легушка
{
    [Serializable]
    class Lily : Item
    {
        public Lily(int x, int y, Image img, Bitmap screen) : base(x, y, img, screen)
{ }
        public override void Collision(Message msg)
        {
            msg.key = "";
```

```
        }
        public override void Refresh(Image img, Bitmap screen)
        {
            this.img = img;
            this.graphics = Graphics.FromImage(screen);
        }
    }
}
```

Tree.cs

```
using System;
using System.Drawing;

namespace Легушка
{
    [Serializable]
    class Tree : Item
    {
        public Tree(int x, int y, Image img, Bitmap screen) : base(x, y, img, screen)
{ }
        public override void Collision(Message msg)
        {
            msg.key = "STOP";
        }
        public override void Refresh(Image img, Bitmap screen)
        {
            this.img = img;
            this.graphics = Graphics.FromImage(screen);
        }
    }
}
```

Coin.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Легушка
{
    [Serializable]
    class Coin : Item
    {
        [NonSerialized]
        Timer t;
        [NonSerialized]
```

```
        ImageList imgl;
        int index = 0;
        public Coin(int x, int y, Image img, Bitmap screen, ImageList imgl) :
base(x, y, img, screen)
        {
            this.imgl = imgl;
            t = new Timer();
            t.Tick += new EventHandler(t_tick);
            t.Interval = 200;
            t.Start();
        }
        public override void Refresh(Image img, Bitmap screen)
        {
            imgl = new ImageList();
            imgl.ImageSize = new Size(64, 64);
            imgl.Images.Add(Properties.Resources.coin);
            imgl.Images.Add(Properties.Resources.coin1);
            imgl.Images.Add(Properties.Resources.coin2);
            imgl.Images.Add(Properties.Resources.coin1);
            this.img = imgl.Images[index];
            this.graphics = Graphics.FromImage(screen);
            t = new Timer();
            t.Tick += new EventHandler(t_tick);
            t.Interval = 200;
            t.Start();
        }

        protected void t_tick(object sender, EventArgs eArgs)
        {
            index++;
            index %= imgl.Images.Count;
            img = imgl.Images[index];
        }

        public override void Collision(Message msg)
        {
            msg.key = "COIN";
        }
    }
}
```

Moving Item.cs

```
using System;
using System.Drawing;
```

```
namespace Легушка
{
    [Serializable]
    abstract class Moving_Item : Item
    {
        protected bool direction; // 0 - влево, 1 - вправо
        protected int speed;
        protected int width; // Измеряется в блоках
        [NonSerialized]
        protected Rectangle rect;
        public Moving_Item(int x, int y, Image img, Bitmap screen, bool direction,
int speed, int width) : base(x, y, img, screen)
        {
            this.direction = direction;
            this.speed = speed;
            this.width = width;
        }

        public int GetSpeed() { return speed; }
        public void ChangeSpeed(int speed) { this.speed = speed; }
        public int GetY() { return y; }
        public int GetX() { return x; }
        public abstract Rectangle GetRect();

        override public void Draw()
        {
            if (direction == false)
            {
                img.RotateFlip(RotateFlipType.Rotate180FlipNone);
                graphics.DrawImage(img, new Point(x, y));
                img.RotateFlip(RotateFlipType.Rotate180FlipNone);
            }
            else graphics.DrawImage(img, new Point(x, y));
        }

        public override void Update()
        {
            y += 64;
        }
        virtual public void Move()
        {
            if (direction == false) x -= speed;
            else x += speed;
        }
    }
```

```
}
```

Car.cs

```
using System;
using System.Drawing;

namespace Легушка
{
   [Serializable]
   class Car : Moving_Item
   {
      public Car(int x, int y, Image img, Bitmap screen, bool direction, int speed,
int width) : base(x, y, img, screen, direction, speed, width) { }
      public override void Collision(Message msg)
      {
         rect.X = x + 15;
         rect.Y = y;
         rect.Width = width - 30;
         rect.Height = 64;
         rect.Intersect(msg.rect);
         if (!rect.IsEmpty) msg.key = "DIE";
      }
      public override void Refresh(Image img, Bitmap screen)
      {
         this.img = Properties.Resources.car;
         this.graphics = Graphics.FromImage(screen);
         this.rect = new Rectangle();
      }
      public override Rectangle GetRect()
      {
         rect.X = x;
         rect.Y = y;
         rect.Width = width;
         rect.Height = 64;
         return rect;
      }
   }
}
```

Log.cs

```
using System;
using System.Drawing;

namespace Легушка
{
```

```
    [Serializable]
    class Log : Moving_Item
    {
        public Log(int x, int y, Image img, Bitmap screen, bool direction, int speed,
int width) : base(x, y, img, screen, direction, speed, width) { }
        public override void Collision(Message msg)
        {
            rect.X = x + 10;
            rect.Y = y;
            rect.Width = width - 20;
            rect.Height = 64;
            rect.Intersect(msg.rect);
            if (!rect.IsEmpty)
            {
                if (direction == false) msg.key = "LOGL";
                else msg.key = "LOGR";
                msg.movingspeed = speed;
            }
        }
        public override void Refresh(Image img, Bitmap screen)
        {
            this.img = Properties.Resources.log;
            this.graphics = Graphics.FromImage(screen);
            this.rect = new Rectangle();
        }
        public override Rectangle GetRect()
        {
            rect.X = x + 8;
            rect.Y = y;
            rect.Width = width - 16;
            rect.Height = 64;
            return rect;
        }
    }
}
```

Train.cs

```
using System;
using System.Drawing;

namespace Легушка
{
    [Serializable]
    class Train : Moving_Item
    {
```

```
        int cd = 120;
        public Train(int x, int y, Image img, Bitmap screen, bool direction, int
speed, int width) : base(x, y, img, screen, direction, speed, width) { }
        public override void Collision(Message msg)
        {
            rect.X = x + 20;
            rect.Y = y;
            rect.Width = width - 40;
            rect.Height = 64;
            rect.Intersect(msg.rect);
            if (!rect.IsEmpty) msg.key = "DIE";
        }
        public override void Refresh(Image img, Bitmap screen)
        {
            this.img = Properties.Resources.train;
            this.graphics = Graphics.FromImage(screen);
            this.rect = new Rectangle();
        }
        public override void Move()
        {
            if (cd > 0) cd--;
            else
            {
                if (direction == false) x -= speed;
                else x += speed;
            }
        }
        public override Rectangle GetRect()
        {
            rect.X = x;
            rect.Y = y;
            rect.Width = width;
            rect.Height = 64;
            return rect;
        }
    }
}
```

Truck.cs

```
using System;
using System.Drawing;

namespace Легушка
{
    [Serializable]
```

```
    class Truck : Moving_Item
    {
        public Truck(int x, int y, Image img, Bitmap screen, bool direction, int
speed, int width) : base(x, y, img, screen, direction, speed, width) { }
        public override void Collision(Message msg)
        {
            rect.X = x + 20;
            rect.Y = y;
            rect.Width = width - 40;
            rect.Height = 64;
            rect.Intersect(msg.rect);
            if (!rect.IsEmpty) msg.key = "DIE";
        }
        public override void Refresh(Image img, Bitmap screen)
        {
            this.img = Properties.Resources.truck;
            this.graphics = Graphics.FromImage(screen);
            this.rect = new Rectangle();
        }

        public override Rectangle GetRect()
        {
            rect.X = x;
            rect.Y = y;
            rect.Width = width;
            rect.Height = 64;
            return rect;
        }
    }
}
```

Frog.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Легушка
{
    [Serializable]
    class Frog
    {
        protected int x, y;
        protected int direction; // 0 - вверх, 1 - вправо, 2 - вниз, 3 - влево
        [NonSerialized]
        protected ImageList imgl;
```

```
[NonSerialized]
protected Image img;
[NonSerialized]
protected Graphics graphics;
protected int startx, starty;
protected bool dead;
[NonSerialized]
Timer t;

public Frog(int x, int y, ImageList imgl, Bitmap screen)
{
   this.startx = x;
   this.starty = y;
   this.x = x;
   this.y = y;
   this.imgl = imgl;
   this.img = this.imgl.Images[0];
   dead = false;
   t = new Timer();
   t.Tick += new EventHandler(t_tick);
   t.Interval = 200;
   t.Stop();


   graphics = Graphics.FromImage(screen);
}

public void Refresh(ImageList imgl, Bitmap screen)
{
   this.imgl = imgl;
   graphics = Graphics.FromImage(screen);
   this.img = this.imgl.Images[0];
   t = new Timer();
   t.Tick += new EventHandler(t_tick);
   t.Interval = 200;
   t.Stop();
}

protected void t_tick(object sender, EventArgs eArgs)
{
   img = imgl.Images[direction];
   t.Stop();
}

public void Round()
```

```
{
   this.x = (int)Math.Round((double)x / 64) * 64;
}

public void Shrink(ImageList imgl)
{
   this.x = startx;
   this.y = starty;
   dead = false;
   this.imgl = imgl;
   img = imgl.Images[0];
}

public void Direction()
{
   direction = 0;
   img = imgl.Images[direction + 4];
   t.Start();
}

public void Event(Message msg)
{
   if (msg.key == "" || dead) return;

   msg.x = this.x;
   msg.y = this.y;
   if (msg.key == "KeyUp")
   {
      direction = 0;
      img = imgl.Images[direction + 4];
      t.Start();
      Round();
      y -= 64;
   }
   else if (msg.key == "KeyDown")
   {
      direction = 2;
      img = imgl.Images[direction + 4];
      t.Start();
      Round();
      y += 64;
   }
   else if (msg.key == "KeyLeft")
   {
      direction = 3;
```

```
            img = imgl.Images[direction + 4];
            t.Start();
            x -= 64;
        }
        else if (msg.key == "KeyRight")
        {
            direction = 1;
            img = imgl.Images[direction + 4];
            t.Start();
            x += 64;
        }
        else if (msg.key == "LOGR") x += msg.movingspeed;
        else if (msg.key == "LOGL") x -= msg.movingspeed;
        else if (msg.key == "DIE")
        {
            dead = true;
            t.Stop();
            img = imgl.Images[direction + 8];
        }
    }

    public void GetCoordinates(Message msg)
    {
        msg.x = this.x;
        msg.y = this.y;
        msg.roundx = (int)Math.Round((double)x / 64);
        msg.roundy = y / 64;
        msg.rect = new Rectangle(x + 16, y + 16, 32, 32);
    }

    public void Draw()
    {
        graphics.DrawImage(img, new Point(x, y));
    }
  }
}
```

Map.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

namespace Легушка
{
```

```
[Serializable]
class Map
{
    protected Block[,] blockmap;
    protected Item[,] itemmap;
    protected List<Moving_Item> moving_items;
    protected List<int[]> templates;

    [NonSerialized]
    Form1 form;
    [NonSerialized]
    Bitmap screen;
    int[,] startmap;
    [NonSerialized]
    Random rnd;
    protected bool direction;
    protected int car_type;

    [NonSerialized]
    Image earthimage;
    [NonSerialized]
    Image waterimage;
    [NonSerialized]
    Image treeimage;
    [NonSerialized]
    Image lilyimage;
    [NonSerialized]
    Image roadimage;
    [NonSerialized]
    Image trainroadimage;
    [NonSerialized]
    Image redtrainroadimage;
    [NonSerialized]
    Image carimage;
    [NonSerialized]
    Image logimage;
    [NonSerialized]
    Image truckimage;
    [NonSerialized]
    Image trainimage;
    [NonSerialized]
    ImageList coinlist;
    [NonSerialized]
    Rectangle r;
    Message msg;
```

```
int index;

public Map(Form1 form, Bitmap screen)
{
   rnd = new Random();
   startmap = new int[12, 9]
   {
      { 0,0,0,0,0,0,0,0,0 },
      { 0,1,0,0,0,0,1,0,0 },
      { 0,1,0,0,1,0,0,0,0 },
      { 0,1,0,0,0,0,0,0,0 },
      { 0,0,1,0,0,1,0,1,0 },
      { 0,0,1,0,0,0,0,0,0 },
      { 0,1,0,0,0,0,0,0,0 },
      { 0,0,0,0,0,0,0,0,0 },
      { 0,0,0,1,0,0,0,1,0 },
      { 0,0,0,0,0,1,0,0,0 },
      { 0,1,0,1,0,0,0,1,0 },
      { 1,0,0,0,1,0,1,0,1 }
   };

   this.form = form;
   this.screen = screen;
   moving_items = new List<Moving_Item>();
   blockmap = new Block[startmap.GetLength(0), startmap.GetLength(1)];
   itemmap = new Item[startmap.GetLength(0), startmap.GetLength(1)];
   templates = new List<int[]>();

   earthimage = Properties.Resources.earth;
   treeimage = Properties.Resources.tree;
   waterimage = Properties.Resources.water;
   lilyimage = Properties.Resources.lily;
   roadimage = Properties.Resources.road;
   trainroadimage = Properties.Resources.trainroad;
   redtrainroadimage = Properties.Resources.redtrainroad;
   carimage = Properties.Resources.car;
   logimage = Properties.Resources.log;
   truckimage = Properties.Resources.truck;
   trainimage = Properties.Resources.train;
   coinlist = new ImageList();
   coinlist.ImageSize = new Size(64, 64);
   coinlist.Images.Add(Properties.Resources.coin);
   coinlist.Images.Add(Properties.Resources.coin1);
   coinlist.Images.Add(Properties.Resources.coin2);
   coinlist.Images.Add(Properties.Resources.coin1);
```

```
    r = new Rectangle();
    msg = new Message();

    templates.Add(new int[9] { 0, 0, 0, 0, 0, 0, 0, 0, 0 });
    templates.Add(new int[9] { 6, 0, 0, 0, 0, 0, 0, 0, 0 });
    templates.Add(new int[9] { 0, 0, 0, 0, 0, 0, 0, 0, 6 });
    templates.Add(new int[9] { 0, 0, 0, 0, 0, 0, 0, 0, 0 });
    templates.Add(new int[9] { 0, 0, 0, 0, 0, 0, 0, 0, 0 });

    templates.Add(new int[9] { 0, 1, 0, 0, 0, 1, 0, 0, 0 });
    templates.Add(new int[9] { 0, 0, 0, 1, 0, 0, 1, 0, 0 });
    templates.Add(new int[9] { 0, 0, 1, 0, 0, 0, 0, 1, 0 });
    templates.Add(new int[9] { 6, 1, 0, 0, 0, 1, 0, 0, 1 });
    templates.Add(new int[9] { 1, 0, 0, 1, 0, 0, 1, 0, 6 });
    templates.Add(new int[9] { 0, 0, 1, 0, 1, 0, 0, 1, 0 });

    templates.Add(new int[9] { 2, 2, 2, 2, 2, 2, 2, 2, 2 });
    templates.Add(new int[9] { 7, 2, 2, 2, 2, 2, 2, 2, 2 });
    templates.Add(new int[9] { 2, 2, 2, 2, 2, 2, 2, 2, 7 });
    templates.Add(new int[9] { 2, 2, 2, 2, 2, 2, 2, 2, 2 });
    templates.Add(new int[9] { 2, 2, 2, 2, 2, 2, 2, 2, 2 });

    templates.Add(new int[9] { 2, 3, 2, 3, 2, 3, 3, 2, 3 });
    templates.Add(new int[9] { 3, 3, 2, 3, 2, 2, 3, 2, 2 });
    templates.Add(new int[9] { 2, 3, 3, 3, 3, 2, 3, 3, 2 });

    templates.Add(new int[9] { 4, 4, 4, 4, 4, 4, 4, 4, 4 });
    templates.Add(new int[9] { 4, 8, 4, 4, 4, 4, 4, 4, 4 });
    templates.Add(new int[9] { 4, 4, 4, 4, 4, 4, 4, 8, 4 });
    templates.Add(new int[9] { 4, 4, 4, 4, 4, 4, 4, 4, 4 });
    templates.Add(new int[9] { 4, 4, 4, 4, 4, 4, 4, 4, 4 });

    templates.Add(new int[9] { 5, 5, 5, 5, 5, 5, 5, 5, 5 });
    templates.Add(new int[9] { 5, 9, 5, 5, 5, 5, 5, 5, 5 });
    templates.Add(new int[9] { 5, 5, 5, 5, 5, 5, 5, 9, 5 });
    templates.Add(new int[9] { 5, 5, 5, 5, 5, 5, 5, 5, 5 });
    templates.Add(new int[9] { 5, 5, 5, 5, 5, 5, 5, 5, 5 });

    Shrink();
}

public void Refresh(Form1 form, Bitmap screen)
{
    this.form = form;
    this.screen = screen;
```

```
            rnd = new Random();
            earthimage = Properties.Resources.earth;
            treeimage = Properties.Resources.tree;
            waterimage = Properties.Resources.water;
            lilyimage = Properties.Resources.lily;
            roadimage = Properties.Resources.road;
            trainroadimage = Properties.Resources.trainroad;
            redtrainroadimage = Properties.Resources.redtrainroad;
            carimage = Properties.Resources.car;
            logimage = Properties.Resources.log;
            truckimage = Properties.Resources.truck;
            trainimage = Properties.Resources.train;
            coinlist = new ImageList();
            coinlist.ImageSize = new Size(64, 64);
            coinlist.Images.Add(Properties.Resources.coin);
            coinlist.Images.Add(Properties.Resources.coin1);
            coinlist.Images.Add(Properties.Resources.coin2);
            coinlist.Images.Add(Properties.Resources.coin1);
            r = new Rectangle();
            msg = new Message();

            for (int i = 0; i < startmap.GetLength(0); i++)
            {
               for (int j = 0; j < startmap.GetLength(1); j++)
               {
                  blockmap[i, j].Collision(msg);
                  if (msg.key == "E") blockmap[i, j].Refresh(earthimage, screen);
                  else if (msg.key == "R") blockmap[i, j].Refresh(roadimage, screen);
                  else if (msg.key == "T") blockmap[i, j].Refresh(trainroadimage,
screen);
                  else if (msg.key == "DIE") blockmap[i, j].Refresh(waterimage,
screen);
                  if (itemmap[i, j] != null)
                  {
                     itemmap[i, j].Collision(msg);
                     if (msg.key == "COIN") itemmap[i,
j].Refresh(coinlist.Images[0], screen);
                     else if (msg.key == "STOP") itemmap[i, j].Refresh(treeimage,
screen);
                     else if (msg.key == "") itemmap[i, j].Refresh(lilyimage, screen);
                  }
               }
            }
```

```
            for (int i = 0; i < moving_items.Count; i++)
moving_items[i].Refresh(treeimage, screen);
        }

        public void Shrink()
        {
            moving_items.Clear();
            for (int i = 0; i < startmap.GetLength(0); i++)
            {
                for (int j = 0; j < startmap.GetLength(1); j++)
                {
                    blockmap[i, j] = null;
                    itemmap[i, j] = null;
                    if (startmap[i, j] == 0)
                    {
                        blockmap[i, j] = new Earth(j, i, earthimage, screen);
                    }
                    else if (startmap[i, j] == 1)
                    {
                        itemmap[i, j] = new Tree(j, i, treeimage, screen);
                        blockmap[i, j] = new Earth(j, i, earthimage, screen);
                    }
                }
            }
        }

        public void Collision(Message msg, ref int coins)
        {
            if (msg.roundx > 8 || msg.roundx < 0 || msg.roundy > 11)
            {
                msg.key = "DIE";
            }
            else
            {
                blockmap[msg.roundy, msg.roundx].Collision(msg);
                if (itemmap[msg.roundy, msg.roundx] != null) itemmap[msg.roundy,
msg.roundx].Collision(msg);
                if (msg.key == "COIN")
                {
                    coins++;
                    itemmap[msg.roundy, msg.roundx] = null;
                }
                for (int i = 0; i < moving_items.Count; i++)
moving_items[i].Collision(msg);
            }
```

```
}

public void DrawMoving()
{
   for (int i = 0; i < moving_items.Count; i++) moving_items[i].Draw();
}

public void Draw()
{
   for (int i = 0; i < blockmap.GetLength(0); i++)
   {
      for (int j = 0; j < blockmap.GetLength(1); j++)
      {
         blockmap[i, j].Draw();
      }
   }
   DrawMoving();
   for (int i = 0; i < blockmap.GetLength(0); i++)
   {
      for (int j = 0; j < blockmap.GetLength(1); j++)
      {
         if (itemmap[i, j] != null) itemmap[i, j].Draw();
      }
   }
}

public void Update()
{
   for (int i = blockmap.GetLength(0) - 1; i > 0; i--)
   {
      for (int j = 0; j < blockmap.GetLength(1); j++)
      {

         blockmap[i, j] = blockmap[i - 1, j];
         blockmap[i, j].Update();
         itemmap[i, j] = itemmap[i - 1, j];
         if (itemmap[i, j] != null) itemmap[i, j].Update();
      }
   }
   for (int i = 0; i < moving_items.Count; i++) moving_items[i].Update();

   GetTemplate();
}

public void Move()
```

```
        {
            for (int i = 0; i < moving_items.Count; i++)
            {
                for (int j = i + 1; j < moving_items.Count; j++)
                {
                    if (moving_items[i].GetY() == moving_items[j].GetY() &&
moving_items[i].GetSpeed() < moving_items[j].GetSpeed())
                    {
                        r = moving_items[i].GetRect();
                        r.Intersect(moving_items[j].GetRect());
                        if (!r.IsEmpty)
moving_items[j].ChangeSpeed(moving_items[i].GetSpeed());
                    }
                }
            }
            for (int i = 0; i < moving_items.Count; i++)
            {
                moving_items[i].Move();
                if (moving_items[i].GetY() > 704 || moving_items[i].GetX() > 576 ||
moving_items[i].GetX() < -512)
                    moving_items.RemoveAt(i);
            }

            for (int i = 0; i < blockmap.GetLength(0); i++)
            {
                direction = false;
                for (int j = 0; j < blockmap.GetLength(1); j++)
                {
                    if (itemmap[i, j] != null)
                    {
                        itemmap[i, j].Collision(msg);
                        if (msg.key == "" || msg.key == "STOP")
                        {
                            direction = true;
                            break;
                        }
                    }
                }
                if (direction) continue;
                car_type = 0;
                car_type = blockmap[i, 0].PlaceMoving(ref direction);
                if (car_type == 0) continue;
                if (car_type == 1)
                {
```

```
                    if (direction) moving_items.Add(new Log(-128, i * 64, logimage,
screen, direction, rnd.Next(2) * 4 + rnd.Next(3) + 2, 128));
                    else moving_items.Add(new Log(576, i * 64, logimage, screen,
direction, rnd.Next(2) * 4 + rnd.Next(3) + 2, 128));
                }
                else if (car_type == 2)
                {
                    if (direction) moving_items.Add(new Car(-64, i * 64, carimage,
screen, direction, rnd.Next(2) * 6 + 2, 64));
                    else moving_items.Add(new Car(576, i * 64, carimage, screen,
direction, rnd.Next(2) * 6 + 2, 64));
                }
                else if (car_type == 3)
                {
                    if (direction) moving_items.Add(new Truck(-128, i * 64,
truckimage, screen, direction, rnd.Next(2) * 6 + 2, 128));
                    else moving_items.Add(new Truck(576, i * 64, truckimage, screen,
direction, rnd.Next(2) * 6 + 2, 128));
                }
                else if (car_type == 4)
                {
                    if (direction) moving_items.Add(new Train(-512, i * 64,
trainimage, screen, direction, 32, 512));
                    else moving_items.Add(new Train(576, i * 64, trainimage, screen,
direction, 32, 512));
                }
            }
        }

        protected void GetTemplate()
        {
            index = rnd.Next() % templates.Count;
            for (int i = 0; i < 9; i++)
            {
                blockmap[0, i] = null;
                itemmap[0, i] = null;
                if (templates[index][i] == 0)
                {
                    blockmap[0, i] = new Earth(i, 0, earthimage, screen);
                }
                else if (templates[index][i] == 1)
                {
                    itemmap[0, i] = new Tree(i, 0, treeimage, screen);
                    blockmap[0, i] = new Earth(i, 0, earthimage, screen);
                }
```

```
                else if (templates[index][i] == 2)
                {
                    blockmap[0, i] = new Water(i, 0, waterimage, screen, new
Random().Next(2) == 1 ? true : false);
                }
                else if (templates[index][i] == 3)
                {
                    itemmap[0, i] = new Lily(i, 0, lilyimage, screen);
                    blockmap[0, i] = new Water(i, 0, waterimage, screen, new
Random().Next(2) == 1 ? true : false);
                }
                else if (templates[index][i] == 4)
                {
                    blockmap[0, i] = new Road(i, 0, roadimage, screen, new
Random().Next(2) == 1 ? true : false);
                }
                else if (templates[index][i] == 5)
                {
                    blockmap[0, i] = new Train_road(i, 0, trainroadimage, screen, new
Random().Next(2) == 1 ? true : false, redtrainroadimage);
                }
                else if (templates[index][i] == 6)
                {
                    blockmap[0, i] = new Earth(i, 0, earthimage, screen);
                    itemmap[0, i] = new Coin(i, 0, coinlist.Images[0], screen, coinlist);
                }
                else if (templates[index][i] == 7)
                {
                    blockmap[0, i] = new Water(i, 0, waterimage, screen, new
Random().Next(2) == 1 ? true : false);
                    itemmap[0, i] = new Coin(i, 0, coinlist.Images[0], screen, coinlist);
                }
                else if (templates[index][i] == 8)
                {
                    blockmap[0, i] = new Road(i, 0, roadimage, screen, new
Random().Next(2) == 1 ? true : false);
                    itemmap[0, i] = new Coin(i, 0, coinlist.Images[0], screen, coinlist);
                }
                else if (templates[index][i] == 9)
                {
                    blockmap[0, i] = new Train_road(i, 0, trainroadimage, screen, new
Random().Next(2) == 1 ? true : false, redtrainroadimage);
                    itemmap[0, i] = new Coin(i, 0, coinlist.Images[0], screen, coinlist);
                }
            }
```

```
        }

        public void CheckForMove(ref Message msg)
        {
            if (msg.key == "KeyUp") msg.roundy--;
            else if (msg.key == "KeyDown") msg.roundy++;
            else if (msg.key == "KeyLeft") msg.roundx--;
            else if (msg.key == "KeyRight") msg.roundx++;

            if (msg.roundx >= 0 && msg.roundx < blockmap.GetLength(1) &&
msg.roundy >= 0 && msg.roundy < blockmap.GetLength(0))
            {
                if (itemmap[msg.roundy, msg.roundx] != null) itemmap[msg.roundy,
msg.roundx].Collision(msg);
                else msg.key = "";
            }
            else msg.key = "STOP";
        }
    }
}
```

Menu.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections.Generic;

namespace Легушка
{
    [Serializable]
    class Menu
    {
        [NonSerialized]
        Button play;
        [NonSerialized]
        Button best;
        [NonSerialized]
        Button store;
        [NonSerialized]
        Button info;
        [NonSerialized]
        Button exit;
        [NonSerialized]
        Button back;
        [NonSerialized]
```

```
        Button deletebase;
        [NonSerialized]
        Button t;
        public MenuMessage msg;
        [NonSerialized]
        public Form1 form;
        [NonSerialized]
        Image background;
        [NonSerialized]
        Image buttonimage;
        [NonSerialized]
        Image deleteimage;
        [NonSerialized]
        Image infoimage;
        public LeaderBoard leaders;
        public Shop shop;

        public Menu(MenuMessage msg, Form1 form, string[] filebuf)
        {
            this.msg = msg;
            this.form = form;
            leaders = new LeaderBoard(form, filebuf);
            shop = new Shop(msg, form, filebuf);

            background = Properties.Resources.background;
            buttonimage = Properties.Resources.button;
            deleteimage = Properties.Resources.trashbin;
            infoimage = Properties.Resources.info;

            play = new Button();
            play.BackgroundImage = buttonimage;
            play.FlatAppearance.BorderSize = 0;
            play.FlatStyle = FlatStyle.Flat;
            play.BackColor = Color.Transparent;
            play.FlatAppearance.MouseDownBackColor =
Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            play.FlatAppearance.MouseOverBackColor =
Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            play.Text = "Новая игра";
            play.Font = new Font("Arial", 20.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            play.Location = new Point(138, 200);
            play.Size = new Size(300, 50);
```

```
            play.TabIndex = 0;
            play.UseVisualStyleBackColor = true;
            play.Click += new System.EventHandler(play_click);

            best = new Button();
            best.BackgroundImage = buttonimage;
            best.FlatAppearance.BorderSize = 0;
            best.FlatStyle = FlatStyle.Flat;
            best.BackColor = Color.Transparent;
            best.FlatAppearance.MouseDownBackColor =
Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            best.FlatAppearance.MouseOverBackColor =
Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            best.Text = "Лидеры";
            best.Font = new Font("Arial", 20.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            best.Location = new Point(138, 300);
            best.Size = new Size(300, 50);
            best.TabIndex = 0;
            best.UseVisualStyleBackColor = true;
            best.Click += new System.EventHandler(best_click);

            store = new Button();
            store.BackgroundImage = buttonimage;
            store.FlatAppearance.BorderSize = 0;
            store.FlatStyle = FlatStyle.Flat;
            store.BackColor = System.Drawing.Color.Transparent;
            store.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            store.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            store.Text = "Магазин";
            store.Font = new Font("Arial", 20.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            store.Location = new Point(138, 400);
            store.Size = new Size(300, 50);
            store.TabIndex = 0;
            store.UseVisualStyleBackColor = true;
            store.Click += new System.EventHandler(store_click);

            info = new Button();
```

```
            info.BackgroundImage = buttonimage;
            info.FlatAppearance.BorderSize = 0;
            info.FlatStyle = FlatStyle.Flat;
            info.BackColor = System.Drawing.Color.Transparent;
            info.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            info.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            info.Text = "Инфо";
            info.Font = new Font("Arial", 20.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            info.Location = new Point(138, 500);
            info.Size = new Size(300, 50);
            info.TabIndex = 0;
            info.UseVisualStyleBackColor = true;
            info.Click += new System.EventHandler(info_click);

            exit = new Button();
            exit.BackgroundImage = buttonimage;
            exit.FlatAppearance.BorderSize = 0;
            exit.FlatStyle = FlatStyle.Flat;
            exit.BackColor = System.Drawing.Color.Transparent;
            exit.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            exit.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            exit.Text = "Выход";
            exit.Font = new Font("Arial", 20.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            exit.Location = new Point(138, 600);
            exit.Size = new Size(300, 50);
            exit.TabIndex = 0;
            exit.UseVisualStyleBackColor = true;
            exit.Click += new System.EventHandler(exit_click);

            back = new Button();
            back.BackgroundImage = Image.FromFile("img/arrow.png");
            back.FlatAppearance.BorderSize = 0;
            back.FlatStyle = FlatStyle.Flat;
            back.BackColor = System.Drawing.Color.Transparent;
```

```
            back.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            back.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            back.Text = "";
            back.Font = new Font("Arial", 20.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            back.Location = new Point(20, 697);
            back.Size = new Size(50, 50);
            back.TabIndex = 0;
            back.UseVisualStyleBackColor = true;
            back.Click += new System.EventHandler(back_click);

            deletebase = new Button();
            deletebase.BackgroundImage = deleteimage;
            deletebase.FlatAppearance.BorderSize = 0;
            deletebase.FlatStyle = FlatStyle.Flat;
            deletebase.BackColor = Color.Transparent;
            deletebase.FlatAppearance.MouseDownBackColor =
Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            deletebase.FlatAppearance.MouseOverBackColor =
Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            deletebase.Text = "";
            deletebase.Font = new Font("Arial", 20.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            deletebase.Location = new Point(512, 704);
            deletebase.Size = new Size(32, 32);
            deletebase.TabIndex = 0;
            deletebase.UseVisualStyleBackColor = true;
            deletebase.Click += new System.EventHandler(deletebase_click);

            t = new Button();
            t.Size = new Size(0, 0);
            t.Location = new Point(-100, -100);

            form.Controls.Add(play);
            form.Controls.Add(best);
            form.Controls.Add(store);
            form.Controls.Add(info);
            form.Controls.Add(exit);
            form.Controls.Add(back);
```

```
    form.Controls.Add(deletebase);
    form.Controls.Add(t);

    back.Hide();
}



public string[] GetLeaders()
{
    return leaders.GetInfo();
}

public string[] GetCosts()
{
    return shop.GetCosts();
}

public void Draw()
{
    form.BackgroundImage = background;
    play.Show();
    best.Show();
    store.Show();
    info.Show();
    exit.Show();
    deletebase.Show();
    back.Hide();
}

public void SetFocus()
{
    t.Focus();
}

public void Hide()
{
    play.Hide();
    best.Hide();
    store.Hide();
    info.Hide();
    exit.Hide();
    deletebase.Hide();
    back.Focus();
    form.Refresh();
```

```
}
private void play_click(object sender, EventArgs e)
{
   msg.activegame = false;
   msg.menuchoice = 1;
   Hide();
}

private void best_click(object sender, EventArgs e)
{
   back.Show();
   Hide();
   leaders.Draw();
   msg.menuchoice = 2;
}

private void store_click(object sender, EventArgs e)
{
   back.Show();
   Hide();
   shop.Middle();
   shop.DrawButtons();
   msg.menuchoice = 3;
}

private void info_click(object sender, EventArgs e)
{
   msg.menuchoice = 4;
   Hide();
   form.CreateGraphics().DrawImage(infoimage, 0, 0);
   back.Show();
}

private void exit_click(object sender, EventArgs e)
{
   msg.menuchoice = 5;
}
private void deletebase_click(object sender, EventArgs e)
{
   msg.menuchoice = 6;
}
private void back_click(object sender, EventArgs e)
{
   msg.menuchoice = 0;
   shop.Hide();
```

```
            Draw();
        }
    }
}
```

Game.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

namespace Легушка
{
    [Serializable]
    class Game
    {
        protected Message msg;
        protected Map map;
        protected Frog frog;
        [NonSerialized]
        protected Form1 form;
        [NonSerialized]
        protected Bitmap screen;
        protected bool dead;
        protected int score;
        [NonSerialized]
        protected Timer t, t2;
        protected Menu menu;
        protected MenuMessage menumsg;
        protected int coordinate;
        [NonSerialized]
        protected TextBox input;
        [NonSerialized]
        protected List<ImageList> img;
        [NonSerialized]
        protected Graphics gscreen, gform;
        [NonSerialized]
        protected Image plateimage;
        protected string[] filebuf;

        public Game(Form1 form)
        {
            menumsg = new MenuMessage();
```

```
filebuf = File.ReadAllLines("data.txt");
menumsg.coins = Convert.ToInt32(filebuf[0]);
menumsg.skinchoice = Convert.ToInt32(filebuf[filebuf.Length - 1]);

gform = form.CreateGraphics();
screen = new Bitmap(576, 768);
gscreen = Graphics.FromImage(screen);
plateimage = Properties.Resources.plate;
this.form = form;
msg = new Message();
msg.Shrink();
map = new Map(form, screen);
img = new List<ImageList>();
img.Add(new ImageList());
img[0].ImageSize = new Size(64, 64);
img[0].Images.Add(Properties.Resources.frogt);
img[0].Images.Add(Properties.Resources.frogr);
img[0].Images.Add(Properties.Resources.frogb);
img[0].Images.Add(Properties.Resources.frogl);
img[0].Images.Add(Properties.Resources.frogjumpt);
img[0].Images.Add(Properties.Resources.frogjumpr);
img[0].Images.Add(Properties.Resources.frogjumpb);
img[0].Images.Add(Properties.Resources.frogjumpl);
img[0].Images.Add(Properties.Resources.frogdeadt);
img[0].Images.Add(Properties.Resources.frogdeadr);
img[0].Images.Add(Properties.Resources.frogdeadb);
img[0].Images.Add(Properties.Resources.frogdeadl);

img.Add(new ImageList());
img[1].ImageSize = new Size(64, 64);
img[1].Images.Add(Properties.Resources.redfrogt);
img[1].Images.Add(Properties.Resources.redfrogr);
img[1].Images.Add(Properties.Resources.redfrogb);
img[1].Images.Add(Properties.Resources.redfrogl);
img[1].Images.Add(Properties.Resources.redfrogjumpt);
img[1].Images.Add(Properties.Resources.redfrogjumpr);
img[1].Images.Add(Properties.Resources.redfrogjumpb);
img[1].Images.Add(Properties.Resources.redfrogjumpl);
img[1].Images.Add(Properties.Resources.redfrogdeadt);
img[1].Images.Add(Properties.Resources.redfrogdeadr);
img[1].Images.Add(Properties.Resources.redfrogdeadb);
img[1].Images.Add(Properties.Resources.redfrogdeadl);

img.Add(new ImageList());
```

```
            img[2].ImageSize = new Size(64, 64);
            img[2].Images.Add(Properties.Resources.bluefrogt);
            img[2].Images.Add(Properties.Resources.bluefrogr);
            img[2].Images.Add(Properties.Resources.bluefrogb);
            img[2].Images.Add(Properties.Resources.bluefrogl);
            img[2].Images.Add(Properties.Resources.bluefrogjumpt);
            img[2].Images.Add(Properties.Resources.bluefrogjumpr);
            img[2].Images.Add(Properties.Resources.bluefrogjumpb);
            img[2].Images.Add(Properties.Resources.bluefrogjumpl);
            img[2].Images.Add(Properties.Resources.bluefrogdeadt);
            img[2].Images.Add(Properties.Resources.bluefrogdeadr);
            img[2].Images.Add(Properties.Resources.bluefrogdeadb);
            img[2].Images.Add(Properties.Resources.bluefrogdeadl);

            img.Add(new ImageList());
            img[3].ImageSize = new Size(64, 64);
            img[3].Images.Add(Properties.Resources.kingt);
            img[3].Images.Add(Properties.Resources.kingr);
            img[3].Images.Add(Properties.Resources.kingb);
            img[3].Images.Add(Properties.Resources.kingl);
            img[3].Images.Add(Properties.Resources.kingjumpt);
            img[3].Images.Add(Properties.Resources.kingjumpr);
            img[3].Images.Add(Properties.Resources.kingjumpb);
            img[3].Images.Add(Properties.Resources.kingjumpl);
            img[3].Images.Add(Properties.Resources.kingdeadt);
            img[3].Images.Add(Properties.Resources.kingdeadr);
            img[3].Images.Add(Properties.Resources.kingdeadb);
            img[3].Images.Add(Properties.Resources.kingdeadl);

            frog = new Frog(256, 384, img[menumsg.skinchoice], screen);
            dead = false;
            score = 0;
            t = new Timer();
            t.Interval = 16;
            t.Tick += new EventHandler(t_tick);
            t.Start();
            t2 = new Timer();
            t2.Interval = 100;
            t2.Tick += new EventHandler(t2_tick);
            this.form.KeyDown += new
System.Windows.Forms.KeyEventHandler(KeyDown);
            menu = new Menu(menumsg, form, filebuf);

            input = new TextBox();
            input.BorderStyle = BorderStyle.None;
```

```
            input.TextAlign = HorizontalAlignment.Center;
            input.MaxLength = 9;
            input.Text = null;
            input.Font = new Font("Arial", 20.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point);
            input.Location = new Point(148, 300);
            input.Size = new Size(280, 70);
            input.TabIndex = 0;
            input.Hide();
            form.Controls.Add(input);

            this.form.FormClosed += Form_FormClosed;
            DeSerialization();
        }

        private void Form_FormClosed(object sender, FormClosedEventArgs e)
        {
            filebuf[0] = Convert.ToString(menumsg.coins);
            string[] costs = menu.GetCosts();
            string[] leaders = menu.GetLeaders();
            for (int i = 1; i < 9; i++) filebuf[i] = costs[i - 1];
            for (int i = 9; i < 29; i++) filebuf[i] = leaders[i - 9];
            filebuf[29] = Convert.ToString(menumsg.skinchoice);
            File.WriteAllLines("data.txt", filebuf);
            if (score != 0)
            {
                Serialization();
            }
            else
            {
                File.Delete("game.dat");
            }
        }

        public void Serialization()
        {
            BinaryFormatter formatter = new BinaryFormatter();
            using (FileStream fs = new FileStream("game.dat",
FileMode.OpenOrCreate))
            {
                formatter.Serialize(fs, this);
            }
        }
        public void DeSerialization()
        {
```

```
    BinaryFormatter formatter = new BinaryFormatter();
    if (System.IO.File.Exists("game.dat"))
    {
        Game gameHelper;
        using (FileStream fs = new FileStream("game.dat", FileMode.Open))
        {
            if (fs.Length != 0)
            {
                gameHelper = (Game)formatter.Deserialize(fs);
                this.dead = gameHelper.dead;
                if (this.dead)
                {
                    this.input.Show();
                }
                this.score = gameHelper.score;
                this.map = gameHelper.map;
                this.map.Refresh(form, screen);
                this.frog = gameHelper.frog;
                this.menumsg.skinchoice = gameHelper.menumsg.skinchoice;
                this.menu.msg.skinchoice = gameHelper.menu.msg.skinchoice;
                frog.Refresh(img[menumsg.skinchoice], screen);
                this.menumsg.menuchoice = 1;
                this.menumsg.activegame = true;
                this.menu.msg.menuchoice = 1;
                this.menu.msg.activegame = true;
                this.menu.Hide();
            }
        }
    }
}

protected void t_tick(object sender, EventArgs eArgs)
{
    Play();
}

protected void t2_tick(object sender, EventArgs eArgs)
{
    t2.Stop();
}

protected void KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.W && !t2.Enabled)
    {
```

```
                Event("KeyUp");
            }
            else if (e.KeyCode == Keys.A && !t2.Enabled)
            {
                Event("KeyLeft");
            }
            else if (e.KeyCode == Keys.S && !t2.Enabled)
            {
                Event("KeyDown");
            }
            else if (e.KeyCode == Keys.D && !t2.Enabled)
            {
                Event("KeyRight");
            }
            else if (e.KeyCode == Keys.Space && !t2.Enabled)
            {
                Event("Space");
            }
            else if (e.KeyCode == Keys.Enter && !t2.Enabled)
            {
                Event("Enter");
            }
            else if (e.KeyCode == Keys.Escape && !t2.Enabled)
            {
                Event("Escape");
            }
            t2.Start();
        }

        protected void Draw()
        {
            if (menumsg.menuchoice == 0)
            {
                menu.Draw();
            }
            else if (menumsg.menuchoice == 1)
            {
                map.Draw();
                frog.Draw();
                if (dead) map.DrawMoving();
                gscreen.DrawString("Счет: " + score.ToString(), new Font(new
FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black), new
Point(0, 0));
                coordinate = 0;
```

```
                for (int i = menumsg.coins.ToString().Length; i < 5; i++)
coordinate++;
                gscreen.DrawString("Монет: " + menumsg.coins.ToString(), new
Font(new FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black),
new Point(392 + coordinate * 15, 0));

                if (input.Visible)
                {
                   gscreen.DrawImage(plateimage, new Point(128, 210));
                   gscreen.DrawString("Новый рекорд!", new Font(new
FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black), new
Point(180, 220));
                   gscreen.DrawString("Введите своё имя:", new Font(new
FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black), new
Point(160, 250));
                }

                gform.DrawImage(screen, new Point(0, 0));
             }
             else if (menumsg.menuchoice == 6)
             {
                gscreen.DrawImage(plateimage, new Point(128, 210));
                gscreen.DrawString("Все данные удалятся!", new Font(new
FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black), new
Point(134, 220));
                gscreen.DrawString("Введите пароль:", new Font(new
FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black), new
Point(165, 250));
                gform.DrawImage(screen, new Point(0, 0));
             }
          }

          protected void Collision()
          {
             frog.GetCoordinates(msg);
             map.Collision(msg, ref menumsg.coins);
             if (msg.key == "DIE")
             {
                dead = true;
                if (!input.Visible && menu.leaders.Check(score))
                {
                   input.Text = null;
                   input.Show();
                   input.Focus();
                }
```

```
            }
            frog.Event(msg);
        }

        protected void Shrink()
        {
            dead = false;
            map.Shrink();
            frog.Shrink(img[menumsg.skinchoice]);
            score = 0;
        }

        protected void Play()
        {
            if (menumsg.menuchoice == 1 && menumsg.activegame == true)
            {
                map.Move();
                Collision();
            }
            else if (menumsg.menuchoice == 1 && menumsg.activegame == false)
            {
                menumsg.activegame = true;
                Shrink();
            }
            Draw();

            if (menumsg.menuchoice == 6 && !input.Visible)
            {
                menu.Hide();
                input.Text = null;
                input.Show();
                input.Focus();
                form.Refresh();
            }
            if (menumsg.menuchoice == 5)
            {
                form.Close();
            }
        }

        protected void Event(string str)
        {
            if (dead == false && (str == "KeyUp" || str == "KeyDown" || str ==
"KeyLeft" || str == "KeyRight"))
            {
```

```
        frog.GetCoordinates(msg);
        msg.key = str;
        map.CheckForMove(ref msg);
        if (msg.key != "STOP")
        {
            frog.GetCoordinates(msg);
            if (msg.roundy == 6 && str == "KeyUp")
            {
                score++;
                map.Update();
                frog.Round();
                frog.Direction();
            }
            else
            {
                msg.key = str;
                frog.Event(msg);
            }
        }
    }
    else if (menumsg.menuchoice == 6 && input.Visible && str == "Enter")
    {
        if (input.Text == "stimsly")
        {
            menumsg.coins = 0;
            filebuf[0] = Convert.ToString(menumsg.coins);
            menu.leaders.DeleteBase();
            menu.shop.DeleteBase();
            filebuf[0] = Convert.ToString(menumsg.coins);
            string[] costs = menu.GetCosts();
            string[] leaders = menu.GetLeaders();
            for (int i = 1; i < 9; i++) filebuf[i] = costs[i - 1];
            for (int i = 9; i < 28; i++) filebuf[i] = leaders[i - 9];
            filebuf[28] = "0";
            File.WriteAllLines("data.txt", filebuf);
            menumsg.menuchoice = 0;
            form.Refresh();
            menumsg.activegame = false;
            menumsg.skinchoice = 0;
            input.Hide();
            menu.SetFocus();
            Shrink();
        }
        else
        {
```

```
                input.Text = "Неверно!";
            }
        }
        else if (menumsg.menuchoice == 6 && input.Visible && str ==
"Escape")
        {
            input.Hide();
            menu.SetFocus();
            if (menumsg.activegame == true) menumsg.menuchoice = 1;
            else
            {
                menumsg.menuchoice = 0;
            }
            form.Refresh();
        }
        else if (!input.Visible && dead == true && str == "Space")
        {
            Shrink();
        }
        else if (input.Visible && str == "Enter")
        {
            menu.leaders.Add(input.Text, score);
            input.Hide();
            menu.SetFocus();
            Shrink();
        }
        else if (str == "Escape" && menumsg.menuchoice == 0 &&
menumsg.activegame == true && !t2.Enabled)
        {
            menu.Hide();
            menumsg.menuchoice = 1;
        }
        else if (str == "Escape" && menumsg.menuchoice == 1 &&
!t2.Enabled)
        {
            menumsg.menuchoice = 0;
            form.Refresh();
            menumsg.activegame = true;
        }
        else if (str == "Escape" && menumsg.menuchoice == 0 &&
menumsg.activegame == false && !t2.Enabled)
        {
            menumsg.menuchoice = 5;
        }
    }
```

```
    }
}
```

LeaderBoard.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.Drawing;

namespace Легушка
{
    [Serializable]
    class LeaderBoard
    {
        protected string[] names;
        protected int[] scores;
        [NonSerialized]
        protected Graphics g;

        public LeaderBoard(Form1 form, string[] filebuf)
        {
            names = new string[10];
            scores = new int[10];
            Array.Fill<int>(scores, 0);

            for (int i = 9; i < 28; i+=2)
            {
                names[(i - 9) / 2] = filebuf[i];
                scores[(i - 8) / 2] = Convert.ToInt32(filebuf[i + 1]);
            }

            this.g = form.CreateGraphics();
        }

        public bool Check(int score)
        {
            if (score > scores[9]) return true;
            return false;
        }

        public void DeleteBase()
        {
            for (int i = 0; i < 10;i++)
            {
```

```
            names[i] = "";
            scores[i] = 0;
        }
    }

    public string[] GetInfo()
    {
        string[] s = new string[20];
        for (int i = 0; i < 20; i += 2)
        {
            s[i] = names[i / 2];
            s[i + 1] = Convert.ToString(scores[i / 2]);
        }
        return s;
    }

    public void Add(string name, int score)
    {
        for (int i = 0; i < 10; i++)
        {
            if (score > scores[i] || i == 9)
            {
                for (int j = 9; j > i; j--)
                {
                    scores[j] = scores[j - 1];
                    names[j] = names[j - 1];
                }
                if (name.Length == 0) name = "аноним";
                names[i] = name;
                scores[i] = score;
                break;
            }
        }
    }

    public void Draw()
    {
        g.DrawImage(Properties.Resources.leaders, 0, 0);
        for (int i = 0; i < 10; i++)
        {
            if (scores[i] != 0)
            {
                g.DrawString(names[i], new Font(new FontFamily("Arial"), 20,
FontStyle.Bold), new SolidBrush(Color.Black), new Point(120, 200 + i * 50));
```

```
                    g.DrawString(scores[i].ToString(), new Font(new
FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black), new
Point(400, 200 + i * 50));
                }
            }
        }
    }
}
```

Shop.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace Легушка
{
    [Serializable]
    class Shop
    {
        [NonSerialized]
        List<Image> images2;
        [NonSerialized]
        List<Image> images4;
        [NonSerialized]
        Button goleft;
        [NonSerialized]
        Button goright;
        [NonSerialized]
        Button choose;
        int l, m, r;
        [NonSerialized]
        Form1 form;
        MenuMessage msg;
        [NonSerialized]
        Graphics g;
        int[] costs;
        bool[] bought;

        public Shop(MenuMessage msg, Form1 form, string[] costs)
        {
            this.form = form;
```

```
            this.msg = msg;

            this.costs = new int[4];
            this.bought = new bool[4];
            for (int i = 0; i < 4; i++) this.costs[i] = Convert.ToInt32(costs[i + 1]);
            for (int i = 0; i < 4; i++) this.bought[i] =
Convert.ToBoolean(Convert.ToInt32(costs[i + 5]));
            images2 = new List<Image>();
            images4 = new List<Image>();
            images2.Add(Properties.Resources.frogt2);
            images2.Add(Properties.Resources.redfrogt2);
            images2.Add(Properties.Resources.bluefrogt2);
            images2.Add(Properties.Resources.kingt2);
            images4.Add(Properties.Resources.frogt4);
            images4.Add(Properties.Resources.redfrogt4);
            images4.Add(Properties.Resources.bluefrogt4);
            images4.Add(Properties.Resources.kingt4);

            g = form.CreateGraphics();

            Middle();

            goleft = new Button();
            goleft.BackgroundImage = Properties.Resources.arrow;
            goleft.FlatAppearance.BorderSize = 0;
            goleft.FlatStyle = FlatStyle.Flat;
            goleft.BackColor = System.Drawing.Color.Transparent;
            goleft.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            goleft.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            goleft.Text = "";
            goleft.Font = new Font("Arial", 20.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            goleft.Location = new Point(119, 500);
            goleft.Size = new Size(50, 50);
            goleft.TabIndex = 0;
            goleft.UseVisualStyleBackColor = true;
            goleft.Click += new System.EventHandler(goleft_click);

            goright = new Button();
            goright.BackgroundImage = Properties.Resources.arrowreverse;
            goright.FlatAppearance.BorderSize = 0;
```

```
            goright.FlatStyle = FlatStyle.Flat;
            goright.BackColor = System.Drawing.Color.Transparent;
            goright.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            goright.FlatAppearance.MouseOverBackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            goright.Text = "";
            goright.Font = new Font("Arial", 20.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            goright.Location = new Point(407, 500);
            goright.Size = new Size(50, 50);
            goright.TabIndex = 0;
            goright.UseVisualStyleBackColor = true;
            goright.Click += new System.EventHandler(goright_click);

            choose = new Button();
            choose.BackgroundImage = Properties.Resources.button;
            choose.FlatAppearance.BorderSize = 0;
            choose.FlatStyle = FlatStyle.Flat;
            choose.BackColor = Color.Transparent;
            choose.FlatAppearance.MouseDownBackColor =
Color.FromArgb(((int)(((byte)(50)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            choose.FlatAppearance.MouseOverBackColor =
Color.FromArgb(((int)(((byte)(25)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            choose.Text = "Выбрать";
            choose.Font = new Font("Arial", 20.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);
            choose.Location = new Point(138, 500);
            choose.Size = new Size(300, 50);
            choose.TabIndex = 0;
            choose.UseVisualStyleBackColor = true;
            choose.Click += new System.EventHandler(choose_click);

            form.Controls.Add(choose);
            form.Controls.Add(goleft);
            form.Controls.Add(goright);
            Hide();
        }

        public string[] GetCosts()
        {
```

```
    string[] s = new string[costs.Length * 2];
    for (int i = 0; i < costs.Length; i++)
    {
        s[i] = Convert.ToString(costs[i]);
        s[i + 4] = Convert.ToString(Convert.ToInt32(bought[i]));
    }
    return s;
}
public void DeleteBase()
{
    for (int i = 1; i < bought.Length; i++)
    {
        bought[i] = false;
    }
}
private void choose_click(object sender, EventArgs e)
{
    if (bought[m] == true)
    {
        msg.skinchoice = m;
        msg.menuchoice = 0;
        Hide();
    }
    else if (bought[m] == false && msg.coins >= costs[m])
    {
        bought[m] = true;
        msg.coins -= costs[m];
        choose.Text = "Выбрать";
        form.Refresh();
        Draw();
    }
}

private void goright_click(object sender, EventArgs e)
{
    l++; m++; r++;
    if (l == images2.Count) l = 0;
    else if (m == images2.Count) m = 0;
    else if (r == images2.Count) r = 0;
    if (bought[m] == true) choose.Text = "Выбрать";
    else choose.Text = costs[m].ToString() + " монет";
    Draw();
}

private void goleft_click(object sender, EventArgs e)
```

```
        {
            l--; m--; r--;
            if (l < 0) l = images2.Count - 1;
            else if (m < 0) m = images2.Count - 1;
            else if (r < 0) r = images2.Count - 1;
            if (bought[m] == true) choose.Text = "Выбрать";
            else choose.Text = costs[m].ToString() + " монет";
            Draw();
        }

        public void Middle()
        {
            m = msg.skinchoice;
            l = m - 1;
            r = m + 1;
            if (l < 0) l = images2.Count - 1;
            if (r == images2.Count) r = 0;
        }

        public void DrawButtons()
        {
            choose.Text = "Выбрать";
            goleft.Show();
            goright.Show();
            choose.Show();
            Draw();
        }

        public void Draw()
        {
            form.Refresh();
            g.DrawImage(images2[l], new Point(16, 328));
            g.DrawImage(images4[m], new Point(160, 200));
            g.DrawImage(images2[r], new Point(432, 328));
            g.DrawString("Монет: " + msg.coins.ToString(), new Font(new
FontFamily("Arial"), 20, FontStyle.Bold), new SolidBrush(Color.Black), new
Point(400, 707));
        }

        public void Hide()
        {
            choose.Hide();
            goleft.Hide();
            goright.Hide();
            form.Refresh();
```

```
        }
    }
}
```

Message.cs

```
using System;
using System.Drawing;

namespace Легушка
{
    [Serializable]
    class Message
    {
        public string key;
        public int x, y;
        public int roundx, roundy;
        public Rectangle rect;
        public int movingspeed;
        public void Shrink()
        {
            key = "";
            x = -1;
            y = -1;
            roundx = -1;
            roundy = -1;
            movingspeed = 0;
        }
    }
}
```

MenuMessage.cs

```
using System;

namespace Легушка
{
    [Serializable]
    class MenuMessage
    {
        public int menuchoice;
        public bool activegame;
        public int skinchoice;
        public int coins;
        public MenuMessage()
        {
            menuchoice = 0;
```

```
            activegame = false;
            skinchoice = 0;
            coins = 0;
        }
    }
}
```