Министерство образования и науки РФ

Государственное бюджетное образовательное учреждение высшего профессионального образования

Ивановский государственный энергетический университет им В.И Ленина

Кафедра ПОКС

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**для курсовой работы по курсу**

**«Объектно-ориентированное программирование»**

**Тема:**

**«Разработка игры "Doodle Jump"»**

Работу выполнил студент

2 курса 41 группы

Дао Минь Хоанг

Проверила: Алыкова А.Л.

Иваново 2018

**План пояснительной записки**

[Постановка задачи 4](#_Toc515963008)

[Общее описание задачи 4](#_Toc515963009)

[Техническое задание 4](#_Toc515963010)

[Название программного продукта 4](#_Toc515963011)

[Основание для разработки 4](#_Toc515963012)

[Требования к системе в целом 5](#_Toc515963013)

[Диаграмма вариантов использования 5](#_Toc515963014)

[Требования к функциональным характеристикам 5](#_Toc515963015)

[Требования к математическому обеспечению 6](#_Toc515963016)

[Требования к информационному обеспечению 6](#_Toc515963017)

[Требования к лингвистическому обеспечению 6](#_Toc515963018)

[Требования к программному обеспечению 6](#_Toc515963019)

[Требования к техническому обеспечению 6](#_Toc515963020)

[Объектно-ориентированный анализ предметной области 7](#_Toc515963021)

[Объектно-ориентированное проектирование 7](#_Toc515963022)

[Разработка статической объектной модели 7](#_Toc515963023)

[Диаграмма классов 7](#_Toc515963024)

[Диаграмма агрегации 8](#_Toc515963025)

[Разработка динамической модели 8](#_Toc515963026)

[Диаграмма последовательности 9](#_Toc515963027)

[Реализация объектов 10](#_Toc515963028)

[class Player 10](#_Toc515963029)

[Раздел private 10](#_Toc515963030)

[Раздел public 10](#_Toc515963031)

[class Platforms 10](#_Toc515963032)

[Раздел private 11](#_Toc515963033)

[Раздел public 11](#_Toc515963034)

[class Green: Platforms 11](#_Toc515963035)

[Раздел public 11](#_Toc515963036)

[class Red: Platforms 11](#_Toc515963037)

[Раздел public 11](#_Toc515963038)

[class Green: Blue 12](#_Toc515963039)

[Раздел public 12](#_Toc515963040)

[class Green: White 12](#_Toc515963041)

[Раздел public 12](#_Toc515963042)

[class Bullet 12](#_Toc515963043)

[Раздел private 12](#_Toc515963044)

[Раздел public 12](#_Toc515963045)

[class Bonus 12](#_Toc515963046)

[Раздел private 12](#_Toc515963047)

[Раздел public 12](#_Toc515963048)

[class Monsters 13](#_Toc515963049)

[Раздел private 13](#_Toc515963050)

[Раздел public 13](#_Toc515963051)

[class Form1: Form 13](#_Toc515963052)

[Раздел private 13](#_Toc515963053)

[Раздел public 14](#_Toc515963054)

[Раздел public 14](#_Toc515963055)

[Руководство пользователя 15](#_Toc515963056)

[Рекомендация по установке 15](#_Toc515963057)

[Инструкция по использованию 15](#_Toc515963058)

[Приложение. Тексты программ 18](#_Toc515963059)

[Class Form1 18](#_Toc515963060)

[Class Player 27](#_Toc515963061)

[Class Platforms 30](#_Toc515963062)

[Class Green 33](#_Toc515963063)

[Class Red 34](#_Toc515963064)

[Class Blue 34](#_Toc515963065)

[Class White 34](#_Toc515963066)

[Class Bonus 34](#_Toc515963067)

[Class Bullet 36](#_Toc515963068)

[Class Monsters 37](#_Toc515963069)

[Class Level 38](#_Toc515963070)

[Class Menu 39](#_Toc515963071)

[Class Scores 40](#_Toc515963072)

[Class GameEnd 43](#_Toc515963073)

[Class CheckRecord 46](#_Toc515963074)

[Class Help 46](#_Toc515963075)

[Class About 46](#_Toc515963076)

# Постановка задачи

## Общее описание задачи

«Doodle Jump» - это игра платформер. Главная цель игры — помочь «Дудлеру» (Doodle) добраться как можно выше по платформам и набрать как можно большее количество очков.

Игрок, прыгая по платформам вверх, должен избегать различные опасности, чтобы добраться как можно выше. Он может, стрелять по врагам, прыгать на них, тем самым убивая, а также использовать бонусы для более высокого прыжка.

В игре 4 типа различных платформ и врагов, а также 2 вида бонусов.

Игра является бесконечной и заканчивается только в том случае, когда пользователь упал с платформ или столкнулся с одним из препятствий.

# Техническое задание

## Название программного продукта

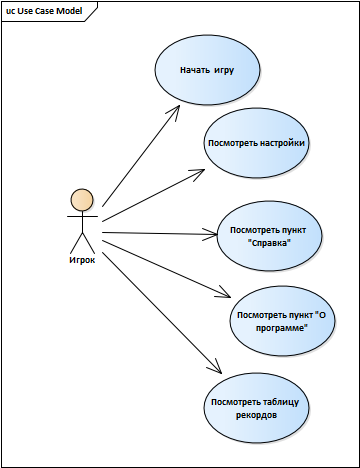
Название "Space Invaders" заимствовано у известной игры, разработанной в 2009 компанией Lima Sky.

## Основание для разработки

Несмотря на то, что игра «Doodle Jump» довольно старая, она до сих пор популярна. Эта игра помогает скоротать время и посоревноваться с друзьями.

## Требования к системе в целом

### Диаграмма вариантов использования



### Требования к функциональным характеристикам

1. На экране должны присутствовать: игрок, платформы, монстры, бонусы и счет игрока.
2. Должна присутствовать возможность передвигать «Дудла» по вертикали, горизонтали и производить выстрелы.
3. На экране всегда должны быть игрок и платформы.
4. При нажатии клавиши "P" все движущиеся объекты останавливаются, и игрок может передохнуть. Чтобы выйти из режима паузы достаточно снова нажать клавишу "P".
5. После проигрыша, игрок может сохранить свой рекорд и посмотреть его в таблице рекордов.
6. При столкновении игрока с монстрами игрок проигрывает.
7. Игрок может уничтожить монстра, прыгнув на него сверху.

### Требования к математическому обеспечению

Сила прыжка зависит от игровой «силы притяжения».

### Требования к информационному обеспечению

При сохранении рекорда в папке с исполнительным файлом появляется бинарный файл, хранящий информацию о результате игрока, а также файл с его именем. Из этого файлов и происходит считывание при запуске игры и загрузке таблицы рекордов.

### Требования к лингвистическому обеспечению

Интерфейс игры написан на английском языке.

### Требования к программному обеспечению

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

### Требования к техническому обеспечению

Для работы программы необходимо 43 Мб свободной оперативной памяти (рекомендуется больше).

Для игры необходима клавиатура и мышь.

# Объектно-ориентированный анализ предметной области

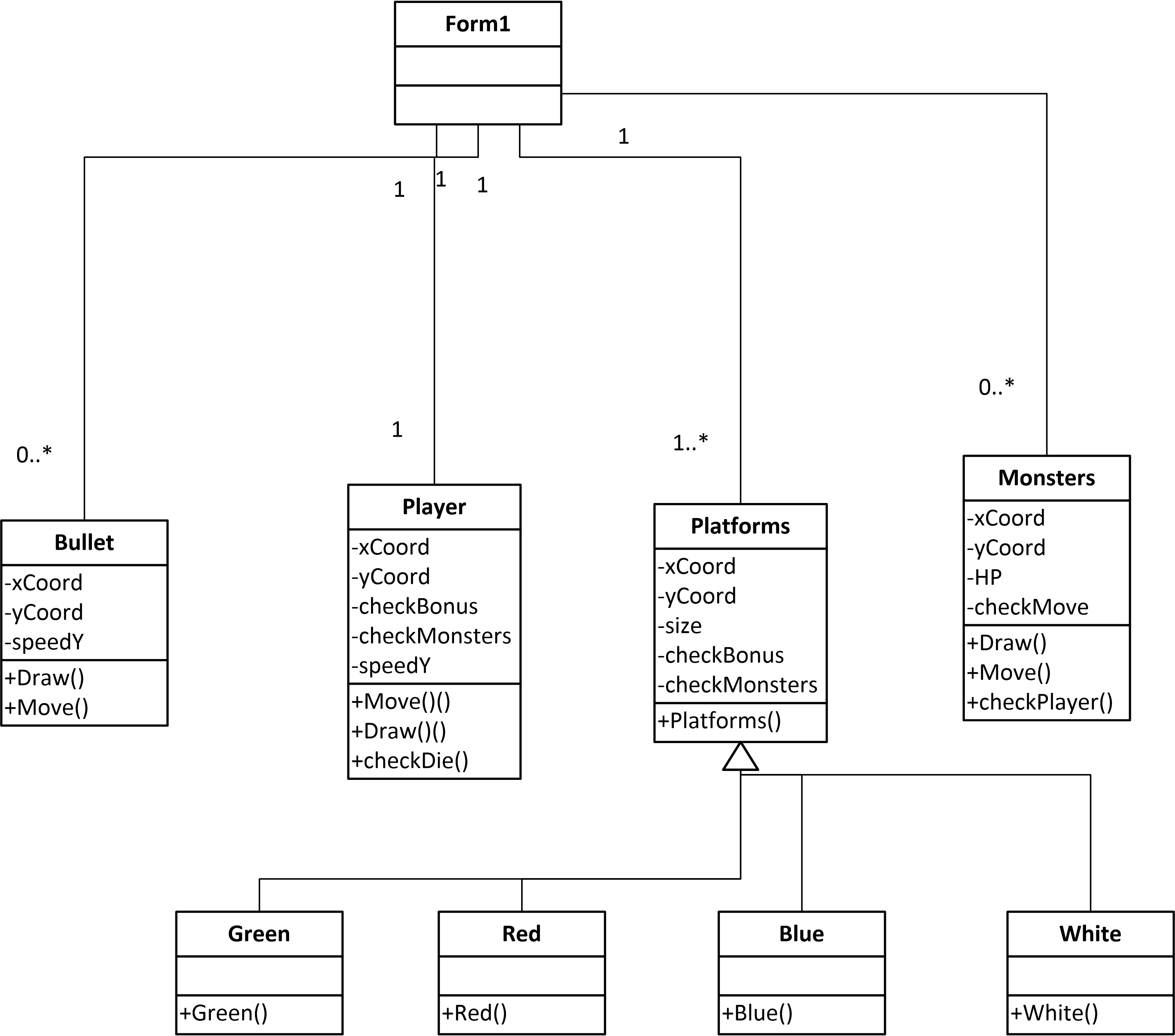
Все элементы, которые мы видим, находятся на полотне. Примем их за объекты. Игрок управляет **персонажем** (дудлом). Дудл прыгает по **платформам**, на которых иногда попадаются **бонусы**. Игрок может стрелять **пулями** по **монстрам**. Для сохранения информации об уровне используется объект **уровень**.

# Объектно-ориентированное проектирование

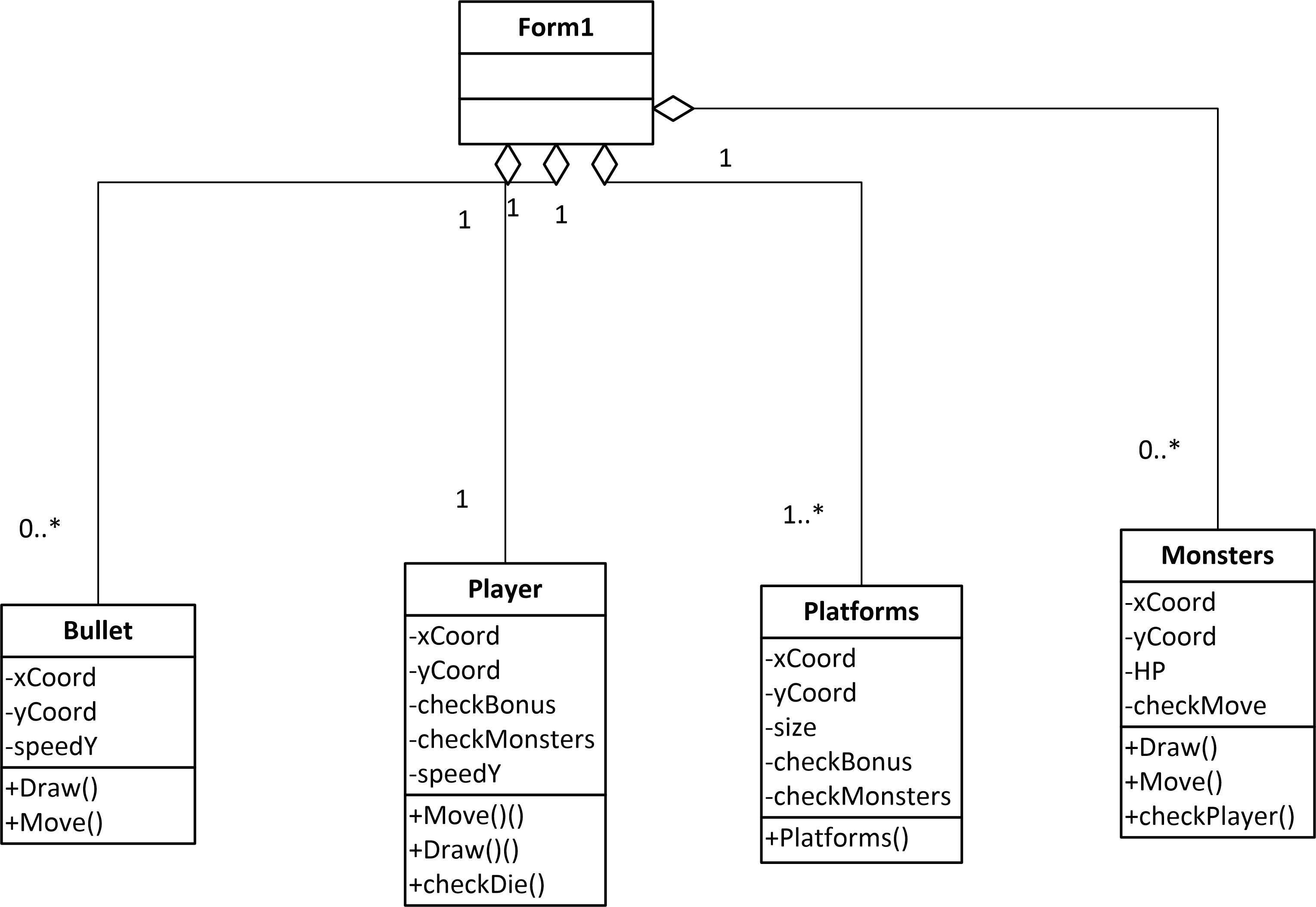
## Разработка статической объектной модели

За полотно примем основной класс Windows Forms, который называется Form. Форма будет содержать в себе некоторые статические объекты WF, такие как: текстовое поле, фоновое изображение и некоторые другие визуальные составляющие, не требующие их вынесения в отдельный класс.

### Диаграмма классов



### Диаграмма агрегации



## Разработка динамической модели

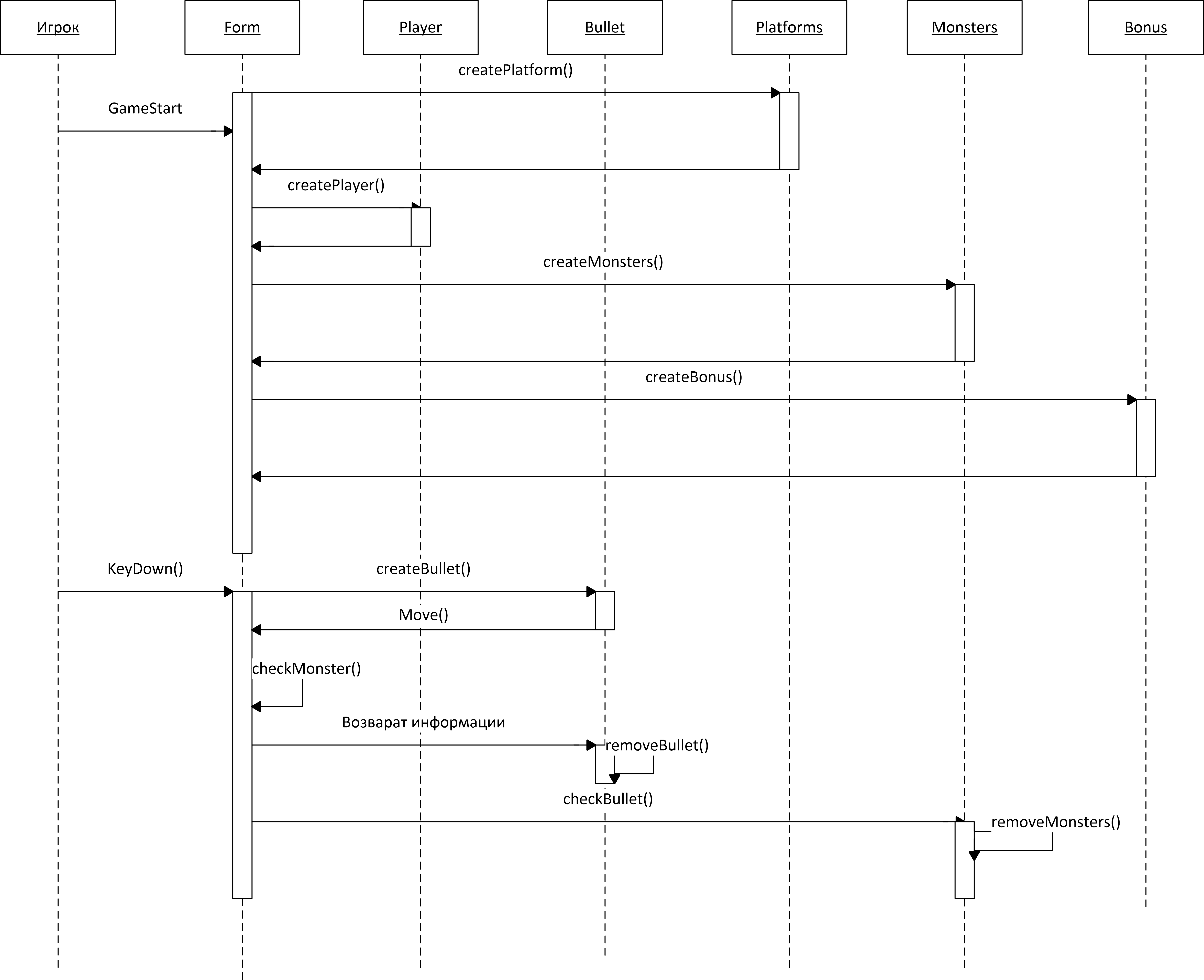
Встроенный в WF обработчик нажатия клавиш посылает форме событие нажатия клавиши, которое обрабатывается определенным образом. При нажатии клавиш стрелок "влево" и "вправо" персонаж игрока передвигается в соответствующую сторону.

При нажатии клавиши "вверх" персонаж производит выстрел пулей, которая движется строго вертикально из точки, в которой находился персонаж во время выстрела.

Передвижение монстров и платформ происходит за счет таймера и random-элемента, которые инициализированы в форме.

При пересечении объекта пули с объектом «монстр», у него отнимается одна жизнь. Как только у монстра станет ноль жизней, он исчезает.

### Диаграмма последовательности



# Реализация объектов

**Основные объекты**

## class Player

Персонаж.

### Раздел private

bool goleft;//флаг влево

bool goright;//флаг вправо

bool isPressed;//флаг нажатия клавиши

bool checkLeftRight;//флаг поворота

bool check;//флаг поворота(скин)

bool checkGun = false;//флаг стрельбы

bool live = true;//флаг жизни

int size = 60;//размер

int speedX = 15;//скороть по горизонтали

int speedY;// скорость по вертикали

int xCoord;// координата х

int yCoord;// координата у

int speed = -23;// сила прыжка

int gravitation = 1;//сила притяжения

Bitmap skin;// скин

Rectangle rectPlayer;// поле персонажа

### Раздел public

public delegate void Method1();//делегат

public delegate void Method2();//делегат

public event Method1 onCheckBonus;//событие пересечения с бонусом

public event Method1 onCheckDie;//событие смерти

public void checkDie()//проверка на смерть

public void checkBonus(int k)//проверка на бонус

public Player(int x, int y, Bitmap sk)//конструктор

public Rectangle PlBounds()//возвращает поле персонажа

public void Draw(Graphics dc) //отрисовка персонажа

public void setSkin(Bitmap sk)//установка скина

public void setCoordX(int x)установка координаты х

public void setCoordY(int y)установка координаты у

public int getCoordX()возвращает координату х

public int getCoordY()возвращает координату у

public int getSpeedY()возвращает скорость по вертикали

public int getSpeedJump()возвращает силу прыжка

public void setSpeedY(int y)установка скорости по вертикали

public void setGravitation(int g)установка гравитации

public bool getCheckGun()проверка на выстрел

public void setLive(bool t)установка жизни/смерти

public bool getLive()возвращает жизнь/смерть

public void checkKeyDown(Form1 form, KeyEventArgs btn)проверка на нажатие

public void checkKeyUp(Form1 form, KeyEventArgs btn)проверка на отпуск клавиши

public void move(Form1 form)движение персонажа

## class Platforms

Платформы, по которым прыгает игрок.

### Раздел private

bool Right = true;//флаг движения по горизонтали

bool Up = true;//флаг движения по вертикали

int HP;//количество прыжков

int speedLeftRight = 5;//скорость по горизонтали

int speedUpDown = 2;//скорость по вертикали

int xCoord;//координата х

int yCoord;//координата у

int sizeX = 60;//ширина

int sizeY = 20;//высота

int UpDown;//точка перемещения вверх-вниз

Bitmap skin;//скин

Rectangle rectPlatform;//поле платформы

int rndDir;//рандомное направление

int checkBonus;//проверка на бонус

int checkMonster;//проверка на монстров

int bonusX;//координата х бонуса

int bonusY;//координата у бонуса

Random rnd = new Random();//рандом

### Раздел public

public Platforms()конструктор

public void Draw(Graphics dc)отрисовка

public void moveLeftRight(Form1 form)движение по горизонтали

public void moveUpDown()по вертикали

public Rectangle PlatBounds()возвращает поле платформы

public void setrndDir(int dir)устанавливает рандомное направление

public void setRect(Rectangle r)устанавливает поле платформы

public void setSkin(Bitmap sk)устанавливает скин

public void setCoordX(int x)устанавливает координату х

public void setCoordY(int y)устанавливает координату у

public void setUpDown(int y)устанавливает координату опоры для вверх-вниз

public void setHP(int hp)устанавливает жизни платформы

public void setBonus(int b)устанавливает бонус на платформу

public void setBonusX(int b)координата х бонуса на платформе

public void setBonusY(int b)координата у бонуса на платформе

public void setMonster(int b)установка монстра на платформу

public int getBonusY()возвращает координату у бонуса

public int getBonusX()возвращает координату х бонуса

public int getBonus()возвращает бонус

public int getMonster()возвращает монстра

public int getHP()возвращает жизни

public int getrndDir()возвращает рандомноне направление

public int getCoordX()возвращает х

public int getCoordY()возвращает у

public int getSizeX()возвращает ширину

public int getSizeY()возвращает высоту

## class Green: Platforms

Зеленая платформа.

### Раздел public

public Green()//конструктор

## class Red: Platforms

Красная платформа.

### Раздел public

public Red()//конструктор

## class Green: Blue

Синяя платформа.

### Раздел public

public Blue()//конструктор

## class Green: White

Белая платформа.

### Раздел public

public White()//конструктор

## class Bullet

Пули.

### Раздел private

int speedY = 20;//скорость пули

int xCoord;//координата х

int yCoord;//координата у

int size = 10;//размер

Rectangle rectBullet;//поле пули

Bitmap skin = Properties.Resources.bullet;//скин

### Раздел public

public void checkDel()проверка на удаление

public Bullet(int x, int y)конструктор

public void Move()движение

public void Draw(Graphics dc)отрисовка

public void setCoordX(int x)установка х

public void setCoordY(int y)установка у

public int getCoordX()возвращает х

public int getCoordY()возвращает у

public Rectangle boundRect()возвращает поле пули

## class Bonus

Бонус.

### Раздел private

int xCoord;//координата х

int yCoord;//координата у

int sizeX;//ширина

int sizeY;//высота

int speedBonus;//бонусная скорость

int checkBonus;//проверка бонуса

int timeSpeed;//время действия

Bitmap skin;//скин

Rectangle rectBonus;//поле бонуса

### Раздел public

public delegate void Method3();//делегат

public event Method3 onCheckPlayer;//событие пересечения с игроком

public void checkPlayer()инициализация события

public Bonus(int f, int x, int y, int speed, int time, int sx, int sy, Bitmap sk)конструктор

public void Draw(Graphics dc)отрисовка

public void setSizeY(int y)установка высоту

public void setSkin(Bitmap sk)установка скина

public void setCoordX(int x)установка координаты х

public void setTimeSpeed(int t)установка времени действия

public void setCoordY(int y)установка координаты у

public int getTimeSpeed()возвращает время действия

public int getSpeedBonus()возвращает бонусную скорость

public int getSizeY()возвращает высоту

public int getCoordX()возвращает координату х

public int getCoordY()возвращает координату у

public int check()проверка бонуса

public Rectangle boundsBonus()возвращает поле бонуса

## class Monsters

### Раздел private

bool Right = true;//флаг движения по горизонтали

int speedLeftRight = 3;//скорость по горизонтали

int xCoord;//координата х

int yCoord;//координата у

int sizeX;//ширина

int sizeY;//высота

int checkMonster;//проверка монстра

int HP;//жизни

bool moveStand;//флаг на движение

Rectangle rectMonster;//поле монстра

Bitmap skin;//скин

### Раздел public

public delegate void Method4();//делегат

public event Method4 onCheckHP;//событие уменьшение жизней

public delegate void Method6();//делегат

public event Method6 onCheckDieMonster;//событие смерти монстра

public void checkDieMon()инициализация события

public void checkHP()инициализация события

public Monsters(int h, int f, int x, int y, int sx, int sy, bool ms, Bitmap s)конструктор

public void Draw(Graphics dc)отрисовка

public void setCoordX(int x)установка х

public void setCoordY(int y)установка у

public void setHP(int hp)установка жизней

public int getCoordX()возвращает х

public int getCoordY()возвращает у

public int getHP()возвращает жизни

public int check()проверка монстра

public Rectangle boundMonster()возвращает поле монстра

public void moveLeftRight(Form1 form)движение по горизнтали

## class Form1: Form

### Раздел private

private Bitmap fon, fonPause;//фон

private Bitmap fon1, fonPause1;//фон

Graphics dc, dcfon1;//графика

PrivateFontCollection custom\_font = new PrivateFontCollection();//шрифт

Player pl;//игрок

List<Platforms> listPlatform = new List<Platforms>();//список платформ

List<Bonus> listBonus = new List<Bonus>();//список бонусов

List<Bullet> listBullet = new List<Bullet>();//список пуль

List<Monsters> listMonsters = new List<Monsters>();//список монстров

Random rnd = new Random();//рандом

int dy = 0;//расстояние по оси у

int rndColor;//рандомный цвет

int g;//джи

int scores = 0;//очки

bool checkStopStart = false;//флаг паузы

Rectangle pause;//поле паузы

Bitmap pauseSkin = Properties.Resources.Pause;//скин паузы

string fileScores = "Scores.txt";//имя файла

string[] scoresfile;//массив результатов

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

private void drawPauseButton()отрисовка паузы

private void FontLoad()проверка на выстрел

private void Bullet\_onCheckDelBullet()событие удаление пули

private void createMonsters(int h, int f, int x, int y, int sx, int sy, bool ms, Bitmap s)создание монстров

private void Monster\_onCheckDieMonster()событие смерть монстра

private void Monster\_onCheckHP()событие уменьшение жизней монстра

private void drawScores(int s)//отрисовка очков

private void checkdy(Platforms pl)проверка расстояний между платформами

private void createPlat()создание платформ

private void moveLevel()движение всей игры

private void deletePlat(int i)удаление платформы

private void deleteBonus(int i)удаление бонуса

private void deleteMonsters(int i)удаление монстра

private void checkPlat()проверка платформ

private void createBonus(int f,int x, int y, int k, int t, int sx, int sy, Bitmap sk)создание бонуса

private void Bonus\_onCheckPlayer()событие пересечение бонуса с игроком

private void createLevel()создание игры

private void Pl\_onCheckDie()событие смерть игрока

private void Pl\_onCheckBonus()подбор бонуса игроком

private void drawLevel()отрисовка игры

private void keyisdown(object sender, KeyEventArgs e)зажата клавиша

private void keyisup(object sender, KeyEventArgs e)не зажата клавиша

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)выход из формы

private void Form1\_MouseMove(object sender, MouseEventArgs e)движение мыши

private void Form1\_MouseClick(object sender, MouseEventArgs e)клик мыши

private void checkEnd()конец игры

### Раздел public

public Form1()конструктор

**Class Level**

### Раздел public

public int score; очки

public Player pl; игрок

public List<Platforms> listPlatform = new List<Platforms>(); список платформ

public List<Bonus> listBonus = new List<Bonus>();список бонусов

public List<Bullet> listBullet = new List<Bullet>(); список пуль

public List<Monsters> listMonsters = new List<Monsters>(); список монстров

public Level(int sc, Player p, List<Platforms> l, List<Bonus> b, List<Bullet> bul, List<Monsters> m) конструктор

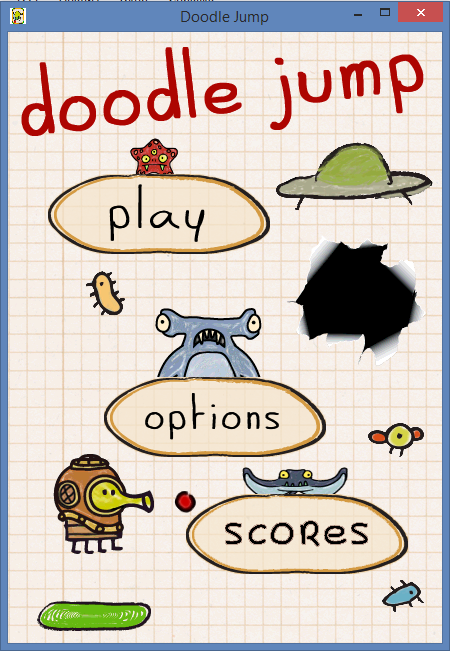
# Руководство пользователя

## Рекомендация по установке

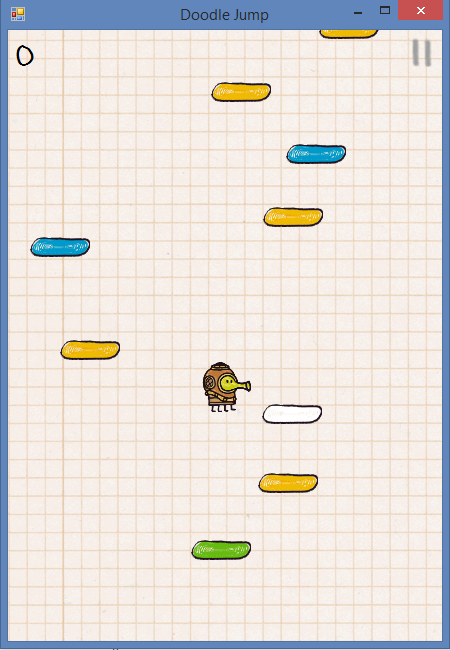
Для работы программы достаточно скопировать в отдельную папку .exe исполнительный файл игры и запустить его. При сохранении результата в папке с исполнительным файлом появится бинарный Scores.bin и Names.bin файл сохранения.

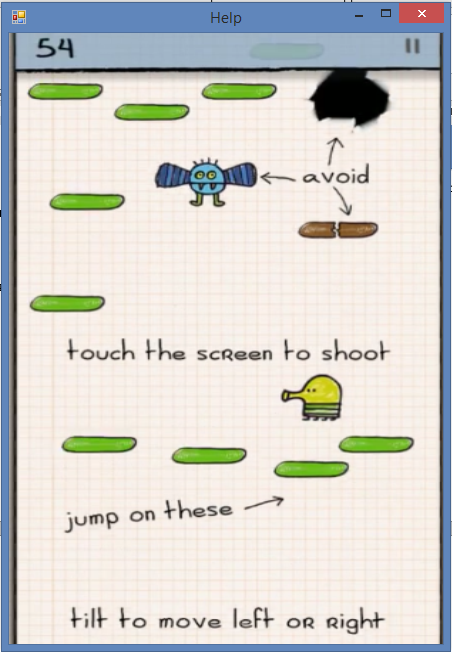
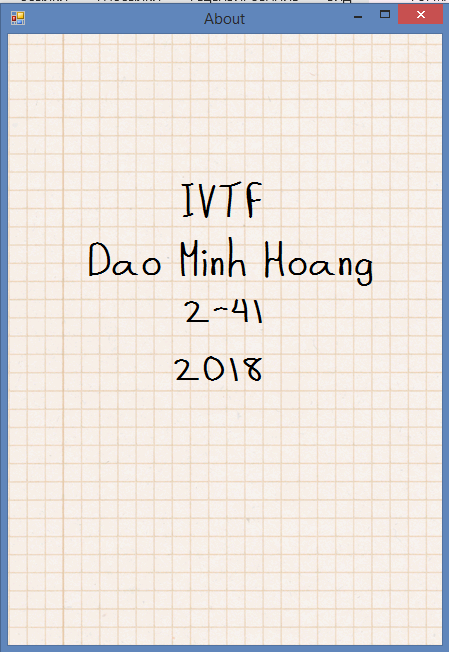
## Инструкция по использованию

Для запуска игры откройте исполнительный файл testJump.exe

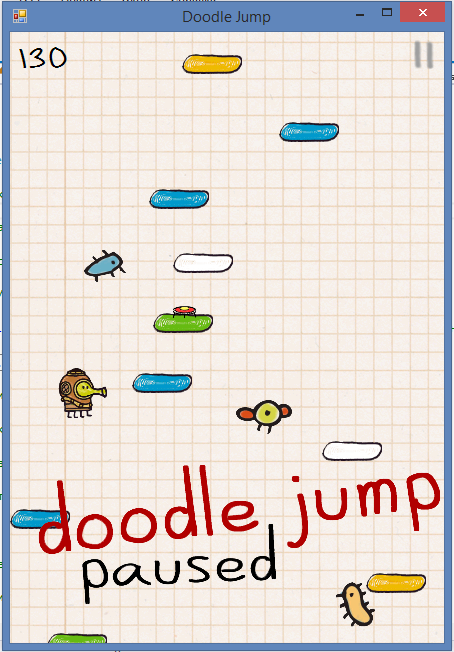


Для начала игры нажмите «play».

 Чтобы посмотреть список рекордов нажмите «scores». Пункт «options» находится в разработке.

Для того чтобы получить справку нажмите клавишу «F1», а для того чтобы узнать об создателе игре нажмите клавишу «F2». Для возврата в главное меню, нажмите клавишу «Esc».

Также во время игры можно немного передохнуть, поставив игру на паузу, для этого нажмите клавишу «P» или кликните мышкой по значку паузы.



# Приложение. Тексты программ

# Class Form1

public partial class Form1 : Form

{

private Bitmap fon, fonPause;//фон

private Bitmap fon1, fonPause1;//фон

Graphics dc, dcfon1;//графика

PrivateFontCollection custom\_font = new PrivateFontCollection();//шрифт

Player pl;//игрок

List<Platforms> listPlatform = new List<Platforms>();//список платформ

List<Bonus> listBonus = new List<Bonus>();//список бонусов

List<Bullet> listBullet = new List<Bullet>();//список пуль

List<Monsters> listMonsters = new List<Monsters>();//список монстров

Random rnd = new Random();//рандом

int dy = 0;//расстояние по оси у

int rndColor;//рандомный цвет

int g;//джи

int scores = 0;//очки

bool checkStopStart = false;//флаг паузы

Rectangle pause;//поле паузы

Bitmap pauseSkin = Properties.Resources.Pause;//скин паузы

string fileScores = "Scores.txt";//имя файла

string[] scoresfile;//массив результатов

public Form1()

{

InitializeComponent();

scoresfile = File.ReadAllLines(fileScores);

FontLoad();

dc = this.CreateGraphics();

fon = new Bitmap(Properties.Resources.bck\_2x);

fon1 = new Bitmap(Properties.Resources.bck\_2x);

fonPause = new Bitmap(Properties.Resources.pauseOnBck);

fonPause1 = new Bitmap(Properties.Resources.pauseOnBck);

dcfon1 = Graphics.FromImage(fon1);

createLevel();

}//конструктор

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

{

for(int i = 0; i < listMonsters.Count; i++)

{

if (listMonsters[i].check() == 2 || listMonsters[i].check() == 3)

{

listMonsters[i].moveLeftRight(this);

}

if (pl.PlBounds().IntersectsWith(listMonsters[i].boundMonster()) && pl.getSpeedY() > 0 && pl.PlBounds().Bottom > listMonsters[i].getCoordY() && pl.getLive())

{

listMonsters[i].checkDieMon();

pl.setSpeedY(pl.getSpeedJump() - 5);

}

else if (pl.PlBounds().IntersectsWith(listMonsters[i].boundMonster()) && pl.getLive())

{

pl.checkDie();

}

else for (int j = 0; j < listBullet.Count; j++)

{

if(listBullet[j].boundRect().IntersectsWith(listMonsters[i].boundMonster()))

{

listMonsters[i].checkHP();

listBullet[i].checkDel();

}

}

}

for (int i = 0; i < listBonus.Count; i++)

{

if (pl.PlBounds().IntersectsWith(listBonus[i].boundsBonus()) && pl.getSpeedY() > 0 && pl.getLive())

{

pl.checkBonus(listBonus[i].check());

listBonus[i].checkPlayer();

}

}

for (int i = 0; i < listPlatform.Count; i++)

{

//if(pl.PlBounds().IntersectsWith(plat.PlatListBounds()[i].PlatBounds()) && ((pl.PlBounds().Bottom > plat.PlatListBounds()[i].getCoordY() && pl.PlBounds().Bottom < plat.PlatListBounds()[i].PlatBounds().Bottom) || (pl.PlBounds().Bottom <= plat.PlatListBounds()[i].getCoordY() && pl.PlBounds().Bottom+pl.getSpeedY() >= plat.PlatListBounds()[i].PlatBounds().Bottom)))

if (pl.getLive() && pl.PlBounds().IntersectsWith(listPlatform[i].PlatBounds()) && pl.getSpeedY() > 0 && pl.PlBounds().Right-8 > listPlatform[i].getCoordX()+2 && pl.getCoordX()+6 < listPlatform[i].PlatBounds().Right-2 &&

((pl.PlBounds().Bottom >= listPlatform[i].PlatBounds().Top && pl.PlBounds().Bottom <= listPlatform[i].PlatBounds().Bottom) ||

(pl.PlBounds().Bottom <= listPlatform[i].PlatBounds().Top) ||

(pl.PlBounds().Bottom + pl.getSpeedY() >= listPlatform[i].PlatBounds().Bottom && pl.PlBounds().Bottom <= listPlatform[i].PlatBounds().Bottom) ||

(pl.PlBounds().Bottom >= listPlatform[i].PlatBounds().Bottom && pl.PlBounds().Bottom + pl.getSpeedY() <= listPlatform[i].PlatBounds().Bottom) ||

(pl.PlBounds().Bottom + pl.getSpeedY() >= listPlatform[i].PlatBounds().Bottom && pl.PlBounds().Bottom + pl.getSpeedY() <= listPlatform[i].PlatBounds().Bottom) ||

(pl.PlBounds().Bottom - pl.getSpeedY() >= listPlatform[i].PlatBounds().Bottom && pl.PlBounds().Bottom <= listPlatform[i].PlatBounds().Bottom) ||

(pl.PlBounds().Bottom > listPlatform[i].PlatBounds().Bottom && pl.PlBounds().Bottom - pl.getSpeedY() <= listPlatform[i].PlatBounds().Bottom) ||

(pl.PlBounds().Bottom - pl.getSpeedY() >= listPlatform[i].PlatBounds().Bottom && pl.PlBounds().Bottom - pl.getSpeedY() <= listPlatform[i].PlatBounds().Bottom))

)

{

pl.setCoordY(listPlatform[i].PlatBounds().Bottom - pl.PlBounds().Height);

pl.setSpeedY(pl.getSpeedJump());

if (listPlatform[i] is White)

{

deletePlat(i);

}

if (listPlatform[i] is Red)

{

if (listPlatform[i].getHP() != 0)

{

g++;

if (listPlatform[i].getHP() == 4) g = 0;

listPlatform[i].setHP(listPlatform[i].getHP() - 1);

listPlatform[i].setSkin(Properties.Resources.red.Clone(new Rectangle(0, ((g+1) \* 35) + g, 114, 30), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

}

else

{

deletePlat(i);

g = 0;

}

}

}

if (listPlatform[i] is Blue || listPlatform[i] is White)

{

if (listPlatform[i].getrndDir() % 3 == 0)

listPlatform[i].moveUpDown();

if (listPlatform[i].getrndDir() % 5 == 0)

listPlatform[i].moveLeftRight(this);

}

}

if (pl.getCoordY() < this.ClientSize.Height/2)

{

pl.setCoordY(this.ClientSize.Height / 2);

moveLevel();

//drawScores(scores);

}

for (int i = 0; i < listBullet.Count;i++)

{

listBullet[i].Move();

listBullet[i].Draw(dcfon1);

if (listBullet[i].getCoordY() < 0)

{

listBullet[i].checkDel();

}

}

drawPauseButton();

checkPlat();

checkEnd();

pl.move(this);

drawLevel();

drawScores(scores);

pl.Draw(dcfon1);

dc.DrawImage(fon1, 0, 0);

dcfon1.DrawImage(fon, 0, 0);

}//тик таймера

private void drawPauseButton()

{

pause = new Rectangle(this.ClientSize.Width - 30, 8, 20, 30);

dcfon1.DrawImage(pauseSkin, pause);

}//отрисовка паузы

private void FontLoad()

{

string fileName = Path.GetTempFileName();

File.WriteAllBytes(fileName, Properties.Resources.DoodleJump);

custom\_font.AddFontFile(fileName);

}//загрузка шрифта

private void checkGun()

{

if(pl.getCheckGun())

{

Bullet bullet = new Bullet(pl.getCoordX() - 4 + pl.PlBounds().Width / 2, pl.getCoordY() - 20);

//bullet.Draw(dcfon1);

listBullet.Add(bullet);

bullet.onCheckDelBullet += Bullet\_onCheckDelBullet;

}

}//проверка на выстрел

private void Bullet\_onCheckDelBullet()

{

listBullet.Remove(listBullet[0]);

}//событие удаление пули

private void createMonsters(int h, int f, int x, int y, int sx, int sy, bool ms, Bitmap s)

{

Monsters monster = new Monsters(h, f, x, y, sx, sy, ms, s);

listMonsters.Add(monster);

monster.onCheckHP += Monster\_onCheckHP;

monster.onCheckDieMonster += Monster\_onCheckDieMonster;

}//создание монстров

private void Monster\_onCheckDieMonster()

{

listMonsters.Remove(listMonsters[0]);

}//событие смерть монстра

private void Monster\_onCheckHP()

{

if (listMonsters[0].getHP() != 0)

{

listMonsters[0].setHP(listMonsters[0].getHP() - 1);

}

else

{

listMonsters[0].checkDieMon();

}

}//событие уменьшение жизней монстра

private void drawScores(int s)

{

dcfon1.DrawString(Convert.ToString(s), new Font(custom\_font.Families[0], 30, FontStyle.Bold), Brushes.Black, new Point(0, 0));

}//отрисовка очков

private void checkdy(Platforms pl)

{

if (dy == 0)

{

dy = 650 - 140;

}

else

{

dy = listPlatform[listPlatform.Count - 1].getCoordY() - 10 - rnd.Next(50, 60);

}

pl.setCoordX(rnd.Next(0, this.ClientSize.Width - pl.getSizeX() + 1));

pl.setCoordY(dy);

pl.setUpDown(dy);

pl.setrndDir(rnd.Next());

pl.setBonus(rnd.Next(0,100));

pl.setMonster(rnd.Next(0, 100));

pl.setBonusX(rnd.Next(0, 50));

pl.setRect(new Rectangle(pl.getCoordX(), pl.getCoordY(), pl.getSizeX(), pl.getSizeY()));

listPlatform.Add(pl);

}//проверка расстояний между платформами

private void createPlat()

{

rndColor = rnd.Next(0, 4);

switch (rndColor)

{

case 0:

Green pl = new Green();

checkdy(pl);

if(pl.getBonus() % 2 == 0 && pl.getBonus() % 3 == 0)

{

int a = rnd.Next(0, 2);

switch (a)

{

case 0:

createBonus(a,pl.getBonusX(), pl.getCoordY(), 30, 0, 10, 15, Properties.Resources.pruzhina.Clone(new Rectangle(0, 0, 34, 33), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

break;

case 1:

createBonus(a,pl.getBonusX(), pl.getCoordY(), 35, 0, 25, 15, Properties.Resources.batut.Clone(new Rectangle(74, 0, 74, 34), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

break;

}

}

if(pl.getMonster() % 2 == 0 && pl.getMonster() % 7 == 0)

{

int e = rnd.Next(0, 4);

switch (e)

{

case 0:

createMonsters(2, e, pl.getCoordX() - 5, pl.getCoordY() - 45, 70, 50, false, Properties.Resources.m1);

break;

case 1:

createMonsters(0, e, pl.getCoordX(), pl.getCoordY() - 40, 50, 40, false, Properties.Resources.m2);

break;

case 2:

createMonsters(1, e, pl.getCoordX(), pl.getCoordY() - 40, 50, 70, true, Properties.Resources.m3);

break;

case 3:

createMonsters(0, e, pl.getCoordX(), pl.getCoordY() - 40, 60, 40, true, Properties.Resources.m4);

break;

}

}

break;

case 1:

Blue pl1 = new Blue();

checkdy(pl1);

break;

case 2:

Red pl2 = new Red();

checkdy(pl2);

if (pl2.getBonus() % 2 == 0 && pl2.getBonus() % 3 == 0)

{

int b = rnd.Next(0, 2);

switch (b)

{

case 0:

createBonus(b,pl2.getBonusX(), pl2.getCoordY(), 30, 0, 10, 15, Properties.Resources.pruzhina.Clone(new Rectangle(0, 0, 34, 33), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

break;

case 1:

createBonus(b,pl2.getBonusX(), pl2.getCoordY(), 35, 0, 25, 15, Properties.Resources.batut.Clone(new Rectangle(74, 0, 74, 34), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

break;

}

}

break;

case 3:

White pl3 = new White();

checkdy(pl3);

break;

}

}//создание платформ

private void moveLevel()

{

for (int i = 0; i < listPlatform.Count; i++)

{

if (i % 2 == 0)

scores += 1;

listPlatform[i].setCoordY(listPlatform[i].getCoordY() - pl.getSpeedY());

listPlatform[i].setUpDown(listPlatform[i].getCoordY() - pl.getSpeedY());

//listPlatform[i].setBonusY(listPlatform[i].getCoordY() - pl.getSpeedY());

}

for (int i = 0; i < listBonus.Count; i++)

{

listBonus[i].setCoordY(listBonus[i].getCoordY() - pl.getSpeedY());

}

for (int i = 0; i < listMonsters.Count; i++)

{

listMonsters[i].setCoordY(listMonsters[i].getCoordY() - pl.getSpeedY());

}

}//движение всей игры

private void deletePlat(int i)

{

listPlatform.Remove(listPlatform[i]);

createPlat();

}//удаление платформы

private void deleteBonus(int i)

{

listBonus.Remove(listBonus[i]);

}//удаление бонуса

private void deleteMonsters(int i)

{

listMonsters.Remove(listMonsters[i]);

}//удаление монстра

private void checkPlat()

{

for (int i = 0; i < listPlatform.Count; i++)

{

if (listPlatform[i].getCoordY() >= this.ClientSize.Height)

{

deletePlat(i);

}

}

for (int i = 0; i < listBonus.Count; i++)

{

if (listBonus[i].getCoordY() >= this.ClientSize.Height)

{

deleteBonus(i);

}

}

for (int i = 0; i < listMonsters.Count; i++)

{

if (listMonsters[i].getCoordY() >= this.ClientSize.Height)

{

deleteMonsters(i);

}

}

}//проверка платформ

private void createBonus(int f,int x, int y, int k, int t, int sx, int sy, Bitmap sk)

{

Bonus bonus = new Bonus(f, x, y, k, t, sx, sy, sk);

bonus.onCheckPlayer += Bonus\_onCheckPlayer;

listBonus.Add(bonus);

}//создание бонуса

private void Bonus\_onCheckPlayer()

{

if(listBonus[0].check()==0)

{

listBonus[0].setSkin(Properties.Resources.pruzhina.Clone(new Rectangle(0, 0, 34, 23), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

}

else if(listBonus[0].check()==1)

{

listBonus[0].setSkin(Properties.Resources.batut.Clone(new Rectangle(0, 0, 74, 34), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

}

}//событие пересечение бонуса с игроком

private void createLevel()

{

for (int i = 0; i < 20; i++)

{

createPlat();

//createLevelBonus(i);

}

pl = new Player(listPlatform[0].getCoordX(), listPlatform[0].getCoordY()-100, Properties.Resources.underwater\_right\_2x);

pl.onCheckBonus += Pl\_onCheckBonus;

pl.onCheckDie += Pl\_onCheckDie;

}//создание игры

private void Pl\_onCheckDie()

{

pl.setLive(false);

}//событие смерть игрока

private void Pl\_onCheckBonus()

{

pl.setSpeedY(listBonus[0].getSpeedBonus());

}//событие подбор бонуса игроком

private void drawLevel()

{

for (int i = 0; i < listPlatform.Count; i++)

{

listPlatform[i].Draw(dcfon1);

}

for (int i = 0; i < listBonus.Count; i++)

{

listBonus[i].Draw(dcfon1);

}

for (int i = 0; i < listMonsters.Count; i++)

{

listMonsters[i].Draw(dcfon1);

}

}//отрисовка игры

private void keyisdown(object sender, KeyEventArgs e)

{

pl.checkKeyDown(this, e);

checkGun();

if (e.KeyCode == Keys.Escape)

{

timer1.Stop();

Menu mn = new Menu();

mn.Show();

Hide();

}

//if(e.KeyCode == Keys.S)

//{

// Level lev = new Level(scores, pl, listPlatform, listBonus, listBullet, listMonsters);

// FileStream stream = File.Create("level.bin");

// var serializer = new BinaryFormatter();

// serializer.Serialize(stream, lev);

// stream.Close();

//}

//if (e.KeyCode == Keys.R)

//{

// Level lev;

// FileStream stream = File.Open("level.bin", FileMode.Open);

// var deserializer = new BinaryFormatter();

// lev = deserializer.Deserialize(stream) as Level;

// stream.Close();

// scores = lev.score;

// pl = lev.pl;

// listPlatform = lev.listPlatform;

// listBonus = lev.listBonus;

// listBullet = lev.listBullet;

// listMonsters = lev.listMonsters;

//}

if (e.KeyCode == Keys.P)

{

if (!checkStopStart)

{

checkStopStart = true;

dc.DrawImage(fonPause1, 0, 0);

dcfon1.DrawImage(fonPause, 0, 0);

timer1.Stop();

}

else

{

checkStopStart = false;

timer1.Start();

}

}

}//зажата клавиша

private void keyisup(object sender, KeyEventArgs e)

{

pl.checkKeyUp(this, e);

}//не зажата клавиша

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}//выход из формы

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

if(e.X > pause.X && e.X < pause.X + pause.Width && e.Y > pause.Y && e.Y < pause.Y + pause.Height)

{

pauseSkin = Properties.Resources.PauseOn;

}

else

{

pauseSkin = Properties.Resources.Pause;

}

}//движение мыши

private void Form1\_MouseClick(object sender, MouseEventArgs e)

{

if (e.X > pause.X && e.X < pause.X + pause.Width && e.Y > pause.Y && e.Y < pause.Y + pause.Height)

{

if (!checkStopStart)

{

checkStopStart = true;

dc.DrawImage(fonPause1, 0, 0);

dcfon1.DrawImage(fonPause, 0, 0);

timer1.Stop();

}

else

{

checkStopStart = false;

timer1.Start();

}

}

}//клик мыши

private void checkEnd()

{

if (pl.getCoordY() > this.ClientSize.Height)

{

if (Convert.ToInt32(scoresfile[scoresfile.Length - 1]) < scores)

{

scoresfile[scoresfile.Length - 1] = Convert.ToString(scores);

CheckRecord.Value = true;

}

else

{

CheckRecord.Value = false;

}

File.WriteAllLines(fileScores, scoresfile);

timer1.Stop();

GameEnd gm = new GameEnd();

gm.Show();

Hide();

}

}//конец игры

}

# Class Player

class Player

{

bool goleft;//флаг влево

bool goright;//флаг вправо

bool isPressed;//флаг нажатия клавиши

bool checkLeftRight;//флаг поворота

bool check;//флаг поворота(скин)

bool checkGun = false;//флаг стрельбы

bool live = true;//флаг жизни

int size = 60;//размер

int speedX = 15;//скороть по горизонтали

int speedY;// скорость по вертикали

int xCoord;// координата х

int yCoord;// координата у

int speed = -23;// сила прыжка

int gravitation = 1;//сила притяжения

Bitmap skin;// скин

Rectangle rectPlayer;// поле персонажа

public delegate void Method1();//делегат

public delegate void Method2();//делегат

public event Method1 onCheckBonus;//событие пересечения с бонусом

public event Method1 onCheckDie;//событие смерти

public void checkDie()

{

onCheckDie();

}//проверка на смерть

public void checkBonus(int k)

{

onCheckBonus();

}//проверка на бонус

public Player(int x, int y, Bitmap sk)

{

xCoord = x;

yCoord = y;

skin = new Bitmap(sk);

}//конструктор

public Rectangle PlBounds()

{

return rectPlayer;

}//возвращает поле персонажа

public void Draw(Graphics dc)

{

rectPlayer = new Rectangle(xCoord,yCoord,size,size);

dc.DrawImage(skin, rectPlayer);

//dc.DrawImage(skin, xCoord, yCoord);

}//отрисовка персонажа

public void setSkin(Bitmap sk)

{

skin = new Bitmap(sk);

}//установка скина

public void setCoordX(int x)

{

xCoord = x;

}//установка координаты х

public void setCoordY(int y)

{

yCoord = y;

}//установка координаты у

public int getCoordX()

{

return xCoord;

}//возвращает координату х

public int getCoordY()

{

return yCoord;

}//возвращает координату у

public int getSpeedY()

{

return speedY;

}//возвращает скорость по вертикали

public int getSpeedJump()

{

return speed;

}//возвращает силу прыжка

public void setSpeedY(int y)

{

speedY = y;

}//установка скорости по вертикали

public void setGravitation(int g)

{

gravitation = g;

}//установка гравитации

public bool getCheckGun()

{

return checkGun;

}//проверка на выстрел

public void setLive(bool t)

{

live = t;

}//установка жизни/смерти

public bool getLive()

{

return live;

}//возвращает жизнь/смерть

public void checkKeyDown(Form1 form, KeyEventArgs btn)

{

if (btn.KeyCode == Keys.Left)

{

checkLeftRight = false;

goleft = true;

skin = new Bitmap(Properties.Resources.underwater\_left\_2x);

}

if (btn.KeyCode == Keys.Right)

{

checkLeftRight = true;

goright = true;

skin = new Bitmap(Properties.Resources.underwater\_right\_2x);

}

if (btn.KeyCode == Keys.Up && !isPressed)

{

isPressed = true;

if (!checkLeftRight) check = false;

else check = true;

skin = new Bitmap(Properties.Resources.underwater\_up\_2x);

checkGun = true;

}

}//проверка на нажатие

public void checkKeyUp(Form1 form, KeyEventArgs btn)

{

if (btn.KeyCode == Keys.Left)

{

goleft = false;

}

if (btn.KeyCode == Keys.Right)

{

goright = false;

}

if (isPressed)

{

isPressed = false;

if (check == false)

skin = new Bitmap(Properties.Resources.underwater\_left\_2x);

else

skin = new Bitmap(Properties.Resources.underwater\_right\_2x);

checkGun = false;

}

}//проверка на отпуск клавиши

public void move(Form1 form)

{

speedY += gravitation;

yCoord += speedY;

if (goleft)

{

xCoord -= speedX;

if (getCoordX() < -rectPlayer.Width)

{

xCoord = form.ClientSize.Width;

}

}

else if (goright)

{

xCoord += speedX;

if (getCoordX() > form.ClientSize.Width)

{

xCoord = -rectPlayer.Width;

}

}

}//движение персонажа

}

# Class Platforms

class Platforms

{

bool Right = true;//флаг движения по горизонтали

bool Up = true;//флаг движения по вертикали

int HP;//количество прыжков

int speedLeftRight = 5;//скорость по горизонтали

int speedUpDown = 2;//скорость по вертикали

int xCoord;//координата х

int yCoord;//координата у

int sizeX = 60;//ширина

int sizeY = 20;//высота

int UpDown;//точка перемещения вверх-вниз

Bitmap skin;//скин

Rectangle rectPlatform;//поле платформы

int rndDir;//рандомное направление

int checkBonus;//проверка на бонус

int checkMonster;//проверка на монстров

int bonusX;//координата х бонуса

int bonusY;//координата у бонуса

Random rnd = new Random();//рандом

//Platforms()

//{

// rndColor = rnd.Next(0, 4);

// switch (rndColor)

// {

// case 0:

// skin = Properties.Resources.green;

// break;

// case 1:

// skin = Properties.Resources.blue;

// break;

// case 2:

// skin = Properties.Resources.red.Clone(new Rectangle(0, 145, 114, 30), System.Drawing.Imaging.PixelFormat.Format32bppArgb);

// break;

// case 3:

// skin = Properties.Resources.white;

// break;

// }

//}

public Platforms()

{

}//конструктор

public void Draw(Graphics dc)

{

dc.DrawImage(skin, rectPlatform);

}//отрисовка

public void moveLeftRight(Form1 form)

{

if (Right) setCoordX(getCoordX() + speedLeftRight);

else setCoordX(getCoordX() - speedLeftRight);

if (xCoord > form.ClientSize.Width - sizeX)

{

Right = false;

setCoordX(getCoordX() - speedLeftRight);

}

else if (xCoord < 0)

{

Right = true;

setCoordX(getCoordX() + speedLeftRight);

}

}//движение по горизонтали

public void moveUpDown()

{

if (Up) setCoordY(getCoordY() - speedUpDown);

else setCoordY(getCoordY() + speedUpDown);

if (yCoord < UpDown - 50)

{

Up = false;

setCoordY(getCoordY() + speedUpDown);

}

else if (yCoord > UpDown + 50)

{

Up = true;

setCoordY(getCoordY() - speedUpDown);

}

}//по вертикали

public Rectangle PlatBounds()

{

return rectPlatform;

}//возвращает поле платформы

public void setrndDir(int dir)

{

rndDir = dir;

}//устанавливает рандомное направление

public void setRect(Rectangle r)

{

rectPlatform = r;

}//устанавливает поле платформы

public void setSkin(Bitmap sk)

{

skin = new Bitmap(sk);

}//устанавливает скин

public void setCoordX(int x)

{

xCoord = x;

rectPlatform.X = x;

}//устанавливает координату х

public void setCoordY(int y)

{

yCoord = y;

rectPlatform.Y = y;

}//устанавливает координату у

public void setUpDown(int y)

{

UpDown = y;

}//устанавливает координату опоры для вверх-вниз

public void setHP(int hp)

{

HP = hp;

}//устанавливает жизни платформы

public void setBonus(int b)

{

checkBonus = b;

}//устанавливает бонус на платформу

public void setBonusX(int b)

{

bonusX = xCoord + b;

}//координата х бонуса на платформе

public void setBonusY(int b)

{

bonusY = b;

}//координата у бонуса на платформе

public void setMonster(int b)

{

checkMonster = b;

}//установка монстра на платформу

public int getBonusY()

{

return bonusY;

}//возвращает координату у бонуса

public int getBonusX()

{

return bonusX;

}//возвращает координату х бонуса

public int getBonus()

{

return checkBonus;

}//возвращает бонус

public int getMonster()

{

return checkMonster;

}//возвращает монстра

public int getHP()

{

return HP;

}//возвращает жизни

public int getrndDir()

{

return rndDir;

}//возвращает рандомноне направление

public int getCoordX()

{

return xCoord;

}//возвращает х

public int getCoordY()

{

return yCoord;

}//возвращает у

public int getSizeX()

{

return sizeX;

}//возвращает ширину

public int getSizeY()

{

return sizeY;

}//возвращает высоту

}

# Class Green

class Green : Platforms

{

public Green()//конструктор

{

setSkin(Properties.Resources.green);

}

}

# Class Red

class Red : Platforms

{

public Red()

{

setSkin(Properties.Resources.red.Clone(new Rectangle(0, 0, 114, 30), System.Drawing.Imaging.PixelFormat.Format32bppArgb));

setHP(4);

}

}

# Class Blue

class Blue : Platforms

{

public Blue()

{

setSkin(Properties.Resources.blue);

}

}

# Class White

class White : Platforms

{

public White()

{

setSkin(Properties.Resources.white);

}

}

# Class Bonus

class Bonus

{

int xCoord;//координата х

int yCoord;//координата у

int sizeX;//ширина

int sizeY;//высота

int speedBonus;//бонусная скорость

int checkBonus;//проверка бонуса

int timeSpeed;//время действия

Bitmap skin;//скин

Rectangle rectBonus;//поле бонуса

public delegate void Method3();//делегат

public event Method3 onCheckPlayer;//событие пересечения с игроком

public void checkPlayer()

{

onCheckPlayer();

}//инициализация события

public Bonus(int f, int x, int y, int speed, int time, int sx, int sy, Bitmap sk)

{

checkBonus = f;

skin = sk;

speedBonus = -speed;

timeSpeed = time;

sizeX = sx;

sizeY = sy;

xCoord = x;

yCoord = y - sizeY + 5;

}//конструктор

public void Draw(Graphics dc)

{

rectBonus = new Rectangle(xCoord, yCoord, sizeX, sizeY);

dc.DrawImage(skin,rectBonus);

}//отрисовка

public void setSizeY(int y)

{

sizeY = y;

}//установка высоту

public void setSkin(Bitmap sk)

{

skin = new Bitmap(sk);

}//установка скина

public void setCoordX(int x)

{

xCoord = x;

rectBonus.X = x;

}//установка координаты х

public void setTimeSpeed(int t)

{

timeSpeed = t;

}//установка времени действия

public void setCoordY(int y)

{

yCoord = y;

rectBonus.Y = y;

}//установка координаты у

public int getTimeSpeed()

{

return timeSpeed;

}//возвращает время действия

public int getSpeedBonus()

{

return speedBonus;

}//возвращает бонусную скорость

public int getSizeY()

{

return sizeY;

}//возвращает высоту

public int getCoordX()

{

return xCoord;

}//возвращает координату х

public int getCoordY()

{

return yCoord;

}//возвращает координату у

public int check()

{

return checkBonus;

}//проверка бонуса

public Rectangle boundsBonus()

{

return rectBonus;

}//возвращает поле бонуса

}

# Class Bullet

class Bullet

{

int speedY = 20;//скорость пули

int xCoord;//координата х

int yCoord;//координата у

int size = 10;//размер

Rectangle rectBullet;//поле пули

Bitmap skin = Properties.Resources.bullet;//скин

public delegate void Method5();//делегат

public event Method5 onCheckDelBullet;//событие удаления пули

public void checkDel()

{

onCheckDelBullet();

}//проверка на удаление

public Bullet(int x, int y)

{

xCoord = x;

yCoord = y;

}//конструктор

public void Move()

{

yCoord -= speedY;

}//движение

public void Draw(Graphics dc)

{

rectBullet = new Rectangle(xCoord, yCoord, size, size);

dc.DrawImage(skin, rectBullet);

}//отрисовка

public void setCoordX(int x)

{

xCoord = x;

rectBullet.X = x;

}//установка х

public void setCoordY(int y)

{

yCoord = y;

rectBullet.Y = y;

}//установка у

public int getCoordX()

{

return xCoord;

}//возвращает х

public int getCoordY()

{

return yCoord;

}//возвращает у

public Rectangle boundRect()

{

return rectBullet;

}//возвращает поле пули

}

# Class Monsters

class Monsters

{

bool Right = true;//флаг движения по горизонтали

int speedLeftRight = 3;//скорость по горизонтали

int xCoord;//координата х

int yCoord;//координата у

int sizeX;//ширина

int sizeY;//высота

int checkMonster;//проверка монстра

int HP;//жизни

bool moveStand;//флаг на движение

Rectangle rectMonster;//поле монстра

Bitmap skin;//скин

public delegate void Method4();//делегат

public event Method4 onCheckHP;//событие уменьшение жизней

public delegate void Method6();//делегат

public event Method6 onCheckDieMonster;//событие смерти монстра

public void checkDieMon()

{

onCheckDieMonster();

}//инициализация события

public void checkHP()

{

onCheckHP();

}//инициализация события

public Monsters(int h, int f, int x, int y, int sx, int sy, bool ms, Bitmap s)

{

HP = h;

checkMonster = f;

xCoord = x;

yCoord = y;

skin = s;

moveStand = ms;

sizeX = sx;

sizeY = sy;

}//конструктор

public void Draw(Graphics dc)

{

rectMonster = new Rectangle(xCoord, yCoord, sizeX, sizeY);

dc.DrawImage(skin, rectMonster);

}//отрисовка

public void setCoordX(int x)

{

xCoord = x;

rectMonster.X = x;

}//установка х

public void setCoordY(int y)

{

yCoord = y;

rectMonster.Y = y;

}//установка у

public void setHP(int hp)

{

HP = hp;

}//установка жизней

public int getCoordX()

{

return xCoord;

}//возвращает х

public int getCoordY()

{

return yCoord;

}//возвращает у

public int getHP()

{

return HP;

}//возвращает жизни

public int check()

{

return checkMonster;

}//проверка монстра

public Rectangle boundMonster()

{

return rectMonster;

}//возвращает поле монстра

public void moveLeftRight(Form1 form)

{

if (Right) setCoordX(getCoordX() + speedLeftRight);

else setCoordX(getCoordX() - speedLeftRight);

if (xCoord > form.ClientSize.Width - sizeX)

{

Right = false;

setCoordX(getCoordX() - speedLeftRight);

}

else if (xCoord < 0)

{

Right = true;

setCoordX(getCoordX() + speedLeftRight);

}

}//движение по горизнтали

}

# Class Level

class Level

{

public int score;

public Player pl;

public List<Platforms> listPlatform = new List<Platforms>();

public List<Bonus> listBonus = new List<Bonus>();

public List<Bullet> listBullet = new List<Bullet>();

public List<Monsters> listMonsters = new List<Monsters>();

public Level(int sc, Player p, List<Platforms> l, List<Bonus> b, List<Bullet> bul, List<Monsters> m)

{

score = sc;

pl = p;

listPlatform = l;

listBonus = b;

listBullet = bul;

listMonsters = m;

}

}

# Class Menu

public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

buttonPlay.Parent = pictureBox14;

buttonOptions.Parent = pictureBox14;

pictureBox13.Parent = pictureBox14;

buttonScores.Parent = pictureBox14;

pictureBox1.Parent = pictureBox14;

pictureBox2.Parent = pictureBox14;

pictureBox3.Parent = pictureBox14;

pictureBox4.Parent = pictureBox14;

pictureBox5.Parent = pictureBox14;

pictureBox6.Parent = pictureBox14;

pictureBox7.Parent = pictureBox14;

pictureBox8.Parent = pictureBox14;

pictureBox9.Parent = pictureBox14;

pictureBox10.Parent = pictureBox14;

pictureBox11.Parent = pictureBox14;

pictureBox12.Parent = pictureBox14;

}

private void buttonPlay\_MouseHover(object sender, EventArgs e)

{

buttonPlay.Image = testJump.Properties.Resources.play\_on\_2x;

buttonPlay.Invalidate();

}

private void buttonPlay\_MouseLeave(object sender, EventArgs e)

{

buttonPlay.Image = testJump.Properties.Resources.play\_2x;

buttonPlay.Invalidate();

}

private void buttonPlay\_MouseClick(object sender, MouseEventArgs e)

{

Form1 frm = new Form1();

frm.Show();

frm.Refresh();

Hide();

}

private void Menu\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void buttonOptions\_MouseClick(object sender, MouseEventArgs e)

{

//Options opt = new Options();

//opt.Show();

//Hide();

}

private void buttonOptions\_MouseLeave(object sender, EventArgs e)

{

buttonOptions.Image = testJump.Properties.Resources.options\_2x;

buttonOptions.Invalidate();

}

private void buttonScores\_MouseClick(object sender, MouseEventArgs e)

{

testJump.Scores scr = new testJump.Scores();

scr.Show();

Hide();

//CheckRecord.CheckForm = false;

}

private void buttonScores\_MouseLeave(object sender, EventArgs e)

{

buttonScores.Image = testJump.Properties.Resources.score2x;

buttonScores.Invalidate();

}

private void buttonOptions\_MouseHover(object sender, EventArgs e)

{

buttonOptions.Image = testJump.Properties.Resources.options\_on\_2x;

buttonOptions.Invalidate();

}

private void buttonScores\_MouseHover(object sender, EventArgs e)

{

buttonScores.Image = testJump.Properties.Resources.scoreOn\_2x;

buttonScores.Invalidate();

}

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

{

if (e.KeyCode == Keys.F1)

{

Help h = new Help();

h.Show();

Hide();

}

if (e.KeyCode == Keys.F2)

{

About ab = new About();

ab.Show();

Hide();

}

}

}

# Class Scores

public partial class Scores : Form

{

Font myFont;

Font myFont2;

string fileScores = "Scores.txt";

string fileNames = "Names.txt";

string[] scores;

string[] names;

int xEnd = 400;

int yStart = 145;

public Scores()

{

InitializeComponent();

FontLoad2();

label1.Parent = pictureBox7;

label2.Parent = pictureBox7;

pictureBox3.Parent = pictureBox7;

pictureBox4.Parent = pictureBox7;

pictureBox5.Parent = pictureBox7;

pictureBox6.Parent = pictureBox7;

buttonCancel.Parent = pictureBox7;

pictureBox1.Parent = pictureBox7;

pictureBox2.Parent = pictureBox7;

label1.Font = myFont2;

label2.Font = myFont2;

scores = File.ReadAllLines(fileScores);

names = File.ReadAllLines(fileNames);

FontLoad();

printScores();

}

private void printScores()

{

int k = yStart;

for (int i = 0; i < 10; i++)

{

if (Convert.ToInt32(scores[i]) != 0)

{

createNames(k, Convert.ToString(i + 1) + ". " + names[i]);

createScores(xEnd, k, scores[i]);

}

else

{

break;

}

k += 40;

}

}

private void createNames(int y, string t)

{

Label lb = new Label();

lb.Font = myFont;

lb.AutoSize = true;

lb.Left = 36;

lb.Top = y;

lb.Text = t;

this.Controls.Add(lb);

lb.Parent = pictureBox7;

lb.BringToFront();

lb.BackColor = Color.Transparent;

}

private void createScores(int x, int y, string t)

{

Label lb1 = new Label();

lb1.Font = myFont;

lb1.AutoSize = true;

lb1.Text = t;

lb1.Left = x - lb1.PreferredWidth;

lb1.Top = y;

this.Controls.Add(lb1);

lb1.Parent = pictureBox7;

lb1.BringToFront();

lb1.BackColor = Color.Transparent;

}

private void FontLoad()

{

string fileName = Path.GetTempFileName();

File.WriteAllBytes(fileName, Properties.Resources.DoodleJump);

PrivateFontCollection custom\_font = new PrivateFontCollection();

custom\_font.AddFontFile(fileName);

myFont = new Font(custom\_font.Families[0], 28F, System.Drawing.FontStyle.Bold);

}

private void FontLoad2()

{

string fileName = Path.GetTempFileName();

File.WriteAllBytes(fileName, Properties.Resources.DoodleJump);

PrivateFontCollection custom\_font = new PrivateFontCollection();

custom\_font.AddFontFile(fileName);

myFont2 = new Font(custom\_font.Families[0], 22.5F, System.Drawing.FontStyle.Bold);

}

private void Scores\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void buttonCancel\_MouseHover(object sender, EventArgs e)

{

buttonCancel.Image = Properties.Resources.cancel\_on\_2x;

buttonCancel.Invalidate();

}

private void buttonCancel\_MouseLeave(object sender, EventArgs e)

{

buttonCancel.Image = Properties.Resources.cancel\_2x;

buttonCancel.Invalidate();

}

private void buttonCancel\_MouseClick(object sender, MouseEventArgs e)

{

if (!CheckRecord.CheckForm)

{

Menu mn = new Menu();

mn.Show();

Hide();

}

else

{

GameEnd gm = new GameEnd();

gm.Show();

Hide();

}

}

}

# Class GameEnd

public partial class GameEnd : Form

{

Font myFont;

[DllImport("user32.dll")]

static extern bool HideCaret(IntPtr hWnd);

[DllImport("user32.dll")]

static extern bool ShowCaret(IntPtr hWnd);

string fileScores = "Scores.txt";

string fileNames = "Names.txt";

string[] scores;

string[] names;

public GameEnd()

{

InitializeComponent();

FontLoad();

buttonScores.Parent = pictureBox2;

buttonResume.Parent = pictureBox2;

buttonMenu.Parent = pictureBox2;

pictureBox1.Parent = pictureBox2;

yourscore.Parent = pictureBox2;

highscore.Parent = pictureBox2;

label3.Parent = pictureBox2;

textBox1.Parent = pictureBox2;

label1.Parent = pictureBox2;

label2.Parent = pictureBox2;

yourscore.Font = myFont;

highscore.Font = myFont;

label3.Font = myFont;

label2.Font = myFont;

textBox1.Font = myFont;

label1.Font = myFont;

textBox1.Text = "doodler";

label2.Visible = false;

textBox1.MaxLength = 10;

if (CheckRecord.Value)

{

scores = File.ReadAllLines(fileScores);

names = File.ReadAllLines(fileNames);

yourscore.Text = "your score: " + scores[scores.Length - 1];

if (Convert.ToInt32(scores[scores.Length - 1]) > Convert.ToInt32(scores[0]))

highscore.Text = "high score: " + scores[scores.Length - 1];

else

highscore.Text = "high score: " + scores[0];

if (CheckRecord.CheckBack)

{

textBox1.Text = CheckRecord.Name;

textBox1.Enabled = false;

}

textBox1.Visible = true;

label2.Visible = true;

}

else

{

yourscore.Visible = false;

highscore.Visible = false;

label3.Visible = false;

textBox1.Visible = false;

label1.Visible = true;

}

}

private void GameEnd\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void Swap(string[] s, int i, int j)

{

string glass = s[i];

s[i] = s[j];

s[j] = glass;

}

private void ShakerSort(string[] s, string[] n)

{

int left = 0,

right = s.Length - 1,

count = 0;

while (left <= right)

{

for (int i = left; i < right; i++)

{

count++;

if (Convert.ToInt32(s[i]) < Convert.ToInt32(s[i + 1]))

{

Swap(s, i, i + 1);

Swap(n, i, i + 1);

}

}

right--;

for (int i = right; i > left; i--)

{

count++;

if (Convert.ToInt32(s[i - 1]) < Convert.ToInt32(s[i]))

{

Swap(s, i - 1, i);

Swap(n, i - 1, i);

}

}

left++;

}

}

private void textBox1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

textBox1.Enabled = false;

names[names.Length - 1] = textBox1.Text;

ShakerSort(scores, names);

File.WriteAllLines(fileScores, scores);

File.WriteAllLines(fileNames, names);

CheckRecord.Name = textBox1.Text;

CheckRecord.CheckBack = true;

label2.ForeColor = Color.Gray;

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

label2.Text = textBox1.Text;

if (textBox1.Text.Length == 10)

HideCaret(textBox1.Handle);

else

ShowCaret(textBox1.Handle);

}

private void buttonMenu\_MouseHover(object sender, EventArgs e)

{

buttonMenu.Image = Properties.Resources.menu\_on\_2x;

buttonMenu.Invalidate();

}

private void buttonMenu\_MouseLeave(object sender, EventArgs e)

{

buttonMenu.Image = Properties.Resources.menu\_2x;

buttonMenu.Invalidate();

}

private void buttonResume\_MouseHover(object sender, EventArgs e)

{

buttonResume.Image = Properties.Resources.play\_again\_on\_2x;

buttonResume.Invalidate();

}

private void buttonResume\_MouseLeave(object sender, EventArgs e)

{

buttonResume.Image = Properties.Resources.play\_again\_2x;

buttonResume.Invalidate();

}

private void buttonMenu\_MouseClick(object sender, MouseEventArgs e)

{

Menu mn = new Menu();

mn.Show();

Hide();

}

private void buttonResume\_MouseClick(object sender, MouseEventArgs e)

{

Form1 frm = new Form1();

frm.Show();

Hide();

}

private void FontLoad()

{

string fileName = Path.GetTempFileName();

File.WriteAllBytes(fileName, Properties.Resources.DoodleJump);

PrivateFontCollection custom\_font = new PrivateFontCollection();

custom\_font.AddFontFile(fileName);

myFont = new Font(custom\_font.Families[0], 28F, System.Drawing.FontStyle.Bold);

}

private void buttonScores\_MouseHover(object sender, EventArgs e)

{

buttonScores.Image = Properties.Resources.scoreOn\_2x;

buttonScores.Invalidate();

}

private void buttonScores\_MouseLeave(object sender, EventArgs e)

{

buttonScores.Image = Properties.Resources.score2x;

buttonScores.Invalidate();

}

private void buttonScores\_MouseClick(object sender, MouseEventArgs e)

{

Scores scr = new Scores();

scr.Show();

Hide();

CheckRecord.CheckForm = true;

}

}

# Class CheckRecord

static class CheckRecord

{

public static bool Value { get; set; }

public static bool CheckForm { get; set; }

public static bool CheckBack { get; set; }

public static string Name { get; set; }

}

# Class Help

public partial class Help : Form

{

public Help()

{

InitializeComponent();

}

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

{

if (e.KeyCode == Keys.Escape)

{

Menu mn = new Menu();

mn.Show();

Hide();

}

}

private void Help\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

# Class About

public partial class About : Form

{

Font myFont;

public About()

{

InitializeComponent();

FontLoad();

label1.Font = myFont;

label2.Font = myFont;

label3.Font = myFont;

label4.Font = myFont;

}

private void FontLoad()

{

string fileName = Path.GetTempFileName();

File.WriteAllBytes(fileName, Properties.Resources.DoodleJump);

PrivateFontCollection custom\_font = new PrivateFontCollection();

custom\_font.AddFontFile(fileName);

myFont = new Font(custom\_font.Families[0], 36F, System.Drawing.FontStyle.Bold);

}

private void About\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

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

{

if (e.KeyCode == Keys.Escape)

{

Menu mn = new Menu();

mn.Show();

Hide();

}

}

}