Федеральное казенное профессиональное образовательное учреждение

«Новокузнецкий государственный гуманитарно-технический колледж-интернат»

Министерства Труда и социальной защиты Российской Федерации

Специальность: 09.02.04 «Информационные системы (по отраслям)»

КУРСОВОЙ ПРОЕКТ

по дисциплине: МДК 02.02 Управление проектами

на тему: «Разработка программного продукта «Игра МАРИО» с использованием СУБД»

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Студент Пестов К.С.

дата личная подпись расшифровка подписи

Руководитель проекта Грисман С.С.

дата личная подпись расшифровка подписи

Нормоконтроль Михайлов Н.Н.

дата личная подпись расшифровка подписи

Оценка

Новокузнецк, 2019 г.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc27552458)

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА 4](#_Toc27552459)

[2. Общая часть 5](#_Toc27552460)

[2.1 Выбор и обоснования средств 5](#_Toc27552461)

[2.2 Выбор и обоснование средств разработки программного продукта 6](#_Toc27552462)

[Для разработки курсового проекта был выбран язык программирования C#. 6](#_Toc27552463)

[2.3 анализ прототипов 7](#_Toc27552464)

[3. СПЕЦИАЛЬНАЯ ЧАСТЬ 8](#_Toc27552465)

[3.1 Разработка алгоритма и проектирование структуры базы данных 8](#_Toc27552466)

[3.2 Разработка программного продукта 8](#_Toc27552467)

[3.3 Руководство пользователя 22](#_Toc27552468)

[ЗАКЛЮЧЕНИЕ 25](#_Toc27552469)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc27552470)

[ПРИЛОЖЕНИЯ 27](#_Toc27552471)

# ВВЕДЕНИЕ

В последнее время компьютерные технологии начали свое стремительное развитие, появление мощнейших графических ускорителей и центральных процессоров способствовало не менее бурному развитию индустрии компьютерных игр. Выдающиеся разработки этой отрасли – это сложнейшие программы, как правило, с очень высокими требованиями к аппаратной части компьютера. Но, и существует великое множество небольших компьютерных игр, которые также популярны. Для возможности отдохнуть в перерыве от выполнения какой-либо работы оператору компьютера не всегда требуется новейшая компьютерная игра, а зачастую использовать её не позволяет маломощное оборудование офисного компьютера. Именно этой цели – отдыху от монотонной работы служит разрабатываемая в рамках данного курсового проекта игра «МАРИО».

Целью данного курсового проекта, является разработка программного продукта "Игра МАРИО" с применением СУБД.

Актуальность темы обуславливается необходимостью создания продукта, который будет помогать развивать реакцию, внимание и приносить удовольствие от игры, с использованием информационных технологий, которые прочно вошли в нашу повседневную жизнь.

Выдвигается гипотеза, что подобные игры помогают развивать концентрацию, внимание, реакцию принятия решений и помогает отдохнуть от повседневности.

Объектом в данной работе является игра «МАРИО».

Предметом в данной работе является автоматизация процессов игры.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* Разработать техническое задание;
* Проанализировать прототипы;
* Произвести анализ и проектирование структуры базы данных;
* Выбрать и обосновать программные средства реализации;
* Разработать программный продукт;
* Разработать руководство пользователя.

# 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА

Разработка приложения для автоматизации игрового процесса

Прототипом данной системы можно считать игру «МАРИО»

Требования к функциональным характеристикам, следующие:

1. Необходимо предусмотреть два режима работы – режим администратора и режим пользователя.

2. В режиме пользователя должна быть возможность зарегистрироваться, заполнить свою анкету (Nickname, password). Реализовывать возможность сохранения прогресса игры.

3. В режиме пользователя можно полноценно играть, а также посмотреть историю своих достижений.

4. В режиме администратора должна быть возможность просматривать, удалять старые сохранения.

5. В режиме администратора должна быть реализована возможность регистрации пользователя в игре.

6. В игре должно присутствовать: препятствия, враги, бонусы, награда, счетчик очков.

Требования к информационному обеспечению, следующие:

К приложению должна быть предоставлена документация, содержащая техническое описание продукта и руководство пользователя.

Требования к лингвистическому обеспечению, следующие:

Приложение должно иметь интерфейс и документацию на русском языке.

Требования к программному обеспечению, следующие:

Разработка программного продукта предусматривается под OS Windows не ниже версии 7.

Требования к техническому обеспечению, следующие:

Для работы приложения требуется MS SQL Server, Visual Studio С#. Также, необходимо не менее 2Гб оперативной памяти и 800Мб свободного места на жестком диске.

# 2. Общая часть

# 2.1 Выбор и обоснования средств

Для реализации проекта были выбраны программы Microsoft SQL Server и Visual Studio 2017.

**Microsoft SQL Server** — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

**База данных MS SQL** – это система, предназначенная для хранения и обработки информации. Комплекс таблиц, взаимосвязанных между собой, для доступа к которым применяется система управления базами данных (СУБД) MS SQL. MS SQL – это специальная программа с открытым кодом, которая используется на сервере SQL. Данная программа не способна обрабатывать большое количество информации, однако она идеальна для небольших и крупных веб-ресурсов.

**Microsoft Visual Studio** - это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Интуитивный стиль кодирования. По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам. Применяемые Visual Studio автоматически параметры форматирования можно даже настраивать, что очень удобно в случаях, когда разработчик предпочитает другой стиль размещения скобок (например, стиль K&R, при котором открывающая скобка размещается на той же строке, что и объявление, которому она предшествует).

Более высокая скорость разработки. Многие из функциональных возможностей Visual Studio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense (которая умеет перехватывать ошибки и предлагать правильные варианты), функции поиска и замены (которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте) и функции автоматического добавления и удаления комментариев (которая может временно скрывать блоки кода), позволяют разработчику работать быстро и эффективно.

Возможности отладки. Предлагаемые в Visual Studio инструменты отладки являются наилучшим средством для отслеживания загадочных ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

## 2.2 Выбор и обоснование средств разработки программного продукта

## Для разработки курсового проекта был выбран язык программирования C#.

C# (произносится как "си шарп")— простой, современный объектно-ориентированный и тип безопасный язык программирования. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

Язык C#, разработанный компанией Майкрософт, один из самых популярных современных языков программирования. Он востребован на рынке разработки, C# применяют при работе с программами для ПК, при создании сложных веб-сервисов или мобильных приложений.

Приложение разрабатывалось в программе Microsoft Visual Studio 2017, она имеет графический интерфейс для построения приложения и программную оболочку для работы непосредственно с кодом. Работать в программе удобно и просто.

В Visual Studio можно легко и быстро подключить внешнюю базу данных используя мастер подключений, связав тем самым базу и разрабатываемую вами программу. Язык C# достаточно прост и популярен у начинающих программистов. Разработка приложения в Visual Studio путем графического добавления элементов на создаваемую форму и с последующим добавлением кода в программной оболочке на нужный вам элемент это довольно удобно, так как все находится в одной программе и не нужно переключаться, что исключает снижение концентрации на выполняемой работе.

## 2.3 анализ прототипов

Прототип игры - это модель будущей игры, в которую можно поиграть и прочувствовать геймплей. На основании прототипа можно сказать есть будущее у проекта или его нету. Прототип дает возможность геймдизайнеру сразу начать манипулировать с игровым миром, физикой внутри игры, экономикой и процессами. Основной смысл прототипа это проверить идею на практике и уменьшив риски, продолжить разработку проекта.

Super Mario bros.([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Супер братья Марио)— [видеоигра](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80%D0%B0) в жанре [платформера](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D1%80), разработанная и выпущенная в 1985 году японской компанией [Nintendo](https://ru.wikipedia.org/wiki/Nintendo) для платформы [Famicom](https://ru.wikipedia.org/wiki/Famicom). Занесена в «[Книгу рекордов Гиннесса](https://ru.wikipedia.org/wiki/%D0%9A%D0%BD%D0%B8%D0%B3%D0%B0_%D1%80%D0%B5%D0%BA%D0%BE%D1%80%D0%B4%D0%BE%D0%B2_%D0%93%D0%B8%D0%BD%D0%BD%D0%B5%D1%81%D1%81%D0%B0)» как самая продаваемая игра в истории. Внешний вид прототипа представлен на рисунке 1.



Рисунок 1 – «Прототип МАРИО»

# 3. СПЕЦИАЛЬНАЯ ЧАСТЬ

## 3.1 Разработка алгоритма и проектирование структуры базы данных

Модель базы данных «Mario» создана в программе Microsoft SQL Server Management Studio 17.

При создании таблиц базы данных использовался режим конструктора MS SQL представлен на рисунке 2, где были установлены свойства полей всех таблиц данной базы данных.

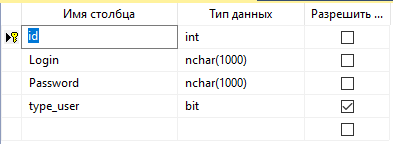


Рисунок 2 – «Создание таблиц»

База данных «Mario» состоит из двух таблиц:

Модель базы данных «Mario» состоит из двух таблиц, каждая из которых имеет свою структуру, необходимую для установления связи между всеми таблицами:

* «Avto» (Пользователи);
* «User1» (Игроки);

Таблица «Avto» предназначена для входа в систему пользователей, при вводе логина и пароля. Ключевым полем является id (Код пользователя), тип данных int. Также имеются поля login (Логин пользователя), тип данных nvarchar(50) и password (Пароль пользователя), тип данных nvarchar(50).

Таблица «User1» имеет связь с таблицей «Avto». Ключевым полем является поле id (Код) тип данных int. Также имеются поля id\_user (Код игрока), по которому проходит соединение с таблицей «Avto».

## 3.2 Разработка программного продукта

При разработке данного программного продукта было реализовано несколько подсистем. Первая из них – система авторизации форма представлена на рисунке 3. В данной подсистеме реализован механизм, который делает проверку на регистрацию в базе программы. Если пользователь уже зарегистрирован, то происходит авторизация, а в противном случае, необходимо пройти регистрацию. Данный механизм реализован посредством простой проверкой существования записи в таблице. Проверка идет SQL запросом в присоединенную к программе базу данных. Код данной части программного продукта представлен на листинге 1.

Листинг 1. Поиск юзера

//создаем подключение к БД

SqlConnection connect = new SqlConnection(Program.st\_connect);

//открываем подключение к БД

connect.Open();

//запрос на поиск зарегистрированных юзеров

string s = "select \* from Avto where login= '" + login + "' and password='" + pass + "' ";

Если пользователь не зарегистрирован в системе, то всплывает сообщение об ошибке представлено на листинге 2.

Листинг 2. Сообщение об ошибке

else

{

MessageBox.Show("Пользователь не найден");

}

Следующая подсистема — это регистрация, пользователь должен заполнить нужные поля для регистрации и последующего входа в систему для начала игры, код представлен на листинге 3.

Листинг 3. Регистрация пользователя

try

{

if (

textBox1.Text == "" ||

textBox2.Text == "")

{

MessageBox.Show("Не все поля заполнены!!");

}

else

{

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "insert into Avto " + " " + " (Login, Password, type\_user) values " +

" ('" + textBox1.Text.Trim() + "' , " +

" '" + textBox2.Text.Trim() + "' , 0 )";

SqlCommand comm = new SqlCommand(s, conn);

comm.ExecuteScalar();

conn.Close();

this.Close();

{

MessageBox.Show("Регистрация прошла успешно! Удачи в игре!!! =)");

}

}

После выполнения команды данные о регистрации пользователя заносятся в таблицу базы данных.

Далее идут две формы, для «Администратора» и «Пользователя», при авторизации происходит проверка типа пользователя представлено на листинге 4.

Листинг 4. Запрос на проверку типа пользователя

if (reader.HasRows)

{

//читаем одну строку из набора данных

reader.Read();

//получаем данные, в скобках это номера столбцов как в запросе!!!

Program.id\_user = reader.GetInt32(0);

Program.type\_user = reader.GetBoolean(3);

if (Program.type\_user == false)

{

MessageBox.Show("Простой юзер");

Form5 f = new Form5();

f.Show();

}

else

{

MessageBox.Show("Привет Админ");

Form6 f = new Form6();

f.Show();

}

}

Из этой формы происходит переход на разные формы, форма «Админа» представлена на рисунке 4, дает ему право на добавление, удаление и изменения данных о пользователях.

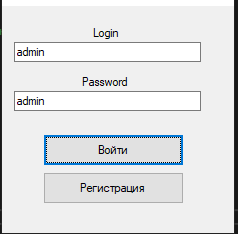


Рисунок 3 – «Форма входа в систему»

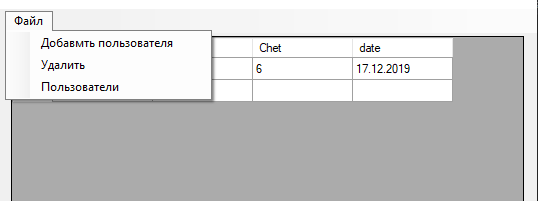


Рисунок 4 – «Форма в режиме админа»

А форма «Пользователя» дает право зарегистрированному пользователю начать играть представлено на рисунке 5.

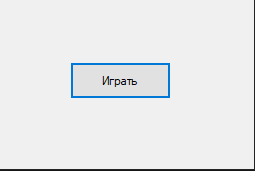


Рисунок 5 – «Форма начала игры»

Реализована форма изменения данных пользователя представлена на листинге 5 и рисунках 6-7.

Листинг 5. Изменение данных

SqlConnection connect = new SqlConnection(Program.st\_connect);

connect.Open();

string s = "update Avto set Login = '" + textBox1.Text.Trim() + "', " +

"Password = '" + textBox2.Text.Trim() +

"' where id = " + Program.id\_user.ToString();

SqlCommand comm = new SqlCommand(s, connect);

comm.ExecuteScalar();

connect.Close();

button1.Visible = false;

button2.Visible = true;

textBox1.Enabled = false;

textBox2.Enabled = false;

{

MessageBox.Show("Ваши данные успешно изменены!");

}

connect.Close();

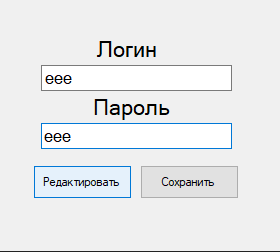


Рисунок 6 – «редактирование данных»

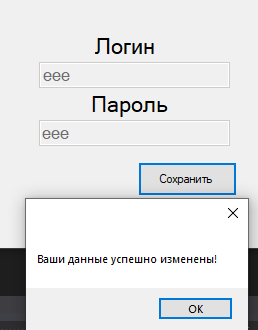


Рисунок 7 – «изменение и сохранение данных»

Далее идет форма, которая доступна только в режиме «Администратора» позволяющая изменять информацию игроков представлена на листинге 6.

Листинг 6. Изменение информации о пользователях.

private void Form8\_Load(object sender, EventArgs e)

{ // TODO: данная строка кода позволяет загрузить данные в таблицу "marioDataSet1.Avto". При необходимости она может быть перемещена или удалена.

this.avtoTableAdapter.Fill(this.marioDataSet1.Avto);

}

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

{

Form4 f = new Form4();

f.Show();

}

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

{

int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "delete from User1 where id = " + Program.id\_user.ToString();

SqlCommand comm = new SqlCommand(s, conn);

{

MessageBox.Show("Пользователь успешно удален!");

}

comm.ExecuteScalar();

conn.Close();

}

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

{ int k = dataGridView1.CurrentRow.Index;

Program.id\_user = Convert.ToInt32(dataGridView1[0, k].Value);

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "select \* from Avto where id = " + Program.id\_user.ToString();

SqlCommand comm = new SqlCommand(s, conn);

SqlDataReader read = comm.ExecuteReader();

read.Read();

Form7 f = new Form7();

f.textBox1.Text = read.GetString(1);

f.textBox2.Text = read.GetString(2);

f.Show();

}

Игра представляет собой бегающего человека «Марио», который уклоняется от падающий бомб с неба и ползающих сов на земле, в тоже самое время с неба падает бонусы в виде монет, которые дает персонажу дополнительные очки, как только игрок набирает 120 очков, бомбы начинают преследовать персонажа и ему становится все сложнее передвигаться по игровому полю не задевая падающие бомбы и других врагов.

Начнем с добавления игровых объектов и системы управления персонажем представлено на листинге 7.

Листинг 7. Добавление объектов и движения

PictureBox[] Bombs = new PictureBox[10];

PictureBox[] Explosives = new PictureBox[10];

PictureBox[] WorldObjects = new PictureBox[10];

Control[] DebugMenu = new Control[9];

PictureBox[] NPC = new PictureBox[2];

Random rng = new Random();

Boolean Player\_Jump = false; //прыжок

Boolean Player\_Left = false; //движение в лево

Boolean Player\_Right = false; //движение в право

Boolean LastDirRight = true;

Boolean GameOn = false; //игра началась?

Boolean GodMode = false;

Boolean Debug = false;

string DebugLog = "STARTED: " + DateTime.Now + "\n";

int Gravity = 20;

int Anim = 0;

int Force = 0;

int BombSize = 16;

int Speed\_Movement = 3;

int Speed\_Jump = 3;

int Speed\_Fall = 3;

int Score = 0;

Далее идет настройка игрового мира с помощью элементов «PictureBox» как видно на листинге 8.

Листинг 8. Настройка PictureBox

public Boolean InAirNoCollision(PictureBox tar)

{ //Проверяет, находится ли целевой Picturebox за пределами кадра

if (!OutsideWorldFrame(tar))

{

foreach (PictureBox Obj in WorldObjects)

{ //Или если он ни с чем не сталкивается

if (!tar.Bounds.IntersectsWith(Obj.Bounds))

{if (tar.Location.Y < WorldFrame.Width)

{ return true;

}

}

}

}

return false;

}

public Boolean OutsideWorldFrame(PictureBox tar)

{ if (tar.Location.X < 0) //Это за пределами левой стороны?

return true;

if (tar.Location.X > WorldFrame.Width) //Это за пределами правой стороны?

return true;

if (tar.Location.Y + tar.Height > WorldFrame.Height - 3)

return true; //или выше предела высоты?

foreach (PictureBox Obj in WorldObjects)

{

if (Obj != null)

{ //Или, пересекаясь с любым мировым объектом

if (tar.Bounds.IntersectsWith(Obj.Bounds))

return true;

}

}

return false;

}

public Boolean Collision\_Top(PictureBox tar)

{

foreach (PictureBox ob in WorldObjects)

{

if (ob != null)

{

PictureBox temp1 = new PictureBox();//Создает один пиксель над целевой картинкой, спрашивает, не сталкивается ли что-либо с ним

temp1.Bounds = ob.Bounds;

//PaintBox(temp1.Location.X, temp1.Location.Y - 1, temp1.Width, 1, Color.Blue);

temp1.SetBounds(temp1.Location.X - 3, temp1.Location.Y - 1, temp1.Width + 6, 1);

if (tar.Bounds.IntersectsWith(temp1.Bounds))

return true;

}

}

Происходит настройка игрового поля, чтобы персонаж не мог выйти за пределы поля расположено на листинге 9.

Листинг 9. Настройка игрового поля

public Boolean Collision\_Left(PictureBox tar)

{

foreach (PictureBox ob in WorldObjects)

{

if (ob != null)

{

PictureBox temp1 = new PictureBox();

temp1.Bounds = ob.Bounds;

//PaintBox(temp1.Location.X - 1, temp1.Location.Y + 1, 1, temp1.Height - 1, Color.Green);

temp1.SetBounds(temp1.Location.X - 1, temp1.Location.Y + 1, 1, temp1.Height - 1);

if (tar.Bounds.IntersectsWith(temp1.Bounds))

return true;

}

}

return false;

}

public Boolean Collision\_Right(PictureBox tar)

{

foreach (PictureBox ob in WorldObjects)

{

if (ob != null)

{

PictureBox temp2 = new PictureBox();

temp2.Bounds = ob.Bounds;

//PaintBox(temp2.Location.X + temp2.Width, temp2.Location.Y + 1, 1, temp2.Height - 1, Color.Yellow);

temp2.SetBounds(temp2.Location.X + temp2.Width, temp2.Location.Y + 1, 1, temp2.Height - 1);

if (tar.Bounds.IntersectsWith(temp2.Bounds))

return true;

}

}

return false;

}

private void timer\_Jump\_Tick(object sender, EventArgs e)

{

if (GameOn)

{

if (Player\_Right && pb\_Player.Right <= WorldFrame.Width - 3 && !Collision\_Left(pb\_Player))

{ //Останавливает игрока от выхода из экрана

pb\_Player.Left += Speed\_Movement; //движение в право

}

if (Player\_Left && pb\_Player.Location.X >= 3 && !Collision\_Right(pb\_Player))

{ //Останавливает игрока от выхода из экрана

pb\_Player.Left -= Speed\_Movement; //движение в лево

}

Настройка движение персонажа, бег в лево, бег в право, прыжок и гравитация расположено на листинге 10.

Листинг 10. Движение персонажа

private void debug\_PGravity\_Click(object sender, EventArgs e)

{

Gravity++;

}

private void debug\_MGravity\_Click(object sender, EventArgs e)

{

Gravity--;

}

private void debug\_PJump\_Click(object sender, EventArgs e)

{

Speed\_Jump++;

}

private void debug\_MJump\_Click(object sender, EventArgs e)

{

Speed\_Jump--;

}

private void debug\_PSpeed\_Click(object sender, EventArgs e)

{

Speed\_Movement++;

}

private void debug\_MSpeed\_Click(object sender, EventArgs e)

{

Speed\_Movement--;

}

pb\_Player.Visible = true; //делает игрока видимым и перемещает его в стартовую локацию

pb\_Player.Location = new System.Drawing.Point(167, WorldFrame.Size.Height - 10 - pb\_Player.Height);

pb\_NPC1.Location = new System.Drawing.Point(1, WorldFrame.Size.Height - 1 - pb\_NPC1.Height);

pb\_NPC2.Location = new System.Drawing.Point(WorldFrame.Width-10, WorldFrame.Size.Height - 1 - pb\_NPC2.Height);

pb\_Player.Image = Character.stand\_r;

case Keys.Left: // нажатие на левую клавишу

if (GameOn)

{

LastDirRight = false; //Для анимации жмите направо или налево

Player\_Left = true; //движение в лево

}

break;

case Keys.Right: // нажатие на правую клавишу

if (GameOn)

{

LastDirRight = true;

Player\_Right = true;

}

break;

case Keys.Space: // нажатие на пробел

if (label\_Dead.Visible && !label\_Dead.Text.Contains("Пауза"))

{ //нажатие на пробел после смерти

Reset(); //перезапускается игра

}

else

{

if (!Player\_Jump && !InAirNoCollision(pb\_Player))

{ //Anti multijump - если игрок не прыгает, находится в воздухе и не сталкивается ни с чем

Создание падающих бомб с неба расположено на листинге 11.

Листинг 11. Падение бомб

public void RemovePictureBoxAt(int x, int y)

{

foreach (PictureBox Boom in Explosives)

{

if (Boom != null)

{

if (Boom.Location.X == x && Boom.Location.Y == y)

{

Boom.Dispose();

}

}

}

}

public void Reset()

{ //Сбрасывает все

label\_Dead.Visible = false;

int x = 0;

foreach (PictureBox Bomb in Bombs)

{

if (Bomb != null)

{ //Удаляет все бомбы

Bomb.Dispose();

Bombs[x] = null; ;

}

}

public void CreateBoom(int x, int y)

{ //Создает картинку для взрывов

PictureBox Boom = new PictureBox();

Boom.Name = "Boom";

Boom.BackColor = Color.Transparent;

Boom.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

Boom.Size = new System.Drawing.Size(BombSize, BombSize);

Boom.Image = World.Boom;

Boom.Location = new System.Drawing.Point(x, y);

WorldFrame.Controls.Add(Boom);

Explosives[0] = Boom;

}

Кнопочный функционал «пауза в игре», «начать заново», «движение в лево и право», «прыжок» расположено на листинге 12.

Листинг 12. Кнопочный функционал

switch (e.KeyCode)

{

case Keys.Z:

Score += 120;

break;

case Keys.X:

pb\_Player.Top+=3;

break;

case Keys.P: // нажатие на клавишу З

if (GameOn)

{

GameOn = false; //пауза в игре

label\_Dead.Text = "Пауза, нажмите З чтобы продолжить";

label\_Dead.Visible = true;

}

else

{

GameOn = true; //Игра возобновляется

label\_Dead.Text = "Вы мертвы, нажмите пробел чтобы начать заново";

label\_Dead.Visible = false;

if (label\_Dead.Visible == false)

{

}

}

break;

case Keys.Left: // нажатие на левую клавишу

if (GameOn)

{

LastDirRight = false; //Для анимации жмите направо или налево

Player\_Left = true; //движение в лево

}

break;

case Keys.Right: // нажатие на правую клавишу

if (GameOn)

{

LastDirRight = true;

Player\_Right = true;

}

break;

case Keys.Space: // нажатие на пробел

if (label\_Dead.Visible && !label\_Dead.Text.Contains("Пауза"))

{ //нажатие на пробел после смерти

Reset(); //перезапускается игра

}

Расположение отображения подсчета очков, рекорда, количество падающих бомб, на приложении находятся в верхнем левом углу, показано на листинге 13.

Листинг 13. Отображение очков, рекорда и падающих бомб

private void timer\_Anim\_Tick(object sender, EventArgs e)

{

//PBSideCollision(pb\_Player, pb\_Pipe);

Anim++;

label1.Text = "Бомбы: " + GetBombsNum(Bombs); //Устанавливает количество меток бомб

label2.Text = "Рекорд: " + Properties.Settings.Default.Highscore;

if (Player\_Right == true && Anim % 15 == 0)

{

pb\_Player.Image = Character.walk\_r;

}

if (Player\_Left == true && Anim % 15 == 0)

{

pb\_Player.Image = Character.walk\_l;

}

foreach (PictureBox Bomb in Bombs)

{ //Взаимодействие бомб, проверяет каждую бомбу на поле

if (Bomb != null)

{ //Если бомба не существует

if (pb\_Player.Bounds.IntersectsWith(Bomb.Bounds))

{

if (Bomb.Name == "Coin")

{ //Если это Монета, добавьте 5 к счету и удалите Монету, чтобы он не получал очки каждую мили секунду

Score++;

Bomb.Dispose();

}

else

{

Dead();

Bomb.Dispose();

}

}

}

}

Расположение персонажа на игровом поле, взаимодействие его с объектами на поле показано на листинге 14.

Листинг 14. Взаимодействие персонажа с объектами на поле

private void timer\_Gravity\_Tick(object sender, EventArgs e)

{

if (!Player\_Jump && pb\_Player.Location.Y + pb\_Player.Height < WorldFrame.Height - 2 && !Collision\_Top(pb\_Player))

{ //Если игрок не прыгает, локация находится над полом или стоит на объекте

pb\_Player.Top += Speed\_Fall; //Player falls

}

if (!Player\_Jump && pb\_Player.Location.Y + pb\_Player.Height > WorldFrame.Height - 1)

{ //Если игрок по какой-то причине окажется под полом, поднимите его

pb\_Player.Top--;

}

int x = 0;

if (GameOn)

{

foreach (PictureBox Bomb in Bombs)

{

if (Bomb != null)

{

try

{

if (!OutsideWorldFrame(Bomb))

{ //если Бомба находится над землей

if (Bomb.Name == "pb" || Bomb.Name == "Coin")

{

Bomb.Top += 3;

}

if (Bomb.Name == "pbR")

{

Bomb.Left += 3;

}

if (Bomb.Name == "pbL")

{

Bomb.Left -= 3;

}

}

else

{

if (OutsideWorldFrame(Bomb))

{

Bombs[x] = null;

Bomb.Dispose();

DebugLog += DateTime.Now + ": Removed rocket at " + x + "\n";

}

Падение бомб в случайном порядке и локации на поле показано на листинге 15.

Листинг 15. Случайное падение бомб

private void timer\_Randombomb\_Tick(object sender, EventArgs e)

{

Random rng = new Random();

if (GameOn || (!GameOn && !label\_Dead.Visible))

if (GetBombsNum(Bombs) == 10)

{

timer\_BombFailsafe.Enabled = true;

}

else

{

timer\_BombFailsafe.Enabled = false;

{

int r = 2;

int NextSpot = NextBomb(Bombs);

if (Score > 20 && Score < 40) r = 12;

if (Score > 40 && Score < 80) r = 13;

if (Score > 80) r = 14;

switch (rng.Next(1, r))

{

case 1:

case 2:

case 3:

case 4:

case 5:

case 6:

case 7:

case 8:

NextSpot = NextBomb(Bombs);

PictureBox pb = new PictureBox();

pb.Name = "pb";

pb.BackColor = Color.Transparent;

pb.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

pb.Size = new System.Drawing.Size(16, 16);

pb.Image = Enemy.Bomb;

if (Score > 120)

{

pb.Location = new System.Drawing.Point(rng.Next(pb\_Player.Location.X-10, pb\_Player.Location.X+10), 0);

}

else

{

pb.Location = new System.Drawing.Point(rng.Next(0, WorldFrame.Width), 0);

}

Ведение подсчета очков, количества бомб показано на листинге 16.

Листинг 16. Ведение подсчета очков

private void timer\_Sec\_Tick(object sender, EventArgs e)

{

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

{

if (Bombs[i] != null && Bombs[i].IsDisposed)

{

Bombs[i] = null;

}

}

label\_Score.Text = "Очки: " + Score;

if (!label\_Dead.Visible)

{

Score++;

BombSize++;

if (timer\_Randombomb.Interval > 1)

{

timer\_Randombomb.Interval--;

}

if (Score > Properties.Settings.Default.Highscore)

{

Properties.Settings.Default.Highscore = Score;

Properties.Settings.Default.Save();

}

}

}

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

{

foreach (Control X in this.Controls)

{

if (X is PictureBox)

{

if (X.Name == "Boom")

{

X.Dispose();

}

}

}

Прорисовка игровых объектов на поле показано на листинге 17.

Листинг 17. Прорисовка объектов

private void Form1\_Load(object sender, EventArgs e)

{

Reset();

WorldObjects[0] = pb\_Pipe;

WorldObjects[1] = pb\_Block1;

WorldObjects[2] = pb\_Block2;

DebugMenu[0] = this.debug\_Log;

DebugMenu[1] = this.debug\_Godmode;

DebugMenu[2] = this.debug\_NoBombs;

DebugMenu[3] = this.debug\_PGravity;

DebugMenu[4] = this.debug\_MGravity;

DebugMenu[5] = this.debug\_PJump;

DebugMenu[6] = this.debug\_MJump;

DebugMenu[7] = this.debug\_PSpeed;

DebugMenu[8] = this.debug\_MSpeed;

NPC[0] = pb\_NPC1;

NPC[1] = pb\_NPC2;

}

Реализация записи с помощью sql запроса в таблицу БД результата игры пользователя показано на листинге 18.

Листинг 18. Запись результата в таблицу

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

{

DateTime dateTimeVariable = DateTime.Now;

SqlConnection conn = new SqlConnection(Program.st\_connect);

conn.Open();

string s = "INSERT INTO USER1 (id\_user, Chet, date) values " +

"('" + Program.id\_user + "'," +

"'" + Score + "'," +

"'" + dateTimeVariable.ToString() + "')";

SqlCommand command = new SqlCommand(s, conn);

SqlDataReader reader = command.ExecuteReader();

conn.Close();

}

Полный код данного продукта предоставлен в приложении Б.

## 3.3 Руководство пользователя

Запуск приложения. Программа запускается через Visual Studio 17, где пользователь при запуске попадает на форму «Авторизация» представлена на рисунке 8.

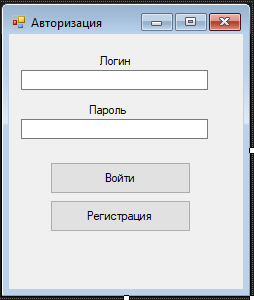


Рисунок 8 – «форма авторизации»

Если пользователь зарегистрирован в системе, он сможет пройти на форму «начала игры» представлена на рисунке 9.

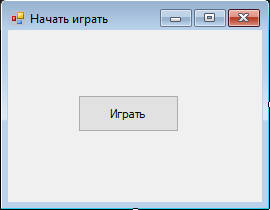


Рисунок 9 – «форма начала игры»

А если пользователь не зарегистрирован в системе, тогда он переходит на форму «Регистрация» представлена на рисунке 10.

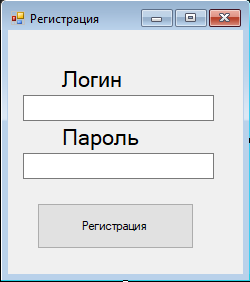


Рисунок 10 – «форма регистрация»

После того, как пользователь зарегистрируется в системе и пройдет авторизацию он может приступить к началу игры представлено на рисунке 11, управление персонажем происходит с помощью клавиатуры, а именно: «стрелки в лево и право» для передвижения по сторонам, кнопка «пробел» для прыжка персонажа, кнопка «З» для паузы в игре.

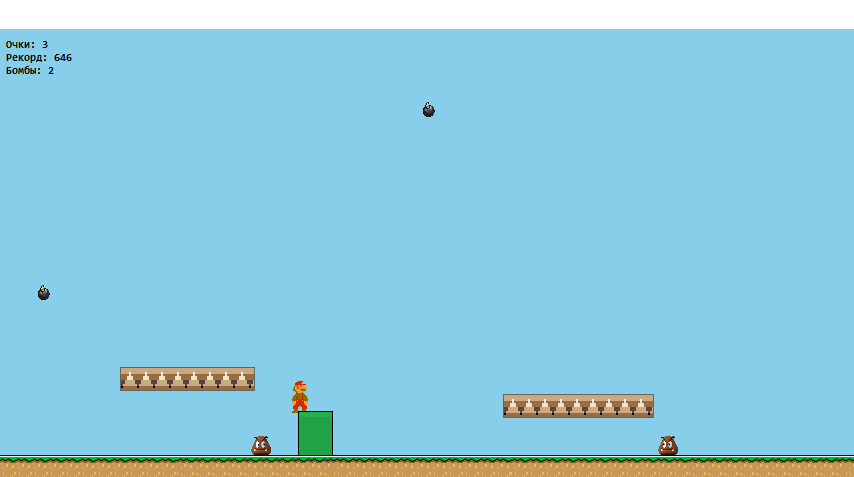


Рисунок 11 – «игровое окно»

После игры, если пользователь умирает ему предлагается начать игру заново, представлено на рисунке 12.

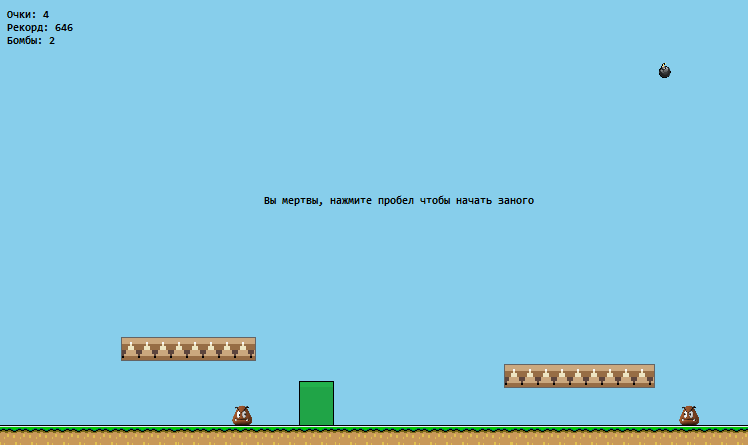


Рисунок 12 – «начала игры заново»

В конце игры весь результат записывается в базу данных и хранится в ней пока Администратор ее не удалит.

# ЗАКЛЮЧЕНИЕ

В результате курсовой работы было проведено исследование предметной области, разработано техническое задание, выбрано и обосновано средство реализации БД, проанализирован прототип, была спроектирована база данных. Так же выбрано и обосновано программное средство реализации данного проекта, была разработана игра «Марио» с использованием СУБД MS SQL-server, изучена среда разработки, произведен сбор данных об объектах и разработано руководство пользователя.

Разработанный проект позволит пользователям играть в игру «Марио».

В дальнейшем планируется провести более детальную проработку интерфейса приложения. Планируется улучшить работоспособность программы, добавить различные уровни сложности.

Цель курсового проекта, а именно разработка игры «Марио» с использованием СУБД MSSQL-server и Visual Studio 17 достигнута.

Программа разработана, процессы, связанные с обработкой информации и ее обменом с пользователем изучены. Регистрация игроков, просмотр и изменение их данных, все данные об игроках попадают в таблицы базы данных MSSQL-server.

Цель достигнута, задачи выполнены.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Васильев, Алексей C#. Объектно-ориентированное программирование / Алексей Васильев. - М.: Питер, 2012. - 320 c.
2. Подбельский, В. В. Язык С#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, 2013. - 408 c.
3. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 633 c
4. Вишневский, Алексей Microsoft SQL Server. Эффективная работа / Алексей Вишневский. - М.: Питер, 2009. - 143 c.
5. Ицик, Бен-Ган Microsoft SQL Server 2012. Высокопроизводительный код T-SQL. Оконные функции / Бен-Ган Ицик. - М.: Русская Редакция, 2013. - 861 c.

# ПРИЛОЖЕНИЯ

Приложение Б.

Полный код курсового проекта

Листинг 7. Добавление объектов и движения

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using BombsAway.Properties;

using System.Data.SqlClient;

namespace BombsAway

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

#region Credits

#endregion

#region Vars

PictureBox[] Bombs = new PictureBox[10];

PictureBox[] Explosives = new PictureBox[10];

PictureBox[] WorldObjects = new PictureBox[10];

Control[] DebugMenu = new Control[9];

PictureBox[] NPC = new PictureBox[2];

Random rng = new Random();

Boolean Player\_Jump = false; //прыжок

Boolean Player\_Left = false; //движение в лево

Boolean Player\_Right = false; //движение в право

Boolean LastDirRight = true;

Boolean GameOn = false; //игра началась?

Boolean GodMode = false;

Boolean Debug = false;

string DebugLog = "STARTED: " + DateTime.Now + "\n";

int Gravity = 20;

int Anim = 0;

int Force = 0;

int BombSize = 16;

int Speed\_Movement = 3;

int Speed\_Jump = 3;

int Speed\_Fall = 3;

int Score = 0;

#endregion

#region Boolean Functions, "Check functions"

Листинг 8. Настройка PictureBox

public Boolean InAirNoCollision(PictureBox tar)

{ //Проверяет, находится ли целевой Picturebox за пределами кадра

if (!OutsideWorldFrame(tar))

{

foreach (PictureBox Obj in WorldObjects)

{ //Или если он ни с чем не сталкивается

if (!tar.Bounds.IntersectsWith(Obj.Bounds))

{

if (tar.Location.Y < WorldFrame.Width)

{

return true;

}

}

}

}

return false;

}

Листинг 9. Настройка игрового поля

public Boolean OutsideWorldFrame(PictureBox tar)

{

if (tar.Location.X < 0) //Это за пределами левой стороны?

return true;

if (tar.Location.X > WorldFrame.Width) //Это за пределами правой стороны?

return true;

if (tar.Location.Y + tar.Height > WorldFrame.Height - 3)

return true; //или выше предела высоты?

foreach (PictureBox Obj in WorldObjects)

{

if (Obj != null)

{ //Или, пересекаясь с любым мировым объектом

if (tar.Bounds.IntersectsWith(Obj.Bounds))

return true;

}

}

return false;

}

public Boolean Collision\_Top(PictureBox tar)

{

foreach (PictureBox ob in WorldObjects)

{

if (ob != null)

{

PictureBox temp1 = new PictureBox();//Создает один пиксель над целевой картинкой, спрашивает, не сталкивается ли что-либо с ним

temp1.Bounds = ob.Bounds;

//PaintBox(temp1.Location.X, temp1.Location.Y - 1, temp1.Width, 1, Color.Blue);

temp1.SetBounds(temp1.Location.X - 3, temp1.Location.Y - 1, temp1.Width + 6, 1);

if (tar.Bounds.IntersectsWith(temp1.Bounds))

return true;

}

}

return false;

}

public Boolean Collision\_Bottom(PictureBox tar)

{

foreach (PictureBox ob in WorldObjects)

{

if (ob != null)

{

PictureBox temp1 = new PictureBox();

temp1.Bounds = ob.Bounds;

//PaintBox(temp1.Location.X, temp1.Location.Y+temp1.Height, temp1.Width, 1, Color.Red);

temp1.SetBounds(temp1.Location.X, temp1.Location.Y + temp1.Height, temp1.Width, 1);

if (tar.Bounds.IntersectsWith(temp1.Bounds))

return true;

}

}

return false;

}

public Boolean Collision\_Left(PictureBox tar)

{

foreach (PictureBox ob in WorldObjects)

{

if (ob != null)

{

PictureBox temp1 = new PictureBox();

temp1.Bounds = ob.Bounds;

//PaintBox(temp1.Location.X - 1, temp1.Location.Y + 1, 1, temp1.Height - 1, Color.Green);

temp1.SetBounds(temp1.Location.X - 1, temp1.Location.Y + 1, 1, temp1.Height - 1);

if (tar.Bounds.IntersectsWith(temp1.Bounds))

return true;

}

}

return false;

}

public Boolean Collision\_Right(PictureBox tar)

{

foreach (PictureBox ob in WorldObjects)

{

if (ob != null)

{

PictureBox temp2 = new PictureBox();

temp2.Bounds = ob.Bounds;

//PaintBox(temp2.Location.X + temp2.Width, temp2.Location.Y + 1, 1, temp2.Height - 1, Color.Yellow);

temp2.SetBounds(temp2.Location.X + temp2.Width, temp2.Location.Y + 1, 1, temp2.Height - 1);

if (tar.Bounds.IntersectsWith(temp2.Bounds))

return true;

}

}

return false;

}

#endregion

#region Clicks

Листинг 10. Движение персонажа

private void debug\_PGravity\_Click(object sender, EventArgs e)

{

Gravity++;

}

private void debug\_MGravity\_Click(object sender, EventArgs e)

{

Gravity--;

}

private void debug\_PJump\_Click(object sender, EventArgs e)

{

Speed\_Jump++;

}

private void debug\_MJump\_Click(object sender, EventArgs e)

{

Speed\_Jump--;

}

private void debug\_PSpeed\_Click(object sender, EventArgs e)

{

Speed\_Movement++;

}

private void debug\_MSpeed\_Click(object sender, EventArgs e)

{

Speed\_Movement--;

}

pb\_Player.Visible = true; //делает игрока видимым и перемещает его в стартовую локацию

pb\_Player.Location = new System.Drawing.Point(167, WorldFrame.Size.Height - 10 - pb\_Player.Height);

pb\_NPC1.Location = new System.Drawing.Point(1, WorldFrame.Size.Height - 1 - pb\_NPC1.Height);

pb\_NPC2.Location = new System.Drawing.Point(WorldFrame.Width-10, WorldFrame.Size.Height - 1 - pb\_NPC2.Height);

pb\_Player.Image = Character.stand\_r;

Score = 0;

BombSize = 16;

GameOn = true;

case Keys.Left: // нажатие на левую клавишу

if (GameOn)

{

LastDirRight = false; //Для анимации жмите направо или налево

Player\_Left = true; //движение в лево

}

break;

case Keys.Right: // нажатие на правую клавишу

if (GameOn)

{

LastDirRight = true;

Player\_Right = true;

}

break;

case Keys.Space: // нажатие на пробел

if (label\_Dead.Visible && !label\_Dead.Text.Contains("Пауза"))

{ //нажатие на пробел после смерти

Reset(); //перезапускается игра

}

else

{

if (!Player\_Jump && !InAirNoCollision(pb\_Player))

{ //Anti multijump - если игрок не прыгает, находится в воздухе и не сталкивается ни с чем

if (LastDirRight) //Проверяет направление, меняет изображение прыжка

{

pb\_Player.Image = Character.jump\_r;

}

else

{

pb\_Player.Image = Character.jump\_l;

}

pb\_Player.Top -= Speed\_Jump; //Игрок немного поднимается

Force = Gravity; //Заставить двигаться вверх изменения

Player\_Jump = true; //Устанавливает переменную, которую прыгает игрок

}

}

break;

}

}

Листинг 11. Падение бомб

public void Reset()

{ //Сбрасывает все

label\_Dead.Visible = false;

int x = 0;

foreach (PictureBox Bomb in Bombs)

{

if (Bomb != null)

{ //Удаляет все бомбы

Bomb.Dispose();

Bombs[x] = null; ;

}

x++;

}

int x2 = 0;

foreach (PictureBox Boom in Explosives)

{

if (Boom != null)

{ //Удаляет все бомбы

Boom.Dispose();

Bombs[x2] = null; ;

}

x2++;

}

public void CreateBoom(int x, int y)

{ //Создает картинку для взрывов

PictureBox Boom = new PictureBox();

Boom.Name = "Boom";

Boom.BackColor = Color.Transparent;

Boom.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

Boom.Size = new System.Drawing.Size(BombSize, BombSize);

Boom.Image = World.Boom;

Boom.Location = new System.Drawing.Point(x, y);

WorldFrame.Controls.Add(Boom);

Explosives[0] = Boom;

}

foreach (PictureBox Bomb in Bombs)

{ //Взаимодействие бомб, проверяет каждую бомбу на поле

if (Bomb != null)

{ //Если бомба не существует

if (pb\_Player.Bounds.IntersectsWith(Bomb.Bounds))

{

if (Bomb.Name == "Coin")

{ //Если это Монета, добавьте 5 к счету и удалите Монету, чтобы он не получал очки каждую мили секунду

Score++;

Bomb.Dispose();

}

else

{

Dead();

Bomb.Dispose();

}

}

}

}

Листинг 12. Кнопочный функционал

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

{

switch (e.KeyCode)

{

case Keys.Z:

Score += 120;

break;

case Keys.X:

pb\_Player.Top+=3;

break;

case Keys.P: // нажатие на клавишу З

if (GameOn)

{

GameOn = false; //пауза в игре

label\_Dead.Text = "Пауза, нажмите З чтобы продолжить";

label\_Dead.Visible = true;

}

else

{

GameOn = true; //Игра возобновляется

label\_Dead.Text = "Вы мертвы, нажмите пробел чтобы начать заново";

label\_Dead.Visible = false;

if (label\_Dead.Visible == false)

{

}

}

Листинг 13. Отображение очков, рекорда и падающих бомб

private void timer\_Anim\_Tick(object sender, EventArgs e)

{

//PBSideCollision(pb\_Player, pb\_Pipe);

Anim++;

label1.Text = "Бомбы: " + GetBombsNum(Bombs); //Устанавливает количество меток бомб

label2.Text = "Рекорд: " + Properties.Settings.Default.Highscore;

if (Player\_Right == true && Anim % 15 == 0)

{

pb\_Player.Image = Character.walk\_r;

}

if (Player\_Left == true && Anim % 15 == 0)

{

pb\_Player.Image = Character.walk\_l;

}

foreach (PictureBox Bomb in Bombs)

{ //Взаимодействие бомб, проверяет каждую бомбу на поле

if (Bomb != null)

{ //Если бомба не существует

if (pb\_Player.Bounds.IntersectsWith(Bomb.Bounds))

{

if (Bomb.Name == "Coin")

{ //Если это Монета, добавьте 5 к счету и удалите Монету, чтобы он не получал очки каждую мили секунду

Score++;

Bomb.Dispose();

}

else

{

Dead();

Bomb.Dispose();

}

}

}

}

Листинг 14. Взаимодействие персонажа с объектами на поле

private void timer\_Gravity\_Tick(object sender, EventArgs e)

{

if (!Player\_Jump && pb\_Player.Location.Y + pb\_Player.Height < WorldFrame.Height - 2 && !Collision\_Top(pb\_Player))

{ //Если игрок не прыгает, локация находится над полом или стоит на объекте

pb\_Player.Top += Speed\_Fall; //Player falls

}

if (!Player\_Jump && pb\_Player.Location.Y + pb\_Player.Height > WorldFrame.Height - 1)

{ //Если игрок по какой-то причине окажется под полом, поднимите его

pb\_Player.Top--;

}

int x = 0;

if (GameOn)

{

foreach (PictureBox Bomb in Bombs)

{

if (Bomb != null)

{

try

{

if (!OutsideWorldFrame(Bomb))

{ //если Бомба находится над землей

if (Bomb.Name == "pb" || Bomb.Name == "Coin")

{

Bomb.Top += 3;

}

if (Bomb.Name == "pbR")

{

Bomb.Left += 3;

}

if (Bomb.Name == "pbL")

{

Bomb.Left -= 3;

}

}

else

{

if (OutsideWorldFrame(Bomb))

{

Bombs[x] = null;

Bomb.Dispose();

DebugLog += DateTime.Now + ": Removed rocket at " + x + "\n";

}