Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ИГРОВОЕ СРЕДСТВО «мОНОПОЛИЯ»**

|  |  |  |
| --- | --- | --- |
| Студент |  | Новиков О.А. |
| Руководитель |  | Шулицкий Д.С. |

Минск 2019

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc28474164)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc28474165)

[Обзор аналогов 7](#_Toc28474166)

[1.1.1 Monopoly Plus 7](#_Toc28474167)

[1.1.2 Monopoly 3D 8](#_Toc28474168)

[Постановка задачи 10](#_Toc28474169)

[2 РАЗРАБОТКА игрового СРЕДСТВА 11](#_Toc28474170)

[2.1 Структура программы 11](#_Toc28474171)

[2.2 Интерфейс программного средства 11](#_Toc28474172)

[2.3 Отрисовка и перемещение фишек игроков 12](#_Toc28474173)

[2.4 Логическая часть программы 16](#_Toc28474174)

[3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 20](#_Toc28474175)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 22](#_Toc28474176)

[**Приложение А** 38](#_Toc28474177)

ВВЕДЕНИЕ

«Монополия» — экономическая и стратегическая настольная игра для двух и более человек. Получила большую популярность в XX веке во многих странах мира, включая СССР; в последнем была также известна под названиями «Менеджер», «Империя», «Бизнесмен».

Цель игры — рационально используя стартовый капитал, добиться банкротства других игроков. Фактически «Монополия» представляет собой игровое поле, состоящее из квадратов, которые проходят по кругу все игроки по очереди. Квадраты разделяются на активы (предприятие, ценная вещь) и события. Когда игроку выпадает очередь ходить, то броском кубика он определяет, какое количество шагов он должен совершить на игровом поле за этот ход (каждый шаг соответствует одному очку на кубике и одному квадрату на игровом поле).

В настоящее время имеются многочисленные компьютерные воспроизведения «Монополии», а также вариации для игровых консолей и сотовых телефонов.

В 1934 году в разгар Великой депрессии Чарльз Дэрроу из Германтауна, штат Пенсильвания, показал представителям компании Паркер Бразерс проект игры Монополия. Проект был отклонен из-за «52 ошибок в дизайне», но это не остановило безработного Чарльза, который решил выпустить игру своими силами.

В 1935 году с помощью друга Чарльз Дэрроу изготовил и продал 5000 самодельных копий «Монополии» в универмаге Филадельфии. Через некоторое время он уже не мог справляться с возросшим спросом, и решил снова обратиться к представителям Паркер Бразерс. Монополия вошла в историю — уже в 1936 году она стала самой продаваемой игрой в США.

На сегодняшний день число играющих в Монополию перевалило за полмиллиарда человек. В мире проводятся коммерческие соревнования (в том числе мировые чемпионаты) по игре Монополия с существенными призовыми фондами.

Игроки поочередно кидают игральные кости и делают соответствующее количество ходов на игровом поле. Встав на поле с активом, игрок может приобрести его, если он свободен, а если он принадлежит другому игроку, то игрок обязан заплатить за посещение данного поля по установленному правилами прейскуранту.

При посещении поля с событиями игрок получает указание следовать выпавшему ему событию. Пример: получить дополнительные средства из Банка (Банк — это ещё один игрок, который взаимодействует с игроками-людьми посредством событий, происходящих в игре), отдать часть своих средств Банку, передвинуться на другое поле, отдать часть своих средств другим игрокам, пропустить свой следующий ход, получить дополнительный ход и другие.

Данное игровое средство представляет собой современную версию настольной игры. В отличии от оригинальной версии игры в качестве полей используются современные мировые компании.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Обзор аналогов

В настоящее время существует огромное количество приложений, каждое из которых имеет свои преимущества и недостатки. На рисунках 1.1 и 1.2 представлена Monopoly Plus.

1.1.1 Monopoly Plus



Рисунок 1.1 – Monopoly Plus

Вы можете менять правила игры или адаптировать их под себя. Также можно активировать режим «Игра на выживание» или выбирать один из 6 вариантов правил, которые признали лучшими поклонники «Монополии» по всему миру. Также есть возможность играть онлайн. Однако данная игра является платной. За 5,46$ эту игру можно приобрести в Steam.



Рисунок 1.2 – Monopoly Plus

1.1.2 Monopoly 3D

Существует ещё аналог - Monopoly 3D. Как из названия можно догадаться, это трёхмерная версия игры. В компьютерной версии, как и в реале, можно играть нескольким игрокам. Игровое поле игры классическое. Здесь есть улицы, предприятия, магазины, фабрики и другие объекты, которые вы можете купить.



Рисунок 1.3 – Monopoly 3D

Есть и другие поля, попадая в которые вы можете лишиться части денег, заплатить штраф, пропустить ход и другие, малоприятные моменты. В центре поля есть карточки: Шанс и Общественный фонд. Открывая ту или иную карточку, вы можете получить бонусы, наследство, или заплатить, к примеру штраф. Эти карточки также влияют на ваше благосостояние. Чем больше количество объектов на поле вы купите, тем больше ваш достаток.



Рисунок 1.4 – Monopoly 3D

Постановка задачи

В рамках данного курсового проекта планируется разработка программного игрового средства «Монополия».

В программном средстве планируется реализовать следующие функции:

отрисовка фишек игроков различного цвета;

добавление новых игроков в игру;

покупка недвижимости, представленной в виде полей;

отрисовка стоимости поля, а также его аренда;

для перемещения игроков по игровому полю будут использоваться игральные кости;

для покупки и аренды полей будет использоваться игровая валюта «монобаксы»;

в качестве бонуса за пройденный круг игрокам будет предоставлено денежное вознаграждение;

в случае проигрыша, то есть банкротства, или при необходимости выйти игрок может сдаться и выйти из игры;

возможность вести диалог между игроками, с помощью специального чата, в котором также по ходу игры отображается история действий игроков;

Для разработки программного средства будет использоваться язык программирования C++, операционная система Windows 10 и среды разработки Borland С++ Builder6 и Visual Studio 2017.

1. РАЗРАБОТКА игрового СРЕДСТВА

2.1 Структура программы

Данное приложение включает в себя единственный модуль Unit1, в котором отображается главное окно с графическим интерфейсом игры и обеспечивается взаимодействие с пользователем, а также логическая часть, отвечающая за функционал программы.

В ходе разработки и создания программного средства были использованы две среды программирования, на это есть ряд причин. Visual Studio 2017 предоставляет огромные возможности для редактирования исходного кода, лексического, синтаксического и даже семантического анализа написанного кода в реальном времени, и удобную подсветку синтаксиса, в [1, 5] описана структура языка. Эти факторы позволяют ускорить процесс разработки и избежать непредвиденных ошибок, что сэкономило значительное количество времени в процессе тестирования серверной части. Недостатком же является отсутствие простого создания интерфейса. Именно поэтому для разработки клиентской части была выбрана среда Borland C++ Builder6 Документация по использованию среды описана в [3, 4].

Данный подход не только позволил эффективно разрабатывать приложение, но и показать совместимость структурных блоков, полученных в результате работы в различных средах.

2.2 Интерфейс программного средства

Удобный пользовательский интерфейс и лаконичный вид являются одними из главных критериев качества программного продукта, в следствии этого необходимо создать максимально интуитивный и простой интерфейс программы.

Весь пользовательский интерфейс был реализован в среде разработки Borland C++ Builder6.

В качестве элементов взаимодействия между программой и пользователем были выбраны понятные и удобные компоненты TBitBtn, TImage, TEdit, TPanel, TLabel, TMemo, TGroupBox, TShape, TTimer, работа с данными компонентами документирована в [2, 6]. Для отрисовки полей игрового пространства используется компонент TImage, позволяющий отображать логотипы компаний. Для отрисовки фишек игроков используется компонент TShape, позволяющий рисовать геометрические фигуры различными цветами. Для отображения стоимости или аренды поля, а также отображения профиля игроков используется компонент TPanel. Для отображения истории игры и ведения диалога между игроками используется компонент TMemo, а для ввода сообщений и имени игроков используется компонент TEdit. Для отображения победителя игры и вспомогательных сообщений используется компонент TLabel. В свою очередь компонент TGroupBox используется для возможности покупки, оплаты полей, а также для генерирования случайных значений кубиков.

На рисунке 2.1 представлено главное окно игры.

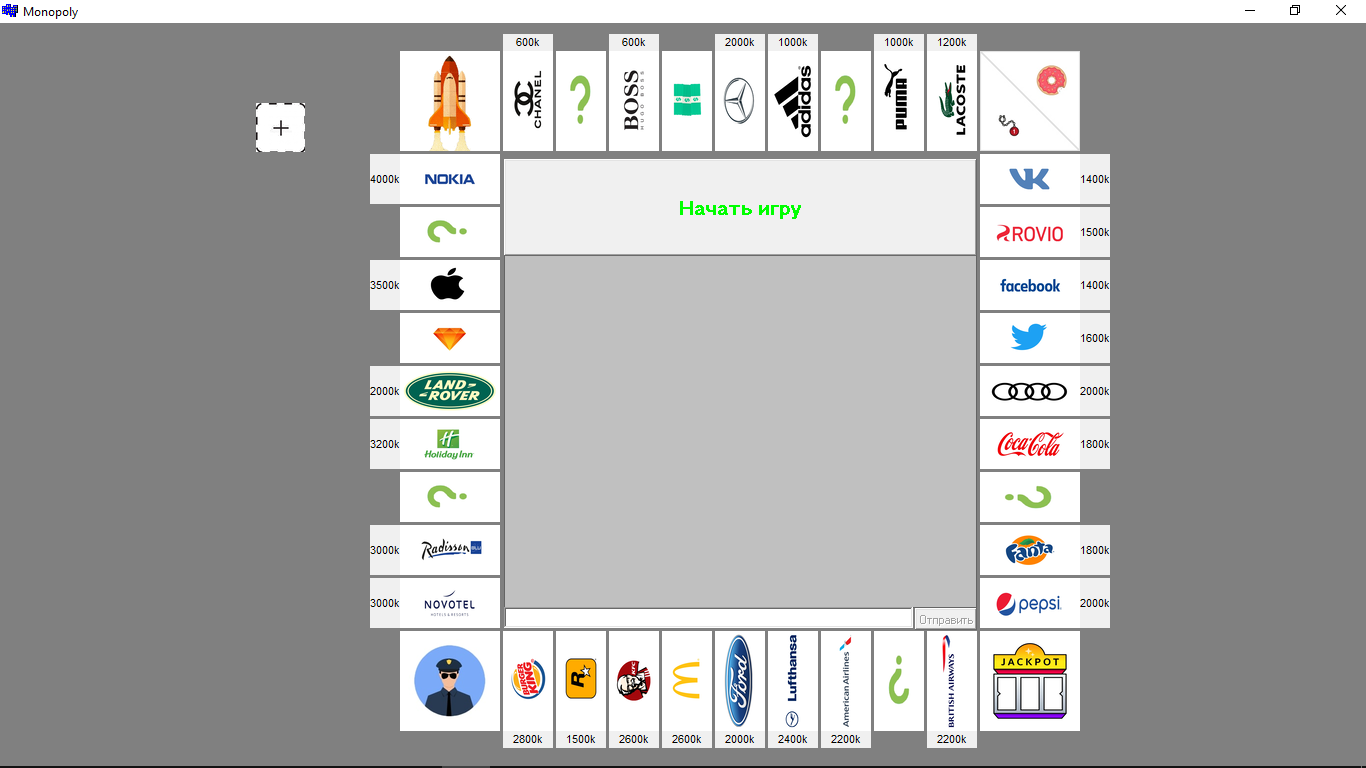


Рисунок 2.1 – Главное окно игры

Интерфейс игрового средства визуально получился достаточно компактным и удобным, по сравнению с аналогами, что позволяет достаточно быстро начать использовать программу новому пользователю, а также достаточно большая игровая зона и привлекательный интерфейс обеспечивает больший комфорт использования программного продукта.

2.3 Отрисовка и перемещение фишек игроков

Отрисовка фигур игроков происходит в зависимости от количества участников игры. В игре может принимать участие от 2 до 4 игроков. Алгоритм отрисовки и выбора количества отрисованных фигур с помощью компонента TShape отображен ниже.

void \_\_fastcall TMonopoly::BitBtn1Click(TObject \*Sender)

{

if (AmountOfPlayers > 1) {

DrawShapes();

if (AmountOfPlayers == 2) {

Player3Shape -> Hide();

Player4Shape -> Hide();

Panel3 -> Enabled = false;

Panel4 -> Enabled = false;

} else if (AmountOfPlayers == 3) {

Player4Shape -> Hide();

Panel4 -> Enabled = false;

}

SendMessageButton -> Enabled = true;

Edit1 -> Enabled = true;

if (AmountOfPlayers == 2) Player3 -> Hide();

else if (AmountOfPlayers == 3) Player4 -> Hide();

pl = head;

pl = pl -> next;

ColorPlayer();

BitBtn1 -> Hide();

} else ShowMessage("Для игры необходимо минимум 2 игрока");

}

Функция DrawShapes() отвечает за отображение в главном окне игры фишек игроков:

void DrawShapes() {

Monopoly -> GroupBox1 -> Show();

Monopoly -> Player1Shape -> Show();

Monopoly -> Player2Shape -> Show();

Monopoly -> Player3Shape -> Show();

Monopoly -> Player4Shape -> Show();

Monopoly -> Panel1 -> Enabled = true;

Monopoly -> Panel2 -> Enabled = true;

Monopoly -> Panel3 -> Enabled = true;

Monopoly -> Panel4 -> Enabled = true;

}

Функция ColorPlayer() отвечает за раскраску фишек игроков в зависимости от очереди бросать кубик:

void ColorPlayer() {

if (pl -> color == 1) Monopoly -> Player1 -> Color = clRed;

else if (pl -> color == 2) Monopoly -> Player2 -> Color = clBlue;

else if (pl -> color == 3) Monopoly -> Player3 -> Color = clLime;

else Monopoly -> Player4 -> Color = clPurple;

}

Для перемещения игроков по игровому полю используются функции MovePlayer(), DrawAndMovePlayer(), которым для работы необходим двумерный массив. В данном массиве отображены координаты каждой клетки поля:

const int FieldCoordinate[40][2] = {{0 ,0}, {512, 63}, {565, 63}, {618, 63}, {671, 63}, {724, 63}, {777, 63}, {830, 63}, {883, 63}, {936, 63}, {0 ,0},

{1015, 140}, {1015, 193}, {1015, 246}, {1015, 299}, {1015, 352}, {1015, 405}, {1015, 458}, {1015, 511}, {1015, 564}, {0 ,0},

{936, 645}, {883, 645}, {830, 645}, {777, 645}, {724, 645}, {671, 645}, {618, 645}, {565, 645}, {512, 645}, {0 ,0},

{432, 565}, {432, 512}, {432, 459}, {432, 406}, {432, 353}, {432, 300}, {432, 247}, {432, 194}, {432, 141}};

Функция MovePlayer() принимает координаты X, Y из массива, описанного выше:

void MovePlayer(int X, int Y) {

switch (pl -> color) {

case 1: Monopoly -> Player1Shape -> Top = Y;

Monopoly -> Player1Shape -> Left = X;

break;

case 2: Monopoly -> Player2Shape -> Top = Y;

Monopoly -> Player2Shape -> Left = X;

break;

case 3: Monopoly -> Player3Shape -> Top = Y;

Monopoly -> Player3Shape -> Left = X;

break;

case 4: Monopoly -> Player4Shape -> Top = Y;

Monopoly -> Player4Shape -> Left = X;

break;

}

}

Функция DrawAndMovePlayer() обрабатывает случаи, когда игроки попадают на угловые поля игровой зоны и также перемещает игроков на соответствующую позицию:

void DrawAndMovePlayer() {

const int jail = 10, jackpot = 20, gotojail = 30, start = 0;

int X = 0, Y = 0;

if (FieldCoordinate[pl -> CurrentField][0] != 0) MovePlayer(FieldCoordinate[pl -> CurrentField][0], FieldCoordinate[pl -> CurrentField][1]);

else if (pl -> CurrentField == start) {

switch (pl -> color) {

case 1: X = 456;

Y = 39;

break;

case 2: X = 456;

Y = 88;

break;

case 3: X = 408;

Y = 88;

break;

case 4: X = 408;

Y = 39;

break;

}

MovePlayer(X, Y);

} else if (pl -> CurrentField == jail) {

switch (pl -> color) {

case 1: X = 1043;

Y = 31;

break;

case 2: X = 1048;

Y = 80;

break;

case 3: X = 1024;

Y = 56;

break;

case 4: X = 992;

Y = 23;

break;

}

MovePlayer(X, Y);

} else if (pl -> CurrentField == jackpot) {

switch (pl -> color) {

case 1: X = 1040;

Y = 615;

break;

case 2: X = 1040;

Y = 664;

break;

case 3: X = 1024;

Y = 664;

break;

case 4: X = 1024;

Y = 615;

break;

}

MovePlayer(X, Y);

} else if (pl -> CurrentField == gotojail) {

pl -> CurrentField = jail;

switch (pl -> color) {

case 1: X = 984;

Y = 87;

break;

case 2: X = 976;

Y = 48;

break;

case 3: X = 1024;

Y = 96;

break;

case 4: X = 1008;

Y = 71;

break;

}

MovePlayer(X, Y);

}

}

2.4 Логическая часть программы

Данная часть программы представляет собой алгоритм покупки полей, оплату аренды, вывод на экран сообщения от игроков в игровой чат, а также вывод событий.

Для перемещения на другое место на игровом поле необходимо бросить кубик. За это отвечает кнопка «Бросить кубики». Далее, помимо отображения в чате происходящего, осуществляется выбор действия, которое может совершить игрок. В зависимости от типа поля, а также факта покупки поля кем-то ранее, игрок может купить поле, отказаться покупать поле или оплатить аренду соответствующему игроку. За покупку полей отвечает кнопка «Заплатить».

После выполнения всех необходимых действий управление переходит к другому игроку. Действия осуществляются до тех пор, пока игрок не будет в состоянии оплатить аренду, налог или посещение поля «Шанс».

При прохождении круга игроков ждёт приятный сюрприз в виде денежного вознаграждения. Присваивание денег игрокам происходит автоматически, тем самым сводит необходимость игроков к каким либо дополнительным действиям к минимуму, что дает возможность игрокам сосредоточиться на стратегии игры.

Параллельно с переходом управления другому игроку происходит визуальное отображение данного действия. Эта функция очень удобна игрокам, которые только начинают играть в эту версию Монополии.

Пример выбора действия приведен ниже:

if (pl -> CurrentField >= 40) {

more = 1;

pl -> CurrentField %= 40;

pl -> cash += Lap;

ViewInChat(LeftCoub, RightCoub);

Chat -> Lines -> Add(AnsiString(pl -> name) + " получает " + IntToStr(Lap) + "k" + " за пройденный круг");

if (pl -> CurrentField == 0) {

Chat -> Lines -> Add(AnsiString(pl -> name) + " получает бонус " + IntToStr(LapBonus) + "k");

pl -> cash += LapBonus;

}

ShowMoney();

}

if (more == 0) ViewInChat(LeftCoub, RightCoub);

DrawAndMovePlayer();

if ((field[pl -> CurrentField].cost != 0) && (pl -> CurrentField != 4) && (pl -> CurrentField != 36)) {

if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner != pl -> color)) {

if ((pl -> CurrentField == 12) || (pl -> CurrentField == 28)) {

field[pl -> CurrentField].pay \*= pl -> CoubSum;

}

if (pl -> cash - field[pl -> CurrentField].pay >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

BitBtn3 -> Caption = "Заплатить " + AnsiString(field[pl -> CurrentField].pay) + "k";

BitBtn3 -> Width = 441;

BitBtn4 -> Hide();

} else if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner == pl -> color)) {

NextPlayer();

HideGroupBox = 1;

}

else if (field[pl -> CurrentField].owner == 0) {

BitBtn3 -> Caption = "Заплатить " + AnsiString(field[pl -> CurrentField].cost) + "k";

if (pl -> cash - field[pl -> CurrentField].cost >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

}

} else if ((pl -> CurrentField == 4) || (pl -> CurrentField == 36)) {

BitBtn3 -> Caption = "Заплатить " + IntToStr(field[pl -> CurrentField].cost) + "k";

BitBtn3 -> Width = 441;

BitBtn4 -> Hide();

if (pl -> cash - field[pl -> CurrentField].cost >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

} else if ((pl -> CurrentField == 2) || (pl -> CurrentField == 7) || (pl -> CurrentField == 17) || (pl -> CurrentField == 22) || (pl -> CurrentField == 33) || (pl -> CurrentField == 38)) {

RandomNumber = rand() % 2;

switch (RandomNumber) {

case 0: field[pl -> CurrentField].pay = 1000;

pl -> cash += field[pl -> CurrentField].pay;

Chat -> Lines -> Add(AnsiString(pl -> name) + " получает " + AnsiString(field[pl -> CurrentField].pay) + "k от банка");

ShowMoney();

NextPlayer();

HideGroupBox = 1;

break;

case 1: field[pl -> CurrentField].pay = 1000;

Chat -> Lines -> Add(AnsiString(pl -> name) + " должен заплатить банку " + IntToStr(field[pl -> CurrentField].pay) + "k");

BitBtn3 -> Caption = "Заплатить " + IntToStr(field[pl -> CurrentField].pay) + "k";

BitBtn3 -> Width = 441;

BitBtn4 -> Hide();

if (pl -> cash - field[pl -> CurrentField].pay >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

GroupBox2 -> Show();

break;

default: ShowMessage("kek, smth wrong!!!");

break;

}

} else if (pl -> CurrentField == jail) {

NextPlayer();

} else if (pl -> CurrentField == gotojail) {

pl -> MovesLeft = 0;

pl -> DoubleMoves = 0;

NextPlayer();

} else if (pl -> CurrentField == jackpot) {

if (pl -> cash - field[pl -> CurrentField].pay >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

GroupBox3 -> Show();

}

*Функция вывода событий в общий чат:*

void ViewInChat(int LeftCoub, int RightCoub) {

const int jail = 10, gotojail = 30, jackpot = 20;

AnsiString str;

str = field[pl -> CurrentField].name;

Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " выбрасывает " + AnsiString(LeftCoub) + ":" + AnsiString(RightCoub));

if ((field[pl -> CurrentField].cost != 0) && (pl -> CurrentField != 4) && (pl -> CurrentField != 36)) {

if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner != pl -> color)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на " + str + " и должен заплатить аренду " + AnsiString(field[pl -> CurrentField].pay) + "k");

else if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner == pl -> color)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на своё поле и ничего не платит");

else Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на " + str + " и задумывается о покупке");

} else if ((pl -> CurrentField == 4) || (pl -> CurrentField == 36)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на Налог и должен заплатить " + AnsiString(field[pl -> CurrentField].cost) + "k");

else if ((pl -> CurrentField == 2) || (pl -> CurrentField == 7) || (pl -> CurrentField == 17) || (pl -> CurrentField == 22) || (pl -> CurrentField == 33) || (pl -> CurrentField == 38)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на поле " + str);

else if (pl -> CurrentField == jail) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на экскурсию в тюрьму");

else if (pl -> CurrentField == gotojail) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " отправляется в тюрьму за мошенничество");

else if (pl -> CurrentField == jackpot) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на Джекпот и может попытать удачу ");

}

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного средства является одной из наиболее важной части разработки программы, что напрямую влияет на его конечное качество. В ходе тестирования проводятся соответствия между ожидаемыми и реальными результатами работы программы.

В процессе тестирования были выявлены и исправлены следующие недостатки разрабатываемого программного средства.

Была обнаружена проблема при закраске поля, которое отвечает за стоимость и аренду. Данная проблема была решена путём обращения к классу TPanel. А необходимый элемент вычислялся из значения тега, присвоенному этому полю и текущего положения фишки игрока.

for (int i = 0; i <= ComponentCount - 1; i++) {

if ((Monopoly -> Components[i] -> ClassNameIs("TPanel")) && (Monopoly -> Components[i] -> Tag == pl -> CurrentField)) {

((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Caption = IntToStr(field[pl -> CurrentField].pay) + "k";

((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Font -> Color = clWhite;

if (pl -> color == 1) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clRed;

else if (pl -> color == 2) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clBlue;

else if (pl -> color == 3) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clLime;

else if (pl -> color == 4) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clPurple;

}

}

Изначально тегам полей, которые можно купить присваиваются значения, соответствующие их номерам на игровом поле. Далее происходит обращение к классу TPanel и отрисовка в определённый цвет с необходимыми данными.

Однако если у игрока отсутствует желание покупки поля, он может отказаться и поле не будет окрашено в цвет игрока, попавшего на клетку.

Ниже приведена таблица, показывающая ожидаемые и реальные результаты тестирования программного средства.

Таблица 3.1 – Ожидаемые и реальные результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестовые случаи | Ожидаемый результат | Полученный результат |
| 1. | Покупка поля | Успешная покупка поля и отображение его стоимости | Успешная покупка поля и отображение его стоимости |
| 2. | Отказ покупки поля | Не произошло окрашивание и отображение стоимости аренды | Не произошло окрашивание и отображение стоимости аренды |

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Интерфейс программы

4.1 Начало игры

Интерфейс программы после запуска игры представлен на рисунке 4.1

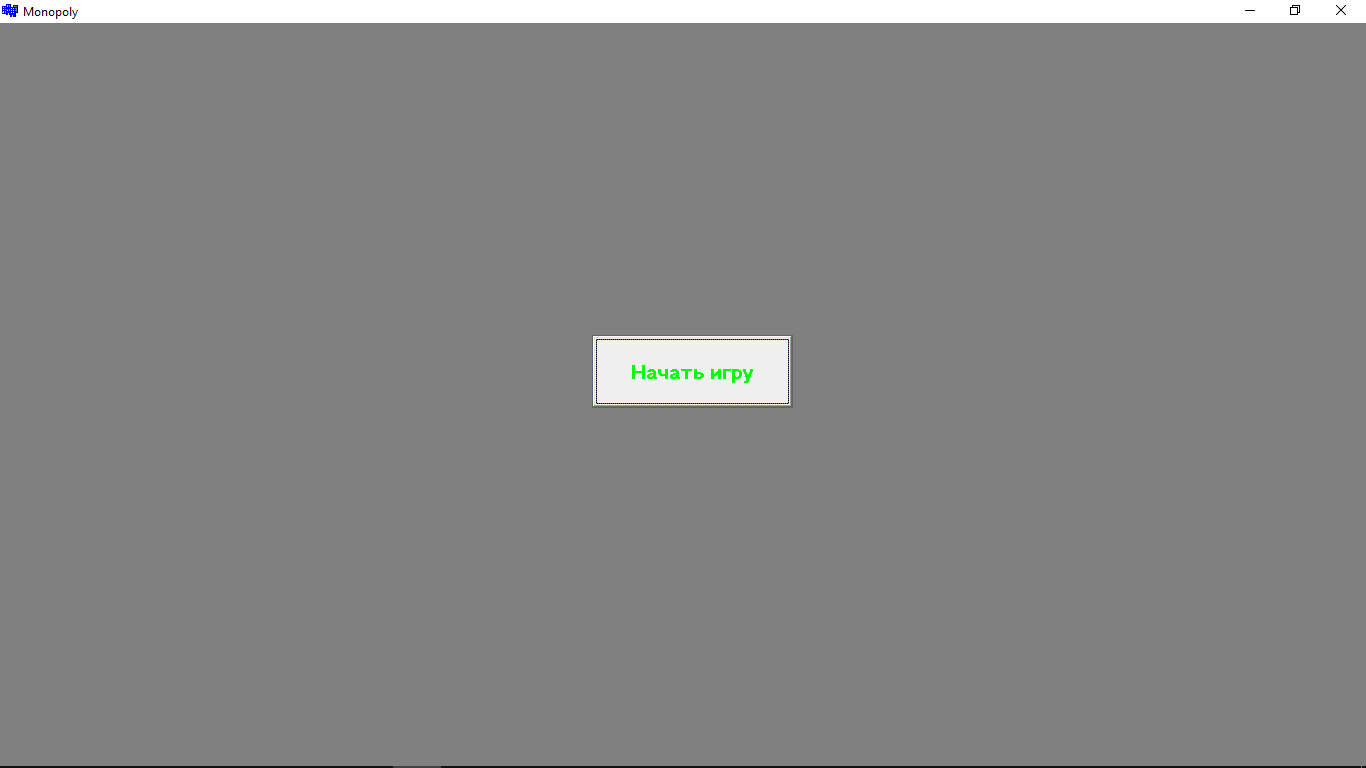


Рисунок 4.1 – Запуск игры

Окно после запуска содержит кнопку «Начать игру».

**4.2 Интерфейс главного окна**

Интерфейс главного окна игры представлен на рисунке 4.2

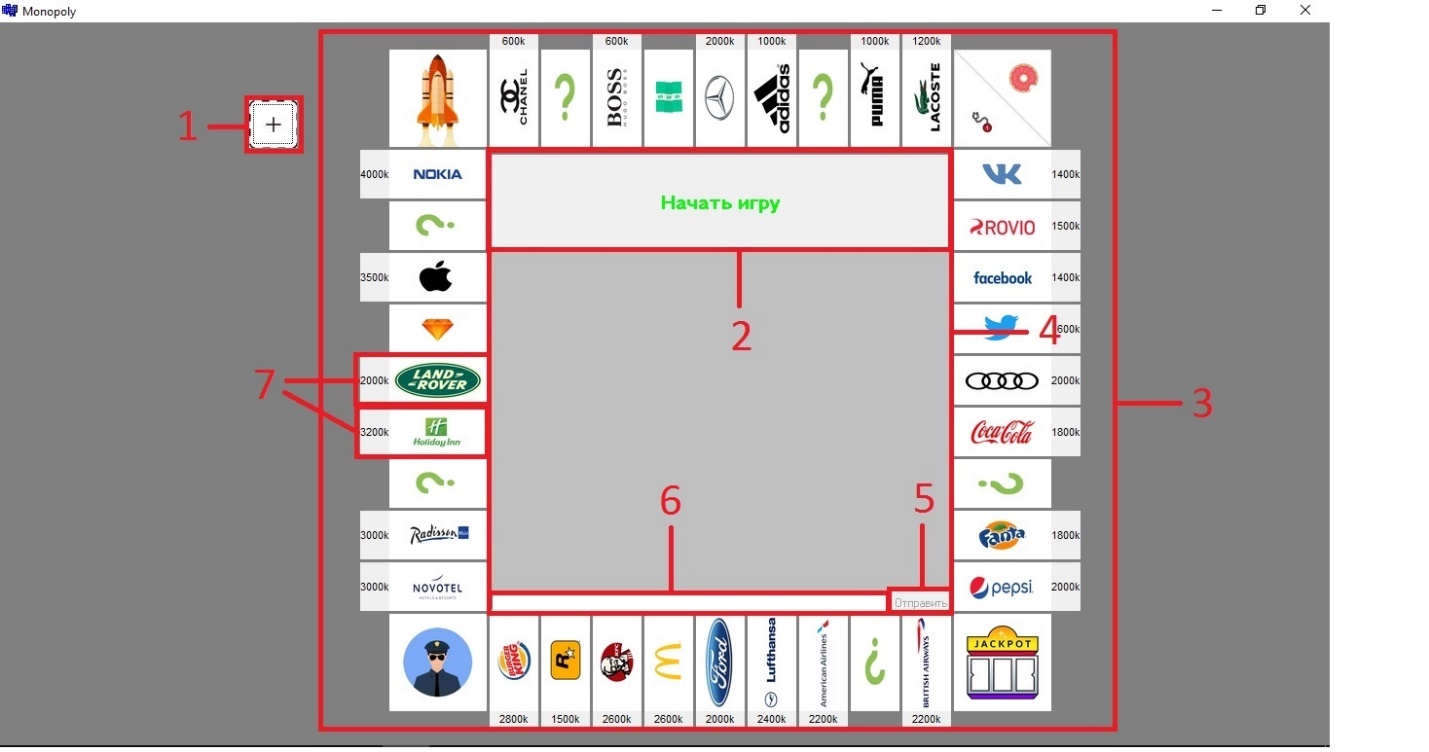


Рисунок 4.2 – Главное окно программы

Главное окно программы, изображенное на рисунке 4.2, содержит следующие элементы:

1. Кнопка «Добавить игрока»
2. Кнопка «Начать игру»
3. Игровое поле
4. Чат
5. Кнопка для отправления сообщений
6. Поле для ввода сообщений
7. Клетки игрового поля

Для добавления игроков в игру необходимо нажать на кнопку «Добавить игрока» из рисунка 4.2.

**4.3 Добавление игрока**

Добавление игрока представлено на рисунке 4.3

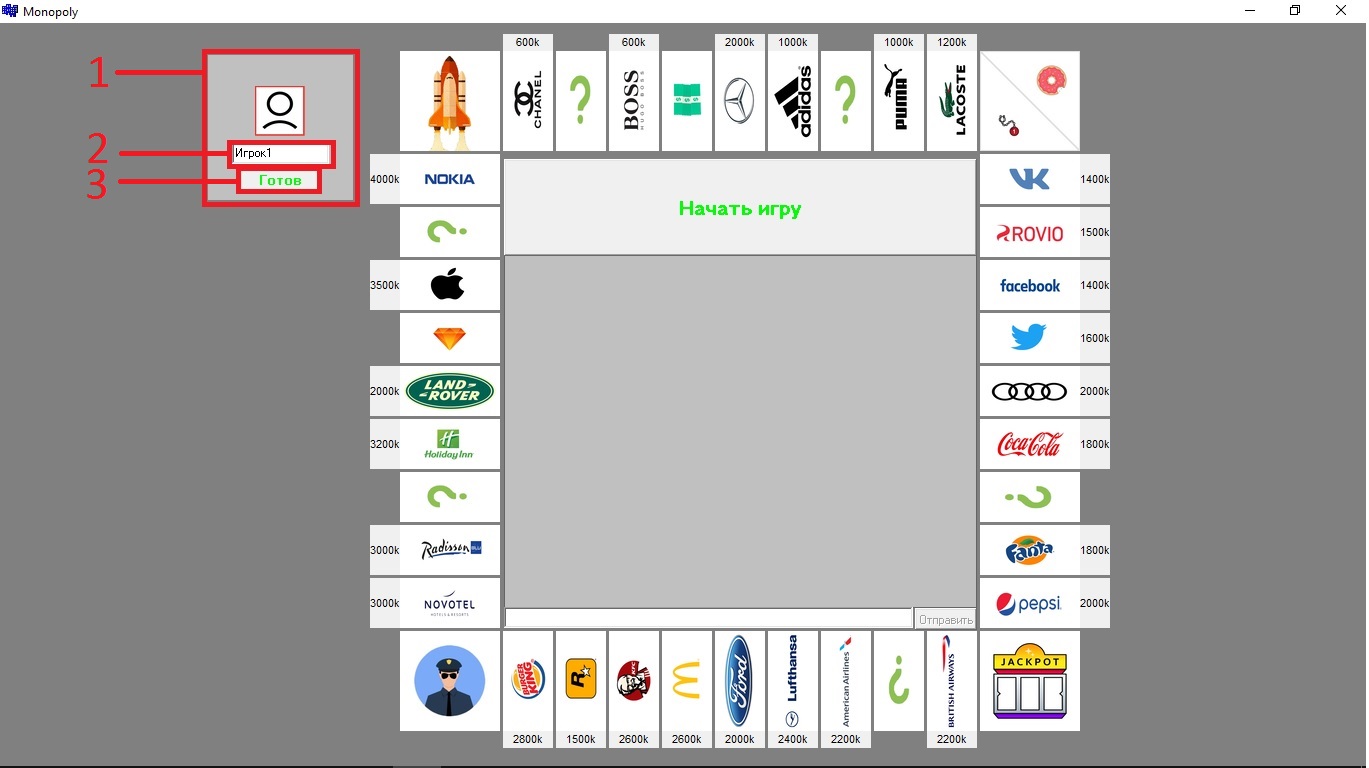


Рисунок 4.3 – Добавление игрока

Главное окно программы, изображенное на рисунке 4.3, содержит следующие элементы:

1. Профиль игрока
2. Поле для ввода игрового имени
3. Кнопка «Готов»

Необходимо заполнить поле игровым именем и нажать кнопку «Готов». Игрок будет добавлен после нажатия на кнопку. После добавления необходимого количества игроков для начала игры необходимо нажать на кнопку «Начать игру» из рисунка 4.2.

**4.4 Перемещение игрока по игровому полю**

Главное игровое окно после начала игры представлено на рисунке 4.4.

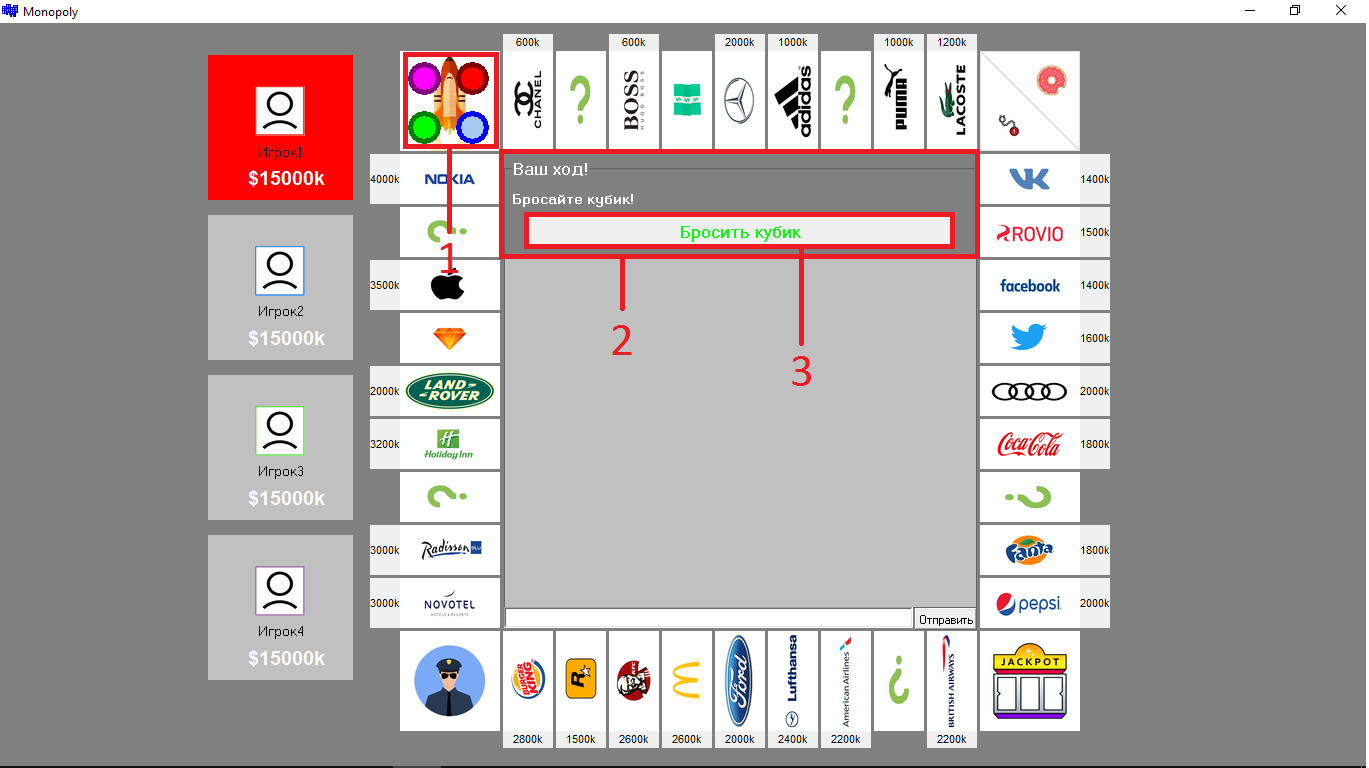


Рисунок 4.4 – Начало движения

Главное окно программы, изображенное на рисунке 4.4, содержит следующие элементы:

1. Фишки игроков
2. Поле для совершения броска
3. Кнопка «Бросить кубик»

Для того, чтобы совершить перемещение по полю, игроку необходимо нажать на кнопку «Бросить кубик».

Передвижение фишки игрока представлено на рисунке 4.5

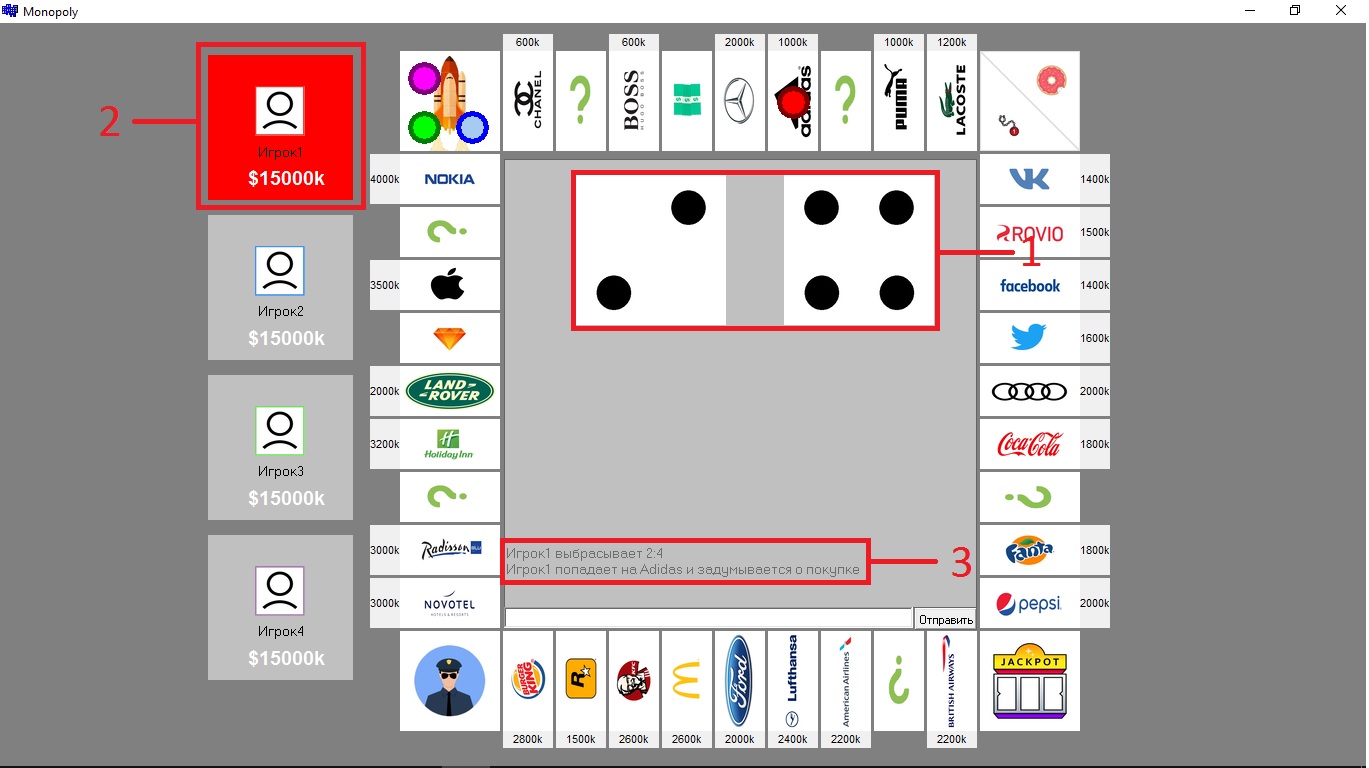


Рисунок 4.5 – Передвижение фишки

Главное окно программы, изображенное на рисунке 4.5, содержит следующие элементы:

1. Значение, на которое нужно передвинуть фишку
2. Очередь игрока
3. История событий

После того, как игрок нажал «Бросить кубик», его фишка перемещается на количество клеток, которое выпало на кубиках. Сколько выпало и куда перемещается фишка отображается в чате.

**4.5 Покупка поля**

Покупка поля представлена на рисунках 4.6 и 4.7

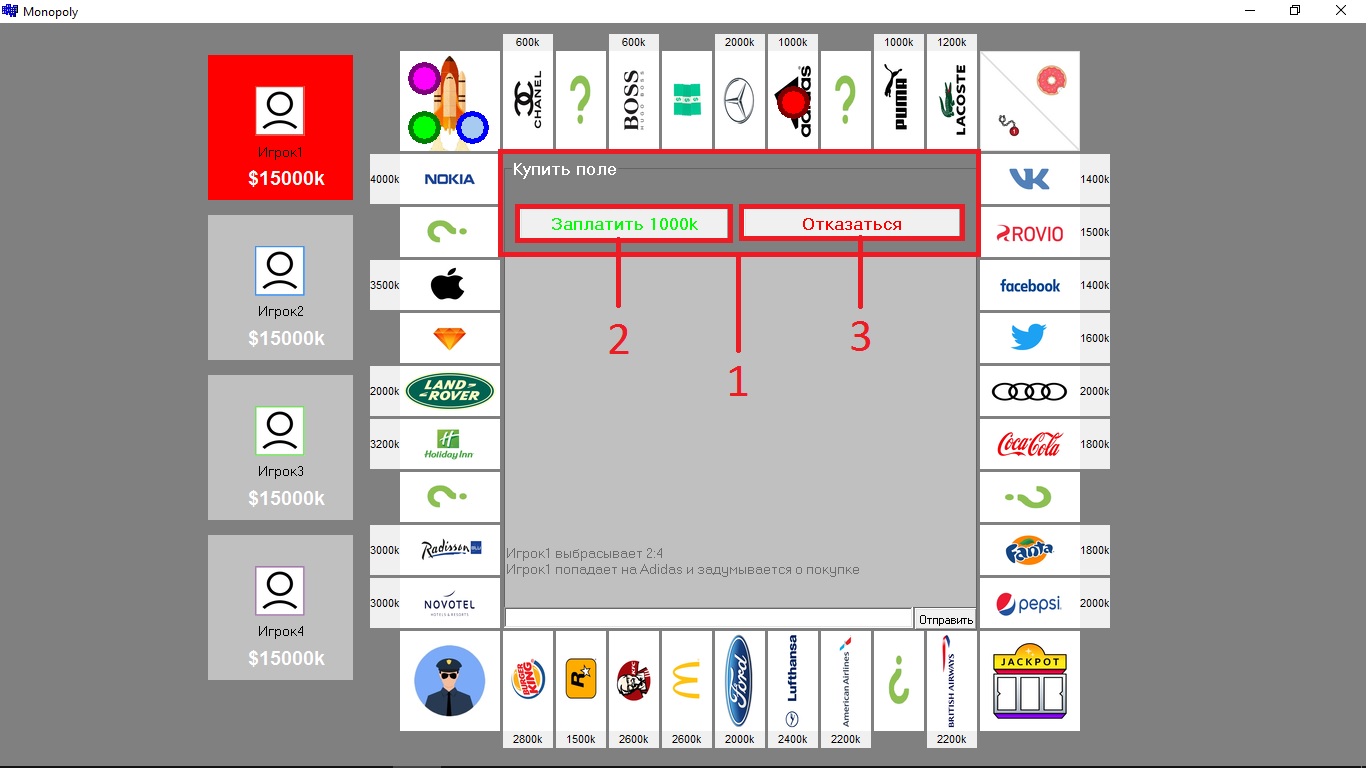


Рисунок 4.6 – Выбор действия

Главное окно программы, изображенное на рисунке 4.6, содержит следующие элементы:

1. Окно выбора действия
2. Кнопка «Купить»
3. Кнопка «Отказаться»

У игрока есть выбор: покупать поле или нет.

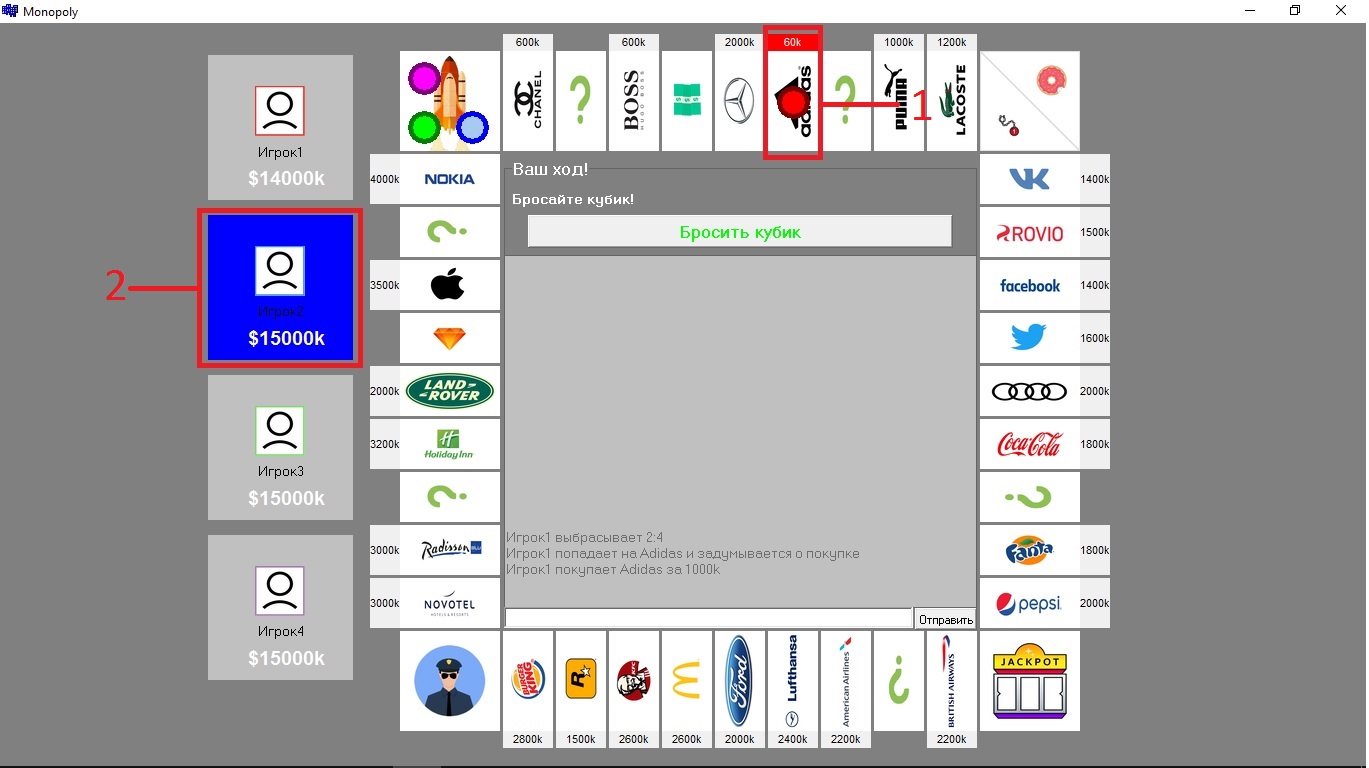


Рисунок 4.7 – Покупка поля

Главное окно программы, изображенное на рисунке 4.7, содержит следующие элементы:

1. Поле, которое купил игрок
2. Ход следующего игрока

Если игрок решил купить поле, то поле окрашивается в цвет фишки игрока, иначе поле не окрашивается. Ход переходит к другому игроку.

**4.6 Аренда поля**

Оплата аренды представлена на рисунке 4.8

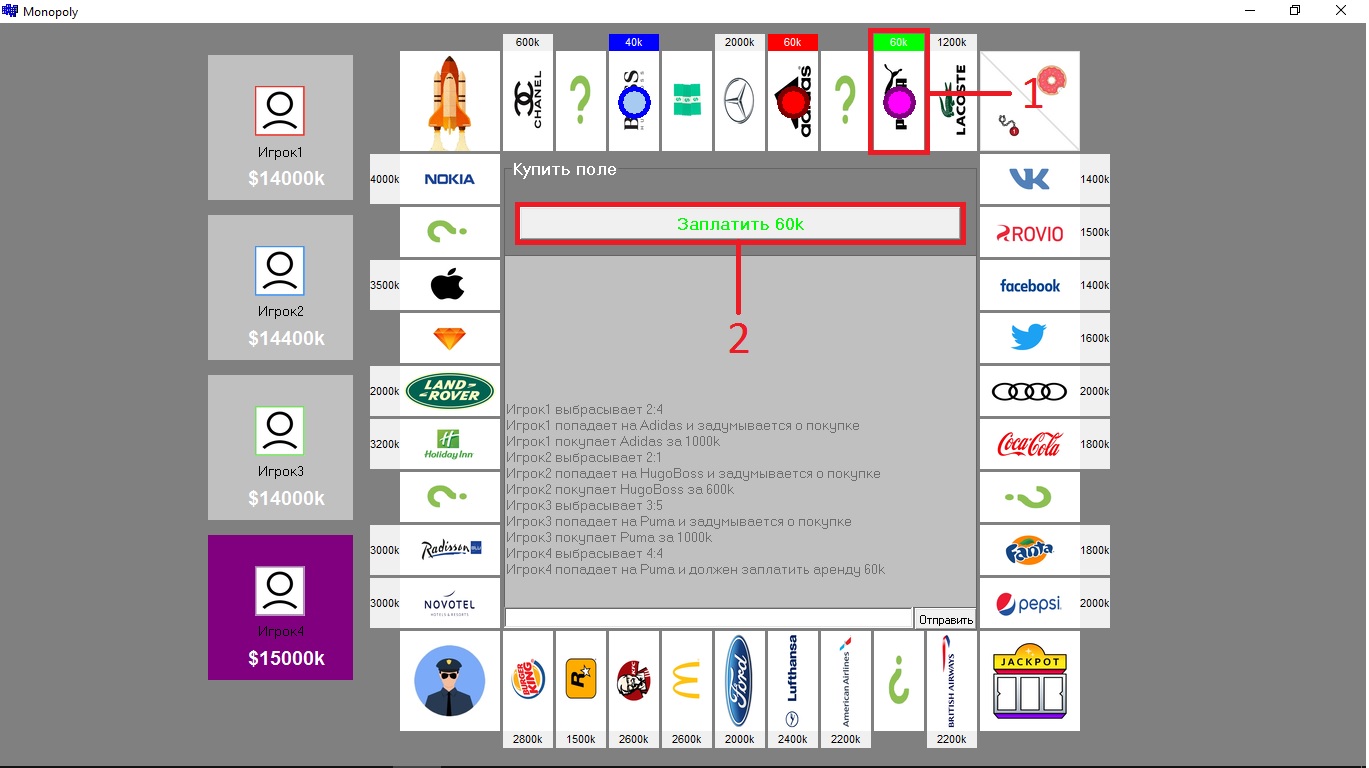


Рисунок 4.8 – Оплата аренды

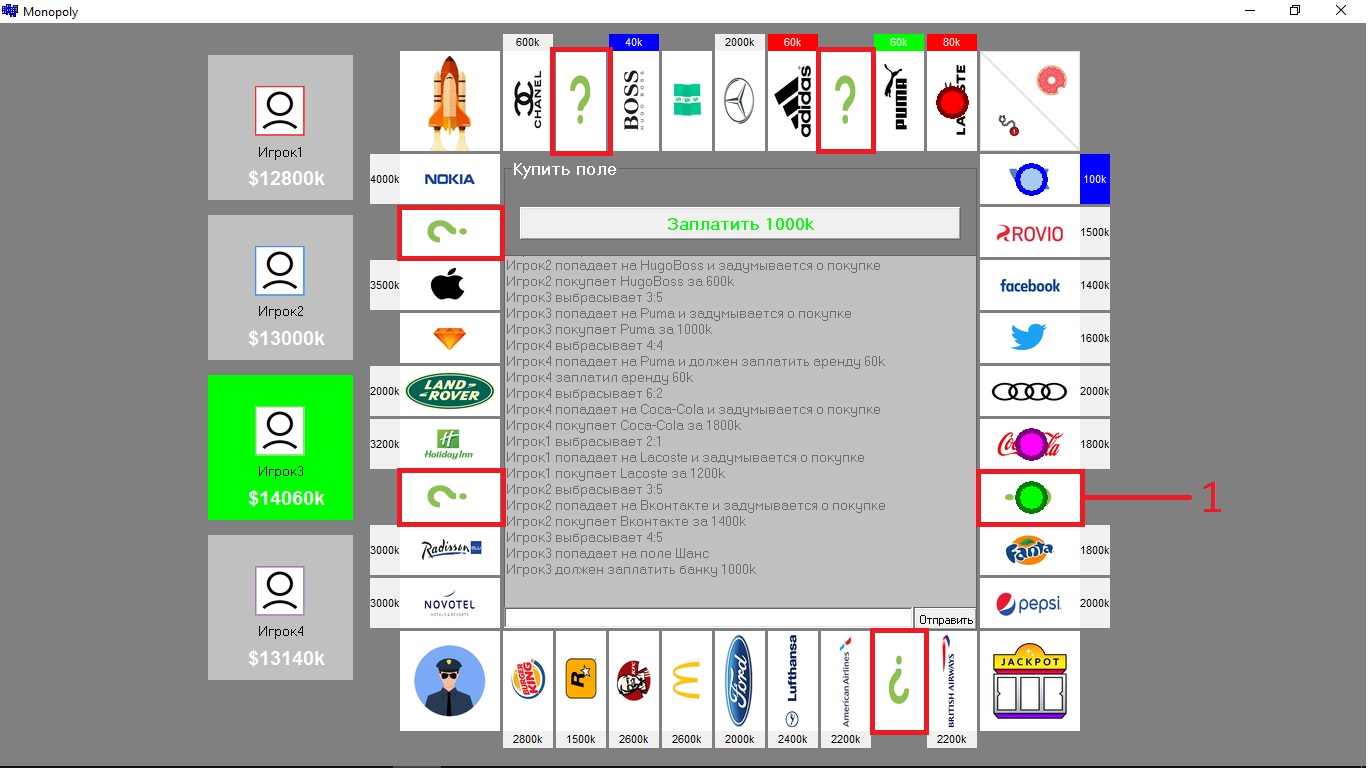
Главное окно программы, изображенное на рисунке 4.8, содержит следующие элементы:

1. Игрок, на поле другого игрока
2. Кнопка для оплаты аренды

Если игрок попадает на поле соперника, то обязан оплатить аренду поля, если он попадает на своё поле, то ничего не платит и ход переходит другому игроку.

**4.7** **Поле «Шанс»**

На карте игры есть такие поля как «Шанс», где игроку заранее неизвестно: получит он деньги или потеряет. Попадание на поле «Шанс» представлено на рисунке 4.9.



Главное окно программы, изображенное на рисунке 4.9, содержит следующие элементы:

1. Поле «Шанс»

**4.8 Поле «Джекпот»**

Попадание на поле «Джекпот» представлено на рисунках 4.10 и 4.11

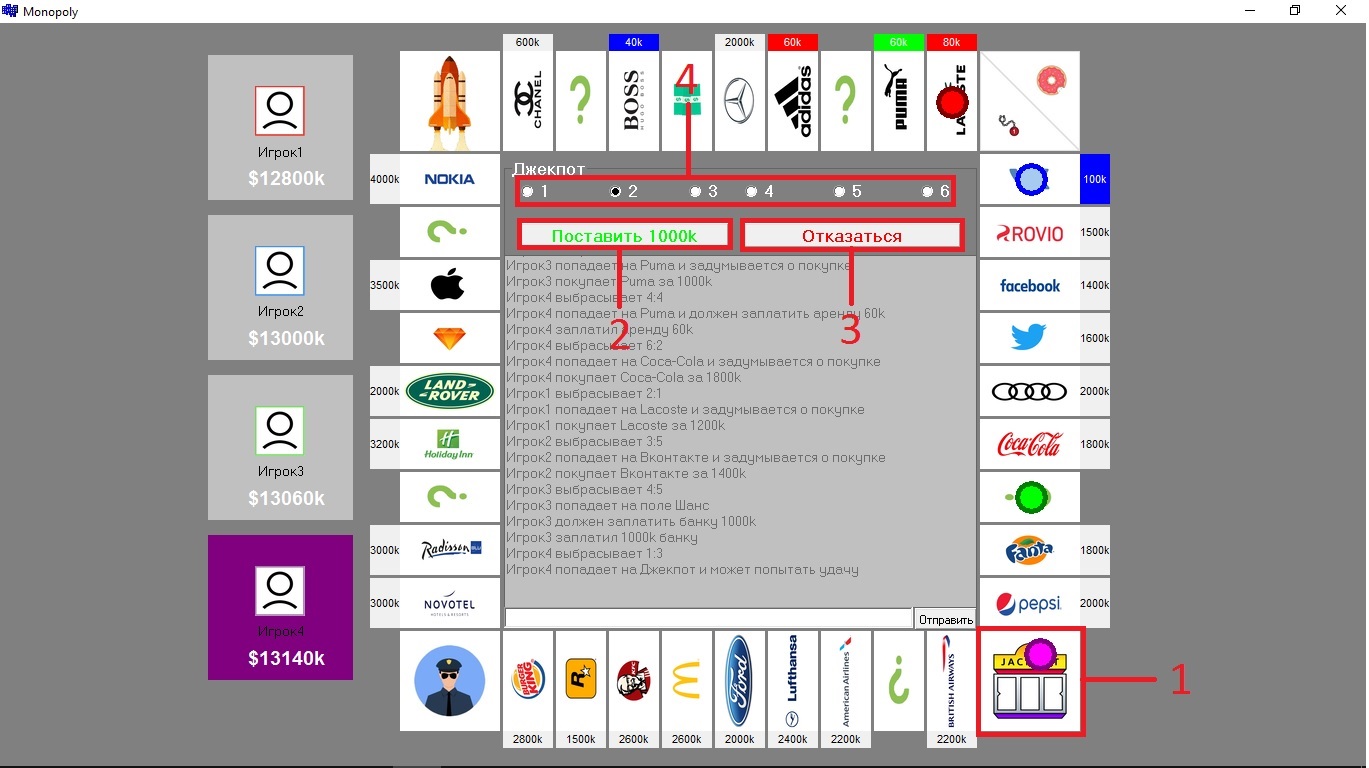


Рисунок 4.10 – Поле Джекпот

Главное окно программы, изображенное на рисунке 4.10, содержит следующие элементы:

1. Поле «Джекпот»
2. Кнопка для оплаты ставки
3. Кнопка отказа от ставки
4. Предполагаемое значение кубика

При попадании на поле «Джекпот» у игрока появляется возможность поставить ставку и заработать. В то же время может отказаться.

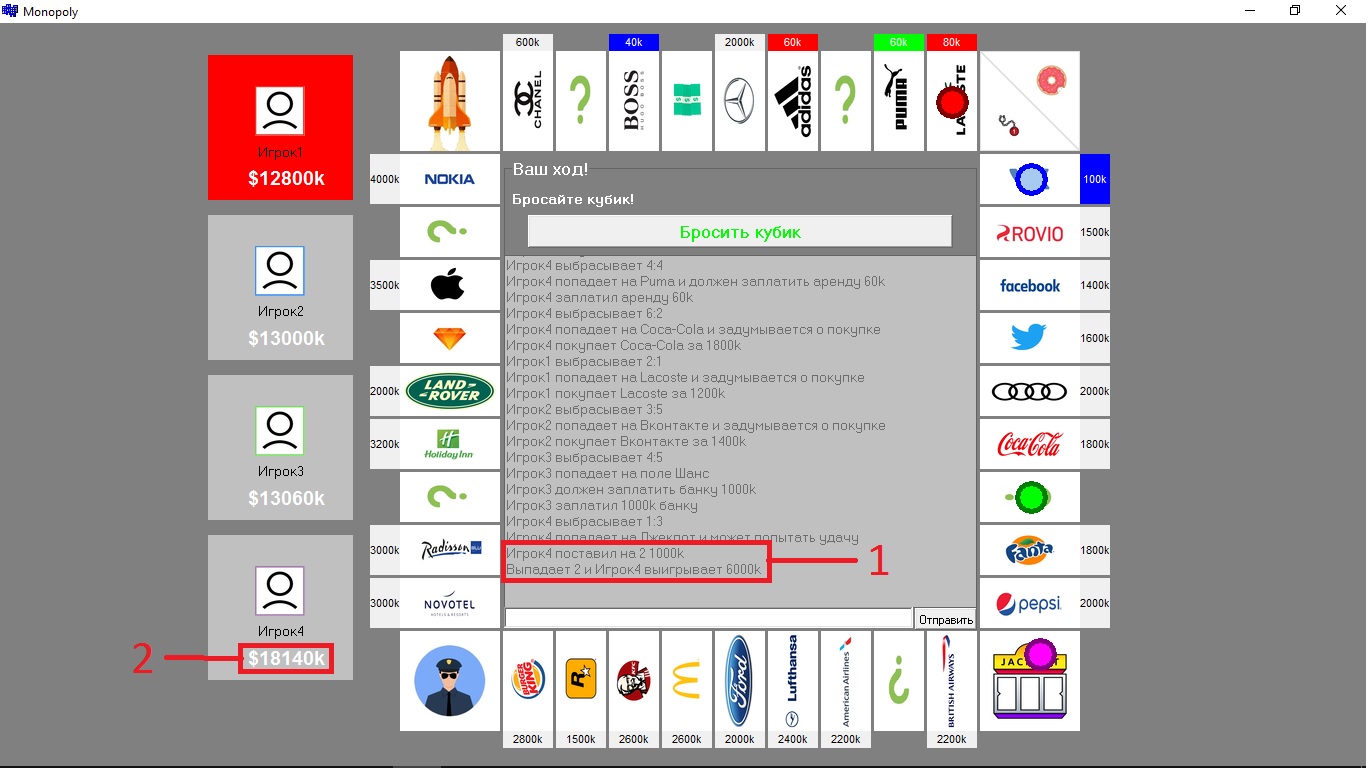


Рисунок 4.11 – Выигрыш после игры на поле Джекпот

Главное окно программы, изображенное на рисунке 4.11, содержит следующие элементы:

1. Сообщение о выигрыше
2. Деньги игрока после выигрыша

**4.9 Поле «Тюрьма»**

Попадание на поле «Тюрьма» представлено на рисунке 4.12

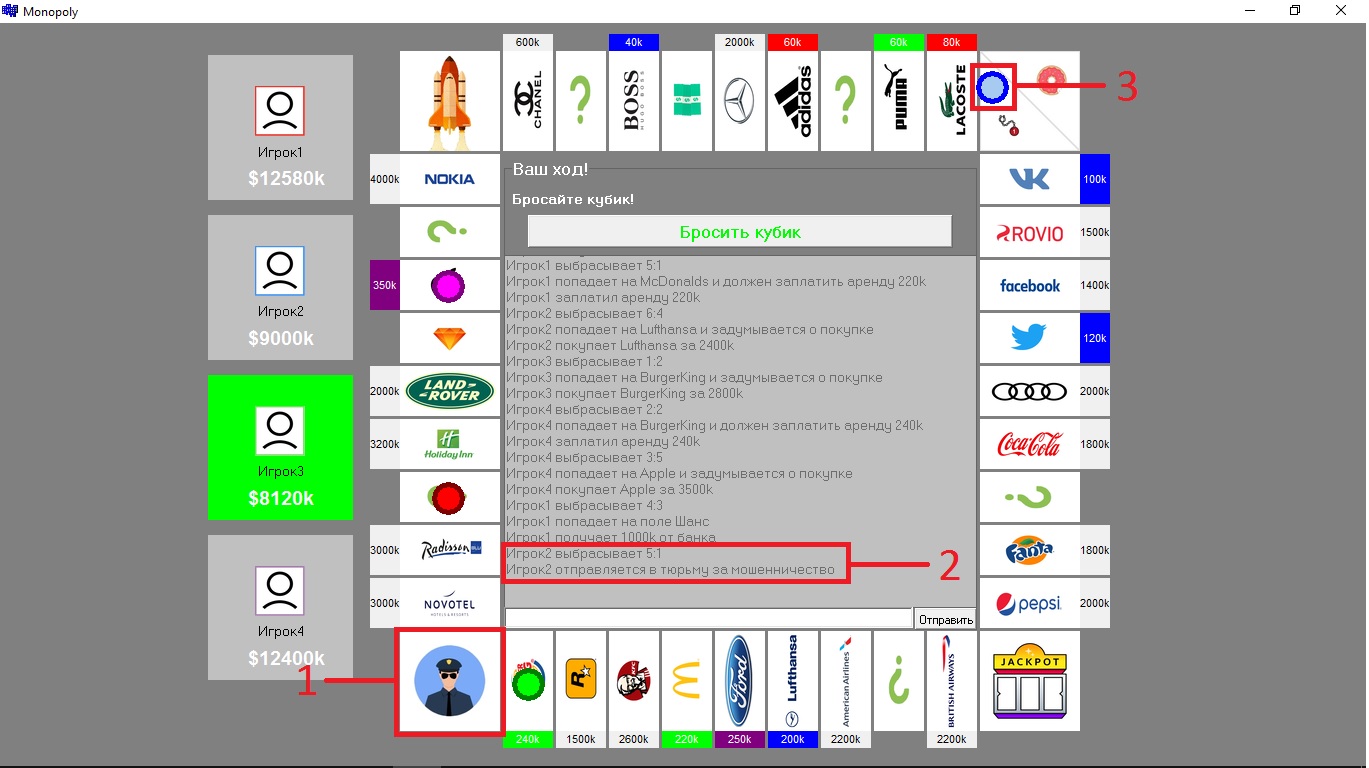


Рисунок 4.12 – Поле «Тюрьма»

Главное окно программы, изображенное на рисунке 4.12, содержит следующие элементы:

1. Поле «Тюрьма»
2. Сообщение о попадании в тюрьму
3. Игрок в тюрьме

**4.10 Поле «Налог»**

Попадание на одно из полей «Налог» представлено на рисунке 4.13

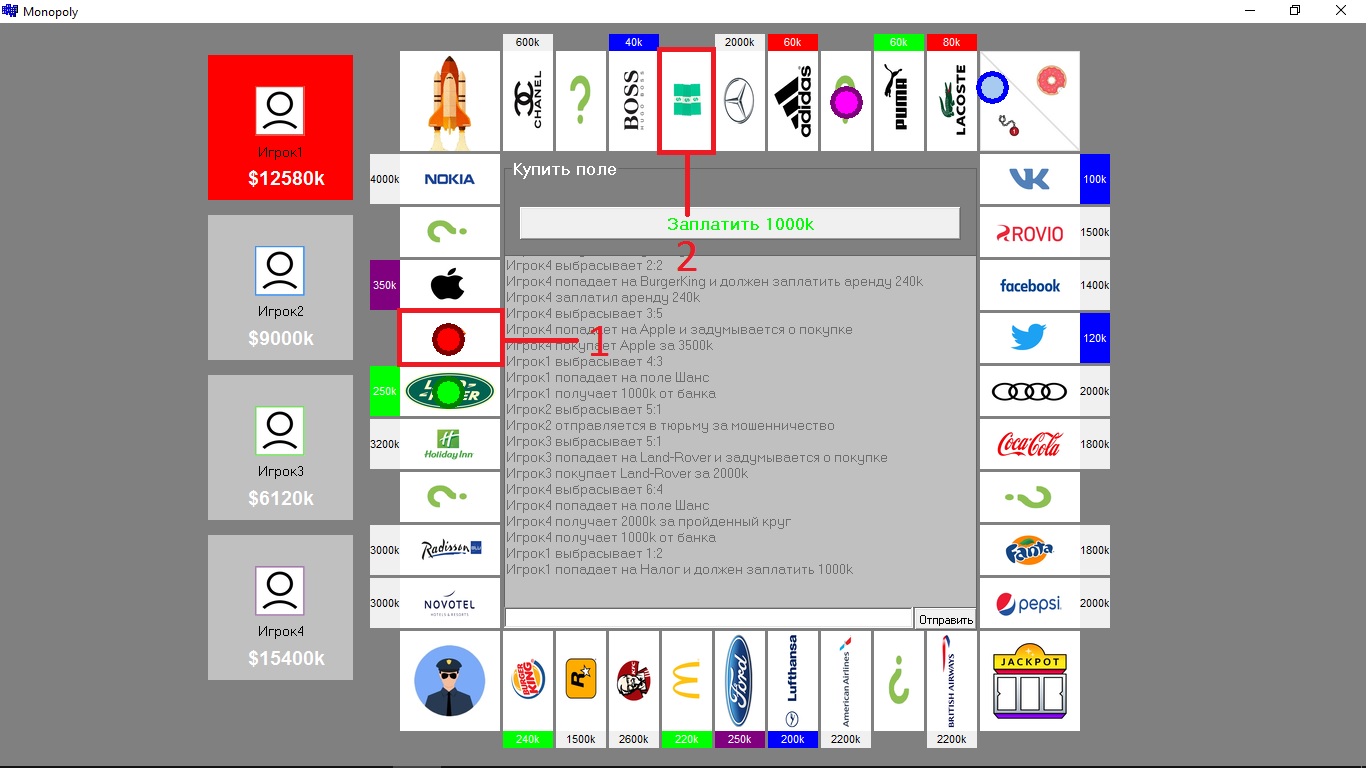


Рисунок 4.13 - Налоги

Главное окно программы, изображенное на рисунке 4.13, содержит следующие элементы:

1. Поле «Налог на драгоценности»
2. Поле «Налог»

При попадании на поле «Налог» игрок обязан оплатить 2000k. При попадании на поле «Налог на драгоценности» игрок обязан оплатить 1000k.

**4.11 Конец игры**

Выход из игры представлен на рисунке 4.14

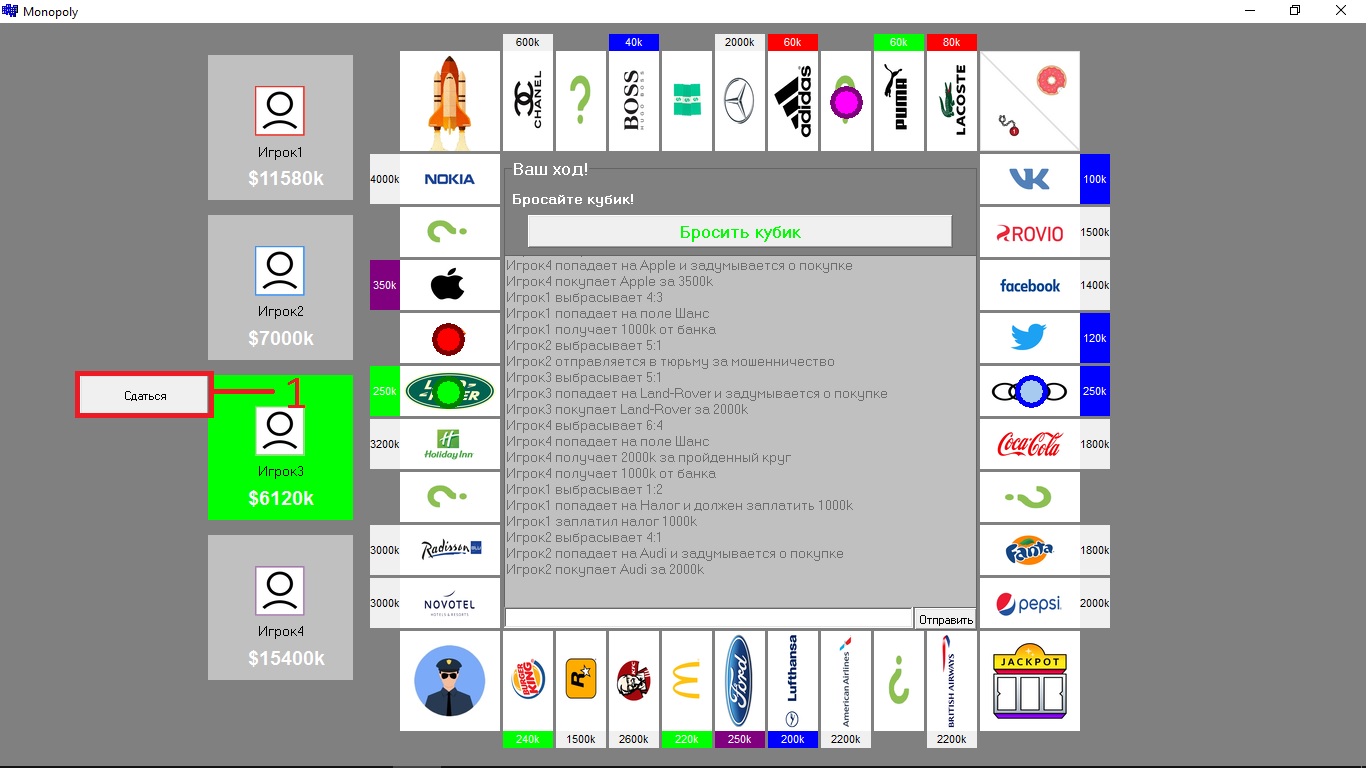


Рисунок 4.14 – Выход из игры

Главное окно программы, изображенное на рисунке 4.14, содержит следующие элементы:

1. Кнопка «Сдаться»

Конец игры представлен на рисунке 4.15

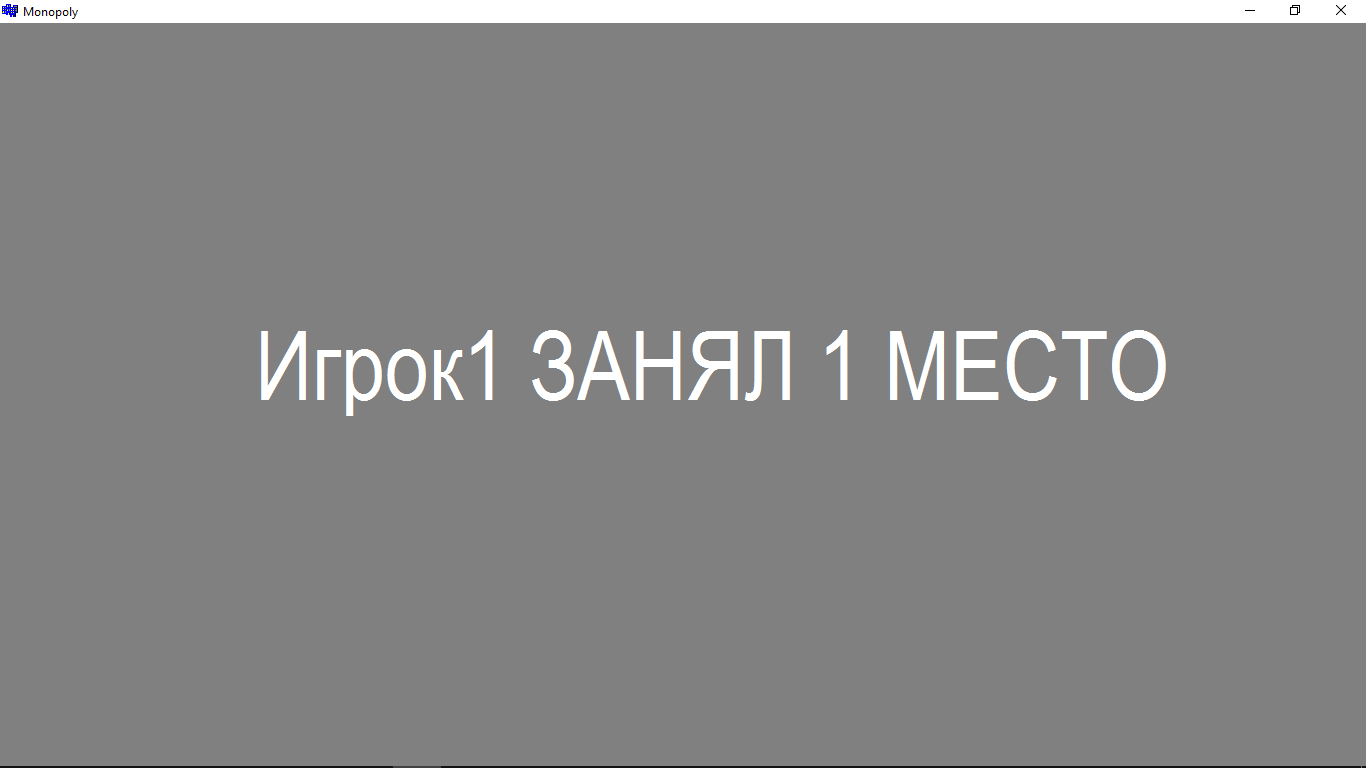


Рисунок 4.15 – Конец игры

ЗАКЛЮЧЕНИЕ

В настоящее время наблюдается развитие игр в сторону 3D – графики, а также виртуальной реальности. Некоторые аналоги игры «Монополия» уже имеют 3D графику.

В рамках данного курсового проекта было разработано игровое средство «Монополия», позволяющее играть до 4 человек. Плюсом данного программного средства является необходимости в интернет - соединении, требование небольшого объёма памяти для установки игры, а также удобный пользовательский интерфейс. Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

отрисовка фишек игроков различного цвета;

добавление новых игроков в игру;

покупка недвижимости, представленной в виде полей;

отрисовка стоимости поля, а также его аренда;

для перемещения игроков по игровому полю будут использоваться игральные кости;

для покупки и аренды полей будет использоваться игровая валюта «монобаксы»;

в качестве бонуса за пройденный круг игрокам будет предоставлено денежное вознаграждение;

в случае проигрыша, то есть банкротства, или при необходимости выйти игрок может сдаться и выйти из игры;

возможность вести диалог между игроками, с помощью специального чата, в котором также по ходу игры отображается история действий игроков;

Для дальнейшего усовершенствования игры имеется огромное количество вариантов. Например, добавление функции обмена полями, возможность строить филиалы, возможность брать кредит в банке, для передвижения фишек игроков сделать анимацию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Программирование на языке C++ / Гирберд Шилдт. Учебный курс: / Гирберд Шилдт. – СПб: изд. С. В. Малгачёва, 2001. – 231 с;
2. Графические возможности Rad Studio (электронный ресурс). – Электронные данные. – Режим доступа: http://embarcadero.gym5cheb.ru.
3. RAD Studio Product Documentation – Embarcadero Technologies [Электронный ресурс]. – Электронные данные. – Режим доступа: http://docs.embarcadero.com/products/rad\_studio;
4. Орлов, С. А. Технологии разработки программного обеспечения: учеб. Пособие. – СПб, 2003;
5. Культин, Н. А. Основы программирования в C++ – учеб. пособие. – Москва: изд. «НТ Пресс», 2008. – 167с;
6. Хлебостроев, В. С. Программирование графики в Rad Studio – учеб. пособие. – Москва: изд. «НТ Пресс», 2008. – 243с;

# **Приложение А**

**Исходный код программы**

**Unit1.h**

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ExtCtrls.hpp>

#include <Graphics.hpp>

#include <Buttons.hpp>

#include <ComCtrls.hpp>

#include <ImgList.hpp>

//---------------------------------------------------------------------------

class TMonopoly : public TForm

{

\_\_published: // IDE-managed Components

TBitBtn \*StartGameButton;

TImage \*Image1;

TImage \*Image2;

TImage \*Image3;

TImage \*Image4;

TImage \*Image5;

TImage \*Image6;

TImage \*Image7;

TImage \*Image8;

TImage \*Image9;

TImage \*Image10;

TImage \*Image11;

TImage \*Image12;

TImage \*Image13;

TImage \*Image20;

TImage \*Image16;

TImage \*Image21;

TImage \*Image23;

TImage \*Image24;

TImage \*Image25;

TImage \*Image26;

TImage \*Image27;

TImage \*Image39;

TImage \*Image30;

TImage \*Image22;

TImage \*Image14;

TImage \*Image17;

TImage \*Image28;

TImage \*Image29;

TImage \*Image18;

TImage \*Image31;

TImage \*Image32;

TImage \*Image33;

TImage \*Image34;

TImage \*Image35;

TImage \*Image36;

TImage \*Image37;

TImage \*Image38;

TImage \*Image15;

TImage \*Image19;

TImage \*Image40;

TEdit \*Edit1;

TBitBtn \*SendMessageButton;

TBitBtn \*AddPlayer1;

TPanel \*Player1;

TImage \*Image41;

TEdit \*Player1Nickname;

TBitBtn \*Player1Ready;

TLabel \*Player1Name;

TBitBtn \*AddPlayer2;

TBitBtn \*AddPlayer3;

TBitBtn \*AddPlayer4;

TPanel \*Player2;

TPanel \*Player3;

TPanel \*Player4;

TImage \*Image42;

TBitBtn \*Player2Ready;

TLabel \*Label2;

TLabel \*Player2Name;

TEdit \*Player2Nickname;

TImage \*Image43;

TBitBtn \*Player3Ready;

TEdit \*Player3Nickname;

TLabel \*Player3Name;

TImage \*Image44;

TBitBtn \*Player4Ready;

TEdit \*Player4Nickname;

TLabel \*Player4Name;

TLabel \*Player1Money;

TLabel \*Player2Money;

TLabel \*Player3Money;

TLabel \*Player4Money;

TMemo \*Chat;

TBitBtn \*BitBtn1;

TGroupBox \*GroupBox1;

TLabel \*Label1;

TBitBtn \*BitBtn2;

TShape \*Player1Shape;

TShape \*Player2Shape;

TShape \*Player3Shape;

TShape \*Player4Shape;

TPanel \*Panel1;

TPanel \*Panel3;

TPanel \*Panel5;

TPanel \*Panel6;

TPanel \*Panel8;

TPanel \*Panel9;

TPanel \*Panel21;

TPanel \*Panel23;

TPanel \*Panel24;

TPanel \*Panel25;

TPanel \*Panel26;

TPanel \*Panel27;

TPanel \*Panel29;

TPanel \*Panel28;

TPanel \*LeftCoubPanel;

TImage \*Coub1;

TImage \*Coub2;

TPanel \*RightCoubPanel;

TTimer \*Timer1;

TPanel \*Panel11;

TPanel \*Panel12;

TPanel \*Panel13;

TPanel \*Panel14;

TPanel \*Panel15;

TPanel \*Panel16;

TPanel \*Panel31;

TPanel \*Panel18;

TPanel \*Panel19;

TPanel \*Panel39;

TPanel \*Panel37;

TPanel \*Panel35;

TPanel \*Panel34;

TPanel \*Panel32;

TGroupBox \*GroupBox2;

TBitBtn \*BitBtn3;

TBitBtn \*BitBtn4;

TShape \*Shape1;

TShape \*Shape2;

TShape \*Shape3;

TShape \*Shape4;

TShape \*Shape5;

TShape \*Shape6;

TShape \*Shape7;

TShape \*Shape8;

TShape \*Shape9;

TShape \*Shape10;

TShape \*Shape11;

TShape \*Shape12;

TShape \*Shape13;

TShape \*Shape14;

TShape \*Shape15;

TShape \*Shape16;

TShape \*Shape17;

TShape \*Shape18;

TShape \*Shape19;

TShape \*Shape20;

TShape \*Shape21;

TShape \*Shape22;

TShape \*Shape23;

TShape \*Shape24;

TShape \*Shape25;

TShape \*Shape26;

TShape \*Shape27;

TShape \*Shape28;

TShape \*Shape29;

TShape \*Shape30;

TShape \*Shape31;

TShape \*Shape37;

TShape \*Shape38;

TShape \*Shape39;

TShape \*Shape40;

TShape \*Shape41;

TShape \*Shape42;

TShape \*Shape43;

TShape \*Shape44;

TShape \*Shape45;

TShape \*Shape46;

TShape \*Shape47;

TShape \*Shape48;

TShape \*Shape49;

TShape \*Shape50;

TShape \*Shape51;

TShape \*Shape52;

TShape \*Shape53;

TShape \*Shape54;

TShape \*Shape55;

TShape \*Shape56;

TShape \*Shape57;

TShape \*Shape58;

TShape \*Shape59;

TShape \*Shape60;

TShape \*Shape61;

TShape \*Shape62;

TShape \*Shape63;

TShape \*Shape64;

TShape \*Shape65;

TShape \*Shape66;

TShape \*Shape67;

TShape \*Shape68;

TShape \*Shape69;

TShape \*Shape70;

TShape \*Shape71;

TShape \*Shape72;

TShape \*Shape77;

TShape \*Shape78;

TShape \*Shape79;

TShape \*Shape80;

TShape \*Shape81;

TShape \*Shape82;

TShape \*Shape83;

TShape \*Shape84;

TShape \*Shape85;

TShape \*Shape86;

TShape \*Shape87;

TShape \*Shape88;

TShape \*Shape89;

TShape \*Shape90;

TShape \*Shape91;

TShape \*Shape92;

TShape \*Shape93;

TShape \*Shape94;

TShape \*Shape95;

TShape \*Shape96;

TShape \*Shape32;

TShape \*Shape97;

TPanel \*DeletePlayer1;

TPanel \*Panel2;

TPanel \*DeletePlayer3;

TPanel \*Panel4;

TBitBtn \*BitBtn5;

TBitBtn \*BitBtn6;

TBitBtn \*BitBtn7;

TBitBtn \*BitBtn8;

TLabel \*Label3;

TGroupBox \*GroupBox3;

TBitBtn \*BitBtn9;

TBitBtn \*BitBtn10;

TRadioButton \*RadioButton1;

TRadioButton \*RadioButton2;

TRadioButton \*RadioButton3;

TRadioButton \*RadioButton4;

TRadioButton \*RadioButton5;

TRadioButton \*RadioButton6;

void \_\_fastcall StartGameButtonClick(TObject \*Sender);

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall AddPlayer1Click(TObject \*Sender);

void \_\_fastcall Player1ReadyClick(TObject \*Sender);

void \_\_fastcall Player2ReadyClick(TObject \*Sender);

void \_\_fastcall Player3ReadyClick(TObject \*Sender);

void \_\_fastcall Player4ReadyClick(TObject \*Sender);

void \_\_fastcall BitBtn2Click(TObject \*Sender);

void \_\_fastcall BitBtn1Click(TObject \*Sender);

void \_\_fastcall SendMessageButtonClick(TObject \*Sender);

void \_\_fastcall Timer1Timer(TObject \*Sender);

void \_\_fastcall BitBtn3Click(TObject \*Sender);

void \_\_fastcall BitBtn4Click(TObject \*Sender);

void \_\_fastcall Player2Click(TObject \*Sender);

void \_\_fastcall Player1Click(TObject \*Sender);

void \_\_fastcall Player3Click(TObject \*Sender);

void \_\_fastcall Player4Click(TObject \*Sender);

void \_\_fastcall FormClick(TObject \*Sender);

void \_\_fastcall BitBtn5Click(TObject \*Sender);

void \_\_fastcall BitBtn6Click(TObject \*Sender);

void \_\_fastcall BitBtn7Click(TObject \*Sender);

void \_\_fastcall BitBtn8Click(TObject \*Sender);

void \_\_fastcall BitBtn9Click(TObject \*Sender);

void \_\_fastcall BitBtn10Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TMonopoly(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TMonopoly \*Monopoly;

//---------------------------------------------------------------------------

#endif

**Unit1.cpp**

//---------------------------------------------------------------------------

#include <vcl.h>

#include <jpeg.hpp>

#include <stdio.h>

#include <string.h>

#include <time.h>

#include <stdlib.h>

#pragma hdrstop

#include "Unit1.h"

typedef

struct PlayerProfile {

AnsiString name;

int cash, color, CoubSum, MovesLeft, DoubleMoves;

int fields[28], CurrentField;

int amountoffields;

struct PlayerProfile \*next;

} player;

typedef

struct Fields {

char name[30];

int cost, number;

int pay, star1, star2, star3, star4, BigStar;

int players, owner;

} F;

const int StartMoney = 15000;

const int Lap = 2000, LapBonus = 1000;

const int FieldCoordinate[40][2] = {{0 ,0}, {512, 63}, {565, 63}, {618, 63}, {671, 63}, {724, 63}, {777, 63}, {830, 63}, {883, 63}, {936, 63}, {0 ,0},

{1015, 140}, {1015, 193}, {1015, 246}, {1015, 299}, {1015, 352}, {1015, 405}, {1015, 458}, {1015, 511}, {1015, 564}, {0 ,0},

{936, 645}, {883, 645}, {830, 645}, {777, 645}, {724, 645}, {671, 645}, {618, 645}, {565, 645}, {512, 645}, {0 ,0},

{432, 565}, {432, 512}, {432, 459}, {432, 406}, {432, 353}, {432, 300}, {432, 247}, {432, 194}, {432, 141}};

int AmountOfPlayers = 0, HideGroupBox = 0;

F field[40];

player \*pl, \*head;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TMonopoly \*Monopoly;

//---------------------------------------------------------------------------

\_\_fastcall TMonopoly::TMonopoly(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void ShowButtons() {

Monopoly -> BitBtn1 -> Show();

Monopoly -> Chat -> Show();

Monopoly -> Chat -> Show();

Monopoly -> StartGameButton -> Show();

Monopoly -> AddPlayer1 -> Show();

Monopoly -> SendMessageButton -> Show();

Monopoly -> SendMessageButton -> Enabled = false;

Monopoly -> Edit1 -> Show();

Monopoly -> Edit1 -> Enabled = false;

Monopoly -> Chat -> Lines -> Text = "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";

Monopoly -> Panel1 -> Show();

Monopoly -> Panel3 -> Show();

Monopoly -> Panel5 -> Show();

Monopoly -> Panel6 -> Show();

Monopoly -> Panel8 -> Show();

Monopoly -> Panel9 -> Show();

Monopoly -> Panel11 -> Show();

Monopoly -> Panel12 -> Show();

Monopoly -> Panel13 -> Show();

Monopoly -> Panel14 -> Show();

Monopoly -> Panel15 -> Show();

Monopoly -> Panel16 -> Show();

Monopoly -> Panel18 -> Show();

Monopoly -> Panel19 -> Show();

Monopoly -> Panel21 -> Show();

Monopoly -> Panel23 -> Show();

Monopoly -> Panel24 -> Show();

Monopoly -> Panel25 -> Show();

Monopoly -> Panel26 -> Show();

Monopoly -> Panel27 -> Show();

Monopoly -> Panel28 -> Show();

Monopoly -> Panel29 -> Show();

Monopoly -> Panel31 -> Show();

Monopoly -> Panel32 -> Show();

Monopoly -> Panel34 -> Show();

Monopoly -> Panel35 -> Show();

Monopoly -> Panel37 -> Show();

Monopoly -> Panel39 -> Show();

}

void HideButtons() {

Monopoly -> BitBtn1 -> Hide();

Monopoly -> Chat -> Hide();

Monopoly -> AddPlayer1 -> Hide();

Monopoly -> AddPlayer2 -> Hide();

Monopoly -> AddPlayer3 -> Hide();

Monopoly -> AddPlayer4 -> Hide();

Monopoly -> Player1Shape -> Hide();

Monopoly -> Player2Shape -> Hide();

Monopoly -> Player3Shape -> Hide();

Monopoly -> Player4Shape -> Hide();

Monopoly -> GroupBox1 -> Hide();

Monopoly -> GroupBox2 -> Hide();

Monopoly -> Panel15 -> Hide();

Monopoly -> Panel16 -> Hide();

Monopoly -> SendMessageButton -> Hide();

Monopoly -> Edit1 -> Hide();

Monopoly -> Panel1 -> Hide();

Monopoly -> Panel3 -> Hide();

Monopoly -> Panel5 -> Hide();

Monopoly -> Panel6 -> Hide();

Monopoly -> Panel8 -> Hide();

Monopoly -> Panel9 -> Hide();

Monopoly -> Panel11 -> Hide();

Monopoly -> Panel12 -> Hide();

Monopoly -> Panel13 -> Hide();

Monopoly -> Panel14 -> Hide();

Monopoly -> Panel15 -> Hide();

Monopoly -> Panel16 -> Hide();

Monopoly -> Panel18 -> Hide();

Monopoly -> Panel19 -> Hide();

Monopoly -> Panel21 -> Hide();

Monopoly -> Panel23 -> Hide();

Monopoly -> Panel24 -> Hide();

Monopoly -> Panel25 -> Hide();

Monopoly -> Panel26 -> Hide();

Monopoly -> Panel27 -> Hide();

Monopoly -> Panel28 -> Hide();

Monopoly -> Panel29 -> Hide();

Monopoly -> Panel31 -> Hide();

Monopoly -> Panel32 -> Hide();

Monopoly -> Panel34 -> Hide();

Monopoly -> Panel35 -> Hide();

Monopoly -> Panel37 -> Hide();

Monopoly -> Panel39 -> Hide();

Monopoly -> Player1 -> Hide();

Monopoly -> Player2 -> Hide();

Monopoly -> Player3 -> Hide();

Monopoly -> Player4 -> Hide();

Monopoly -> Panel1 -> Hide();

Monopoly -> Panel2 -> Hide();

Monopoly -> Panel3 -> Hide();

Monopoly -> Panel4 -> Hide();

Monopoly -> Label3 -> Hide();

Monopoly -> GroupBox3 -> Hide();

Monopoly -> LeftCoubPanel -> Hide();

Monopoly -> RightCoubPanel -> Hide();

Monopoly -> DeletePlayer1 -> Hide();

Monopoly -> DeletePlayer3 -> Hide();

}

void ShowFields() {

Monopoly -> Image1 -> Show();

Monopoly -> Image2 -> Show();

Monopoly -> Image3 -> Show();

Monopoly -> Image4 -> Show();

Monopoly -> Image5 -> Show();

Monopoly -> Image6 -> Show();

Monopoly -> Image7 -> Show();

Monopoly -> Image8 -> Show();

Monopoly -> Image9 -> Show();

Monopoly -> Image10 -> Show();

Monopoly -> Image11 -> Show();

Monopoly -> Image12 -> Show();

Monopoly -> Image13 -> Show();

Monopoly -> Image14 -> Show();

Monopoly -> Image15 -> Show();

Monopoly -> Image16 -> Show();

Monopoly -> Image17 -> Show();

Monopoly -> Image18 -> Show();

Monopoly -> Image19 -> Show();

Monopoly -> Image20 -> Show();

Monopoly -> Image21 -> Show();

Monopoly -> Image22 -> Show();

Monopoly -> Image23 -> Show();

Monopoly -> Image24 -> Show();

Monopoly -> Image25 -> Show();

Monopoly -> Image26 -> Show();

Monopoly -> Image27 -> Show();

Monopoly -> Image28 -> Show();

Monopoly -> Image29 -> Show();

Monopoly -> Image30 -> Show();

Monopoly -> Image31 -> Show();

Monopoly -> Image32 -> Show();

Monopoly -> Image33 -> Show();

Monopoly -> Image34 -> Show();

Monopoly -> Image35 -> Show();

Monopoly -> Image36 -> Show();

Monopoly -> Image37 -> Show();

Monopoly -> Image38 -> Show();

Monopoly -> Image39 -> Show();

Monopoly -> Image40 -> Show();

}

void HideFields() {

Monopoly -> Image1 -> Hide();

Monopoly -> Image2 -> Hide();

Monopoly -> Image3 -> Hide();

Monopoly -> Image4 -> Hide();

Monopoly -> Image5 -> Hide();

Monopoly -> Image6 -> Hide();

Monopoly -> Image7 -> Hide();

Monopoly -> Image8 -> Hide();

Monopoly -> Image9 -> Hide();

Monopoly -> Image10 -> Hide();

Monopoly -> Image11 -> Hide();

Monopoly -> Image12 -> Hide();

Monopoly -> Image13 -> Hide();

Monopoly -> Image14 -> Hide();

Monopoly -> Image15 -> Hide();

Monopoly -> Image16 -> Hide();

Monopoly -> Image17 -> Hide();

Monopoly -> Image18 -> Hide();

Monopoly -> Image19 -> Hide();

Monopoly -> Image20 -> Hide();

Monopoly -> Image21 -> Hide();

Monopoly -> Image22 -> Hide();

Monopoly -> Image23 -> Hide();

Monopoly -> Image24 -> Hide();

Monopoly -> Image25 -> Hide();

Monopoly -> Image26 -> Hide();

Monopoly -> Image27 -> Hide();

Monopoly -> Image28 -> Hide();

Monopoly -> Image29 -> Hide();

Monopoly -> Image30 -> Hide();

Monopoly -> Image31 -> Hide();

Monopoly -> Image32 -> Hide();

Monopoly -> Image33 -> Hide();

Monopoly -> Image34 -> Hide();

Monopoly -> Image35 -> Hide();

Monopoly -> Image36 -> Hide();

Monopoly -> Image37 -> Hide();

Monopoly -> Image38 -> Hide();

Monopoly -> Image39 -> Hide();

Monopoly -> Image40 -> Hide();

}

int GetRandomNumber(void) {

int result = 0;

result = 1 + (rand() % 6);

return result;

}

void ColorPlayer() {

if (pl -> color == 1) Monopoly -> Player1 -> Color = clRed;

else if (pl -> color == 2) Monopoly -> Player2 -> Color = clBlue;

else if (pl -> color == 3) Monopoly -> Player3 -> Color = clLime;

else Monopoly -> Player4 -> Color = clPurple;

}

void DrawShapes() {

Monopoly -> GroupBox1 -> Show();

Monopoly -> Player1Shape -> Show();

Monopoly -> Player2Shape -> Show();

Monopoly -> Player3Shape -> Show();

Monopoly -> Player4Shape -> Show();

Monopoly -> Panel1 -> Enabled = true;

Monopoly -> Panel2 -> Enabled = true;

Monopoly -> Panel3 -> Enabled = true;

Monopoly -> Panel4 -> Enabled = true;

}

void ReadFromFile(F \*field, char \*path) {

int count = 0;

FILE\* f = fopen(path, "r");

while (!(feof(f))){

fscanf(f, "%s %d %d %d %d %d %d %d %d", field[count].name, &field[count].cost, &field[count].number, &field[count].pay, &field[count].star1, &field[count].star2, &field[count].star3, &field[count].star4, &field[count].BigStar);

field[count].players = 0;

field[count].owner = 0;

count++;

}

fclose(f);

}

void ViewInChat(int LeftCoub, int RightCoub) {

const int jail = 10, gotojail = 30, jackpot = 20;

AnsiString str;

str = field[pl -> CurrentField].name;

Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " выбрасывает " + AnsiString(LeftCoub) + ":" + AnsiString(RightCoub));

if ((field[pl -> CurrentField].cost != 0) && (pl -> CurrentField != 4) && (pl -> CurrentField != 36)) {

if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner != pl -> color)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на " + str + " и должен заплатить аренду " + AnsiString(field[pl -> CurrentField].pay) + "k");

else if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner == pl -> color)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на своё поле и ничего не платит");

else Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на " + str + " и задумывается о покупке");

} else if ((pl -> CurrentField == 4) || (pl -> CurrentField == 36)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на Налог и должен заплатить " + AnsiString(field[pl -> CurrentField].cost) + "k");

else if ((pl -> CurrentField == 2) || (pl -> CurrentField == 7) || (pl -> CurrentField == 17) || (pl -> CurrentField == 22) || (pl -> CurrentField == 33) || (pl -> CurrentField == 38)) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на поле " + str);

else if (pl -> CurrentField == jail) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на экскурсию в тюрьму");

else if (pl -> CurrentField == gotojail) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " отправляется в тюрьму за мошенничество");

else if (pl -> CurrentField == jackpot) Monopoly -> Chat -> Lines -> Add(AnsiString(pl -> name) + " попадает на Джекпот и может попытать удачу ");

}

void DrawCoubes(int LeftCoub, int RightCoub) {

switch (LeftCoub) {

case 1: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/1.bmp");

break;

case 2: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/2.bmp");

break;

case 3: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/3.bmp");

break;

case 4: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/4.bmp");

break;

case 5: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/5.bmp");

break;

case 6: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/6.bmp");

break;

}

switch (RightCoub) {

case 1: Monopoly -> Coub2 -> Picture -> LoadFromFile("images/BMP/Coub/1.bmp");

break;

case 2: Monopoly -> Coub2 -> Picture -> LoadFromFile("images/BMP/Coub/2.bmp");

break;

case 3: Monopoly -> Coub2 -> Picture -> LoadFromFile("images/BMP/Coub/3.bmp");

break;

case 4: Monopoly -> Coub2 -> Picture -> LoadFromFile("images/BMP/Coub/4.bmp");

break;

case 5: Monopoly -> Coub2 -> Picture -> LoadFromFile("images/BMP/Coub/5.bmp");

break;

case 6: Monopoly -> Coub2 -> Picture -> LoadFromFile("images/BMP/Coub/6.bmp");

break;

}

}

void DeletePlayerFromList() {

player \*DeleteAdress;

DeleteAdress = pl;

pl = head;

while (pl -> next != DeleteAdress) {

if (pl -> next == NULL) pl = head;

pl = pl -> next;

}

pl -> next = DeleteAdress -> next;

free(DeleteAdress);

AmountOfPlayers--;

pl = head;

pl = pl -> next;

}

void ShowMoney() {

switch (pl -> color) {

case 1: Monopoly -> Player1Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

break;

case 2: Monopoly -> Player2Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

break;

case 3: Monopoly -> Player3Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

break;

case 4: Monopoly -> Player4Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

break;

}

}

void ClearPlayersColor() {

Monopoly -> Player1 -> Color = clSilver;

Monopoly -> Player2 -> Color = clSilver;

Monopoly -> Player3 -> Color = clSilver;

Monopoly -> Player4 -> Color = clSilver;

}

void NextPlayer() {

if (pl -> MovesLeft == 0) {

ClearPlayersColor();

HideGroupBox = 1;

if (pl -> next == NULL) pl = head;

pl = pl -> next;

ColorPlayer();

Monopoly -> GroupBox2 -> Hide();

if (field[pl -> CurrentField].owner == pl -> color) HideGroupBox = 1;

}

}

void MovePlayer(int X, int Y) {

switch (pl -> color) {

case 1: Monopoly -> Player1Shape -> Top = Y;

Monopoly -> Player1Shape -> Left = X;

break;

case 2: Monopoly -> Player2Shape -> Top = Y;

Monopoly -> Player2Shape -> Left = X;

break;

case 3: Monopoly -> Player3Shape -> Top = Y;

Monopoly -> Player3Shape -> Left = X;

break;

case 4: Monopoly -> Player4Shape -> Top = Y;

Monopoly -> Player4Shape -> Left = X;

break;

}

}

void ClearRadioButtons() {

Monopoly -> RadioButton1 -> Checked = false;

Monopoly -> RadioButton2 -> Checked = false;

Monopoly -> RadioButton3 -> Checked = false;

Monopoly -> RadioButton4 -> Checked = false;

Monopoly -> RadioButton5 -> Checked = false;

Monopoly -> RadioButton6 -> Checked = false;

}

int PlayJackpod() {

int result = GetRandomNumber();

switch (result) {

case 1: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/1.bmp");

break;

case 2: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/2.bmp");

break;

case 3: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/3.bmp");

break;

case 4: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/4.bmp");

break;

case 5: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/5.bmp");

break;

case 6: Monopoly -> Coub1 -> Picture -> LoadFromFile("images/BMP/Coub/6.bmp");

break;

}

return result;

}

void HidePanels() {

Monopoly -> DeletePlayer1 -> Hide();

Monopoly -> Panel2 -> Hide();

Monopoly -> DeletePlayer3 -> Hide();

Monopoly -> Panel4 -> Hide();

}

void AddPlayerToList() {

AmountOfPlayers++;

pl -> next = (struct PlayerProfile\*)malloc(sizeof(struct PlayerProfile));

pl = pl -> next;

pl -> cash = StartMoney;

pl -> CoubSum = 0;

pl -> amountoffields = 0;

pl -> MovesLeft = 1;

pl -> DoubleMoves = 0;

pl -> CurrentField = 0;

pl -> next = NULL;

}

void DrawAndMovePlayer() {

const int jail = 10, jackpot = 20, gotojail = 30, start = 0;

int X = 0, Y = 0;

if (FieldCoordinate[pl -> CurrentField][0] != 0) MovePlayer(FieldCoordinate[pl -> CurrentField][0], FieldCoordinate[pl -> CurrentField][1]);

else if (pl -> CurrentField == start) {

switch (pl -> color) {

case 1: X = 456;

Y = 39;

break;

case 2: X = 456;

Y = 88;

break;

case 3: X = 408;

Y = 88;

break;

case 4: X = 408;

Y = 39;

break;

}

MovePlayer(X, Y);

} else if (pl -> CurrentField == jail) {

switch (pl -> color) {

case 1: X = 1043;

Y = 31;

break;

case 2: X = 1048;

Y = 80;

break;

case 3: X = 1024;

Y = 56;

break;

case 4: X = 992;

Y = 23;

break;

}

MovePlayer(X, Y);

} else if (pl -> CurrentField == jackpot) {

switch (pl -> color) {

case 1: X = 1040;

Y = 615;

break;

case 2: X = 1040;

Y = 664;

break;

case 3: X = 1024;

Y = 664;

break;

case 4: X = 1024;

Y = 615;

break;

}

MovePlayer(X, Y);

} else if (pl -> CurrentField == gotojail) {

pl -> CurrentField = jail;

switch (pl -> color) {

case 1: X = 984;

Y = 87;

break;

case 2: X = 976;

Y = 48;

break;

case 3: X = 1024;

Y = 96;

break;

case 4: X = 1008;

Y = 71;

break;

}

MovePlayer(X, Y);

}

}

void \_\_fastcall TMonopoly::StartGameButtonClick(TObject \*Sender)

{

ShowButtons();

ShowFields();

StartGameButton -> Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::FormCreate(TObject \*Sender)

{

char \*path = "Fields.txt";

ReadFromFile(field, path);

HideButtons();

HideFields();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::AddPlayer1Click(TObject \*Sender)

{

Player1 -> Show();

Image41 -> Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Player1ReadyClick(TObject \*Sender)

{

if (Player1Nickname -> Text.Length() != 0) {

head = (struct PlayerProfile\*)malloc(sizeof(struct PlayerProfile));

pl = head;

pl -> next = NULL;

AddPlayerToList();

pl -> color = 1;

pl -> name = Player1Nickname -> Text;

Player1Name -> Caption = Player1Nickname -> Text;

Player1Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

Player1Nickname -> Hide();

Player1Ready -> Hide();

Player1Money -> Show();

Player2 -> Show();

} else ShowMessage("Вы не написали ник");

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Player2ReadyClick(TObject \*Sender)

{

if (Player2Nickname -> Text.Length() != 0) {

AddPlayerToList();

pl -> color = 2;

pl -> name = Player2Nickname -> Text;

Player2Name -> Caption = Player2Nickname -> Text;

Player2Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

Player2Nickname -> Hide();

Player2Ready -> Hide();

Player2Money -> Show();

Player3 -> Show();

} else ShowMessage("Вы не написали ник");

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Player3ReadyClick(TObject \*Sender)

{

if (Player3Nickname -> Text.Length() != 0) {

AddPlayerToList();

pl -> color = 3;

pl -> name = Player3Nickname -> Text;

Player3Name -> Caption = Player3Nickname -> Text;

Player3Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

Player3Nickname -> Hide();

Player3Ready -> Hide();

Player3Money -> Show();

Player4 -> Show();

} else ShowMessage("Вы не написали ник");

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Player4ReadyClick(TObject \*Sender)

{

if (Player4Nickname -> Text.Length() != 0) {

AddPlayerToList();

pl -> color = 4;

pl -> name = Player4Nickname -> Text;

Player4Name -> Caption = Player4Nickname -> Text;

Player4Money -> Caption = "$" + IntToStr(pl -> cash) + "k";

Player4Nickname -> Hide();

Player4Ready -> Hide();

Player4Money -> Show();

} else ShowMessage("Вы не написали ник");

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn2Click(TObject \*Sender)

{

const int jail = 10, jackpot = 20, gotojail = 30;

int LeftCoub = 0, RightCoub = 0;

int RandomNumber = 0;

int more = 0;

HideGroupBox = 0;

HidePanels();

GroupBox1 -> Hide();

srand(time(NULL));

LeftCoub = GetRandomNumber();

RightCoub = GetRandomNumber();

DrawCoubes(LeftCoub, RightCoub); //Загрузка кубиков

//--------------------------------

LeftCoubPanel -> Show(); //Левый кубик

RightCoubPanel -> Show(); //Правый кубик

Timer1 -> Enabled = true;

//--------------------------------

pl -> CoubSum = LeftCoub + RightCoub;

if (LeftCoub == RightCoub) {

pl -> DoubleMoves++;

pl -> MovesLeft = 1;

} else {

pl -> MovesLeft = 0;

pl -> DoubleMoves = 0;

}

pl -> CurrentField += pl -> CoubSum;

if (pl -> CurrentField >= 40) { //Если начнётся новый круг

more = 1; //Для вывода сообщения о прохождении круга в правильном порядке

pl -> CurrentField %= 40;

pl -> cash += Lap;

ViewInChat(LeftCoub, RightCoub); //Вывод сообщения о прохождении круга

Chat -> Lines -> Add(AnsiString(pl -> name) + " получает " + IntToStr(Lap) + "k" + " за пройденный круг");

if (pl -> CurrentField == 0) {

Chat -> Lines -> Add(AnsiString(pl -> name) + " получает бонус " + IntToStr(LapBonus) + "k");

pl -> cash += LapBonus;

}

ShowMoney(); //Отрисовка денег

}

if (more == 0) ViewInChat(LeftCoub, RightCoub); //Если круг не был пересечён, то выводится сообщение в обычном порядке

DrawAndMovePlayer();

if ((field[pl -> CurrentField].cost != 0) && (pl -> CurrentField != 4) && (pl -> CurrentField != 36)) {

if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner != pl -> color)) {

if ((pl -> CurrentField == 12) || (pl -> CurrentField == 28)) {

field[pl -> CurrentField].pay \*= pl -> CoubSum;

}

if (pl -> cash - field[pl -> CurrentField].pay >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

BitBtn3 -> Caption = "Заплатить " + AnsiString(field[pl -> CurrentField].pay) + "k";

BitBtn3 -> Width = 441;

BitBtn4 -> Hide();

} else if ((field[pl -> CurrentField].owner != 0) && (field[pl -> CurrentField].owner == pl -> color)) {

NextPlayer();

HideGroupBox = 1;

}

else if (field[pl -> CurrentField].owner == 0) {

BitBtn3 -> Caption = "Заплатить " + AnsiString(field[pl -> CurrentField].cost) + "k";

if (pl -> cash - field[pl -> CurrentField].cost >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

}

} else if ((pl -> CurrentField == 4) || (pl -> CurrentField == 36)) {

BitBtn3 -> Caption = "Заплатить " + IntToStr(field[pl -> CurrentField].cost) + "k";

BitBtn3 -> Width = 441;

BitBtn4 -> Hide();

if (pl -> cash - field[pl -> CurrentField].cost >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

} else if ((pl -> CurrentField == 2) || (pl -> CurrentField == 7) || (pl -> CurrentField == 17) || (pl -> CurrentField == 22) || (pl -> CurrentField == 33) || (pl -> CurrentField == 38)) {

//поле шанс

RandomNumber = rand() % 2;

switch (RandomNumber) {

case 0: field[pl -> CurrentField].pay = 1000;

pl -> cash += field[pl -> CurrentField].pay;

Chat -> Lines -> Add(AnsiString(pl -> name) + " получает " + AnsiString(field[pl -> CurrentField].pay) + "k от банка");

ShowMoney();

NextPlayer();

HideGroupBox = 1;

break;

case 1: field[pl -> CurrentField].pay = 1000;

Chat -> Lines -> Add(AnsiString(pl -> name) + " должен заплатить банку " + IntToStr(field[pl -> CurrentField].pay) + "k");

BitBtn3 -> Caption = "Заплатить " + IntToStr(field[pl -> CurrentField].pay) + "k";

BitBtn3 -> Width = 441;

BitBtn4 -> Hide();

if (pl -> cash - field[pl -> CurrentField].pay >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

GroupBox2 -> Show();

break;

default: ShowMessage("kek, smth wrong!!!");

break;

}

} else if (pl -> CurrentField == jail) {

//ничего особенного, вывести сообщение и т.д.

NextPlayer();

} else if (pl -> CurrentField == gotojail) {

//переход в тюрьму

pl -> MovesLeft = 0;

pl -> DoubleMoves = 0;

NextPlayer();

} else if (pl -> CurrentField == jackpot) {

if (pl -> cash - field[pl -> CurrentField].pay >= 0) BitBtn3 -> Enabled = true;

else BitBtn3 -> Enabled = false;

GroupBox3 -> Show();

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn1Click(TObject \*Sender)

{

if (AmountOfPlayers > 1) {

DrawShapes();

if (AmountOfPlayers == 2) {

Player3Shape -> Hide();

Player4Shape -> Hide();

Panel3 -> Enabled = false;

Panel4 -> Enabled = false;

} else if (AmountOfPlayers == 3) {

Player4Shape -> Hide();

Panel4 -> Enabled = false;

}

SendMessageButton -> Enabled = true;

Edit1 -> Enabled = true;

if (AmountOfPlayers == 2) Player3 -> Hide();

else if (AmountOfPlayers == 3) Player4 -> Hide();

pl = head;

pl = pl -> next;

ColorPlayer();

BitBtn1 -> Hide();

} else ShowMessage("Для игры необходимо минимум 2 игрока");

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::SendMessageButtonClick(TObject \*Sender)

{

if ((Edit1 -> Text.Length() != 0) && (pl != NULL)) {

Chat -> Lines -> Text = Chat -> Lines -> Text + pl -> name + ": " + Edit1 -> Text;

Chat ->Perform(WM\_VSCROLL, SB\_BOTTOM, SB\_THUMBTRACK);

Edit1 -> Clear();

}

}

//--------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Timer1Timer(TObject \*Sender)

{

Timer1 -> Enabled = false;

LeftCoubPanel -> Hide();

RightCoubPanel -> Hide();

GroupBox1 -> Show();

if ((field[pl -> CurrentField].cost != 0) && (HideGroupBox == 0)) GroupBox2 -> Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn3Click(TObject \*Sender)

{

player \*temp;

int tempnumber = 0, tempcolor = 0, money = 0;

HideGroupBox = 0;

AnsiString str;

HidePanels();

str = field[pl -> CurrentField].name;

if ((field[pl -> CurrentField].owner == 0) && (pl -> CurrentField != 2) && (pl -> CurrentField != 4) && (pl -> CurrentField != 7) && (pl -> CurrentField != 17) && (pl -> CurrentField != 22) && (pl -> CurrentField != 33) && (pl -> CurrentField != 36) && (pl -> CurrentField != 38)) {

pl -> cash -= field[pl -> CurrentField].cost;

field[pl -> CurrentField].owner = pl -> color;

Chat -> Lines -> Add(AnsiString(pl -> name) + " покупает " + str + " за " + AnsiString(field[pl -> CurrentField].cost) + "k");

for (int i = 0; i <= ComponentCount - 1; i++) {

if ((Monopoly -> Components[i] -> ClassNameIs("TPanel")) && (Monopoly -> Components[i] -> Tag == pl -> CurrentField)) {

((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Caption = IntToStr(field[pl -> CurrentField].pay) + "k";

((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Font -> Color = clWhite;

if (pl -> color == 1) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clRed;

else if (pl -> color == 2) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clBlue;

else if (pl -> color == 3) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clLime;

else if (pl -> color == 4) ((TPanel\*)FindComponent("Panel" + IntToStr(pl -> CurrentField))) -> Color = clPurple;

}

}

} else if ((pl -> CurrentField == 4) || (pl -> CurrentField == 36)){

pl -> cash -= field[pl -> CurrentField].cost;

Chat -> Lines -> Add(AnsiString(pl -> name) + " заплатил налог " + AnsiString(field[pl -> CurrentField].cost) + "k");

} else if ((pl -> color != field[pl -> CurrentField].owner) && (field[pl -> CurrentField].owner != 0)) {

pl -> cash -= field[pl -> CurrentField].pay;

Chat -> Lines -> Add(AnsiString(pl -> name) + " заплатил аренду " + AnsiString(field[pl -> CurrentField].pay) + "k");

money = field[pl -> CurrentField].pay;

tempcolor = pl -> color;

tempnumber = field[pl -> CurrentField].owner;

for (int i = 0; i <= AmountOfPlayers - 1; i++) {

if (tempcolor == tempnumber) {

pl -> cash += money;

ShowMoney();

}

if (pl -> next == NULL) pl = head;

pl = pl -> next;

tempcolor = pl -> color;

}

} else if ((pl -> CurrentField == 2) || (pl -> CurrentField == 7) || (pl -> CurrentField == 17) || (pl -> CurrentField == 22) || (pl -> CurrentField == 33) || (pl -> CurrentField == 38)) {

pl -> cash -= field[pl -> CurrentField].pay;

Chat -> Lines -> Add(AnsiString(pl -> name) + " заплатил " + AnsiString(field[pl -> CurrentField].pay) + "k банку ");

}

ShowMoney();

GroupBox2 -> Hide();

NextPlayer();

BitBtn3 -> Width = 209;

BitBtn4 -> Show();

GroupBox2 -> Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn4Click(TObject \*Sender)

{

HideGroupBox = 0;

GroupBox2 -> Hide();

NextPlayer();

}

void \_\_fastcall TMonopoly::Player2Click(TObject \*Sender)

{

if ((pl -> color == 2) && (pl != NULL)) Panel2 -> Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Player1Click(TObject \*Sender)

{

if (AmountOfPlayers > 1)

if ((pl -> color == 1) && (pl != NULL)) DeletePlayer1 -> Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Player3Click(TObject \*Sender)

{

if (AmountOfPlayers > 1)

if ((pl -> color == 3) && (pl != NULL)) DeletePlayer3 -> Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::Player4Click(TObject \*Sender)

{

if (AmountOfPlayers > 1)

if ((pl -> color == 4) && (pl != NULL)) Panel4 -> Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::FormClick(TObject \*Sender)

{

HidePanels();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn5Click(TObject \*Sender)

{

DeletePlayerFromList();

AddPlayer1 -> Hide();

DeletePlayer1 -> Hide();

Player1 -> Hide();

Player1Shape -> Hide();

ColorPlayer();

if (AmountOfPlayers == 1) {

HideButtons();

HideFields();

Label3 -> Show();

Label3 -> Caption = AnsiString(pl -> name) + " ЗАНЯЛ 1 МЕСТО";

} else NextPlayer();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn6Click(TObject \*Sender)

{

DeletePlayerFromList();

Panel2 -> Hide();

Player2 -> Hide();

Player2Shape -> Hide();

ColorPlayer();

if (AmountOfPlayers == 1) {

HideButtons();

HideFields();

Label3 -> Show();

Label3 -> Caption = AnsiString(pl -> name) + " ЗАНЯЛ 1 МЕСТО";

} else NextPlayer();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn7Click(TObject \*Sender)

{

DeletePlayerFromList();

DeletePlayer3 -> Hide();

Player3 -> Hide();

Player3Shape -> Hide();

ColorPlayer();

if (AmountOfPlayers == 1) {

HideButtons();

HideFields();

Label3 -> Show();

Label3 -> Caption = AnsiString(pl -> name) + " ЗАНЯЛ 1 МЕСТО";

} else NextPlayer();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn8Click(TObject \*Sender)

{

DeletePlayerFromList();

Panel4 -> Hide();

Player4 -> Hide();

Player4Shape -> Hide();

ColorPlayer();

if (AmountOfPlayers == 1) {

HideButtons();

HideFields();

Label3 -> Show();

Label3 -> Caption = AnsiString(pl -> name) + " ЗАНЯЛ 1 МЕСТО";

} else NextPlayer();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn9Click(TObject \*Sender)

{

const int prize = 6000, price = 1000;

int MyNumber = 0, GuessNumber;

if ((RadioButton1 -> Checked) || (RadioButton2 -> Checked) || (RadioButton3 -> Checked) || (RadioButton4 -> Checked) || (RadioButton5 -> Checked) || (RadioButton6 -> Checked)) {

GroupBox3 -> Hide();

if (RadioButton1 -> Checked) MyNumber = 1;

else if (RadioButton2 -> Checked) MyNumber = 2;

else if (RadioButton3 -> Checked) MyNumber = 3;

else if (RadioButton4 -> Checked) MyNumber = 4;

else if (RadioButton5 -> Checked) MyNumber = 5;

else if (RadioButton6 -> Checked) MyNumber = 6;

pl -> cash -= price;

Chat -> Lines -> Add(AnsiString(pl -> name) + " поставил на " + AnsiString(MyNumber) + " 1000k" );

GuessNumber = PlayJackpod();

if (MyNumber == GuessNumber) {

Chat -> Lines -> Add("Выпадает " + AnsiString(GuessNumber) + " и " + AnsiString(pl -> name) + " выигрывает 6000k");

pl -> cash += prize;

} else Chat -> Lines -> Add("Выпадает " + AnsiString(GuessNumber) + " и " + AnsiString(pl -> name) + " ничего не выигрывает...");

ShowMoney();

} else ShowMessage("Выберите значение кубика");

ClearRadioButtons();

NextPlayer();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMonopoly::BitBtn10Click(TObject \*Sender)

{

HideGroupBox = 0;

NextPlayer();

}

//---------------------------------------------------------------------------