Министерство науки и образования РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)
Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Отчёт
по «Учебной практике»

Выполнил студент гр. 4306:
Табаков А. В.
Приняли:
Разумовский Г. В.
Павловский М.Г.

Санкт-Петербург
2016

Оглавление

1. Техническое Задание

1.1 Введение

На учебной практике должен быть спроектирован и разработан проект на любом объектно-ориентированном языке. Задание на разработку выбирается из методических указаний либо студент придумывает задание самостоятельно. Проект должен быть разработан, как приложение ОС Windows с оконным интерфейсом.

Обязательными требованиями при разработке кода проекта являются использование следующих принципов ООП:

- Инкапсуляция;
- Наследование;
- Полиморфизм;

В ходе разработки на ООП языке должны быть использованы:

- Конструкторы
- Абстрактные классы
- Обработка исключительных ситуаций

Мой проект заключается в создании игры на языке C++/CLI который основывается на платформе .Net. Игра будет разработана в виде головоломки, где люди должны закрасить всё поле одним цветом за определённое количество кликов.

1.2 Основание для разработки

Многим людям нравятся игры на развитие логики, моя игра пополнит эту категорию игр.

1.3 Назначение разработки

Разрабатываемая игра предназначена для всех людей от 3 до 100 лет. Игра направлена на развитие логики.

1.4 Требование к программному изделию

1.4.1 Требование к функциональным характеристикам

В игре должны быть реализованы разные уровни сложности. На сложных уровнях потребуются бонусы для возможности прохождения. Для поддержания интереса к игре, должны быть введены очки. Сложность каждого уровня зависит от размера поля и количества разных цветов, поэтому необходима автоматическая генерация поля по заданным параметрам. Требуется предусмотреть возможность проигрыша.

Необходима кампания в которой уровень сложности будет расти линейно в зависимости от пройденных уровней. Также людям нужно дать возможность задать параметры игры самостоятельно.

1.4.2 Требования к надёжности

Игра должна удовлетворять требованиям надёжности выполнения программы. Должна быть обеспечена безотказная работа с любым типом игры. В игре нажатие на квадрат, часть игрового поля, должно всегда верно обрабатываться и закрашивать требуемые соседние квадраты. Особых требований к надёжности и отказоустойчивости не предъявляется

1.4.3 Условия эксплуатации

Проект будет использоваться в нормальных условиях при отсутствии повышенных нагрузок и вредоносных внешних воздействий со стороны аппаратных и программных средств. Пользователь игры должен обладать минимальным опытом работы с персональным компьютером.

1.4.4 Требование к составу и параметрам технических средств

Для нормального функционирования игры персональный компьютер должен обладать следующими характеристиками:

- Процессор: Pentium 4 с частотой 1Ггц
- Оперативная память: 512Мб
- Видеоадаптер: VGA 640x480
- Свободное место на жёстком диске: 100Мб
- Манипулятор: мышь или трекбол
- Операционная система: Windows XP и выше

1.5 Требование к программной документации

Документация на игру должна содержать пояснительную записку.
Пояснительная записка проекта должна иметь следующую структуру:

- Техническое задание
- Руководство пользователя
- Исходные тексты программы

1.6 Стадии и этапы разработки

Процесс проектирования и разработки игры должен содержать следующие стадии и этапы:

1. Разработка технического задания
2. Создания прототипа интерфейса пользователя
3. Разработка архитектуры
4. Написание кода

1.7 Порядок контроля и приёмки

Отчёт должен содержать оттестированную программу (игру) и пояснительную записку, содержащую следующие пункты:

1. Техническое задание
2. Руководство пользователя
3. Исходные тексты программы

При сдаче проекта необходимо предъявить работоспособную программу(игру) и продемонстрировать все её функциональные возможности в соответствии с заданием.

2. Создание прототипа интерфейса пользователя

Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных решений, на основе конкретных технологий ввода-вывода информации. Когда речь идёт об интерфейсе пользователя, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с конкретными устройствами. Для каждой экранной формы указываются поля ввода и перечень элементов управления, действия пользователя (нажать кнопку, выбрать пункт меню, ввести данные, нажать правую/левую кнопку мыши) и отклики системы (отобразить данные, вывести подсказку, переместить курсор). Такое описание интерфейса представляется в виде таблицы экранных форм.

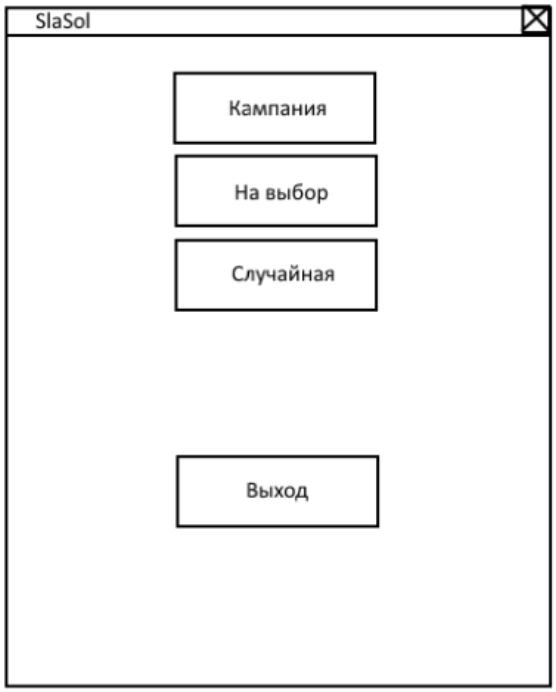

Рис 1. Набросок главного меню

Рис 2. Набросок меню выбора уровня

Рис 3. Набросок слоя задания параметров своей игры

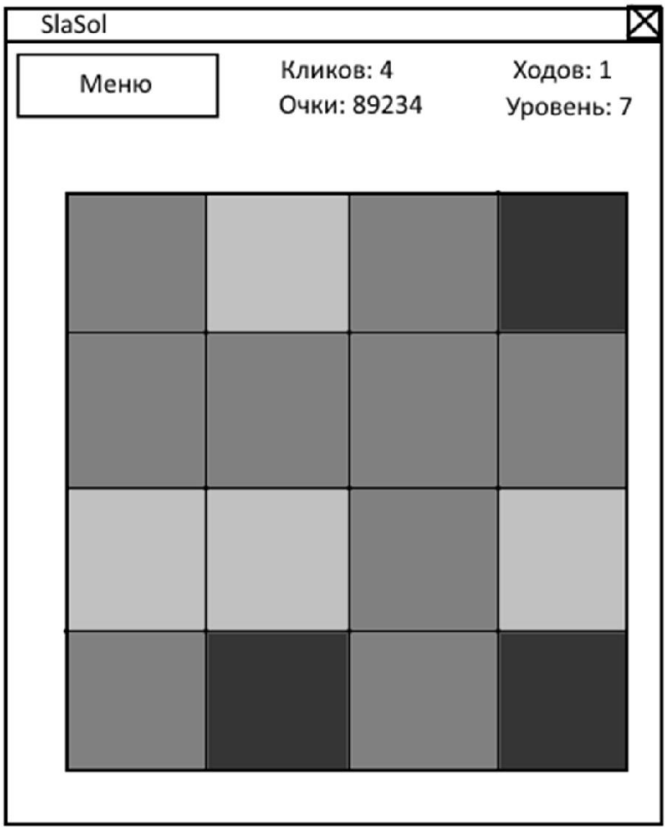

Рис 4. Набросок игрового поля

Экранный слой	Элементы управления	Действия пользователя	Отклик системы
Главное меню	Кнопка «Кампания»	Нажатие левой кнопки мыши	Скроется слой главного меню Появится слой игрового поля Появится метка «Уровень:0»
	Кнопка «На выбор»	Нажатие левой кнопки мыши	Скроется слой главного меню Появится слой меню выбора уровня
	Кнопка «Случайная»	Нажатие левой кнопки мыши	Скроется слой главного меню Появится слой игрового поля
	Кнопка «Выход»	Нажатие левой кнопки мыши	Выведется диалоговое окно выхода, в котором пользователь должен подтвердить или отказаться
Меню выбора уровня	Кнопка «Лёгкий»	Нажатие левой кнопки мыши	Скроется слой меню выбора уровня Появится слой игрового поля
	Кнопка «Средний»	Нажатие левой кнопки мыши	Скроется слой меню выбора уровня Появится слой игрового поля
	Кнопка «Сложный»	Нажатие левой кнопки мыши	Скроется слой меню выбора уровня Появится слой игрового поля
	Кнопка «Задать»	Нажатие левой кнопки мыши	Скроется слой меню выбора уровня Появится слой задания параметров игры
	Кнопка «Главное меню»	Нажатие левой кнопки мыши	Скроется слой меню выбора уровня Появится слой главного меню
Задание параметров игры	Кнопка «Начать»	Нажатие левой кнопки мыши	Скроется слой задания параметров игры Появится слой игрового поля
	Кнопка «Главное меню»	Нажатие левой кнопки мыши	Скроется слой задания параметров игры Появится слой главного меню
	Область задания количества цветов	Нажатие левой кнопки мыши на стрелку вверх или вниз	При нажатии на стрелку вверх значение уменьшится, вниз увеличится

	Область задания размера поля	Нажатие левой кнопки мыши на стрелку вверх или вниз	При нажатии на стрелку вверх значение уменьшится, вниз увеличится
Игровое поле	Таблица панелей (NxN)	-	-
	Разноцветная панель (в количестве клеток таблицы)	Нажатие левой кнопки мыши	Все панели стоящие на заданной горизонтали и вертикали меняют цвет на цвет данной панели
	Кнопка «Меню»	Нажатие левой кнопки мыши	Выведется диалоговое окно выхода, в котором пользователь должен подтвердить или отказаться. Если пользователь подтвердит выход в меню, то: Скроется слоя игрового поля Появится слой главного меню
	Метка «Кликов:N»	-	-
	Метка «Ходов:N»	-	-
	Метка «Очки:N»	-	-

3. Руководство пользователя

3.1 Назначение программы

Игра предназначена для всех людей от 3 до 100 лет. Игра направлена на развитие логики.

3.2 Условия выполнения задачи

Игра должна эксплуатироваться на персональном компьютере со следующими минимальными системными требованиями:

- Процессор: Pentium 4 с частотой 1Ггц
- Оперативная память: 512Мб
- Видеоадаптер: VGA 640x480
- Свободное место на жёстком диске: 100Мб
- Манипулятор: мышь или трекбол
- Операционная система: Windows XP и выше

3.3 Описание программы

Для эксплуатации данной игры требуются минимальный опыт работы за персональным компьютером.

Игра заключается в закрашивании всего игрового поля одним цветом (любым из доступных на экране). Панели закрашиваются по горизонтали и вертикали от той панели на которой была нажата левая кнопка мыши. На сложных уровнях с размером поля более 5x5 закрашиваются и соседние 8 клеток вокруг задействованной панели. Однако данный бонус работает только если задействованная панель находится на расстоянии больше 2 панелей от стены и больше 20% от размера поля, также считая от стены. Количество очков зависит от количества закрашенных панелей за один ход.

В режиме Кампания пользователю предлагаются 15 уровней с нарастанием сложности. Очки за каждый уровень накапливаются. В качестве бонуса неиспользованные клики дают 5000 очков за каждый.

3.4 Выполнение программы

Чтобы запустить данную программу, на персональном компьютере пользователя должны быть распространяемые пакеты Visual C++. Для запуска программы необходимо два раза нажать левой кнопкой мыши на иконку программы SlaSol.exe. После запуска, пользователь увидит главное меню игры, где он сможет:

- Запустить кампанию нажав на кнопку «Кампания»
- Войти в меню выбора уровней нажав на кнопку «На выбор»
- В меню выбора уровней пользователь может: выбрать уровень сложности и начать игру на данном уровне нажав на соответствующую кнопку, перейти в меню задания параметров игры для запуска собственной игры либо выйти в главное меню.
- Запустить случайную игру нажав на кнопку «Случайная», где будет сгенерирована случайная игра
- Выйти из игры нажав кнопку «Выход»

3.5 Проверка программы

Для проверки работоспособности программы, запустим игру.

Первым делом проверим кампанию:

- Нажимаем левой кнопкой мыши на кнопку «Кампания», появляется слой игрового поля с квадратом 2x2 одного цвета.
- Щёлкаем по любой панели. Замечаем, что теперь поле стало больше, добавился цвет, уровень изменился на первый и количество кликов теперь 2.
- Нажимаем на панели, так, чтобы закрасить другие панели в один цвет. После нажатия замечаем, что все панели по вертикали и горизонтали стали цвета, задействованной панели. Количество ходов и очки увеличились, а количество кликов уменьшилось.

- Если всё сделали правильно количество панелей увеличится, уровень изменится на второй и количество кликов станет 3, также добавятся очки за каждую перекрашенную панель.
- Нажимаем несколько раз на одну и ту же панель и проигрываем, получаем сообщение о том, что мы проиграли.

Проверим случайную игру:

- Нажимаем на панели, так, чтобы закрасить другие панели в один цвет. После нажатия замечаем, что все панели по вертикали и горизонтали стали цвета, задействованной панели. Количество ходов и очки увеличились, а количество кликов уменьшилось.

Проверим кнопку меню на игровом поле:

- Нажимаем кнопку «Меню», получаем сообщение нажимаем «нет» и продолжаем игру.
- Нажимаем на панель. После нажатия замечаем, что все панели по вертикали и горизонтали стали цвета, задействованной панели. Количество ходов и очки увеличились, а количество кликов уменьшилось.
- Нажимаем кнопку «Меню», получаем сообщение нажимаем «да» и выходим в главное меню

Проверим меню выбора уровня:

- Выбираем лёгкий уровень, появляется игровое поле с не очень сложным уровнем. Выигрываем, получаем сообщение о выигрыше, иначе о проигрыше.
- Выбираем средний уровень, появляется игровое поле уровнем по сложнее.
- Выбираем сложный уровень, появляется игровое поле с большим количеством панелей и цветов.
- Нажимаем на кнопку «Задать», появляются поля для задания игрового поля, выбираем 4 цвета и размер поля 17x17 и нажимаем «Начать». Видим, что перед нами поле 17x17 и с *4 цветами.
  *Цветов может быть меньше, так как цвет каждой панели генерируется случайно, возможна ситуация, когда один или несколько цветов не будут использованы.
- Нажимаем на кнопку «Главное меню» попадаем в главное меню.

4. Исходные тексты программы

Заголовочные файлы

AbstractGame.h

```
#pragma once
namespace SlaSolGame {
   class AbstractGame {

   public:
      virtual int getSteps() = 0;
      virtual void setSteps(int count) = 0;
      virtual void incSteps() = 0;
      virtual int getClicks() = 0;
      virtual void setClicks(int count) = 0;
      virtual void decClicks() = 0;
      virtual int getScores() = 0;
      virtual void setScores(int value) = 0;
      virtual void addScores(int value) = 0;
      virtual int getFieldSize() = 0;
      virtual void setFieldSize(int value) = 0;
      virtual int getColors() = 0;
      virtual void setColors(int value) = 0;
      virtual void clearCounters() = 0;
      virtual int checkerFinishRound(bool check) = 0;
      virtual void win() = 0;
      virtual void loose() = 0;
      virtual void start() = 0;
      virtual void end(System::Object^ sender) = 0;
      virtual ~AbstractGame() {};
   };
}
```

Campaign.h

```
#pragma once
#include "Game.h"

#define MAXLEVEL 15

namespace SlaSolGame {
   class Campaign : public Game {
      int level;
      int maxLevel;
      int campaignFieldSize[MAXLEVEL] = { 2, 3, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7, 8, 8 };
      int campaignColors[MAXLEVEL] = { 1, 2, 2, 3, 2, 3, 4, 2, 3, 4, 2, 3, 4, 2, 3 };
   public:
      Campaign() : Game(), level(0), maxLevel(MAXLEVEL - 1) {};
      ~Campaign();

      int getLevel();
      void setLevel(int value);
      void incLevel();
      int getFieldSizeByLevel(int level);
      int getCountColorByLevel(int level);
      int getMaxLevel();

      void start();
      void end(System::Object^ sender);
      void win();
      void loose();
   };
}
```

Display.h

```
#pragma once
#include <cliext\vector>
#include "GameWindow.h"
#include "LevelMenu.h"
#include "MainMenu.h"

namespace SlaSol {
   ref class mainWindow;
}
```

```
namespace SlaSolDisplay {
   public ref class Display : public GameWindow {
      //GameWindow gameWindow;
      LevelMenu levelMenu;
      MainMenu mainMenu;

   public:
      static SlaSol::mainWindow^ mainWindow;
      Display();

      static void addControl(System::Windows::Forms::Control^);

      void showMainMenu();
      void hideMainMenu();
      void showLevelMenu();
      void hideLevelMenu();
      void showLevelSetMenu();
      void hideLevelSetMenu();
      void showGameField();
      void hideGameField();

      int getFieldSizeFromLevelMenuDomain();
      int getCountOfColorsFromLevelMenuDomain();

      static void setVisibleVector(cliext::vector<System::Object^> vector, bool value);
   };
}
```

Errors.h

```
#pragma once
class WrongParams {
public:
   int first, second;
   WrongParams(int iFirst, int iSecond) :first(iFirst), second(iSecond) {}
   void getMessage();
};

void errorMessageWithGui();
```

Game.h

```
#pragma once
#include <cliext/vector>
#include "Display.h"
#include "AbstractGame.h"

namespace SlaSolGame {
   class Game : public AbstractGame {
      int steps; //Количество ходов в текущей игре
      int clicks; //Количество кликов которые остались
      int scores; //Очки в игре
      int fieldSize; //Размер игрового поля
      int colors; //Количество цветов для генерации

   public:
      Game() :steps(0), scores(0), fieldSize(0), colors(0) {}
      Game(int iFieldSize, int iColors);
      virtual ~Game() {};

      int getSteps();
      void setSteps(int value);
      void incSteps();
      int getClicks();
      void setClicks(int value);
      void decClicks();
      int getScores();
      void setScores(int value);
      void addScores(int value);
      int getFieldSize();
      void setFieldSize(int value);
      int getColors();
      void setColors(int value);
```

```
      void clearCounters();
      int checkerFinishRound(bool check);

      virtual void win() = 0;
      virtual void loose() = 0;
      virtual void start() = 0;
      virtual void end(System::Object^ sender) = 0;
   };
}
```

GameWindow.h

```
#pragma once
#include <cliext/vector>

namespace SlaSolDisplay {
   public ref class GameWindow {
      /****************Labels*************************/
      System::Windows::Forms::Label^  stepsLabel;
      System::Windows::Forms::Label^  levelLabel;
      System::Windows::Forms::Label^  clicksLeftLabel;
      System::Windows::Forms::Label^  scoreLabel;
      /****************Buttons************************/
      System::Windows::Forms::Button^  menuButton;
      /****************Table Layout*******************/
      cliext::vector<System::Object^> gameVector;
   protected:
      System::Windows::Forms::TableLayoutPanel^ gameField;
   public:
      GameWindow();

      /****************Labels************************/
      void initStepsLabel();
      void initLevelLabel();
      void initClicksLeftLabel();
      void initScoreLabel();

      void setStepsLabel(int steps, bool visible);
      void setLevelLabel(int level, bool visible);
      void setClicksLeftLabel(int clicks, bool visible);
      void setScoreLabel(int scores, bool visible);
      /****************Buttons************************/
      void initMenuButton();
      /****************Table Layout*******************/
      void initGameField();
      void updateGameField(int row, int col);

      void showGameField();
      void hideGameField();

      void writeLabel(System::Windows::Forms::Label^, int);
      int changeColorOnRowColumn(System::Windows::Forms::Control^ pictureBox);
      void fillGameField(int countOfColors);
      bool isOneColor();
   };
}
```

LevelMenu.h

```
#pragma once
#include <cliext\vector>

namespace SlaSolDisplay {
   public ref class LevelSetMenu {
      /****************Labels*************************/
      System::Windows::Forms::Label^  helpLabelColors;
      System::Windows::Forms::Label^  helpLabelFieldSize;
      /****************Buttons************************/
   private:System::Windows::Forms::Button^  startSettedGameButton;

   protected:
      System::Windows::Forms::Button^  mainMenuButton;
      /****************Domains************************/
```

```
      System::Windows::Forms::DomainUpDown^  domainColorsCount;
      System::Windows::Forms::DomainUpDown^  domainFieldSize;
      cliext::vector<System::Object^> levelSetMenuVector;
   public:
      LevelSetMenu();

      /****************Labels************************/
      void initHelpLabelColors();
      void initHelpLabelFieldSize();
      /****************Domains***********************/
      void initDomainColorsCount();
      void initDomainFieldSize();
      /****************Buttons***********************/
      void initStartSettedGameButton();
      void initMainMenuButton();
   };

   public ref class LevelMenu : public LevelSetMenu {
      /****************Buttons***********************/
      System::Windows::Forms::Button^  lightLevelButton;
      System::Windows::Forms::Button^  middleLevelButton;
      System::Windows::Forms::Button^  hardLevelButton;
      System::Windows::Forms::Button^  setGameButton;

      cliext::vector<System::Object^> levelMenuVector;
   public:
      LevelMenu();

      /****************Buttons***********************/
      void initLightLevelButton();
      void initMiddleLevelButton();
      void initHardLevelButton();
      void initSetGameButton();

      int getFieldSizeFromDomain();
      int getCountOfColorsFromDomain();

      void show();
      void hide();
      void showLevelSetMenu();
      void hideLevelSetMenu();
   };
}
```

MainMenu.h

```
#pragma once
#include <cliext\vector>

namespace SlaSolDisplay {
   public ref class MainMenu {
      /****************Buttons***********************/
      System::Windows::Forms::Button^  campaignButton;
      System::Windows::Forms::Button^  settedGameButton;
      System::Windows::Forms::Button^  randomGameButton;
      System::Windows::Forms::Button^  quitButton;

      cliext::vector<System::Object^> mainMenuVector;
   public:
      MainMenu();

      /****************Buttons***********************/
      void initCampaignButton();
      void initChooseGameButton();
      void initRandomGameButton();
      void initQuitButton();

      void show();
      void hide();
   };
}
```

mainWindow.h

```
#pragma once
#include "Errors.h"
#include "Game.h"
#include "Display.h"
#include "Campaign.h"
#include "SettedGame.h"
#include "RandomGame.h"

namespace SlaSol {
   using namespace SlaSolDisplay;
   using namespace SlaSolGame;
   using namespace System;
   using namespace System::Windows::Forms;

   /// <summary>
   /// Сводка для mainWindow
   /// </summary>
   public ref class mainWindow : public System::Windows::Forms::Form {

   private: AbstractGame *game;
   public: static Display^ display;

   public:
      mainWindow(void)
      {
         InitializeComponent();
         Display::mainWindow = this;
         display = gcnew Display();
      }

   protected:
      /// <summary>
      /// Освободить все используемые ресурсы.
      /// </summary>
      ~mainWindow()
      {
         if (components)
         {
            delete components;
         }
      }

   private:
      /// <summary>
      /// Обязательная переменная конструктора.
      /// </summary>
      System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
      /// <summary>
      /// Требуемый метод для поддержки конструктора — не изменяйте
      /// содержимое этого метода с помощью редактора кода.
      /// </summary>
      void InitializeComponent(void)
      {
         System::ComponentModel::ComponentResourceManager^  resources = (gcnew
System::ComponentModel::ComponentResourceManager(mainWindow::typeid));
         //
         // mainWindow
         //
         this->SuspendLayout();
         this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
         this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
         this->BackColor = System::Drawing::Color::MediumTurquoise;
         this->ClientSize = System::Drawing::Size(534, 462);
         this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;
         this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources-
>GetObject(L"$this.Icon")));
         this->MaximizeBox = false;
```

```
        this->Name = L"mainWindow";
        this->Text = L"SlaSol";
        this->ResumeLayout(false);
        this->PerformLayout();

      }
#pragma endregion
   public: System::Void newGameButton_Click(System::Object^  sender, System::EventArgs^  e) {
      try {
         String^ message = "Вы уверены, что хотите начать новую игру?";
         String^ caption = "Новая игра?";
         MessageBoxButtons buttons = MessageBoxButtons::YesNo;
         System::Windows::Forms::DialogResult result;
         result = MessageBox::Show(this, message, caption, buttons, MessageBoxIcon::Question,
MessageBoxDefaultButton::Button1);
         if (result == System::Windows::Forms::DialogResult::Yes) {
            delete game;
            display->hideGameField();
            display->showMainMenu();
         }
      }
      catch (Exception^ e) {
         MessageBox::Show(e->Message);
      }
      catch (...) {
         errorMessageWithGui();
      }
   }
   public: System::Void campaignButton_Click(System::Object^  sender, System::EventArgs^  e) {
      game = new Campaign();
      game->start();
   }

   public: System::Void settedGameButton_Click(System::Object^  sender, System::EventArgs^  e) {
      display->hideMainMenu();
      display->showLevelMenu();
   }

   public: System::Void lightLevelButton_Click(System::Object^  sender, System::EventArgs^  e) {
      display->hideLevelMenu();
      game = new SettedGame(2, 2, 4);
      game->start();
   }
   public: System::Void middleLevelButton_Click(System::Object^  sender, System::EventArgs^  e) {
      display->hideLevelMenu();
      game = new SettedGame(3, 3, 5);
      game->start();
   }

   public: System::Void hardLevelButton_Click(System::Object^  sender, System::EventArgs^  e) {
      display->hideLevelMenu();
      game = new SettedGame(7, 7, 14);
      game->start();
   }

   public: System::Void startSettedGameButton_Click(System::Object^  sender, System::EventArgs^
e) {
      display->hideLevelSetMenu();
      game = new SettedGame(display->getFieldSizeFromLevelMenuDomain(), display-
>getCountOfColorsFromLevelMenuDomain());
      game->start();
   }

   public: System::Void randomGameButton_Click(System::Object^  sender, System::EventArgs^  e) {
      game = new RandomGame();
      game->start();
   }

   public: System::Void quitButton_Click(System::Object^  sender, System::EventArgs^  e) {
      String^ message = "Вы уверены, что хотите выйти? ";
```

```
      String^ caption = "Закрыть игру?";
      MessageBoxButtons buttons = MessageBoxButtons::YesNo;
      System::Windows::Forms::DialogResult result;
      result = MessageBox::Show(this, message, caption, buttons, MessageBoxIcon::Exclamation,
MessageBoxDefaultButton::Button1);
      if (result == System::Windows::Forms::DialogResult::Yes) {
         Application::Exit();
      }
   }

   public: System::Void block_Click(System::Object^ sender, System::EventArgs^  e) {
      game->end(sender);
   }

   public: System::Void setGameButton_Click(System::Object^  sender, System::EventArgs^  e) {
      display->hideLevelMenu();
      display->showLevelSetMenu();
   }

   public: System::Void mainMenuButton_Click(System::Object^  sender, System::EventArgs^  e) {
      display->hideLevelMenu();
      display->hideLevelSetMenu();
      display->showMainMenu();
   }
   };
}
```

RandomGame.h

```
#pragma once
#include "SettedGame.h"

namespace SlaSolGame {
   class RandomGame : public SettedGame {
   public:
      RandomGame() : SettedGame() {};

      void start();
   };
}
```

SettedGame.h

```
#pragma once
#include "Game.h"

namespace SlaSolGame {
   class SettedGame : public Game {
   public:
      SettedGame() : Game() {}
      SettedGame(int iBeginNumber, int iModForRandColor, int iModForRandFieldSize);
      SettedGame(int iFieldSize, int iColors);

      void start();
      void end(System::Object^ sender);
      void win();
      void loose();
   };
}
```

Файлы реализации

Campaign.cpp

```
#include "Campaign.h"
#include "mainWindow.h"

using namespace SlaSolGame;

Campaign::~Campaign() {
   SlaSol::mainWindow::display->setLevelLabel(0, false);
}

int Campaign::getFieldSizeByLevel(int level) {
   return campaignFieldSize[level];
}
```

```
int Campaign::getCountColorByLevel(int level) {
   return campaignColors[level];
}

int Campaign::getMaxLevel() {
   return maxLevel;
}

int Campaign::getLevel() {
   return level;
}

void Campaign::setLevel(int value) {
   level = value;
}

void Campaign::incLevel() {
   ++level;
}

void Campaign::start() {
   clearCounters();
   try {
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), true);
      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), true);
      SlaSol::mainWindow::display->setScoreLabel(getScores(), true);
      SlaSol::mainWindow::display->setLevelLabel(level, true);
      setFieldSize(campaignFieldSize[level]);
      setColors(campaignColors[level]);
      SlaSol::mainWindow::display->updateGameField(getFieldSize(), getFieldSize());
      setClicks(getFieldSize() - 1);

      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), false);
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), false);

      SlaSol::mainWindow::display->showGameField();
      SlaSol::mainWindow::display->hideMainMenu();

      SlaSol::mainWindow::display->fillGameField(getColors());
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
};
void Campaign::end(System::Object^ sender) {
   try {
      incSteps();
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), true);
      decClicks();
      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), true);
      addScores(SlaSol::mainWindow::display-
>changeColorOnRowColumn((cli::safe_cast<System::Windows::Forms::Control^>(sender))));
      SlaSol::mainWindow::display->setScoreLabel(getScores(), true);

      int result = checkerFinishRound(SlaSol::mainWindow::display->isOneColor());
      if (result != 0) {
         if (result > 0) {
            if (level == maxLevel) {
               SlaSol::mainWindow::display->hideGameField();
               SlaSol::mainWindow::display->showMainMenu();
               delete this;
            }
            else {
               addScores(getClicks() * 5000);
               incLevel();
               start();
```

```
            }
         }
         else {
            SlaSol::mainWindow::display->hideGameField();
            SlaSol::mainWindow::display->showMainMenu();
            delete this;
         }
      }
   }
   catch (WrongParams& e) {
      e.getMessage();
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
};

void Campaign::win() {
   if (level != maxLevel && level % 3 != 0 || level == 0)
      return;

   System::String^ message;
   System::String^ caption;

   switch (level)
   {
   case (MAXLEVEL - 1):
      caption = L"ПОБЕДА!";
      message = L"Поздравляю, Вы прошли кампанию!\nВам нет равных ведь эта кампания
непроходима.\n" + \
         "Вы одержали победу над беспощадной судьбой, теперь Вы способны делать ВСЁ! \n" + \
         "Параметр вашей личной удачи повышен на " + (rand() % 20 + 40) + "%\n" + \
         "Логика развита на " + (rand() % 20 + 30) + "%\n" + \
         "Поздравляем!\nХотите повысить ваши параметры ещё? Предлагаю пройти кампанию снова!";
      break;
   case 3:
      caption = L"КМБ";
      message = L"Ого! Вы прошли 3 уровня, чтож, дальше так просто не будет.";
      break;
   case 6:
      caption = L"Высоко";
      message = L"Значит Вы уже прошли 6 уровней? Не думаю что осилите дальше.";
      break;
   case 9:
      caption = L"Всё дальше в лес";
      message = L"Так, Вы преодолели 9 уровней, мозги закипают, да?";
      break;
   case 12:
      caption = L"Под землёй";
      message = L"Хорошо, 12 уровней позади, может стоит остановиться на достигнутом?";
      break;
   default:
      break;
   }

   System::Windows::Forms::MessageBoxButtons buttons =
System::Windows::Forms::MessageBoxButtons::OK;
   System::Windows::Forms::MessageBox::Show(message, caption, buttons,
System::Windows::Forms::MessageBoxIcon::Exclamation,
System::Windows::Forms::MessageBoxDefaultButton::Button1);
}

void Campaign::loose() {
   System::String^ message = "Вы проиграли!";
   System::String^ caption = "Раунд закончен!";
   System::Windows::Forms::MessageBoxButtons buttons =
System::Windows::Forms::MessageBoxButtons::OK;
```

```
   System::Windows::Forms::MessageBox::Show(message, caption, buttons,
System::Windows::Forms::MessageBoxIcon::Error,
System::Windows::Forms::MessageBoxDefaultButton::Button1);
};
```

Display.cpp

```
#include "Display.h"
#include "mainWindow.h"
#include "Errors.h"

using namespace SlaSolDisplay;

Display::Display() {}

void Display::addControl(System::Windows::Forms::Control^ control) {
   mainWindow->Controls->Add(control);
}

void Display::showMainMenu() {
   mainMenu.show();
}
void Display::hideMainMenu() {
   mainMenu.hide();
}
void Display::showLevelMenu() {
   levelMenu.show();
}
void Display::hideLevelMenu() {
   levelMenu.hide();
}
void Display::showLevelSetMenu() {
   levelMenu.showLevelSetMenu();
}
void Display::hideLevelSetMenu() {
   levelMenu.hideLevelSetMenu();
}
void Display::showGameField() {
   GameWindow::showGameField();
}
void Display::hideGameField() {
   GameWindow::hideGameField();
}
int Display::getFieldSizeFromLevelMenuDomain() {
   return levelMenu.getFieldSizeFromDomain();
}
int Display::getCountOfColorsFromLevelMenuDomain() {
   return levelMenu.getCountOfColorsFromDomain();
}

void Display::setVisibleVector(cliext::vector<System::Object^> vector, bool value) {
   try {
      for (auto it = vector.begin(); it != vector.end(); ++it) {
         (cli::safe_cast<System::Windows::Forms::Control^>(*it))->Visible = value;
      }
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}
```

Errors.cpp

```
#include "Errors.h"

void WrongParams::getMessage() {
   System::String^ message = "Первый параметр: " + first + "\nВторой параметр: " + second;
   System::String^ caption = "Неправильный параметр";
   System::Windows::Forms::MessageBoxButtons buttons =
System::Windows::Forms::MessageBoxButtons::OK;
```

```
   System::Windows::Forms::MessageBox::Show(message, caption, buttons,
System::Windows::Forms::MessageBoxIcon::Error,
System::Windows::Forms::MessageBoxDefaultButton::Button1);
}

void errorMessageWithGui() {
   System::String^ message = "Произошла ошибка. Повторите действие";
   System::String^ caption = "Повторите действие";
   System::Windows::Forms::MessageBoxButtons buttons =
System::Windows::Forms::MessageBoxButtons::OK;
   System::Windows::Forms::MessageBox::Show(message, caption, buttons,
System::Windows::Forms::MessageBoxIcon::Error,
System::Windows::Forms::MessageBoxDefaultButton::Button1);
}
```

Game.cpp

```
#include "Game.h"
#include "mainWindow.h"

using namespace SlaSolGame;

SlaSolGame::Game::Game(int iFieldSize, int iColors) :fieldSize(iFieldSize), colors(iColors),
steps(0), scores(0) {}

int Game::getSteps() {
   return steps;
}

void Game::setSteps(int count) {
   steps = count;
}

void Game::incSteps() {
   ++steps;
}

int Game::getClicks() {
   return clicks;
}
void  Game::setClicks(int count) {
   clicks = count;
}

void  Game::decClicks() {
   --clicks;
}

int Game::getScores() {
   return scores;
}

void Game::setScores(int value) {
   scores = value;
}

void Game::addScores(int value) {
   scores += value;
}

int Game::getFieldSize(){
   return fieldSize;
}

void Game::setFieldSize(int value){
   fieldSize = value;
}

int Game::getColors(){
   return colors;
}
```

```
void Game::setColors(int value){
   colors = value;
}

void Game::clearCounters() {
   steps = 0;
   clicks = 0;
}

int Game::checkerFinishRound(bool check) {
   if (check) {
      win();
      return 1;
   }
   if (clicks == 0) {
      loose();
      return -1;
   }
   return 0;
}
```

GameWindow.cpp

```
#include "GameWindow.h"
#include "Errors.h"
#include "mainWindow.h"
#include "Display.h"

using namespace SlaSolDisplay;

GameWindow::GameWindow() {
   initStepsLabel();
   initLevelLabel();
   initClicksLeftLabel();
   initScoreLabel();
   initMenuButton();
   initGameField();
}

void GameWindow::initStepsLabel() {
   try {
      stepsLabel = (gcnew System::Windows::Forms::Label());
      stepsLabel->AutoSize = true;
      stepsLabel->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 13,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
         static_cast<System::Byte>(204)));
      stepsLabel->Location = System::Drawing::Point(402, 0);
      stepsLabel->Name = L"stepsLabel";
      stepsLabel->Size = System::Drawing::Size(88, 21);
      stepsLabel->TabIndex = 1;
      stepsLabel->Text = L"Ходов: 0";
      stepsLabel->Visible = false;
      gameVector.push_back(stepsLabel);
      Display::addControl(stepsLabel);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void GameWindow::initLevelLabel() {
   try {
      levelLabel = (gcnew System::Windows::Forms::Label());
      levelLabel->AutoSize = true;
      levelLabel->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 13,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
         static_cast<System::Byte>(204)));
      levelLabel->Location = System::Drawing::Point(382, 21);
      levelLabel->Name = L"levelLabel";
```

```
      levelLabel->Size = System::Drawing::Size(108, 21);
      levelLabel->TabIndex = 5;
      levelLabel->Text = L"Уровень: 0";
      levelLabel->Visible = false;
      Display::addControl(levelLabel);
      //gameVector.push_back(levelLabel);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void GameWindow::initClicksLeftLabel() {
   try {
      clicksLeftLabel = (gcnew System::Windows::Forms::Label());
      clicksLeftLabel->AutoSize = true;
      clicksLeftLabel->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 13,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
         static_cast<System::Byte>(204)));
      clicksLeftLabel->Location = System::Drawing::Point(224, 0);
      clicksLeftLabel->Name = L"clicksLeftLabel";
      clicksLeftLabel->Size = System::Drawing::Size(96, 21);
      clicksLeftLabel->TabIndex = 7;
      clicksLeftLabel->Text = L"Кликов: 0";
      clicksLeftLabel->Visible = false;
      gameVector.push_back(clicksLeftLabel);
      Display::addControl(clicksLeftLabel);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void GameWindow::initScoreLabel() {
   try
   {
      scoreLabel = (gcnew System::Windows::Forms::Label());
      scoreLabel->AutoSize = true;
      scoreLabel->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 13,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
         static_cast<System::Byte>(204)));
      scoreLabel->Location = System::Drawing::Point(224, 21);
      scoreLabel->Name = L"scoreLabel";
      scoreLabel->Size = System::Drawing::Size(76, 21);
      scoreLabel->TabIndex = 19;
      scoreLabel->Text = L"Очки: 0";
      scoreLabel->Visible = false;
      gameVector.push_back(scoreLabel);
      Display::addControl(scoreLabel);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void GameWindow::setStepsLabel(int steps, bool visible) {
   stepsLabel->Visible = visible;
   writeLabel(stepsLabel, steps);
}

void GameWindow::setLevelLabel(int level, bool visible) {
```

```
   levelLabel->Visible = visible;
   writeLabel(levelLabel, level);
}

void GameWindow::setClicksLeftLabel(int clicks, bool visible) {
   clicksLeftLabel->Visible = visible;
   writeLabel(clicksLeftLabel, clicks);
}

void GameWindow::setScoreLabel(int scores, bool visible) {
   scoreLabel->Visible = visible;
   writeLabel(scoreLabel, scores);
}

void GameWindow::initMenuButton() {
   try {
      menuButton = (gcnew System::Windows::Forms::Button());
      menuButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      menuButton->Cursor = System::Windows::Forms::Cursors::Hand;
      menuButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      menuButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      menuButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      menuButton->ForeColor = System::Drawing::SystemColors::ControlText;
      menuButton->Location = System::Drawing::Point(-1, 0);
      menuButton->Name = L"menuButton";
      menuButton->Size = System::Drawing::Size(138, 40);
      menuButton->TabIndex = 0;
      menuButton->Text = L"Меню";
      menuButton->UseVisualStyleBackColor = false;
      menuButton->Visible = false;
      gameVector.push_back(menuButton);
      Display::addControl(menuButton);
      menuButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::newGameButton_Click);

   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void GameWindow::initGameField() {
   gameField = gcnew System::Windows::Forms::TableLayoutPanel();
   gameVector.push_back(gameField);
   Display::addControl(gameField);
}

void GameWindow::updateGameField(int row, int col) {
   gameField->Controls->Clear();
   gameField->ColumnStyles->Clear();
   gameField->RowStyles->Clear();

   gameField->SuspendLayout();

   float size = (float)100 / col;
   gameField->Size = System::Drawing::Size(370, 370);
   gameField->ColumnCount = col;
   for (int i = 0; i < col; ++i)
      gameField->ColumnStyles->Add((gcnew
System::Windows::Forms::ColumnStyle(System::Windows::Forms::SizeType::Percent, size)));
   gameField->Location = System::Drawing::Point(70, 45);
   gameField->Name = L"gameFieldLayout";
   gameField->RowCount = row;
   for (int i = 0; i < row; ++i)
```

```
      gameField->RowStyles->Add((gcnew
System::Windows::Forms::RowStyle(System::Windows::Forms::SizeType::Percent, size)));
   gameField->TabIndex = 6;
   gameField->ResumeLayout(false);
   gameField->PerformLayout();
}

void GameWindow::showGameField() {
   Display::setVisibleVector(gameVector, true);
}

void GameWindow::hideGameField() {
   Display::setVisibleVector(gameVector, false);
}

void GameWindow::writeLabel(System::Windows::Forms::Label^ label, int value) {
   label->Text = label->Text->Split(L':')[0] + ": " + value;
}

int GameWindow::changeColorOnRowColumn(System::Windows::Forms::Control^ pictureBox) {
   System::Windows::Forms::TableLayoutPanelCellPosition position = gameField-
>GetPositionFromControl(pictureBox);
   int scores = 0;
   if (position.Column < 0 || position.Row < 0)
      throw WrongParams(position.Column, position.Row);
   for (int i = 0; i < gameField->ColumnCount; ++i) {
      for (int j = 0; j < gameField->RowCount; ++j) {
         if (gameField->GetControlFromPosition(position.Column, j)->BackColor != pictureBox-
>BackColor)
            scores += 250;
         if (gameField->GetControlFromPosition(j, position.Row)->BackColor != pictureBox-
>BackColor)
            scores += 250;
         gameField->GetControlFromPosition(position.Column, j)->BackColor = pictureBox->BackColor;
         gameField->GetControlFromPosition(j, position.Row)->BackColor = pictureBox->BackColor;
      }
   }
   if (position.Column > 2 && position.Row > 2 && position.Column < gameField->ColumnCount - 3 &&
position.Row < gameField->RowCount - 3) {
      int fromWall = gameField->ColumnCount * 0.2;
      if (position.Column > fromWall && position.Row > fromWall && position.Column < gameField-
>ColumnCount - fromWall
         && position.Row < gameField->RowCount - fromWall) {
         for (int i = -1; i < 2; i += 2)
            for (int j = -1; j < 2; j += 2) {
               if (gameField->GetControlFromPosition(position.Column + j, position.Row + i)-
>BackColor != pictureBox->BackColor)
                  scores += 250;
               gameField->GetControlFromPosition(position.Column + j, position.Row + i)->BackColor
= pictureBox->BackColor;
            }
      }
   }
   if (scores > 4000) return scores * 6;
   if (scores > 2000) return scores * 4;
   if (scores > 500) return scores * 2;
   return scores;
}

void GameWindow::fillGameField(int countOfColors) {
   unsigned int colors[10] = { 0xFF60004D, 0xFF159300, 0xFF0000A3, 0xFF840000, 0xFFE26FA7,
0xFFD3A900,
      0xFFA6BC56,  0xFF00605D,  0xFFA8A8A8, 0xFF0B161E };
   cliext::vector<System::Windows::Forms::Panel^> panelVector;
   int counter = 0;
   bool tableLayoutVisability = gameField->Visible;
   gameField->Visible = false;
   gameField->SuspendLayout();
   for (int i = 0; i < gameField->ColumnCount; ++i) {
      for (int j = 0; j < gameField->RowCount; ++j, ++counter) {
```

```
        System::Windows::Forms::Panel^ panel = gcnew System::Windows::Forms::Panel();
        panel->BackColor = System::Drawing::Color::FromArgb(colors[rand() % countOfColors]);
        panel->Location = System::Drawing::Point(3, 3);
        panel->Name = L"Block" + counter;
        panel->Size = System::Drawing::Size(gameField->GetColumnWidths()[0], gameField-
>GetRowHeights()[0]);
        panel->TabIndex = counter;
        gameField->Controls->Add(panel, i, j);
        panelVector.push_back(panel);
        panel->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::block_Click);
      }
   }
   gameField->ResumeLayout(false);
   gameField->PerformLayout();
   gameField->Visible = tableLayoutVisability;
}

bool GameWindow::isOneColor() {
   bool check = true;
   System::Drawing::Color blockColor = gameField->GetControlFromPosition(0, 0)->BackColor;
   for (int i = 0; i < gameField->ColumnCount && check; ++i) {
      for (int j = 0; j < gameField->RowCount && check; ++j) {
         if (blockColor != gameField->GetControlFromPosition(i, j)->BackColor)
            check = false;
      }
   }
   return check;
}
```

LevelMenu.cpp

```
#include "LevelMenu.h"
#include "mainWindow.h"
#include "Display.h"

using namespace SlaSolDisplay;

LevelSetMenu::LevelSetMenu() {
   initStartSettedGameButton();
   initMainMenuButton();
   initHelpLabelColors();
   initHelpLabelFieldSize();
   initDomainColorsCount();
   initDomainFieldSize();
}

void LevelSetMenu::initStartSettedGameButton() {
   try {
      startSettedGameButton = (gcnew System::Windows::Forms::Button());
      startSettedGameButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      startSettedGameButton->Cursor = System::Windows::Forms::Cursors::Hand;
      startSettedGameButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      startSettedGameButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      startSettedGameButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      startSettedGameButton->ForeColor = System::Drawing::SystemColors::ControlText;
      startSettedGameButton->Location = System::Drawing::Point(198, 198);
      startSettedGameButton->Name = L"startSettedGameButton";
      startSettedGameButton->Size = System::Drawing::Size(138, 40);
      startSettedGameButton->TabIndex = 18;
      startSettedGameButton->Text = L"Начать";
      startSettedGameButton->UseVisualStyleBackColor = false;
      startSettedGameButton->Visible = false;
      levelSetMenuVector.push_back(startSettedGameButton);
      Display::addControl(startSettedGameButton);
      startSettedGameButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::startSettedGameButton_Click);
   }
   catch (System::Exception^ e) {
```

```
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }

}

void LevelSetMenu::initMainMenuButton() {
   try {
      mainMenuButton = (gcnew System::Windows::Forms::Button());
      mainMenuButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      mainMenuButton->Cursor = System::Windows::Forms::Cursors::Hand;
      mainMenuButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      mainMenuButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      mainMenuButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 13,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      mainMenuButton->ForeColor = System::Drawing::SystemColors::ControlText;
      mainMenuButton->Location = System::Drawing::Point(198, 244);
      mainMenuButton->Name = L"mainMenuButton";
      mainMenuButton->Size = System::Drawing::Size(138, 40);
      mainMenuButton->TabIndex = 20;
      mainMenuButton->Text = L"Главное меню";
      mainMenuButton->UseVisualStyleBackColor = false;
      mainMenuButton->Visible = false;
      levelSetMenuVector.push_back(mainMenuButton);
      Display::addControl(mainMenuButton);
      mainMenuButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::mainMenuButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void LevelSetMenu::initHelpLabelColors() {
   try {
      helpLabelColors = (gcnew System::Windows::Forms::Label());
      helpLabelColors->AutoSize = true;
      helpLabelColors->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 13,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
         static_cast<System::Byte>(204)));
      helpLabelColors->Location = System::Drawing::Point(171, 82);
      helpLabelColors->Name = L"helpLabelColors";
      helpLabelColors->Size = System::Drawing::Size(192, 21);
      helpLabelColors->TabIndex = 13;
      helpLabelColors->Text = L"Количество цветов:";
      helpLabelColors->Visible = false;
      levelSetMenuVector.push_back(helpLabelColors);
      Display::addControl(helpLabelColors);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void LevelSetMenu::initHelpLabelFieldSize() {
   try {
      helpLabelFieldSize = (gcnew System::Windows::Forms::Label());
      helpLabelFieldSize->AutoSize = true;
      helpLabelFieldSize->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 13,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
```

```
         static_cast<System::Byte>(204)));
      helpLabelFieldSize->Location = System::Drawing::Point(237, 143);
      helpLabelFieldSize->Name = L"helpLabelFieldSize";
      helpLabelFieldSize->Size = System::Drawing::Size(63, 21);
      helpLabelFieldSize->TabIndex = 15;
      helpLabelFieldSize->Text = L"Поле:";
      helpLabelFieldSize->Visible = false;
      levelSetMenuVector.push_back(helpLabelFieldSize);
      Display::addControl(helpLabelFieldSize);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void LevelSetMenu::initDomainColorsCount() {
   try {
      domainColorsCount = (gcnew System::Windows::Forms::DomainUpDown());
      for (int i = 2; i < 11; ++i)
         domainColorsCount->Items->Add(L"" + i);
      domainColorsCount->Location = System::Drawing::Point(207, 120);
      domainColorsCount->Name = L"domainColorsCount";
      domainColorsCount->ReadOnly = true;
      domainColorsCount->SelectedIndex = 0;
      domainColorsCount->Size = System::Drawing::Size(120, 20);
      domainColorsCount->TabIndex = 16;
      domainColorsCount->Text = L"2";
      domainColorsCount->Visible = false;
      levelSetMenuVector.push_back(domainColorsCount);
      Display::addControl(domainColorsCount);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void LevelSetMenu::initDomainFieldSize() {
   try {
      domainFieldSize = (gcnew System::Windows::Forms::DomainUpDown());
      for (int i = 2; i < 21; ++i)
         domainFieldSize->Items->Add(System::Convert::ToString(i + "x" + i));
      domainFieldSize->Location = System::Drawing::Point(207, 167);
      domainFieldSize->Name = L"domainFieldSize";
      domainFieldSize->ReadOnly = true;
      domainFieldSize->SelectedIndex = 0;
      domainFieldSize->Size = System::Drawing::Size(120, 20);
      domainFieldSize->TabIndex = 17;
      domainFieldSize->Visible = false;
      levelSetMenuVector.push_back(domainFieldSize);
      Display::addControl(domainFieldSize);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}
//*****************************************************************************
LevelMenu::LevelMenu() {
   initLightLevelButton();
   initMiddleLevelButton();
   initHardLevelButton();
   initSetGameButton();
```

```
}

void LevelMenu::initLightLevelButton() {
   try {
      lightLevelButton = (gcnew System::Windows::Forms::Button());
      lightLevelButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      lightLevelButton->Cursor = System::Windows::Forms::Cursors::Hand;
      lightLevelButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      lightLevelButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      lightLevelButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      lightLevelButton->ForeColor = System::Drawing::SystemColors::ControlText;
      lightLevelButton->Location = System::Drawing::Point(198, 60);
      lightLevelButton->Name = L"lightLevelButton";
      lightLevelButton->Size = System::Drawing::Size(138, 40);
      lightLevelButton->TabIndex = 8;
      lightLevelButton->Text = L"Лёгкий";
      lightLevelButton->UseVisualStyleBackColor = false;
      lightLevelButton->Visible = false;
      levelMenuVector.push_back(lightLevelButton);
      Display::addControl(lightLevelButton);
      lightLevelButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::lightLevelButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void LevelMenu::initMiddleLevelButton() {
   try {
      middleLevelButton = (gcnew System::Windows::Forms::Button());
      middleLevelButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      middleLevelButton->Cursor = System::Windows::Forms::Cursors::Hand;
      middleLevelButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      middleLevelButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      middleLevelButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      middleLevelButton->ForeColor = System::Drawing::SystemColors::ControlText;
      middleLevelButton->Location = System::Drawing::Point(198, 106);
      middleLevelButton->Name = L"middleLevelButton";
      middleLevelButton->Size = System::Drawing::Size(138, 40);
      middleLevelButton->TabIndex = 9;
      middleLevelButton->Text = L"Средний";
      middleLevelButton->UseVisualStyleBackColor = false;
      middleLevelButton->Visible = false;
      levelMenuVector.push_back(middleLevelButton);
      Display::addControl(middleLevelButton);
      middleLevelButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::middleLevelButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void LevelMenu::initHardLevelButton() {
   try {
      hardLevelButton = (gcnew System::Windows::Forms::Button());
      hardLevelButton->BackColor = System::Drawing::Color::MediumSeaGreen;
```

```
      hardLevelButton->Cursor = System::Windows::Forms::Cursors::Hand;
      hardLevelButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      hardLevelButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      hardLevelButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      hardLevelButton->ForeColor = System::Drawing::SystemColors::ControlText;
      hardLevelButton->Location = System::Drawing::Point(198, 152);
      hardLevelButton->Name = L"hardLevelButton";
      hardLevelButton->Size = System::Drawing::Size(138, 40);
      hardLevelButton->TabIndex = 10;
      hardLevelButton->Text = L"Сложный";
      hardLevelButton->UseVisualStyleBackColor = false;
      hardLevelButton->Visible = false;
      levelMenuVector.push_back(hardLevelButton);
      Display::addControl(hardLevelButton);
      hardLevelButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::hardLevelButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void LevelMenu::initSetGameButton() {
   try {
      setGameButton = (gcnew System::Windows::Forms::Button());
      setGameButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      setGameButton->Cursor = System::Windows::Forms::Cursors::Hand;
      setGameButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      setGameButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      setGameButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      setGameButton->ForeColor = System::Drawing::SystemColors::ControlText;
      setGameButton->Location = System::Drawing::Point(198, 198);
      setGameButton->Name = L"setGameButton";
      setGameButton->Size = System::Drawing::Size(138, 40);
      setGameButton->TabIndex = 11;
      setGameButton->Text = L"Задать";
      setGameButton->UseVisualStyleBackColor = false;
      setGameButton->Visible = false;
      levelMenuVector.push_back(setGameButton);
      levelMenuVector.push_back(mainMenuButton);
      Display::addControl(setGameButton);
      Display::addControl(mainMenuButton);
      setGameButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::setGameButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

int SlaSolDisplay::LevelMenu::getFieldSizeFromDomain() {
   return (domainFieldSize->SelectedIndex + 2);
}

int SlaSolDisplay::LevelMenu::getCountOfColorsFromDomain() {
   return (domainColorsCount->SelectedIndex + 2);
}
```

```
void LevelMenu::show() {
   Display::setVisibleVector(levelMenuVector, true);
}

void LevelMenu::hide() {
   Display::setVisibleVector(levelMenuVector, false);
}

void LevelMenu::showLevelSetMenu() {
   Display::setVisibleVector(levelSetMenuVector, true);
}
void LevelMenu::hideLevelSetMenu() {
   Display::setVisibleVector(levelSetMenuVector, false);
}
```

MainMenu.cpp

```
#include "MainMenu.h"
#include "mainWindow.h"
#include "Display.h"

using namespace SlaSolDisplay;

MainMenu::MainMenu() {
   initCampaignButton();
   initChooseGameButton();
   initRandomGameButton();
   initQuitButton();
}

void MainMenu::initCampaignButton() {
   try {
      campaignButton = (gcnew System::Windows::Forms::Button());
      campaignButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      campaignButton->Cursor = System::Windows::Forms::Cursors::Hand;
      campaignButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      campaignButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      campaignButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      campaignButton->ForeColor = System::Drawing::SystemColors::ControlText;
      campaignButton->Location = System::Drawing::Point(198, 60);
      campaignButton->Name = L"campaignButton";
      campaignButton->Size = System::Drawing::Size(138, 40);
      campaignButton->TabIndex = 2;
      campaignButton->Text = L"Кампания";
      campaignButton->UseVisualStyleBackColor = false;
      mainMenuVector.push_back(campaignButton);
      Display::addControl(campaignButton);
      campaignButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::campaignButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void MainMenu::initChooseGameButton() {
   try {
      settedGameButton = (gcnew System::Windows::Forms::Button());
      settedGameButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      settedGameButton->Cursor = System::Windows::Forms::Cursors::Hand;
      settedGameButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      settedGameButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      settedGameButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
```

```
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      settedGameButton->ForeColor = System::Drawing::SystemColors::ControlText;
      settedGameButton->Location = System::Drawing::Point(198, 106);
      settedGameButton->Name = L"chooseGameButton";
      settedGameButton->Size = System::Drawing::Size(138, 40);
      settedGameButton->TabIndex = 4;
      settedGameButton->Text = L"На выбор";
      settedGameButton->UseVisualStyleBackColor = false;
      mainMenuVector.push_back(settedGameButton);
      Display::addControl(settedGameButton);
      settedGameButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::settedGameButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void MainMenu::initRandomGameButton() {
   try {
      randomGameButton = (gcnew System::Windows::Forms::Button());
      randomGameButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      randomGameButton->Cursor = System::Windows::Forms::Cursors::Hand;
      randomGameButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      randomGameButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      randomGameButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      randomGameButton->ForeColor = System::Drawing::SystemColors::ControlText;
      randomGameButton->Location = System::Drawing::Point(198, 152);
      randomGameButton->Name = L"randomGameButton";
      randomGameButton->Size = System::Drawing::Size(138, 40);
      randomGameButton->TabIndex = 6;
      randomGameButton->Text = L"Случайная";
      randomGameButton->UseVisualStyleBackColor = false;
      mainMenuVector.push_back(randomGameButton);
      Display::addControl(randomGameButton);
      randomGameButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::randomGameButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void MainMenu::initQuitButton() {
   try {
      quitButton = (gcnew System::Windows::Forms::Button());
      quitButton->BackColor = System::Drawing::Color::MediumSeaGreen;
      quitButton->Cursor = System::Windows::Forms::Cursors::Hand;
      quitButton->FlatAppearance->BorderColor = System::Drawing::Color::Turquoise;
      quitButton->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
      quitButton->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 15,
         static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
         System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
      quitButton->ForeColor = System::Drawing::SystemColors::ControlText;
      quitButton->Location = System::Drawing::Point(198, 290);
      quitButton->Name = L"quitButton";
      quitButton->Size = System::Drawing::Size(138, 40);
      quitButton->TabIndex = 3;
      quitButton->Text = L"Выход";
      quitButton->UseVisualStyleBackColor = false;
```

```
      mainMenuVector.push_back(quitButton);
      Display::addControl(quitButton);
      quitButton->Click += gcnew System::EventHandler(Display::mainWindow,
&SlaSol::mainWindow::quitButton_Click);
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void MainMenu::show() {
   Display::setVisibleVector(mainMenuVector, true);
}

void MainMenu::hide() {
   Display::setVisibleVector(mainMenuVector, false);
}
```

mainWindow.cpp

```
#include "mainWindow.h"
#include <time.h>

using namespace System;
using namespace System::Windows::Forms;

int main(array<String^>^ args) {
   srand(time(nullptr));
   Application::EnableVisualStyles();
   Application::SetCompatibleTextRenderingDefault(true);
   SlaSol::mainWindow mw;
   Application::Run(%mw);
   return 0;
}
```

RandomGame.cpp

```
#include "RandomGame.h"
#include "mainWindow.h"

using namespace SlaSolGame;

void RandomGame::start() {
   clearCounters();
   try {
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), true);
      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), true);
      SlaSol::mainWindow::display->setScoreLabel(getScores(), true);
      setFieldSize(rand() % 14 + 2);
      setColors(rand() % 10 + 1);
      SlaSol::mainWindow::display->updateGameField(getFieldSize(), getFieldSize());
      setClicks(getFieldSize() - 1);

      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), false);
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), false);

      SlaSol::mainWindow::display->showGameField();
      SlaSol::mainWindow::display->hideMainMenu();

      SlaSol::mainWindow::display->fillGameField(getColors());
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
};
```

SettedGame.cpp

```
#include "SettedGame.h"
#include "mainWindow.h"

using namespace SlaSolGame;

SettedGame::SettedGame(int iBeginNumber, int iModForRandColor, int iModForRandFieldSize) :
   Game(rand() % iModForRandFieldSize + iBeginNumber, rand() % iModForRandColor + iBeginNumber)
{}

SettedGame::SettedGame(int iFieldSize, int iColors) : Game(iFieldSize, iColors) {}


void SettedGame::start() {
   clearCounters();
   try {
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), true);
      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), true);
      SlaSol::mainWindow::display->setScoreLabel(getScores(), true);

      SlaSol::mainWindow::display->updateGameField(getFieldSize(), getFieldSize());
      SlaSol::mainWindow::display->fillGameField(getColors());
      setClicks(getFieldSize() - 1);

      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), false);
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), false);

      SlaSol::mainWindow::display->showGameField();
      SlaSol::mainWindow::display->hideMainMenu();
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void SettedGame::end(System::Object^ sender) {
   try {
      incSteps();
      SlaSol::mainWindow::display->setStepsLabel(getSteps(), true);
      decClicks();
      SlaSol::mainWindow::display->setClicksLeftLabel(getClicks(), true);
      addScores(SlaSol::mainWindow::display-
>changeColorOnRowColumn((cli::safe_cast<System::Windows::Forms::Control^>(sender))));
      SlaSol::mainWindow::display->setScoreLabel(getScores(), true);

      if (checkerFinishRound(SlaSol::mainWindow::display->isOneColor()) != 0) {
         SlaSol::mainWindow::display->hideGameField();
         SlaSol::mainWindow::display->showMainMenu();
         delete this;
      }
   }
   catch (WrongParams& e) {
      e.getMessage();
   }
   catch (System::Exception^ e) {
      System::Windows::Forms::MessageBox::Show(e->Message);
   }
   catch (...) {
      errorMessageWithGui();
   }
}

void SettedGame::win() {
   System::String^ message = L"Поздравляю, вы выиграли!";
   System::String^ caption = L"Игра завершена";
   System::Windows::Forms::MessageBoxButtons buttons =
System::Windows::Forms::MessageBoxButtons::OK;
```

```
   System::Windows::Forms::MessageBox::Show(message, caption, buttons,
System::Windows::Forms::MessageBoxIcon::Exclamation,
System::Windows::Forms::MessageBoxDefaultButton::Button1);
}

void SettedGame::loose() {
   System::String^ message = L"Вы проиграли! Возможно повезёт в следующий раз.";
   System::String^ caption = L"Игра завершена";
   System::Windows::Forms::MessageBoxButtons buttons =
System::Windows::Forms::MessageBoxButtons::OK;
   System::Windows::Forms::MessageBox::Show(message, caption, buttons,
System::Windows::Forms::MessageBoxIcon::Exclamation,
System::Windows::Forms::MessageBoxDefaultButton::Button1);
};
```

Заключение

В ходе учебной практики была разработана игра, развивающая логику. При её написании были использованы основные элементы ООП, такие как: инкапсуляция, наследование, полиморфизм. Также были использованы основные конструкции ООП языков: конструкторы, деструкторы, абстрактные классы, обработка исключительных ситуаций.

Список используемых источников

1. http://stackoverflow.com – Сайт вопросов и ответов по программированию.
2. http://cyberforum.ru – Форум программистов и сисадминов.
3. http://www.cplusplus.com/ - Информация о C++
4. https://msdn.microsoft.com/ - Сеть разработчиков Microsoft