|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Министерство науки и высшего образования Российской Федерации | | | | | | | | | | |
| Федеральное государственное бюджетное образовательное учреждение | | | | | | | | | | |
| высшего образования | | | | | | | | | | |
| **«Тверской государственный технический университет»** | | | | | | | | | | |
| **(ТвГТУ)** | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| Кафедра Программного обеспечения | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **Лабораторная работа № 2** | | | | | | | | | | |
| по дисциплине | | | | | | | | | | |
| **«Объектно-Ориентированное Программирование»** | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Выполнил: | | | | | | | **Лёвкин Д.А** | | | |
| Группа: | | | | | | | **Б.ПИН.ИИ.24.16** | | | |
|  | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Тверь, 2025 | | | | | | | | | | |

### **1. Трудности, с которыми столкнулись при разработке приложения**

#### 1.1. Проектирование архитектуры

* **Сложность выбора структуры данных** для хранения состояния игрового поля 128×128 клеток
* **Оптимизация использования памяти** при хранении истории ходов (каждое состояние содержит список всех фигур)
* **Проектирование системы отката ходов** без потери производительности

#### 1.2. Реализация игровой логики

* **Обработка столкновений фигур** по принципу "камень-ножницы-бумага":
  + Круг > Треугольник
  + Треугольник > Квадрат
  + Квадрат > Круг
* **Эффективный поиск линий** из 3+ одинаковых фигур на большом поле
* **Синхронизация состояния** между логикой игры и визуальным представлением

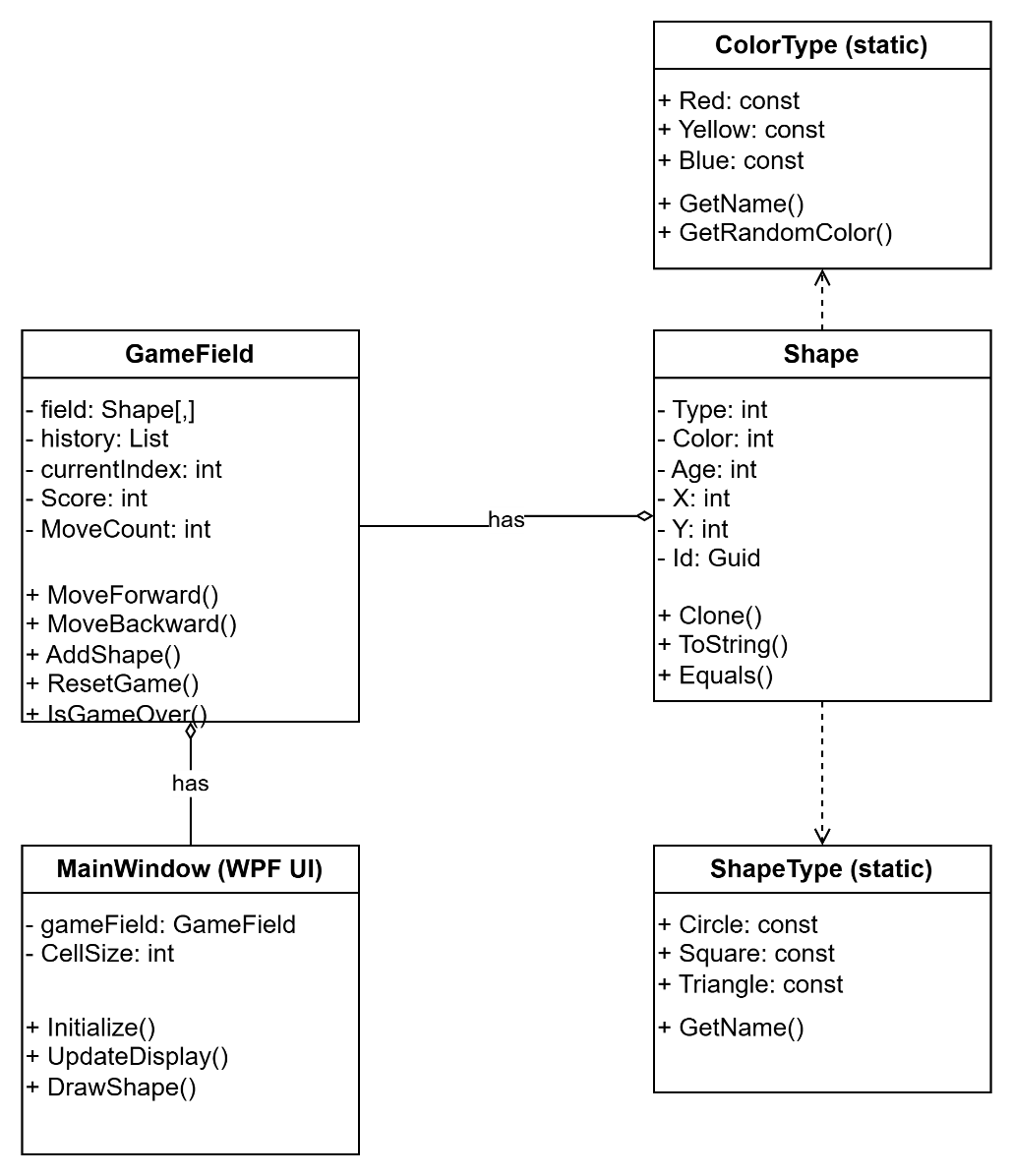
#### 1.3. Реализация пользовательского интерфейса

* **Визуализация большого поля** (128×128 клеток) с возможностью масштабирования
* **Отрисовка различных типов фигур** с сохранением производительности
* **Обработка всех возможных исключительных ситуаций**:
  + Некорректные координаты
  + Ошибки при движении фигур
  + Проблемы с восстановлением состояния

#### 1.4. Технические сложности

* **Реализация глубокого копирования** фигур для системы истории
* **Управление жизненным циклом объектов** при частом создании/удалении фигур
* **Интеграция между уровнями логики и интерфейса**

### **2. Диаграмма классов**



#### Описание классов:

**GameField** - основной класс игровой логики:

* Управляет состоянием игрового поля
* Обрабатывает ходы вперед/назад
* Проверяет условия завершения игры
* Управляет историей состояний

**Shape** - класс фигуры:

* Хранит тип, цвет, возраст и позицию
* Реализует клонирование для системы истории
* Содержит уникальный идентификатор

**ColorType** и **ShapeType** - статические классы-перечисления:

* Определяют константы цветов и типов фигур
* Содержат методы для получения названий

**MainWindow** - WPF окно:

* Визуализирует игровое поле
* Обрабатывает пользовательский ввод
* Отображает игровую статистику

### **3. Этапы тестирования и разработки**

#### 3.1. Этап 1: Проектирование и разработка ядра (3 дня)

**Задачи:**

* Проектирование классов предметной области
* Реализация базовой логики фигур и игрового поля
* Создание системы типов и цветов фигур

**Тестирование:**

* Модульное тестирование создания фигур
* Проверка корректности клонирования объектов
* Тестирование валидации позиций на поле

*// Пример теста создания фигуры*

var shape = new Shape(ShapeType.Circle, ColorType.Red, 10, 10);

Assert.AreEqual(ShapeType.Circle, shape.Type);

Assert.AreEqual(ColorType.Red, shape.Color);

Assert.AreEqual(10, shape.X);

Assert.AreEqual(10, shape.Y);

#### 3.2. Этап 2: Реализация игровой механики (4 дня)

**Задачи:**

* Реализация движения фигур
* Обработка столкновений по правилам игры
* Система возраста и изменения цвета
* Проверка и удаление линий

**Тестирование:**

* Тестирование правил столкновений:
  + Круг заменяет треугольник
  + Треугольник заменяет квадрат
  + Квадрат заменяет круг
* Проверка обнаружения горизонтальных и вертикальных линий
* Тестирование изменения цвета в зависимости от возраста

*// Пример теста столкновений*

var circle = new Shape(ShapeType.Circle, ColorType.Red, 0, 0);

var triangle = new Shape(ShapeType.Triangle, ColorType.Blue, 0, 0);

Assert.IsTrue(gameField.CanReplace(circle, triangle)); *// Круг > Треугольник*

#### 3.3. Этап 3: Система истории ходов (2 дня)

**Задачи:**

* Реализация хранения состояний игры
* Механизм отката ходов назад
* Восстановление состояния при движении вперед после отката

**Тестирование:**

* Проверка корректности сохранения/восстановления состояния
* Тестирование последовательности ходов вперед-назад-вперед
* Проверка сохранения возраста фигур при откате

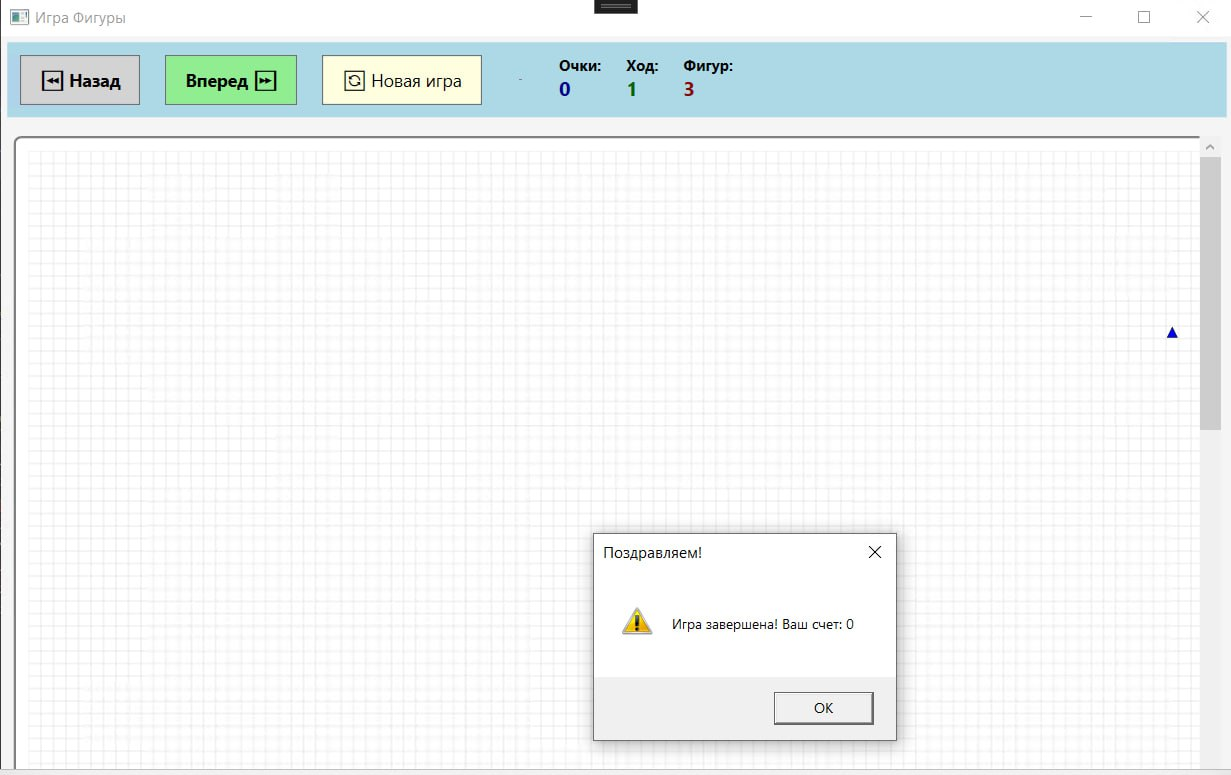
#### 3.4. Этап 4: Пользовательский интерфейс (3 дня)

**Задачи:**

* Создание WPF приложения
* Визуализация игрового поля и фигур
* Реализация элементов управления
* Обработка исключений и валидация

**Тестирование:**

* Тестирование отображения различных типов фигур
* Проверка реакции на пользовательский ввод
* Тестирование обработки исключительных ситуаций
* Проверка обновления статистики игры



**Внесенные исправления:**Были созданы перечисления с атрибутами Description для удобного отображения в UI:

**ShapeType.cs:**

public enum ShapeType

{

[Description("Круг")]

Circle = 0,

[Description("Квадрат")]

Square = 1,

[Description("Треугольник")]

Triangle = 2

}

Создан класс EnumHelper для работы с перечислениями:

public static class EnumHelper

{

public static string Description(this Enum value)

{

FieldInfo field = value.GetType().GetField(value.ToString());

DescriptionAttribute attribute = field.GetCustomAttribute<DescriptionAttribute>();

return attribute?.Description ?? value.ToString();

}

public static T GetRandomEnumValue<T>(Random random) where T : Enum

{

var values = Enum.GetValues(typeof(T));

return (T)values.GetValue(random.Next(values.Length));

}

}

#### 3.5. Этап 5: Интеграция и финальное тестирование (2 дня)

**Задачи:**

* Интеграция логики игры с интерфейсом
* Оптимизация производительности
* Финальное тестирование всего функционала

**Тестирование:**

* Полный игровой цикл (старт игры, несколько ходов, завершение)
* Тестирование граничных условий (пустое поле, полное поле)
* Проверка условий завершения игры (все фигуры одного цвета/типа)
* Тестирование на разных размерах окна и масштабах

### Результаты тестирования:

1. **Функциональность**: Все требования лабораторной работы выполнены
2. **Производительность**: Приложение работает плавно даже при большом количестве фигур
3. **Надежность**: Обработаны все основные исключительные ситуации
4. **Юзабилити**: Интерфейс интуитивно понятен и предоставляет всю необходимую информацию

### **Выводы:**

В ходе выполнения лабораторной работы были успешно реализованы все требуемые функции игры "Фигуры". Приложение демонстрирует хорошую архитектуру, разделяющую игровую логику и пользовательский интерфейс. Все механизмы игры (движение, столкновения, линии, история ходов) работают корректно и эффективно.

**Скриншоты работы приложения**

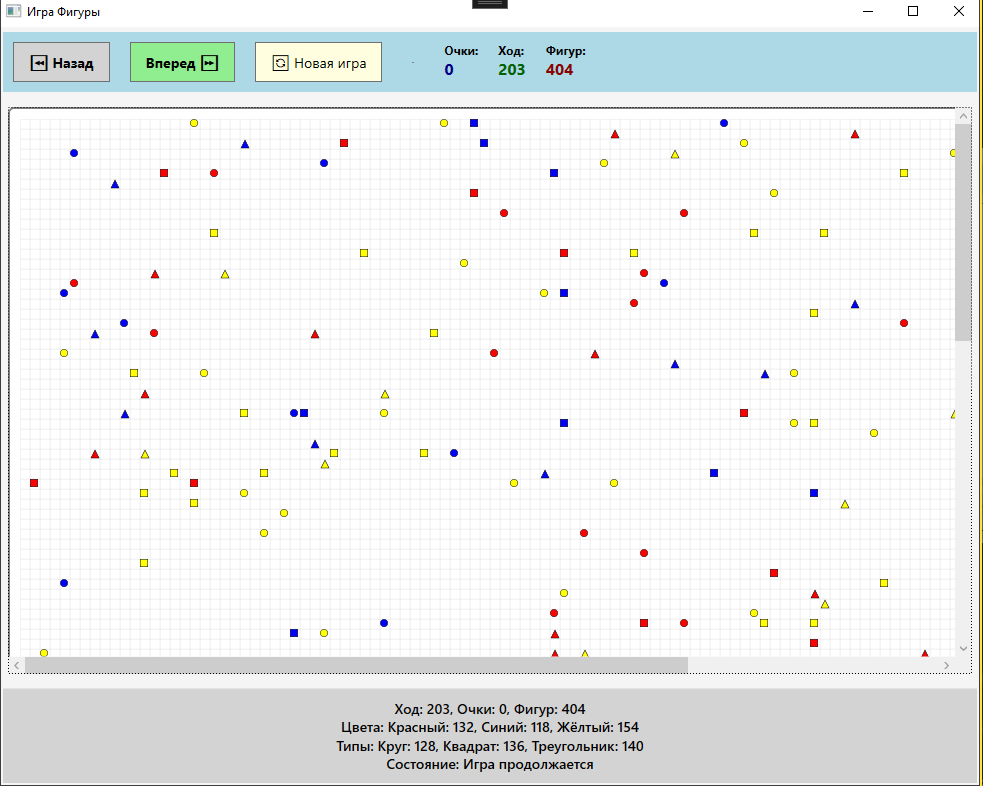
****

Рис. 1 – Главное окно