МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ ИМЕНИ И.Т. ТРУБИЛИНА»

Факультет прикладной информатики (заочного обучения)

Кафедра системного анализа и обработки информации

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине «Алгоритмизация и программирование»

на тему «Программирование игрового приложения „Усложнённые крестики-нолики“»

Направление подготовки 09.03.05 «Прикладная информатика»

Направленность «Менеджмент проектов в области информационных технологий, создание и поддержка информационных систем»

Выполнил:
Сидоров Дмитрий Ильич
группа ПИз1902
Руководитель:
к.т.н., доцент
_____ Орлянская Н.П. (Крамаренко Т.А. – 10 человек)

Дата защиты_____
Оценка _____
_____ Орлянская Н.П. (Крамаренко Т.А. – 10 человек)

Краснодар 2020

РЕФЕРАТ

18 с., 7 рис., 3 библ., N прил.

ПРОГРАММИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ «УСЛОЖНЁННЫЕ КРЕСТИКИ-НОЛИКИ»

Ключевые слова: паттерны проектирования, теория игр, минимакс, альфа-бета отсечение, кроссплатформенность, графика.

Цель работы: разработка графического приложения «Усложнённые крестики-нолики» на языке высокого уровня C++.

Объект исследования: алгоритм принятия решений.

Предмет исследования: средства языка программирования C++ для реализации 2D-графики и алгоритмизации принятия решений.

Полученные результаты: разработанное приложение позволяет осуществлять игру в «Крестики-нолики» между пользователем и компьютером с учётом возможности настраивания размера игрового поля и передачи права первого хода.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

К настоящему времени разработано большое количество различных реализаций игры «Крестики-нолики». Методы, используемые в данной курсовой работе, основаны на использовании паттернов проектирования: состояния, шаблонного метода, синглтона и других.

Для нахождения оптимального хода компьютера используется простейший из методов теории принятия решений – алгоритм минимакс. Для улучшения производительности алгоритма дополнительно применяется альфа-бета отсечение.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 ПОСТАНОВКА ЗАДАЧИ

Необходимо реализовать графическое приложение для игры в «Крестики-нолики».

Игра должна происходить между пользователем и компьютером и иметь настройки размерности поля и возможность передачи права первого хода компьютеру.

Приложение необходимо реализовать на языке высокого уровня C++ с поддержкой кроссплатформенности – возможностью сборки для персональных компьютеров с операционными системами на основе ядра Linux и семейства Windows.

1.2 СВЕДЕНИЯ ИЗ ТЕОРИИ

Процесс игры можно представить в виде дерева вариантов. В каждой конкретной позиции игрок имеет выбор между разными вариантами следующего хода, учитывая при этом дальнейшие варианты развития игровой партии.

Компьютер при выборе исходит из максимизации своей выгоды и минимизации выгоды игрока – такой алгоритм носит название «минимакс». Функция, которая оценивает игровое поле, называется статической функцией оценки (далее – СФО), чем «выгоднее» игровая ситуация, тем большее значение СФО [2, с. 54].

Опишем минимакс как рекурсивную функцию:

1. Если найдено конечное состояние, возвращается результат СФО.
2. Проходит по всем пустым ячейкам игрового поля и вызывает для каждой рекурсивно минимакс.
3. Получает для каждого вызова значения СФО, оценивает их и возвращает лучшее.

Для демонстрации работы рассмотрим пошагово эндшпиль одной из партий (Рисунок 1.1):

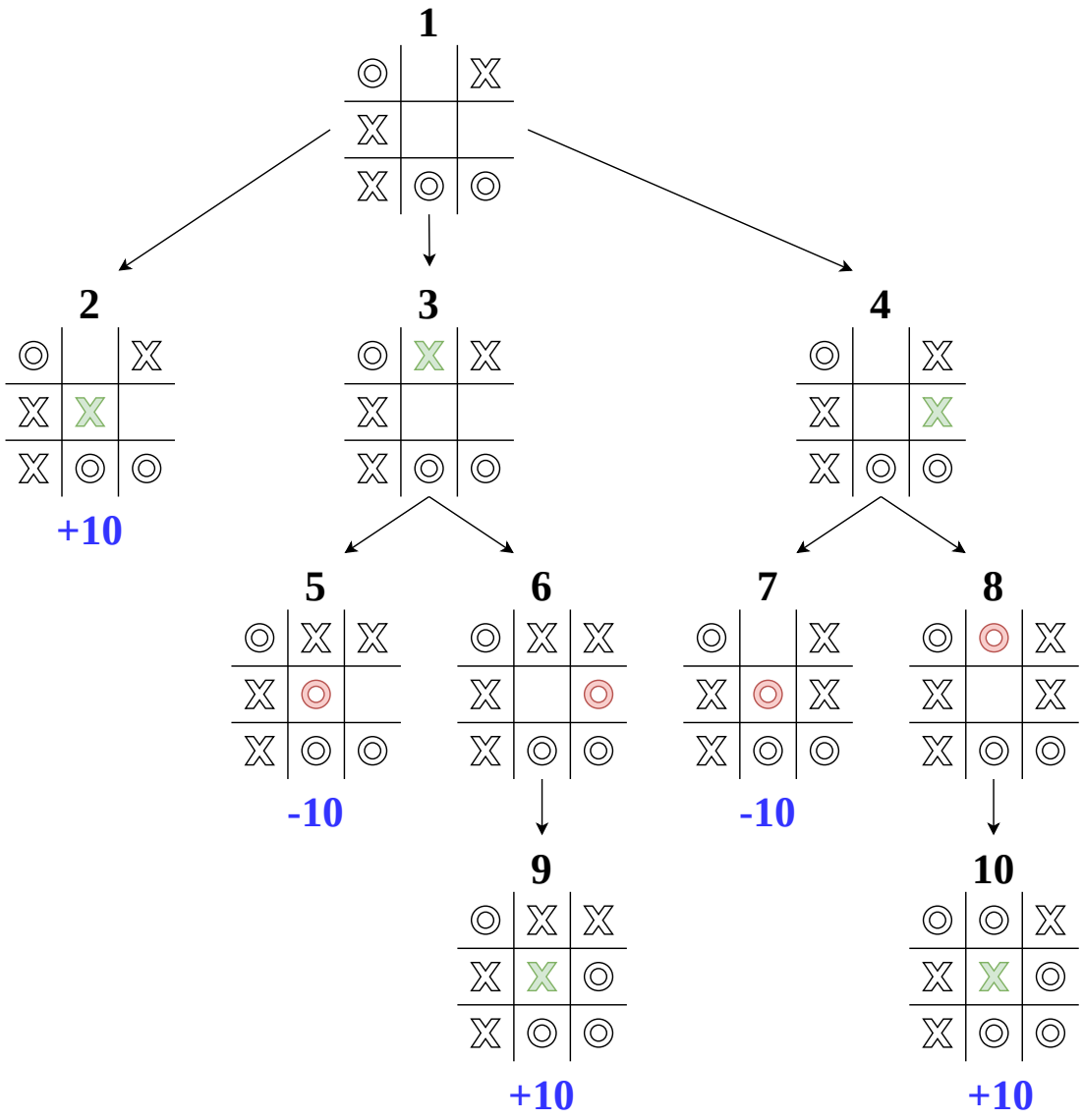

Рисунок 1.1 – Схема вариантов завершения игры

- Состояние 1 для X порождает состояния 2, 3 и 4 и вызывает минимакс на этих состояниях.
- Состояние 2 даёт оценку +10 к состоянию 1, потому что здесь игра заканчивается победой.

- Состояния 3 и 4 не являются конечным состояниями, так что состояние 3 порождает состояния 5 и 6 и вызывает минимакс на них, пока состояние 4 делает то же самое с состояниями 7 и 8.
- Состояние 5 даёт оценку -10 к состоянию 3, также состояние 7 даёт -10 очков к состоянию 4.
- Состояния 6 и 8 порождают по одному возможному конечному состоянию, которые оценку +10 к состояниям 3 и 4.
- Так как в порождённых состояниях состояниями 3 и 4 ходят нолики, нолики ищут минимальную оценку и делают выбор между -10 и +10, состояния 3 и 4 получают -10 каждое.
- Наконец состояния 2, 3 и 4 имеют оценки +10, -10, -10 соответственно, и состояние 1, максимизируя ход, выбирает состояние 2, дающий оценку +10. Партия завершена.

Количество рекурсивных вызовов минимакса экспоненциально растёт при изменении величины игрового поля. Чтобы компенсировать рост числа операций, применяется дополнительное альфа-бета отсечение. Вводятся коэффициенты альфа (минимальное значение СФО в ветке) и бета (максимальное значение СФО в ветке). На каждом уровне происходит сравнение текущей СФО с коэффициентами альфа и бета, что позволяет отсекать заведомо менее выгодные для компьютера или более выгодные для игрока [3, с. 541-543].

1.3 ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

В качестве основного инструмента разработки был выбран бесплатный текстовый редактор с открытым исходным кодом Atom, разработанный компанией GitHub, Inc.

Редактор является легковесным, поддерживает систему управления версиями Git и предоставляет расширяемость с помощью пакетов, устанавливаемых дополнительно.

В частности были использованы следующие из них:

1. atom-ide-ui – расширение интерфейса редактора для поддержки языковых сервисов и возможностей отладчика.

2. autocomplete-clang – автодополнение для C/C++/Objective-C с помощью API, предоставляемого Clang.

3. atom-ide-debugger-native-gdb – интеграция нативного отладчика GDB с интерфейсом редактора.

Для разработки был выбран стандарт C++17, так как имеет место использование синтаксиса распаковки пар, который отсутствует в более ранних версиях языка [1].

Учитывая игровую специфику приложения, для разработки была использована свободная кроссплатформенная мультимедийная библиотека SFML (Simple and Fast Multimedia Library), позволяющая легко взаимодействовать с 2D графикой.

Для автоматизации сборки используется CMake, непосредственно компилятором под Linux-системами выступает GCC, под Windows – MinGW.

2 ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

2.1 АЛГОРИТМ РЕШЕНИЯ

Для решения поставленной задачи реализуется разделение игры на непосредственно игровое поле и графический интерфейс пользователя.

На Рисунке 2.1 представлена общая схема взаимоотношений классов приложения.

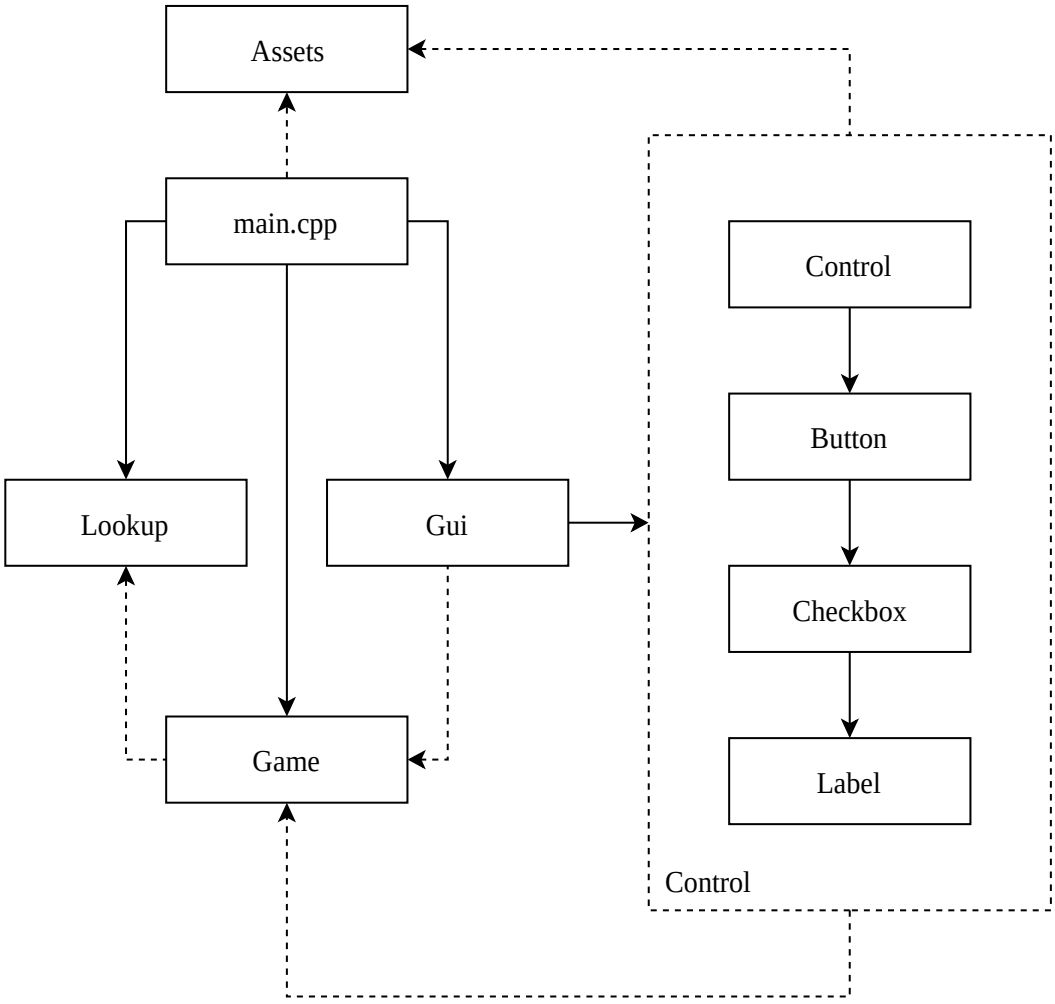

Рисунок 2.1 – Структура приложения

В функции int main() инициализируются объекты типов Lookup, Game, Gui.

Lookup реализует алгоритм нахождения наилучшего хода компьютера, взаимодействие с алгоритмом осуществляет через объект типа Game.

Game реализует игровое поле, взаимодействие с которым может осуществляться с помощью графического интерфейса пользователя, предоставляемого объектом типа Gui.

При вызове конструктора копирования Gui создаются элементы графического интерфейса пользователя, представляемые общим родительским классом Control.

Control имеет доступ к классу Assets, в котором хранятся ресурсы приложения, также имеет доступ к объекту Game в функциях клика по элементам.

В качестве алгоритма Lookup используется минимакс с альфа-бета отсечением. СФО на каждой глубине прохождения дерева вариантов возвращает три состояния: победа, поражение, ничья (или неоконченная игра).

2.2 ОПИСАНИЕ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ

Окно приложения состоит из двух частей: игрового поля слева и графического интерфейса пользователя справа (см. Рисунок 2.2).

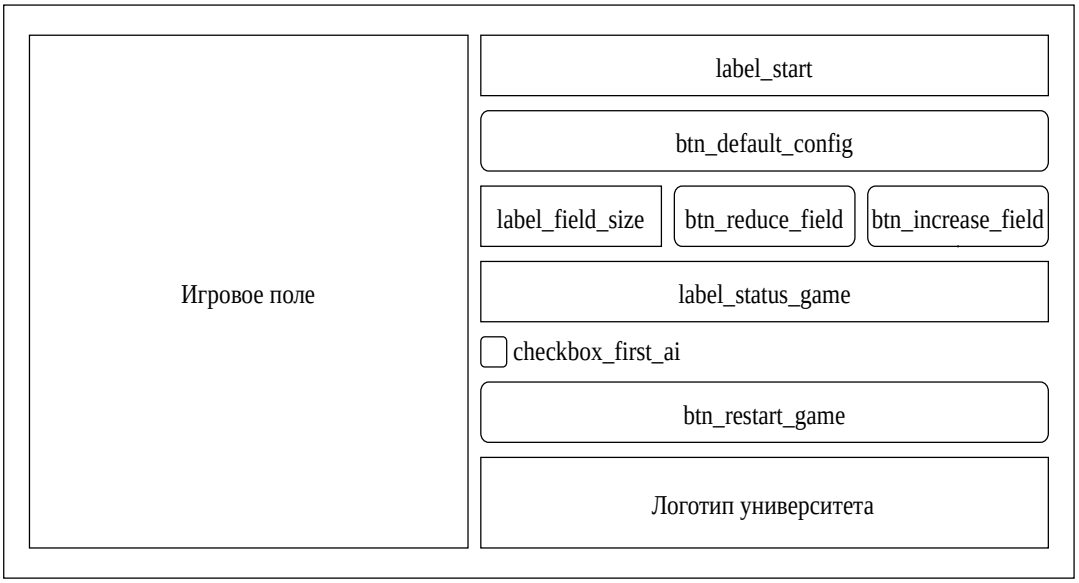

Рисунок 2.2 – Макет приложения

Игровое поле представляет собой матрицу размером NxN без внешнего обрамления, где каждая ячейка может иметь только одно из трёх состояний: пустое, заполненное крестиком, заполненное ноликом.

Крестики отображаются как пересекающиеся диагонали квадрата и имеют синий цвет, нолики отображаются как окружность и имеют красный цвет. Пустая ячейка не имеет специфических выделений.

В случае выстраивания N фигур одного типа в ряд или диагональ отображается выделение этого ряда или диагонали цветной линией, где цвет соответствует «зачёркиваемому» типу фигур.

Для реализации различных настроек и возможности перезапуска игры реализован графический интерфейс пользователя. Графический интерфейс пользователя состоит из элементов трёх типов: текст, кнопка и чекбокс. Информационно в этой области присутствует логотип КубГАУ.

Далее следует описание каждого из элементов:

1. label_start – приветствующее сообщение с названием игры.
2. btn_default_config – кнопка сброса размера игрового поля на стандартный 3x3.
3. label_field_size – информация о текущем размере игрового поля.
4. btn_reduce_field – кнопка уменьшения размерности игрового поля.
5. btn_increase_field – кнопка увеличения размерности игрового поля.
6. label_status_game – информация о текущем состоянии игры (игра не закончена, победа крестиков, победа ноликов, ничья).
7. checkbox_first_ai – чекбокс, передающий право первого хода от игрока компьютеру.
8. btn_restart_game – кнопка перезапуска игры с новыми настройками.

2.3 ОПИСАНИЕ ПРОГРАММЫ

В функции int main() осуществляется инициализация Assets с последующей загрузкой ресурсов (шрифт, логотип университета), происходит инициализация окна приложения, инициализация объектов игры (Game) и графического интерфейса пользователя (Gui). Здесь расположен цикл, обрабатывающий входные сигналы пользователя (перемещение курсора, нажатие кнопок, изменение размера окна) и передающий управление отрисовкой состояний в окне соответствующим элементам приложения.

Для хранения ресурсов игры используется класс Assets, являющийся по своей сути синглтоном Майерса, его использование обуславливается необходимостью обращений к статичным объектам-ресурсам из разных частей приложения, в результате которых могли бы порождаться множественные копии класса Assets.

Объект типа Game отвечает за отрисовку игрового поля, обработку нажатий на соответствующие ячейки внутри него. Через объект такого типа происходит взаимодействие с объектом типа Lookup, реализующий ходы компьютера.

Максимальная глубина вхождения в дерево вариантов в Lookup ограничена константой kMaxDepth (по-умолчанию равной 4). Ограничение установлено для получения результата работы алгоритма в адекватное время ввиду присутствия возможности изменения размерности игрового поля: поле размером 3x3 порождает 19683 (3^9) возможных состояний, 4x4 – уже 43046721 (3^{16}), 5x5 – 847288609443 (3^{25}) и так далее. Основная функция std::pair<int, std::pair<int, int>> Lookup::MinimaxOptimization(

std::vector<std::vector<StateCell>>& board, StateCell marker, int depth,

int alpha, int beta)

есть сам алгоритм минимакс. При первом вызове в board передаётся текущее состояние игрового поля, в marker – идентификатор (крестик StateCell:X или

нолик StateCell:O), под которым играет компьютер, глубина depth равна 0, alpha и beta равны kLoss и kWin соответственно.

Объект типа Gui при инициализации вызывает функцию void Gui::Initialize(), в которой происходит инициализация компонентов пользовательского интерфейса приложения (см. Рисунок 1), заполнение контейнера элементов std::map<std::string, std::shared_ptr<Control>> controls_, для дальнейшего доступа к ним. Здесь обрабатываются перемещения курсора по элементам интерфейса, нажатие на них.

Все элементы интерфейса являются наследниками класса Control, имеющий следующие поля: std::wstring title_text_ – название, bool is_hovered_ – состояние нахождения под указателем, bool is_pressed_ – состояния «нажатия», sf::Vector2f size_ – размер, sf::Vector2f position_ – положение внутри контейнера пользовательского интерфейса. Публичное поле void (*OnClick) (Control& me, Game& game, Gui& gui) предназначено для назначения обработчика нажатия на соответствующий элемент интерфейса.

Каждый элемент интерфейса обязательно имеет виртуальную функцию virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const, которая вызывается всякий раз при отрисовке интерфейса и задаёт внешний вид элемента: контуры, цвета, текст.

В Config.h константно задаются стандартные размеры окна, элементов (если их размер не изменён при инициализации в Gui.cpp), пути к ресурсным файлам, размер шрифта, ограничение глубины минимакса, значения СФО в случае победы, ничьи и проигрыша.

2.4 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

На Рисунке 2.3 представлено окно приложения сразу после запуска.

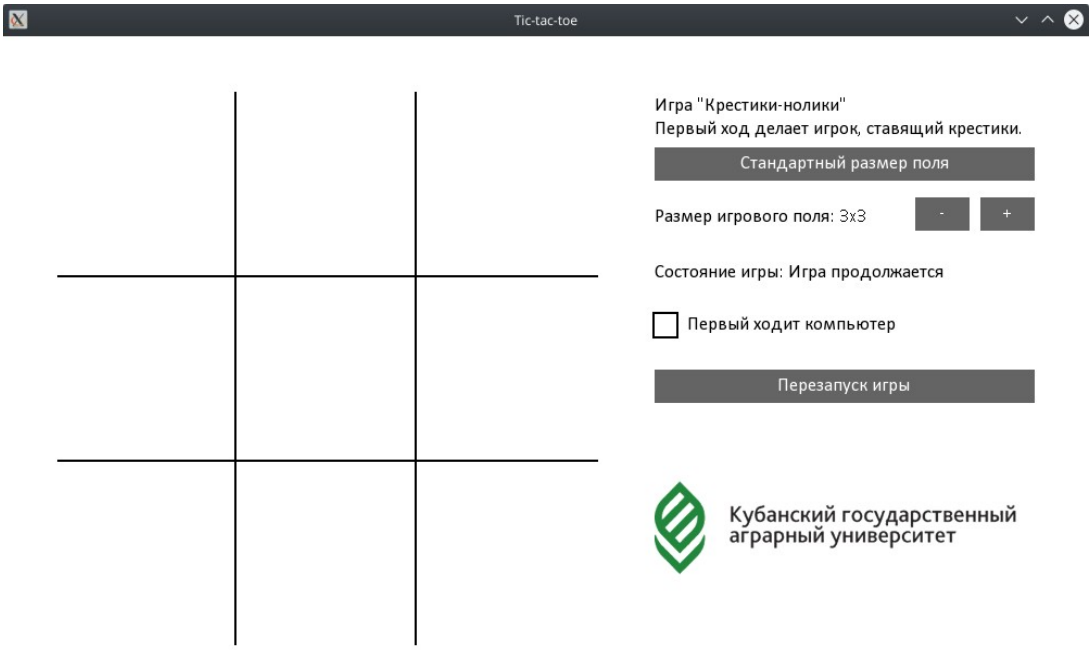

Рисунок 2.3 – Окно приложения после запуска

На Рисунке 2.4 представлена игровая партия с победой ноликов.

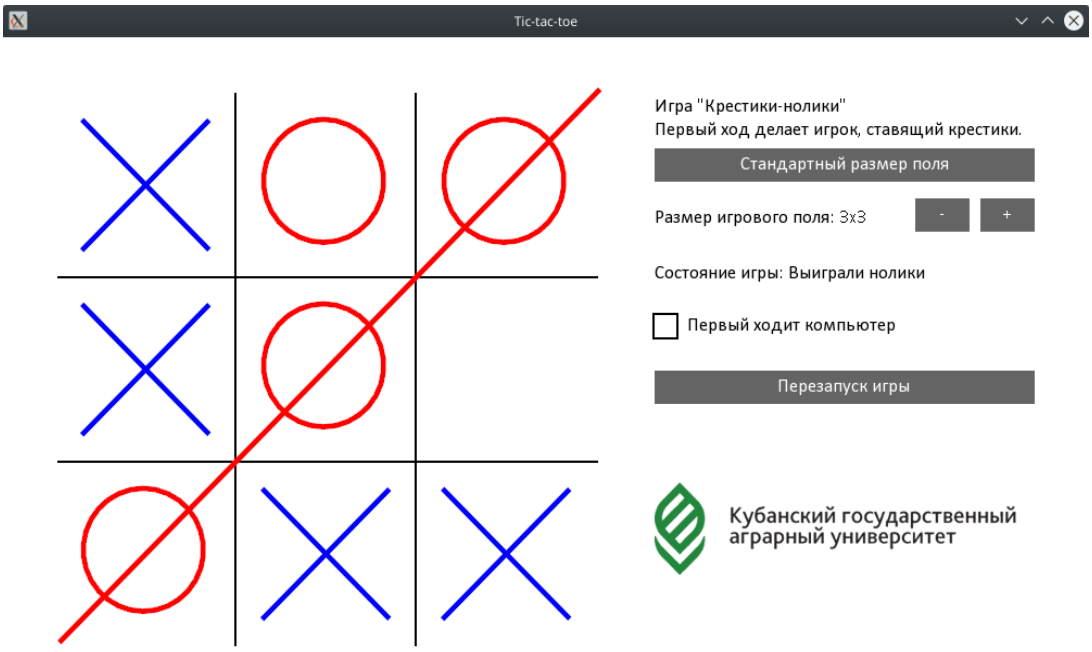

Рисунок 2.4 – Законченная игровая партия

На Рисунке 2.5 представлена законченная игровая партия на поле размером 5x5 с первым ходом компьютера.

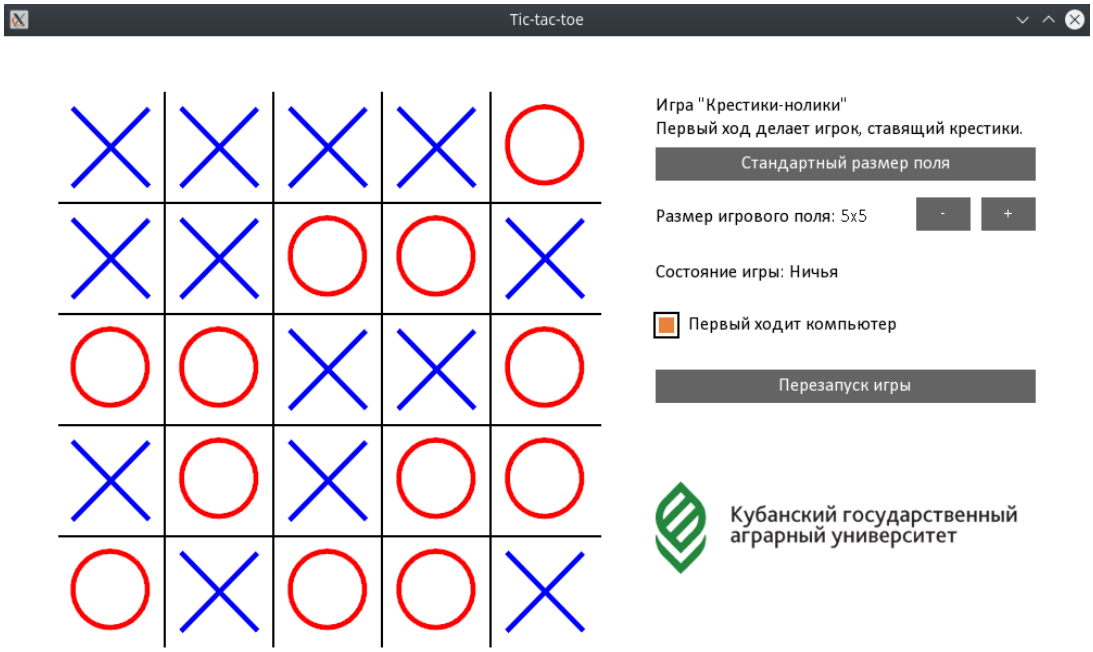

Рисунок 2.5 – Игра с первым ходом компьютера

На Рисунке 2.6 представлено запущенное приложение под операционной системой Windows 10.

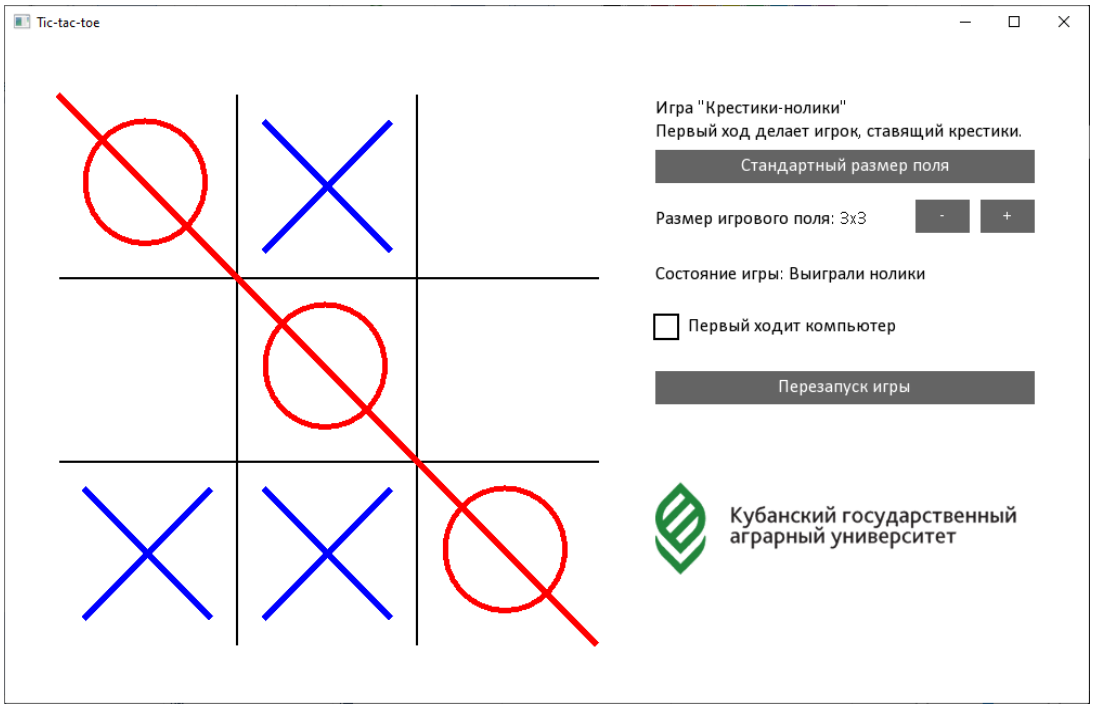

Рисунок 2.6 – Окно приложения в ОС Windows 10

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Приложение изначально разработано для операционных систем на базе ядра Linux, для работы требуется установленный пакет libsfml-dev, доступный во многих репозиториях.

Для операционных систем Windows приложение имеет динамически подключаем библиотеки в комплекте и не требует установки никакого дополнительного программного обеспечения.

Работа в приложении осуществляется с помощью компьютерной мышки.

С помощью кнопок «-» и «+» осуществляется изменение размеров игрового поля. Выставленные размеры указаны слева от данных кнопок. При больших размерах игрового поля может потребоваться ожидание очередного хода компьютера, которое, в зависимости от оборудования, может составлять десятки секунд.

Ниже находится информация о состоянии игры в текущий момент.

Чекбокс «Первый ходит компьютер» позволяет дать возможность совершить первый ход (и дальнейшие) крестиком, при этом игрок будет играть ноликом.

Кнопка «Перезапуск игры» сбрасывает игровое поле и состояние игры, применяет выставленные настройки к новой игровой партии.

В случае чьей-либо победы или заполнении игрового поля полностью дальнейшая простановка крестиков или ноликов на ней невозможна, партия считается завершённой и требуется начало новой партии с помощью кнопки перезапуска.

Выход из приложения осуществляется с помощью нажатия на стандартную иконку крестика вверху окна.

ЗАКЛЮЧЕНИЕ

Разработанное в данной курсовой работе приложение является готовым программным продуктом.

Приложение может быть использовано в исследовательских целях.

Стоит отметить, что реализованный алгоритм минимакс является самым простым, но неэффективным при большой глубине поиска. Используемое альфа-бета отсечение значительно уменьшает количество просматриваемых вариантов, но число операций всё же растёт экспоненциально.

В дальнейшем для улучшения производительности возможно дополнительное использование более сложных эвристических методов, как, например, итерационного углубления и таблицы перестановок.

СПИСОК ЛИТЕРАТУРЫ

1. Structured binding declaration [Электронный ресурс] / cppreference.com – Режим доступа: https://en.cppreference.com/w/cpp/language/structured_binding. (Дата обращения: 08.02.2020)

2. Гончаров Е.Н. Исследование операций [Электронный ресурс]: учебное пособие / Гончаров Е.Н., Ерзин А.И., Залюбовский В.В. – Режим доступа: http://math.nsc.ru/LBRT/k4/or/ – Новосибирск: НГУ, 2005. – 78 с. (Дата обращения: 30.01.2020)

3. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG [Текст] – 3-е изд. : Пер. с англ. – М.: Издательский дом "Вильямс", 2004. – 640 с.