МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

КОНТРОЛЬНАЯ РАБОТА

по дисциплине «Программирование на языке Python»

на тему: «Реализация игры «Крестики-Нолики» против игрока и компьютера»

Студент _____ Шорин В.Д.

Шифр 171406

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 71ПГ

Руководитель _____ Захарова О.В.

Оценка: «_____» Дата 20.12.19

Орел 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные игры требуют достаточно большой производительности от компьютера, и не каждая офисная машина в силах воспроизводить их. Однако для отдыха от монотонной работы зачастую достаточно простой, не требовательной к технике, игры. К таким играм относятся: «Крестики – Нолики», «Сапер», «Пасьянс» и т.д.

Целью контрольной работы является проектирование и реализация игры «Крестики – Нолики» с двумя режимами игры: против игрока (с возможностью выбора размера стороны квадрата игрового поля) и против компьютера.

Для выполнения поставленной цели необходимо:

- описать предметную область;
- выбрать программные средства;
- описать алгоритм, реализующий действия компьютера;
- реализовать приложение.

1 ПОСТАНОВКА ЗАДАЧИ И РАЗРАБОТКА ТРЕБОВАНИЙ

Задачей данной контрольной работы является предоставление пользователю возможности игры в «Крестики – Нолики» в двух режимах. Первый режим реализует игру игрока против игрока с возможностью выбора размера стороны квадрата игрового поля. Второй режим реализует игру игрока против компьютера.

Требования к приложению:

- выбор размера поля (3x3 и более) в режиме «Игрок против Игрока»;
- в режиме «Игрок против Компьютера» действия компьютера основаны на алгоритме «Минимакс»;
- в обоих режимах оповещения о выигрыше одной из сторон или о ничьей.

2 ОПИСАНИЕ РАБОТЫ АЛГОРИТМА «МИНИМАКС»

«Минимакс» — правило принятия решений, используемое в теории игр, теории принятия решений, исследовании операций, статистике и философии для минимизации возможных потерь из тех, которые лицу, принимающему решение, нельзя предотвратить при развитии событий по наихудшему для него сценарию. Как и профессиональный шахматист, алгоритм «Минимакс» просчитывает действия соперника на несколько ходов вперёд — до тех пор, пока не достигнет конца партии, будь то победа, поражение или ничья. Попав в это конечное состояние, алгоритм начислит себе положительное количество очков (в нашем случае +10) за победу, отрицательное (-10) — за поражение, и нейтральное (0) — за ничью. В то же время алгоритм проводит аналогичные расчёты для ходов игрока. Он будет выбирать ход с наиболее высоким баллом, если ходит алгоритм, и ход с наименьшим, если ходит игрок. Используя такую стратегию, минимакс избегает поражения [1].

Алгоритм «Минимакс» проще всего описать в виде рекурсивной функции, которая:

1. Возвращает значение, если найдено конечное состояние (+10, 0, -10).
2. Проходит по всем пустым клеткам на поле.
3. Вызывает минимакс-функцию для каждой из них (рекурсия).
4. Оценивает полученные значения.
5. Возвращает наилучшее из них.

Примерная схема работы данного алгоритма представлена на рисунке 1.

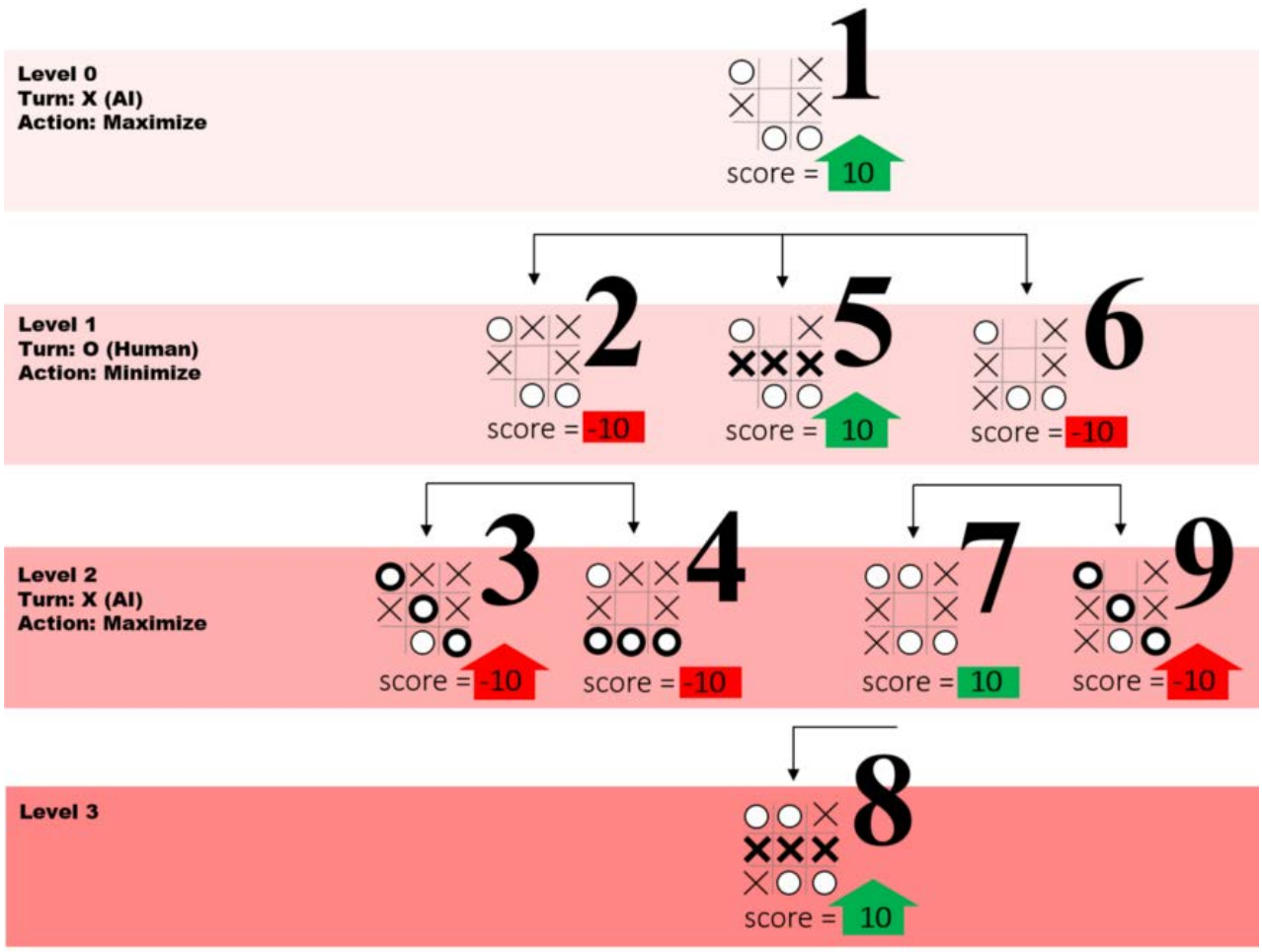

Рисунок 1 – Схема работы алгоритма "Минимакс"

3 ВЫБОР ПРОГРАММНЫХ СРЕДСТВ И РЕАЛИЗАЦИЯ

Для реализации поставленных задач будет использован язык Python.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты [2].

Для создания пользовательского графического интерфейса будет использован встроенный в Python пакет графики tkinter.

Tkinter (от англ. Tk interface) – кросс-платформенная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX-подобных систем, портирована также и на Microsoft Windows), написанная Стином Лумхольтом (Steen Lumholt) и Гвидо ван Россумом. Входит в стандартную библиотеку Python. Библиотека предназначена для организации диалогов в программе с помощью оконного графического интерфейса (GUI). В составе библиотеки присутствуют общие графические компоненты:

- Toplevel. Окно верхнего уровня (корневой виджет);
- Tk, Frame. Рамка. Содержит в себе другие визуальные компоненты;
- Label. Этикетка. Показывает некоторый текст или графическое изображение;
- Entry. Поле ввода текста;
- Button. Кнопка.

Эти и многие другие графические компоненты будут использованы при реализации графического пользовательского интерфейса [3].

В процессе реализации были разработаны классы «PlayerBoard» и «MinimaxBoard», отвечающие за логику работы программы, и «GUI_players» и «GUI_minimax», отвечающие за пользовательский графический интерфейс. В классе «MinimaxBoard» создана функция «minimax», которая реализует алгоритм «Минимакс» и, соответственно, логику поведения компьютера:

```
def __minimax(self, player):
    if self.won():
        if player:
            return (-10, None)
        else:
            return (+10, None)
    elif self.tied():
        return (0, None)
    elif player:
        best = (-2, None)
        for x, y in self.fields:
            if self.fields[x, y] == self.empty:
                value = self.move(x, y).__minimax(not player)[0]
                if value > best[0]:
                    best = (value, (x, y))
        return best
    else:
        best = (+2, None)
        for x, y in self.fields:
            if self.fields[x, y] == self.empty:
                value = self.move(x, y).__minimax(not player)[0]
                if value < best[0]:
                    best = (value, (x, y))
        return best
```

После каждого хода необходимо обновлять интерфейс и проверять поле на чью-либо победу или ничью. Для этого реализована функция «update» (проверка происходит вызовом функции «won» из соответствующих классов «PlayerBoard» и «MinimaxBoard»:

```
def update(self):
    for (x, y) in self.board.fields:
```

```
            text = self.board.fields[x, y]

            self.buttons[x, y]['text'] = text
            self.buttons[x, y]['disabledforeground'] = 'black'

            if text == self.board.empty:
                self.buttons[x, y]['state'] = 'normal'
            else:
                self.buttons[x, y]['state'] = 'disabled'

        winning = self.board.won()
        if winning:
            for x, y in winning:
                self.buttons[x, y]['disabledforeground'] = 'red'
            for x, y in self.buttons:
                self.buttons[x, y]['state'] = 'disabled'
            messagebox.showinfo("Winner", "Player 'O' wins")

        tie = self.board.tied()
        if tie:
            messagebox.showinfo("Tie", "Oops, it`s tie!")

        for (x, y) in self.board.fields:
            self.buttons[x, y].update()
```

4 ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

При запуске приложения, пользователю выводится окно выбора режима игры (рисунок 2).

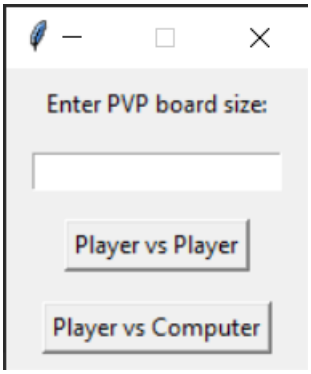

Рисунок 2 – Окно выбора режима игры

В данном окне пользователь выбирает режим, в который он хочет играть: против игрока или против компьютера. Если он выбирает режим игры против игрока, то ему необходимо ввести размер стороны поля (т.е. сколько клеток в ширину и высоту будет игровое поле). Это значение обязательно должно быть больше 2, иначе пользователю выведется сообщение с ошибкой (рисунок 3) и он не сможет начать игру против игрока.

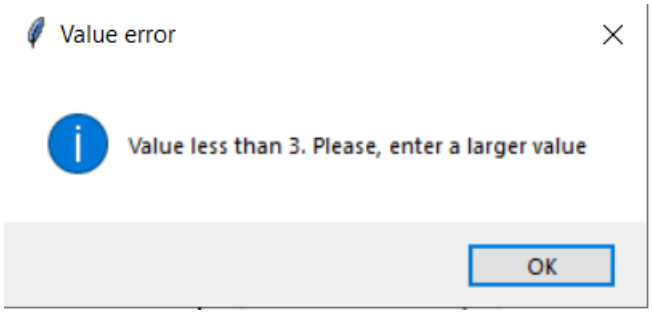

Рисунок 3 – Окно с ошибкой

После ввода корректного значения размера поля, пользователю выводится окно с непосредственно самим игровым полем (рисунок 4).

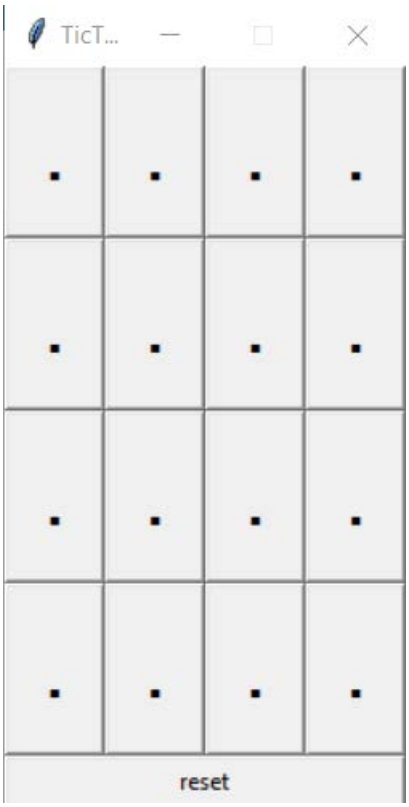

Рисунок 4 – Игровое поле режима «Игрок против игрока» размера 4x4

Далее происходит сама игра. У первого игрока по умолчанию стоит символ «X», у второго, соответственно, «O». Игра продолжается до победы одного из игроков (рисунки 5 - 6) или до ничьей (рисунок 7) с соответствующим оповещением.

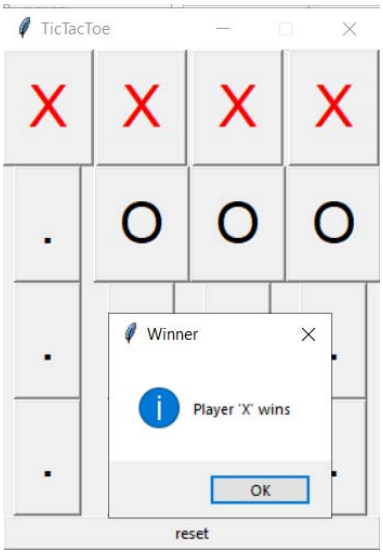

Рисунок 5 – Победа игрока «X»

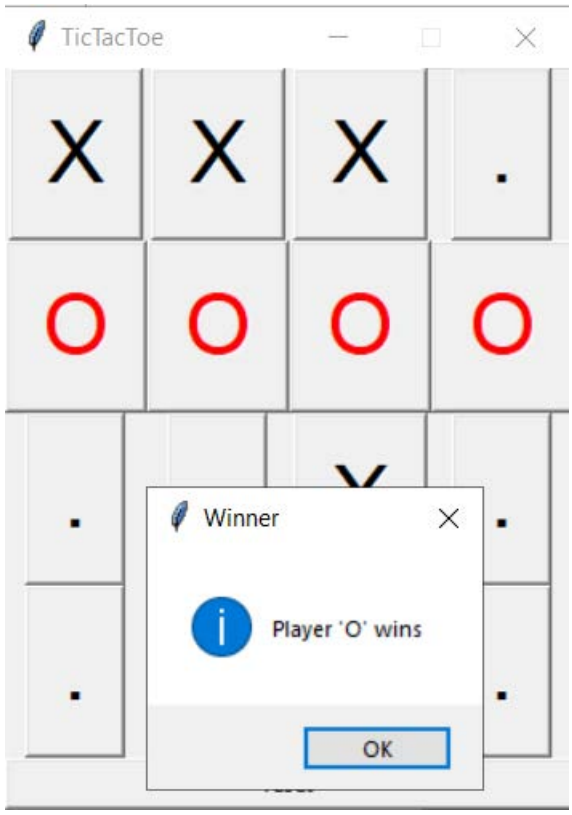

Рисунок 6 – Победа игрока «O»

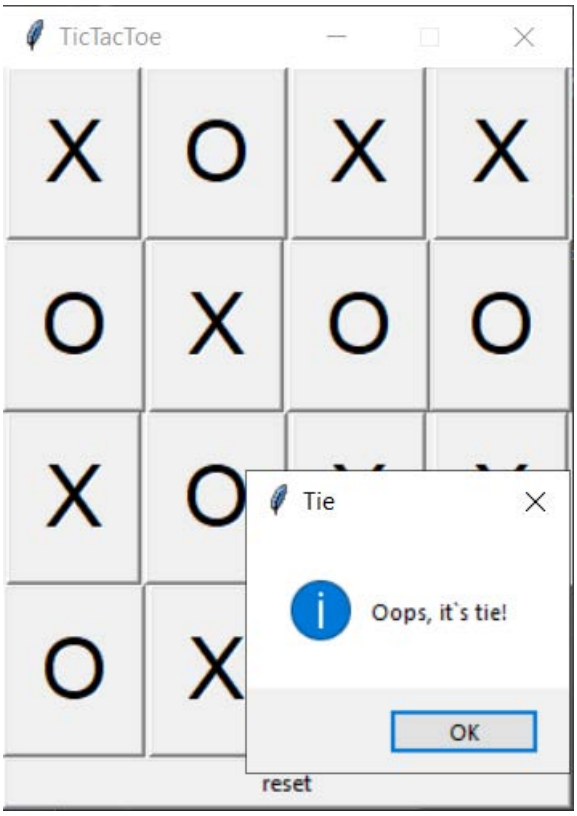

Рисунок 7 – Ничья

Также, в самом низу игрового поля присутствует кнопка «reset», которая отвечает за перезагрузку игрового процесса. Пользователь может нажать данную кнопку в любой момент.

При выборе пользователем режима игры против компьютера, пользователю высвечивается игровое поле для данного режима. Ввиду особенностей реализации алгоритма действий компьютера (алгоритм «Minimax»), игровое поле строго фиксировано и имеет размер 3x3 (рисунок 8).

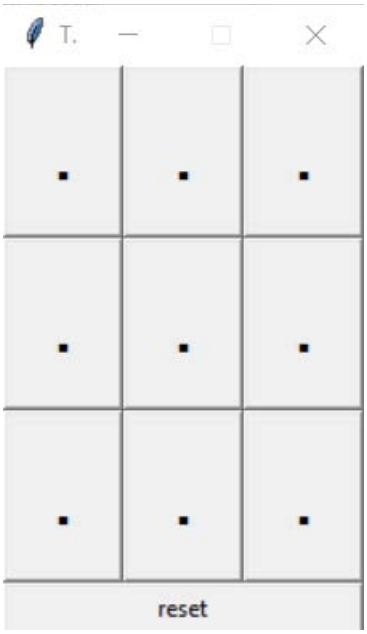

Рисунок 8 – Игровое поле режима игры против компьютера

В данном режиме игру начинает пользователь и ходит с помощью символа «X». Компьютер, соответственно, имеет символ «O». Игра продолжается либо до победы компьютера (рисунок 10), либо до ничьей (рисунок 9).

Игра не может закончиться победой пользователя по той причине, что это невозможно в выбранном алгоритме действий компьютера. В данном режиме также присутствует кнопка «reset», имеющая тот же функционал, что и в режиме игра против игрока.

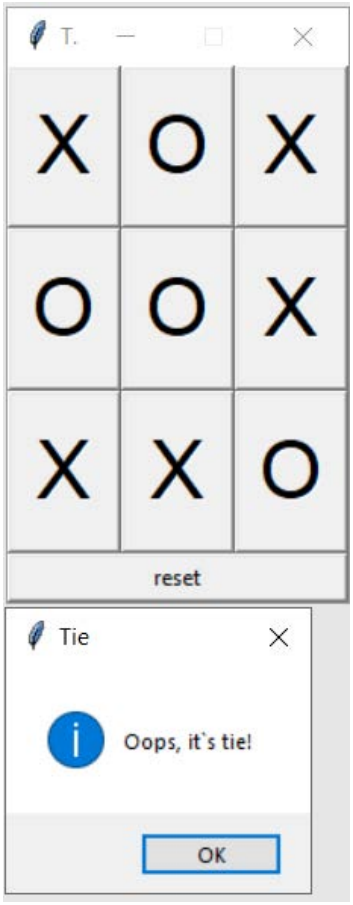

Рисунок 9 – Ничья

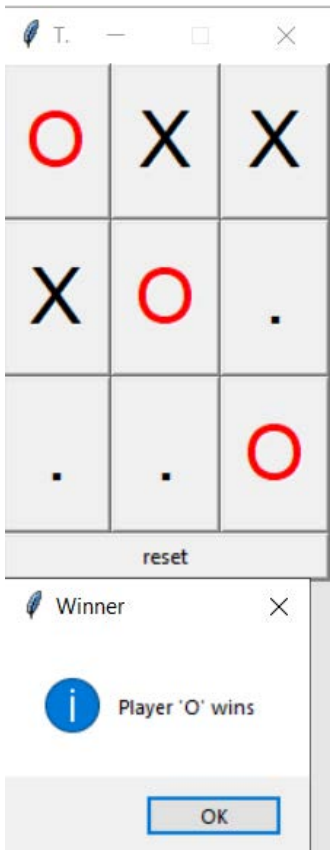

Рисунок 10 – Победа компьютера

ЗАКЛЮЧЕНИЕ

В ходе выполнения контрольной работы была разработана игра «Крестики – Нолики» с двумя режимами игры: против игрока и против компьютера. В режиме против игрока у пользователя есть возможность выбраться размер поля n, который должен быть больше 3. В режиме против компьютера пользователь играет против алгоритма «Minimax», результатом игры которого может быть либо победа компьютера, либо ничья.

Для достижения этой цели была описана предметная область, выбраны программные средства, описан алгоритм, реализующий действия компьютера, реализована логика работы режимов игры против игрока и компьютера.

Так как каждая поставленная задача выполнена, контрольную работу можно считать завершенной.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Демьянов В. Ф. Введение в минимакс [Текст] / В. Ф. Демьянов, В. Н. Малоземов. — М.: Наука, 1972. — 368 с.

2. Лутц Марк. Программирование на Python. [Текст] / Марк Лутц. — СПб.: Символ-Плюс, 2011. — 992 с.

3. Сузи Р. А. Создание приложений с графическим интерфейсом пользователя. Язык программирования Python: учебное пособие [Текст] / Р. А. Сузи. — М.: Интуит, Бином, 2006. — 328 с.