Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

Отделение №2 Информационные технологии и транспорт

**Отчёт**

по МДК.02.01. Технология разработки программного обеспечения

на тему: создание игры «Две лисы и двадцать кур»

Исполнители:Дувакин Андрей Андреевич,

Зырянова Елизавета Петровна, студенты группы ИСпП-21-2

Преподаватель: Коржина В. С.

**Техническое задание**

На поле указанной формы находятся две лисы и 20 кур. Куры могут перемещаться на один шаг вверх, влево или вправо, но не назад и не по диагонали. Лисы также могут перемещаться на один шаг (вверх, вниз, влево и вправо).

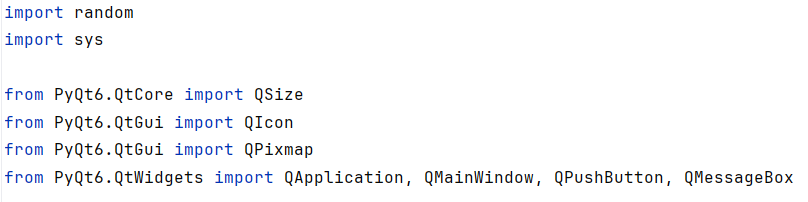
Лиса может съесть курицу: если в горизонтальном или вертикальном направлении за курицей на один шаг следует свободное поле, то лиса перепрыгивает через курицу и берет ее.

Лисы обязаны есть, и когда у них есть выбор. Составить программу, которая играет за лис. Игрок перемещает кур. Партнеры играю по очереди, причем куры начинают.

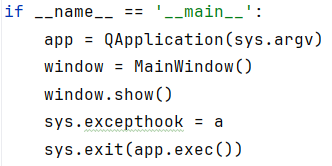
Они выигрывают партию, если девяти из них удается занять 9 полей, образующий верхний квадрат поля. Лисы выигрывают, если им удается съесть 12 кур, т.к. тогда оставшихся кур недостаточно, чтобы занять 9 верхних полей

**Описание работы программы**

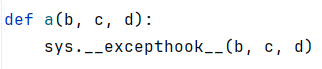
Изначально необходимо импортировать все библиотеки для разработки программы, в данном случае импортируем фреймвор PyQt6 и необходимые объекты из него, встроенный модуль random и sys (Листинг 3).



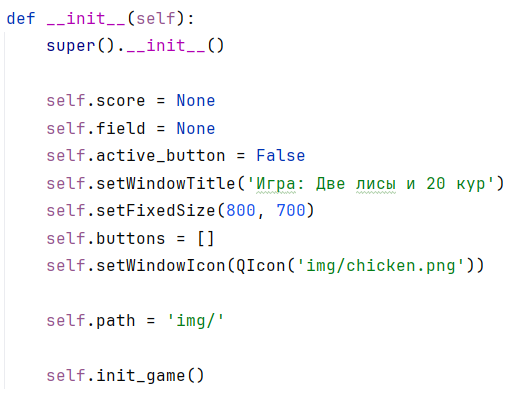
Далее создается объект приложения и объект класса окна, конструкция if необходима для проверки, что данный файл является запущенным, а не вызван для исполнения другим файлом (Листинг 1).



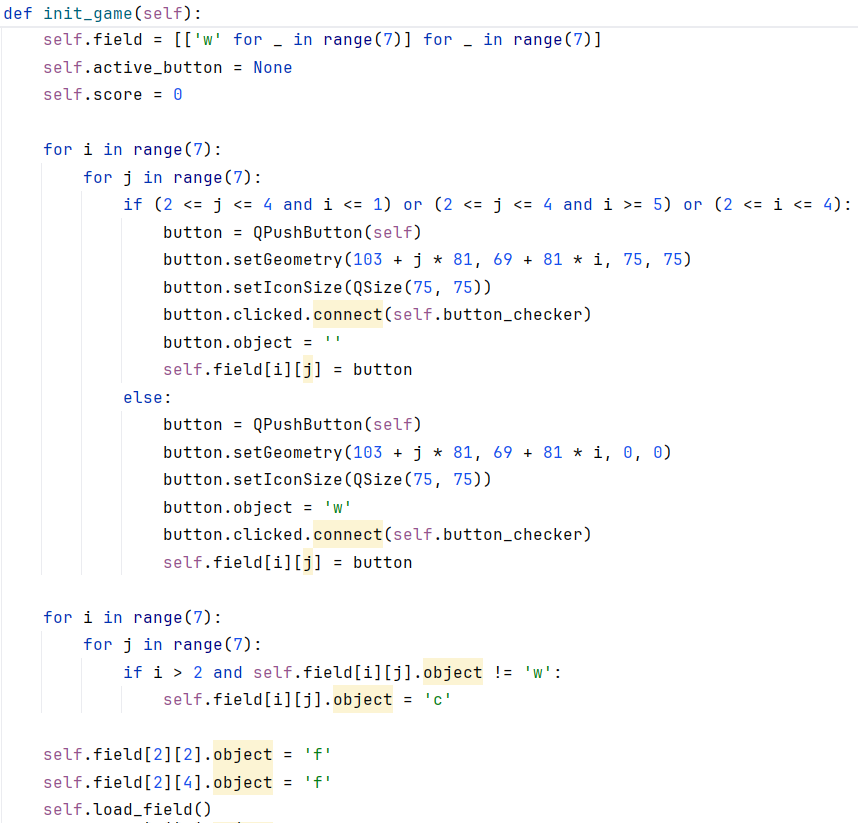
Функция a(b, c, d) необходима для отображения ошибок и исключений в ходе работы программы ь.к. PyQt по умолчанию «замалчивает» ошибки (Листинг 2).



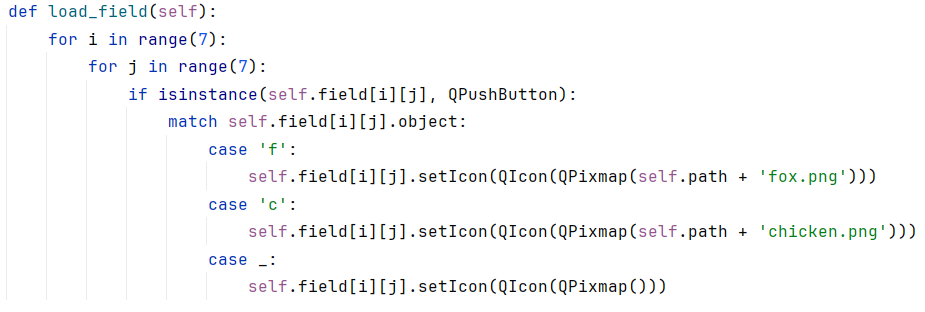
В классе MainWindow определена функция инициализации, которая инициализирует основные параметры окна приложения, такие как заголовок, размер, иконку и т.д. Затем вызывается метод init\_game(), который инициализирует игровое поле и другие элементы интерфейса (Листинг 4).



Происходит инициализация игрового поля. Сначала создается двумерный список self.field, представляющий собой игровое поле. Затем создаются кнопки для каждого элемента поля. Если индексы соответствуют позициям кур, они отображаются на интерфейсе, иначе кнопки создаются невидимыми. Далее происходит установка объектов (кур, лис и пустых клеток) и их начальное расположение. Метод load\_field() вызывается для отображения состояния игрового поля на интерфейсе (Листинг 5).

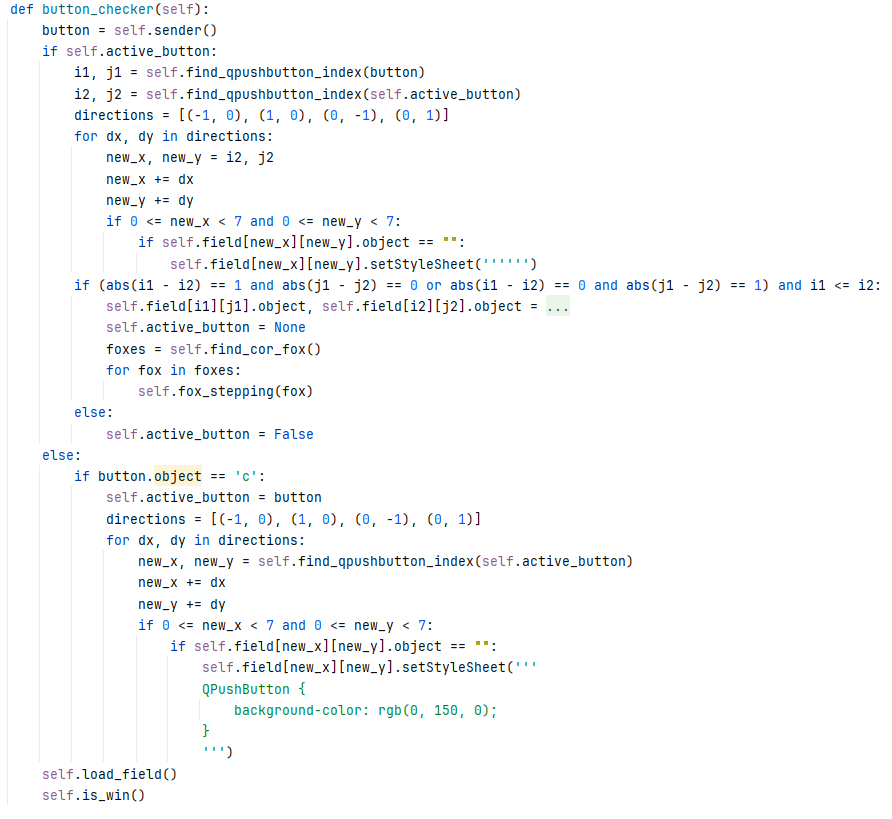


Далее и при каждом ходе игрока или лисы происходит перерисовка объектов на поле с учетом текущего состояния. Все элементы поля (курицы, лисы и пустые клетки) отображаются с помощью иконок на соответствующих кнопках (Листинг 8).

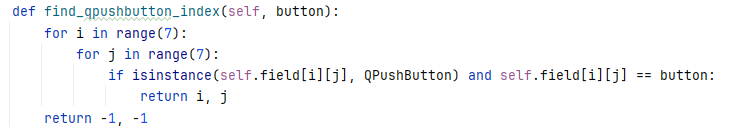


После первичной отрисовки ожидается ход игрока, обработкой действий игрока занимается специальная функция, которая вызывается при нажатии на любую кнопку на поле. Сначала мы получаем объект отправителя сигнала (нажатую кнопку), если в классе определена кнопка active\_button (т.е. сейчас нажата вторая по счету кнопка), то происходит расчет возможности перемещения курицы, в случае успешного перемещения объекты двух кнопок, находящиеся в переменной field меняются местами и происходит перерисовка поля, иначе активная кнопка пропадает.

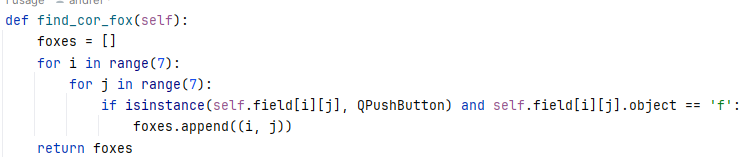
Если кнопка нажата впервые и в ней содержится курица, то подсвечиваются варианты кнопок, в которые игрок может сходить. После выполнения любого варианта происходит перерисовка пол, и проверка условия победы или проигрыша (Листинг 10).



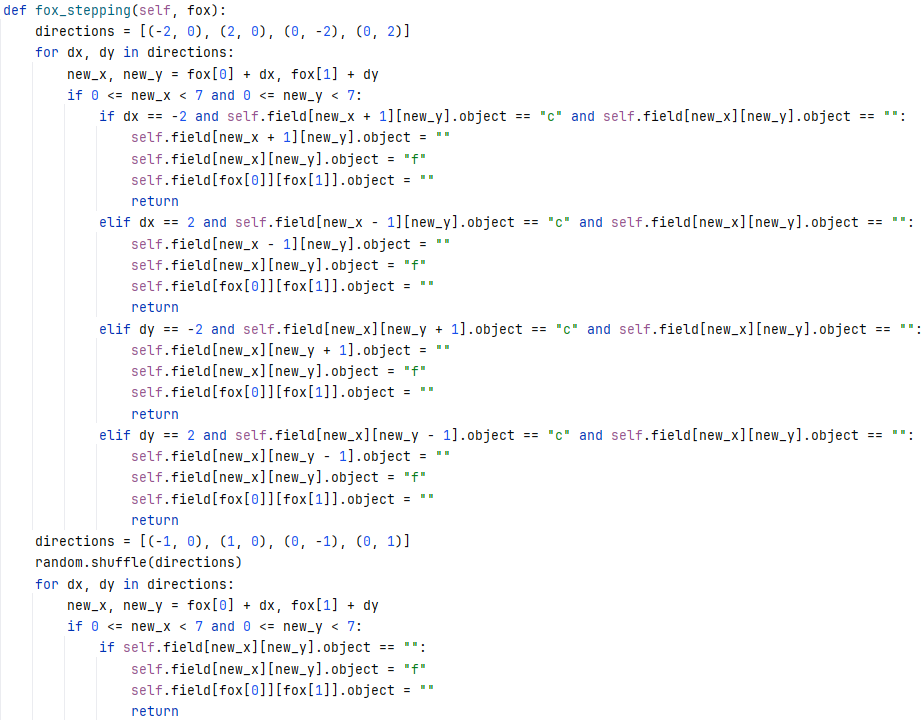
Во время проверки хода игрока вызывается функция поиска координат кнопки. Метод принимает кнопку в качестве аргумента и перебирает все элементы поля, чтобы найти соответствие (Листинг 6).



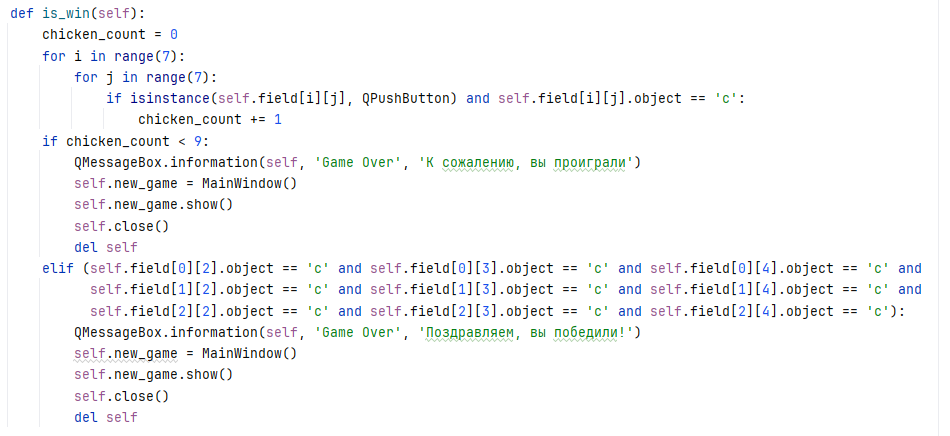
Также во время проверки происходит поиск координат позиции лисы, за это тоже отвечает специальная функция (Листинг 9).



После «хода» игрока происходит расчет хода лисы. Лиса перемещается в свободную клетку вокруг себя или пытается съесть курицу, если это возможно. Пробуя 4 направления и съедает ближайшую возможную курицу. В ином случае лиса пытается сходить случайным образом (Листинг 7).

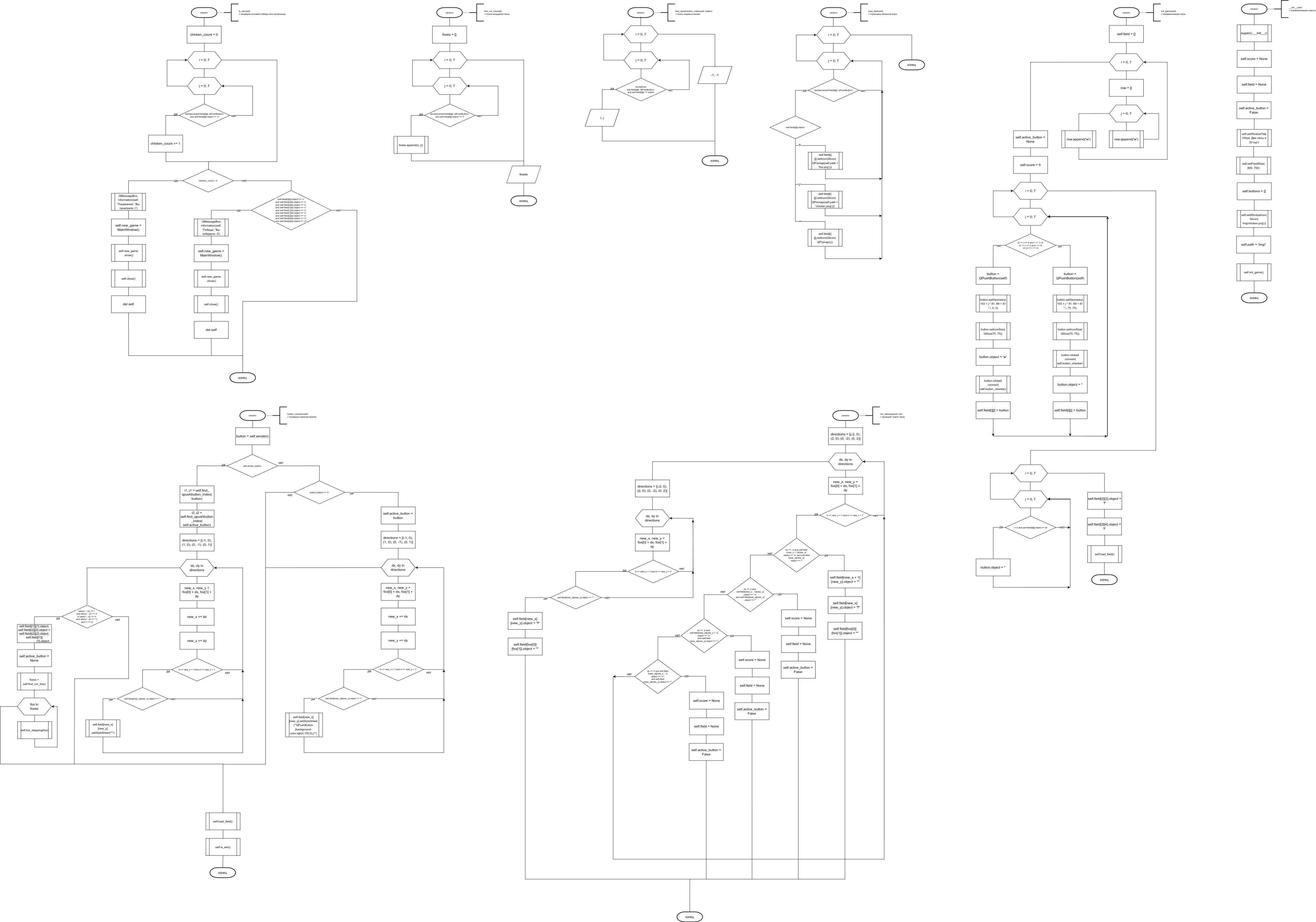


После каждого хода игрока или лисы происходит проверка условия победы или проигрыша. Считается количество куриц на всей карте и в случае выполнения условия (меньше 9 куриц на карте), выводится соответствующее сообщение и игра перезапускается (Листинг 11).



**Репозиторий**

**<https://github.com/AndreiDuvakin/SemestrTask.git>**

****

**Приложение А**

**Листинг 1 (Создание объекта приложения)**

app = QApplication(sys.argv)  
window = MainWindow()  
window.show()  
sys.excepthook = a  
sys.exit(app.exec())

**Листинг 2 (Вывод исключений)**

def a(b, c, d):  
 sys.\_\_excepthook\_\_(b, c, d)

**Листинг 3 (Импорт библиотек)**

import random  
import sys  
  
from PyQt6.QtCore import QSize  
from PyQt6.QtGui import QIcon  
from PyQt6.QtGui import QPixmap  
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton, QMessageBox

**Листинг 4 (Инициализация класса окна приложения)**

def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.score = None  
 self.field = None  
 self.active\_button = False  
 self.setWindowTitle('Игра: Две лисы и 20 кур')  
 self.setFixedSize(800, 700)  
 self.buttons = []  
 self.setWindowIcon(QIcon('img/chicken.png'))  
  
 self.path = 'img/'  
  
 self.init\_game()

**Листинг 5 (Инициализация объектов окна)**

def init\_game(self):  
 self.field = [['w' for \_ in range(7)] for \_ in range(7)]  
 self.active\_button = None  
 self.score = 0  
  
 for i in range(7):  
 for j in range(7):  
 if (2 <= j <= 4 and i <= 1) or (2 <= j <= 4 and i >= 5) or (2 <= i <= 4):  
 button = QPushButton(self)  
 button.setGeometry(103 + j \* 81, 69 + 81 \* i, 75, 75)  
 button.setIconSize(QSize(75, 75))  
 button.clicked.connect(self.cell\_checker)  
 button.object = ''  
 self.field[i][j] = button  
 else:  
 button = QPushButton(self)  
 button.setGeometry(103 + j \* 81, 69 + 81 \* i, 0, 0)  
 button.setIconSize(QSize(75, 75))  
 button.object = 'w'  
 button.clicked.connect(self.cell\_checker)  
 self.field[i][j] = button  
  
 for i in range(7):  
 for j in range(7):  
 if i > 2 and self.field[i][j].object != 'w':  
 self.field[i][j].object = 'c'  
  
 self.field[2][2].object = 'f'  
 self.field[2][4].object = 'f'  
 self.load\_field()

**Листинг 6 (Поиск индексов объекта QpushButton)**

def find\_qpushbutton\_index(self, button):  
 for i in range(7):  
 for j in range(7):  
 if isinstance(self.field[i][j], QPushButton) and self.field[i][j] == button:  
 return i, j  
 return -1, -1

**Листинг 7 (Расчет хода лисы)**

def fox\_stepping(self, fox):  
 directions = [(-2, 0), (2, 0), (0, -2), (0, 2)]  
 for dx, dy in directions:  
 new\_x, new\_y = fox[0] + dx, fox[1] + dy  
 if 0 <= new\_x < 7 and 0 <= new\_y < 7:  
 if dx == -2 and self.field[new\_x + 1][new\_y].object == "c" and self.field[new\_x][new\_y].object == "":  
 self.field[new\_x + 1][new\_y].object = ""  
 self.field[new\_x][new\_y].object = "f"  
 self.field[fox[0]][fox[1]].object = ""  
 return  
 elif dx == 2 and self.field[new\_x - 1][new\_y].object == "c" and self.field[new\_x][new\_y].object == "":  
 self.field[new\_x - 1][new\_y].object = ""  
 self.field[new\_x][new\_y].object = "f"  
 self.field[fox[0]][fox[1]].object = ""  
 return  
 elif dy == -2 and self.field[new\_x][new\_y + 1].object == "c" and self.field[new\_x][new\_y].object == "":  
 self.field[new\_x][new\_y + 1].object = ""  
 self.field[new\_x][new\_y].object = "f"  
 self.field[fox[0]][fox[1]].object = ""  
 return  
 elif dy == 2 and self.field[new\_x][new\_y - 1].object == "c" and self.field[new\_x][new\_y].object == "":  
 self.field[new\_x][new\_y - 1].object = ""  
 self.field[new\_x][new\_y].object = "f"  
 self.field[fox[0]][fox[1]].object = ""  
 return  
 directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]  
 random.shuffle(directions)  
 for dx, dy in directions:  
 new\_x, new\_y = fox[0] + dx, fox[1] + dy  
 if 0 <= new\_x < 7 and 0 <= new\_y < 7:  
 if self.field[new\_x][new\_y].object == "":  
 self.field[new\_x][new\_y].object = "f"  
 self.field[fox[0]][fox[1]].object = ""  
 return

**Листинг 8 (Перерисовка карты)**

def load\_field(self):  
 for i in range(7):  
 for j in range(7):  
 if isinstance(self.field[i][j], QPushButton):  
 match self.field[i][j].object:  
 case 'f':  
 self.field[i][j].setIcon(QIcon(QPixmap(self.path + 'fox.png')))  
 case 'c':  
 self.field[i][j].setIcon(QIcon(QPixmap(self.path + 'chicken.png')))  
 case \_:  
 self.field[i][j].setIcon(QIcon(QPixmap()))

**Листинг 9 (Поиск кардинат лисы)**

def find\_cor\_fox(self):  
 foxes = []  
 for i in range(7):  
 for j in range(7):  
 if isinstance(self.field[i][j], QPushButton) and self.field[i][j].object == 'f':  
 foxes.append((i, j))  
 return foxes

**Листинг 10 (Обработка хода игрока)**

def button\_checker(self):  
 button = self.sender()  
 if self.active\_button:  
 i1, j1 = self.find\_qpushbutton\_index(button)  
 i2, j2 = self.find\_qpushbutton\_index(self.active\_button)  
 directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]  
 for dx, dy in directions:  
 new\_x, new\_y = i2, j2  
 new\_x += dx  
 new\_y += dy  
 if 0 <= new\_x < 7 and 0 <= new\_y < 7:  
 if self.field[new\_x][new\_y].object == "":  
 self.field[new\_x][new\_y].setStyleSheet('''''')  
 if (abs(i1 - i2) == 1 and abs(j1 - j2) == 0 or abs(i1 - i2) == 0 and abs(j1 - j2) == 1) and i1 <= i2:  
 self.field[i1][j1].object, self.field[i2][j2].object = self.field[i2][j2].object, self.field[i1][  
 j1].object  
 self.active\_button = None  
 foxes = self.find\_cor\_fox()  
 for fox in foxes:  
 self.fox\_stepping(fox)  
 else:  
 self.active\_button = False  
 else:  
 if button.object == 'c':  
 self.active\_button = button  
 directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]  
 for dx, dy in directions:  
 new\_x, new\_y = self.find\_qpushbutton\_index(self.active\_button)  
 new\_x += dx  
 new\_y += dy  
 if 0 <= new\_x < 7 and 0 <= new\_y < 7:  
 if self.field[new\_x][new\_y].object == "":  
 self.field[new\_x][new\_y].setStyleSheet('''  
 QPushButton {  
 background-color: rgb(0, 150, 0);  
 }  
 ''')  
 self.load\_field()  
 self.is\_win()

**Листинг 11 (Проверка победы/проигрыша)**

def is\_win(self):  
 chicken\_count = 0  
 for i in range(7):  
 for j in range(7):  
 if isinstance(self.field[i][j], QPushButton) and self.field[i][j].object == 'c':  
 chicken\_count += 1  
 if chicken\_count < 9:  
 QMessageBox.information(self, 'Game Over', 'К сожалению, вы проиграли')  
 self.new\_game = MainWindow()  
 self.new\_game.show()  
 self.close()  
 del self  
 elif (self.field[0][2].object == 'c' and self.field[0][3].object == 'c' and self.field[0][4].object == 'c' and  
 self.field[1][2].object == 'c' and self.field[1][3].object == 'c' and self.field[1][4].object == 'c' and  
 self.field[2][2].object == 'c' and self.field[2][3].object == 'c' and self.field[2][4].object == 'c'):  
 QMessageBox.information(self, 'Game Over', 'Поздравляем, вы победили!')  
 self.new\_game = MainWindow()  
 self.new\_game.show()  
 self.close()  
 del self