Для создания игры «Сапер» на языке Python удобнее всего будет использовать Tkinter, кроссплатформенную событийно-ориентированную графическую библиотеку на основе средств Tk. Она входит в стандартную библиотеку Python.

Чтобы воспользоваться этой библиотекой, нам необходимо ее подключить, написав следующую команду.



Рисунок 1 – Подключение Tkinter

Я решила создать игру с помощью объектно-ориентированного подхода, поэтому, в первую очередь, нам понадобится класс, где будут располагаться ключевые методы для управления. Назовем этот класс MineSweeper (в переводе – «Сапер»).

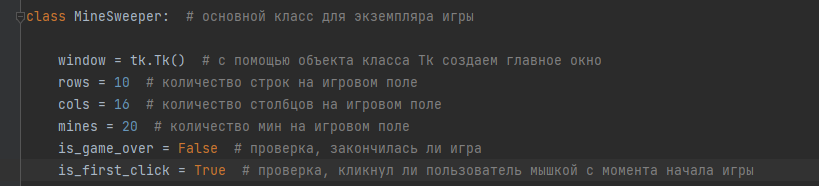


Рисунок 2 – Создание класса MineSweeper

Также нам понадобится класс, хранящий дополнительную информацию о каждой конкретной кнопке, так как в классе tk.Button недостаточно полей для этого. Назовем этот класс MyButton и сделаем его классом, наследуемым от tk.Button. Инициализируем кнопку класса tk.Button, чтобы добавлять к ней дополнительные поля и переопределить инициализацию в новом классе.

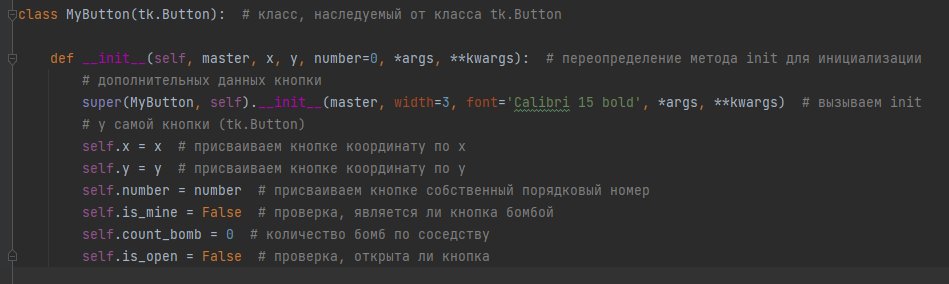


Рисунок 3 – Класс MyButton

Следующим шагом нужно переопределить метод \_\_init\_\_(self) в классе MineSweeper, который запускается каждый раз при создании объекта данного класса. Этот метод необходим для инициализации игры, то есть здесь нужно создать данные, которые мы будем обрабатывать в дальнейшем. Создадим здесь список buttons, хранящий все кнопки игрового поля.

Забегая вперед, уточним небольшую деталь. У крайних и центральных элементов нашей матрицы кнопок разное количество соседних элементов, что очень неудобно для перебора и подсчета соседей каждой кнопки. Поэтому будет удобно создать барьерные (крайние) кнопки, огибающие нашу матрицу со всех сторон. Тогда у всех нужных нам кнопок будет по 8 соседей. Таким образом, сразу создаем двумерный список, где количество рядов и столбцов будет увеличено на 2.

Также здесь мы обрабатываем варианты нажатия на кнопку: левой или правой кнопкой мыши.

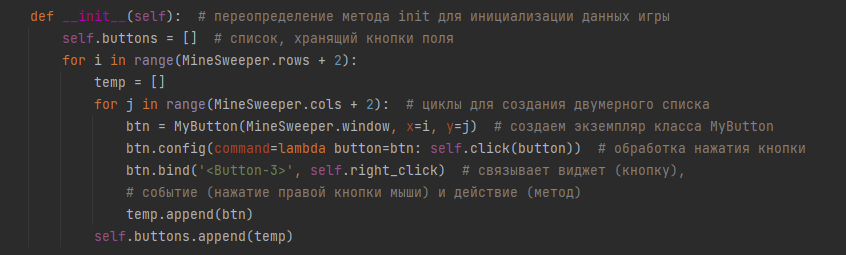


Рисунок 4 – Метод \_\_init\_\_(self) в MineSweeper

Далее создаем виджеты: меню и кнопки, а также присваиваем каждой кнопке порядковый номер. Методы reload(self) и create\_settings\_window(self) опишем ниже.

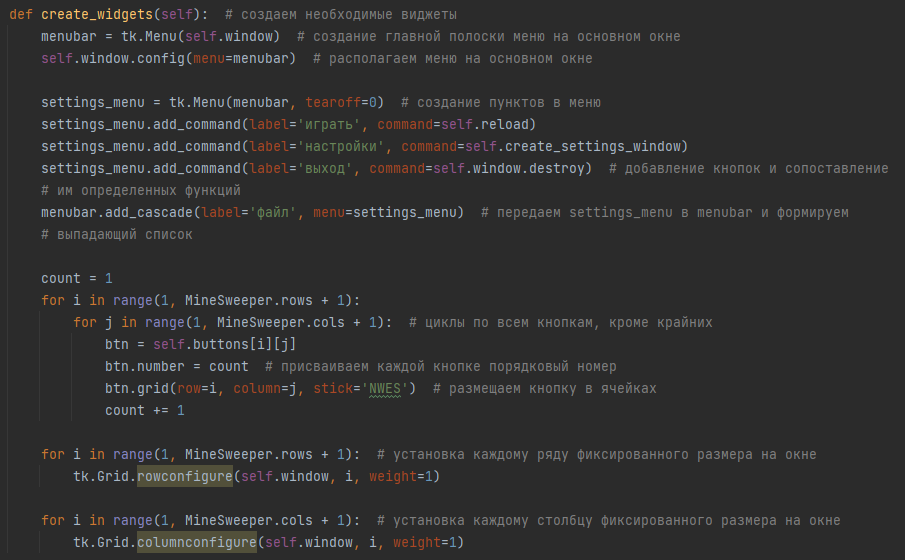


Рисунок 5 – Метод create\_widgets(self) в MineSweeper

Метод reload(self) отвечает за перезапуск игры, в случае выбора этого пункта в меню. С помощью пробега по элементам главного окна, мы удаляем их и заново инициализируем данные и создаем виджеты.

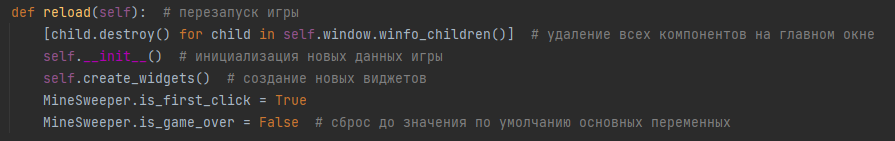


Рисунок 6 – Метод reload(self) в MineSweeper

Метод create\_settings\_window(self) в паре с методом change\_settings(row: tk.Entry, col: tk.Entry, mine: tk.Entry) отвечает настройку интерфейса, которую может производить сам пользователь.

В методе create\_settings\_window(self) создается дочернее окно, в этом окне находятся 3 поля ввода: для количества строк в игровом поле, для количества столбцов в игровом поле и для количества заминированных ячеек. Также здесь создается кнопка, нажав на которую, пользователь может применить все введенные значения к игровому полю.

При нажатии на данную кнопку вызывается метод change\_settings(row: tk.Entry, col: tk.Entry, mine: tk.Entry), где полям класса MineSweeper присваиваются введенные значения.

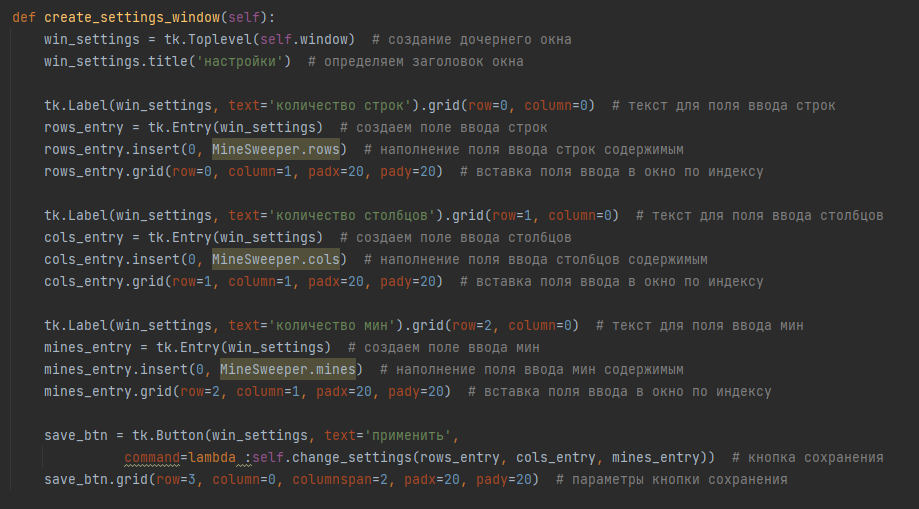


Рисунок 7 – Метод create\_settings\_window(self) в MineSweeper

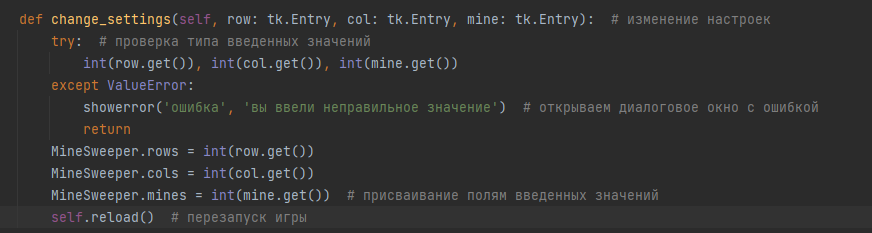


Рисунок 8 – Метод change\_settings(row: tk.Entry, col: tk.Entry, mine: tk.Entry) в MineSweeper

Виджеты готовы, теперь можно написать метод start(self), при вызове которого открывается окно и запускается игровой процесс.

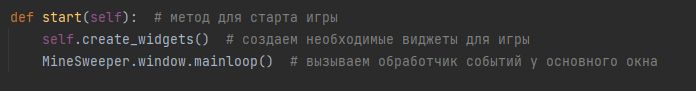


Рисунок 9 – Метод start(self) в MineSweeper

В главной программе создаем экземпляр класса MineSweeper и вызываем у него метод start(self).

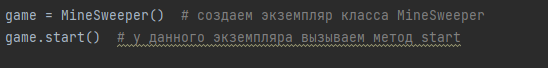


Рисунок 10 – Создание экземпляра класса MineSweeper

Теперь необходимо создать список заминированных кнопок. Для этого понадобятся 2 метода: get\_mines\_places(exclude\_number: int) и insert\_mines(self, number: int), а также нужно импортировать функцию shuffle из модуля random.

В методе get\_mines\_places(exclude\_number: int) с помощью перемешивания элементов и среза мы определяем какие кнопки будем минировать.

В методе insert\_mines(self, number: int) мы уже приступаем к непосредственному минированию кнопок, то есть к изменению их поля is\_mine на значение True. Также для удобства печатаем порядковые номера заминированных кнопок.



Рисунок 11– Подключение функции shuffle из модуля random

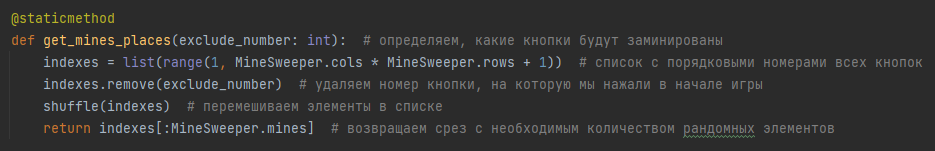


Рисунок 12 – Метод get\_mines\_places(exclude\_number: int) в MineSweeper

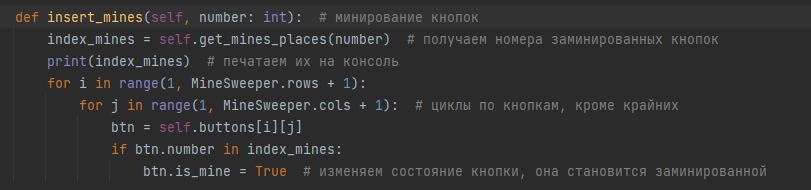


Рисунок 13 – Метод insert\_mines(self, number: int) в MineSweeper

Чтобы выводить количество заминированных соседей на кнопке, нам нужно его подсчитать. Для этого применим метод count\_mines\_in\_buttons(self).

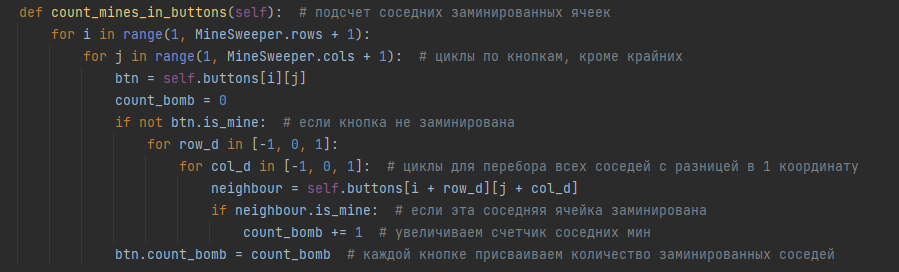


Рисунок 14 – Метод count\_mines\_in\_buttons(self) в MineSweeper

Удобно выводить прототип таблицы игрового поля на консоль. Для этого напишем метод print\_buttons(self).

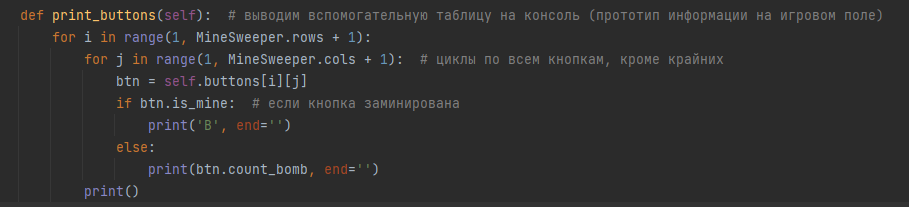


Рисунок 15 – Метод print\_buttons(self) в MineSweeper

Приступаем к обработке нажатий левой клавиши мыши с помощью метода click(self, clicked\_button: MyButton). Также для показа сообщений импортируем функции showinfo и showerror из модуля tkinter.messagebox. Для того, чтобы раскрасить цифры на кнопках, создаем словарь, где каждой цифре сопоставляем цвет.



Рисунок 16 – Подключение showinfo и showerror из модуля tkinter.messagebox

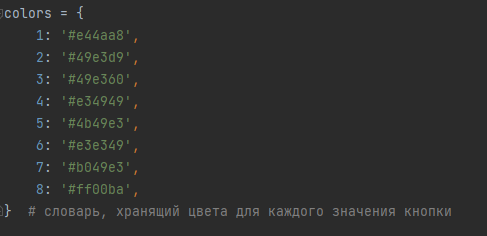


Рисунок 17 – Словарь colors

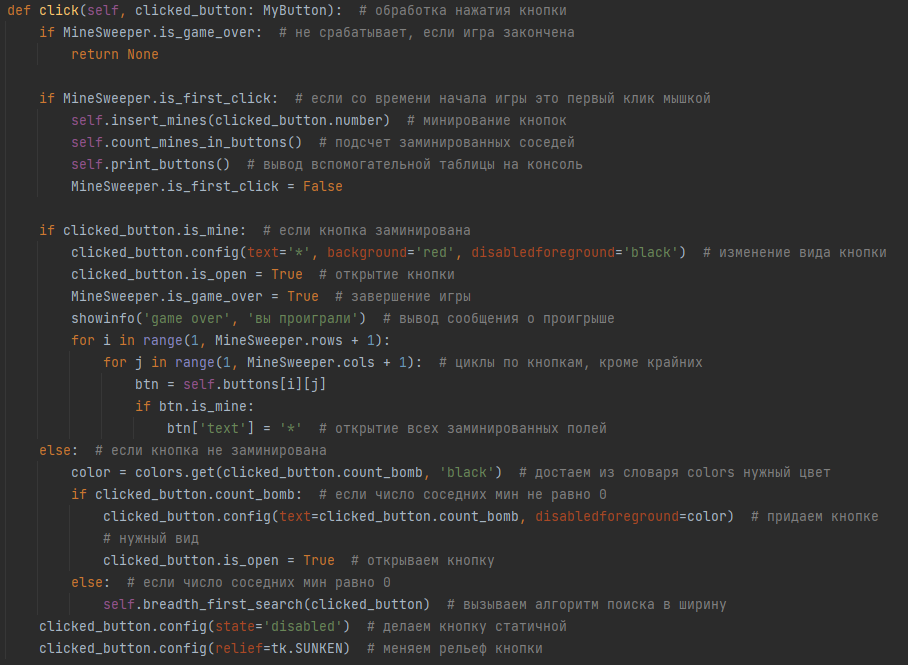


Рисунок 18 – Метод click(self, clicked\_button: MyButton) в MineSweeper

В оригинальной игре «Сапер» в случае, если пользователь нажал на кнопку, у которой нет соседних мин, открывается группа кнопок без мин, которые окружают эту кнопку. Реализуем этот процесс при помощи алгоритма поиска в ширину и выполним его в методе breadth\_first\_search(self, btn: MyButton).

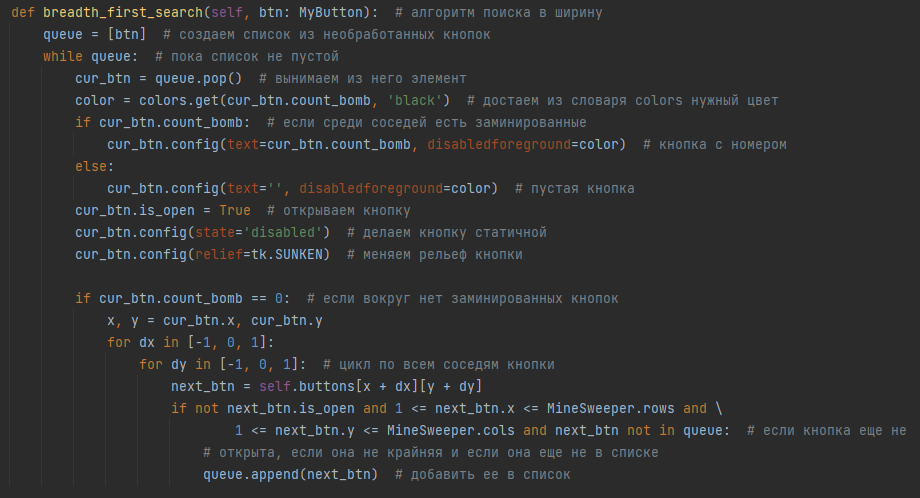


Рисунок 19 – Метод breadth\_first\_search(self, btn: MyButton) в MineSweeper

Чтобы помечать и обезвреживать мины, реализуем метод right\_click(self, event) для установки флажков на кнопки правой кнопкой мыши.

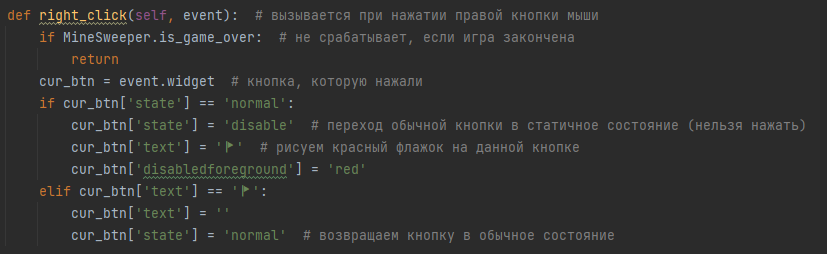


Рисунок 19 – Метод right\_click(self, event) в MineSweeper

Игра полностью готова, теперь можно ее запустить.

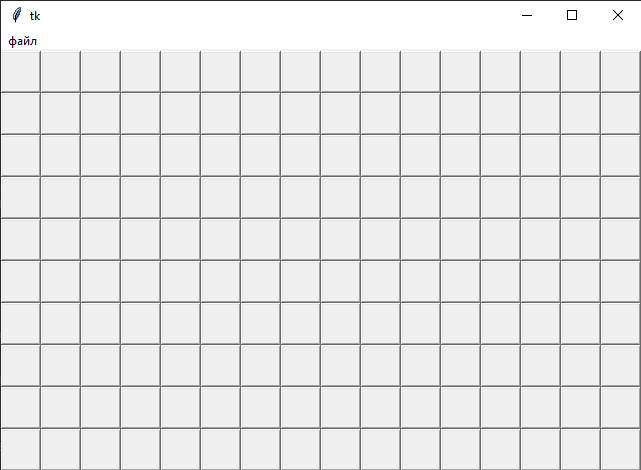


Рисунок 20 – Пустое игровое поле

Играем и убеждаемся, что все работает правильно. Вот как может выглядеть поле в течение игры.

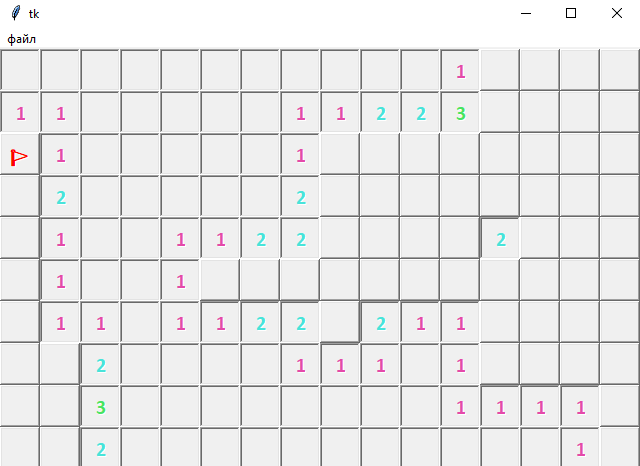


Рисунок 21 – Поле в процессе игры

С помощью вспомогательной таблицы на консоли находим мину и нажимаем на нее.

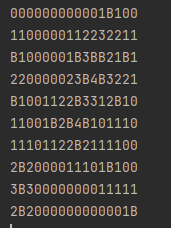


Рисунок 22 – Вспомогательная таблица на консоли

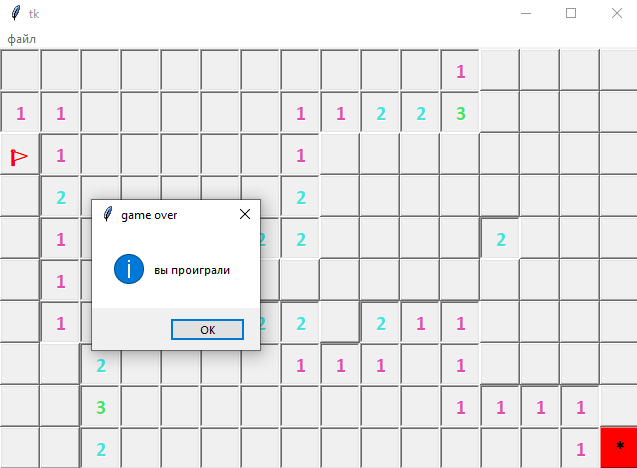


Рисунок 23 – Сообщение о проигрыше

После нажатия «ОК» программа показывает места, где находились все бомбы.

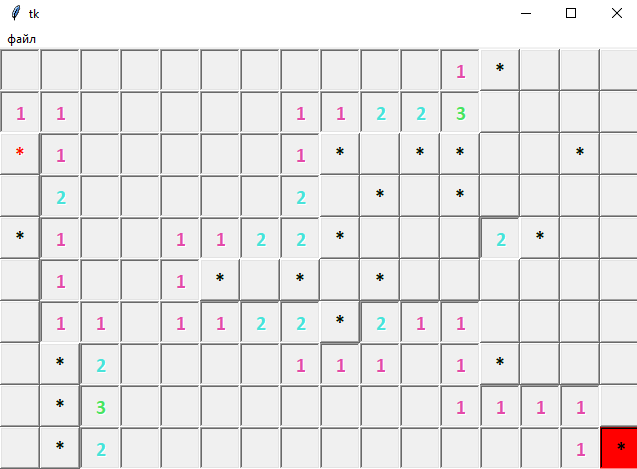


Рисунок 24 – Игровое поле после проигрыша

Открываем меню и нажимаем «играть».

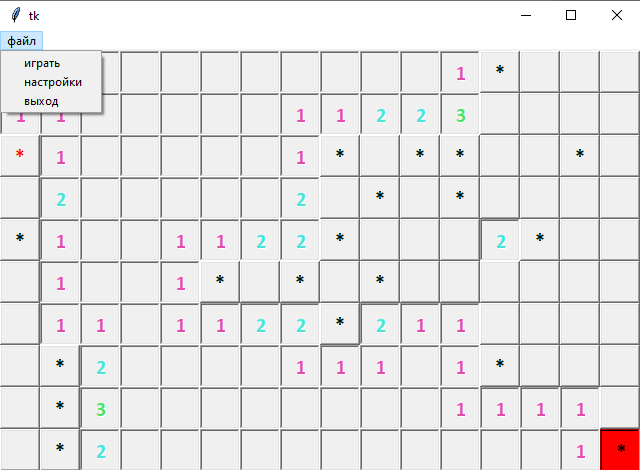


Рисунок 25 – Меню игры

Программа создает новое поле для игры. В нем проверяем работу настроек. Настраиваем параметры таким образом: 3 строки, 4 столбца и 5 мин.

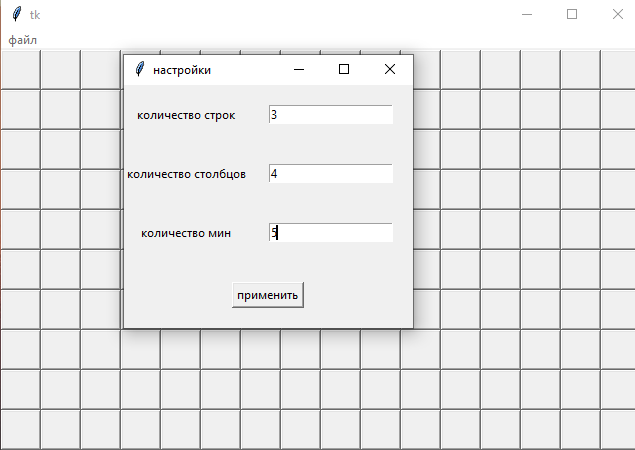


Рисунок 26 – Настройки игры

Программа работает правильно.



Рисунок 27 – Игровое поле с измененными настройками

Закрываем программу через меню.