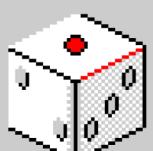
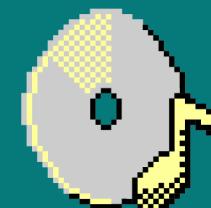
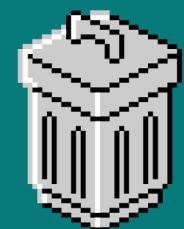
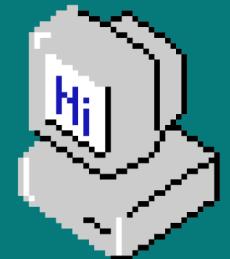


FIRE IN DA HALL



Amelia Rossinskaya



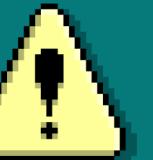
11:11PM

Git Hub



https://github.com/ame-mi/FIRE_GRAPH

H



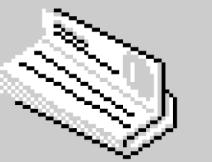
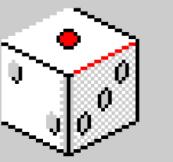
ame-mi/
FIRE_GRAPH

1 Contributor 0 Issues 0 Stars 0 Forks

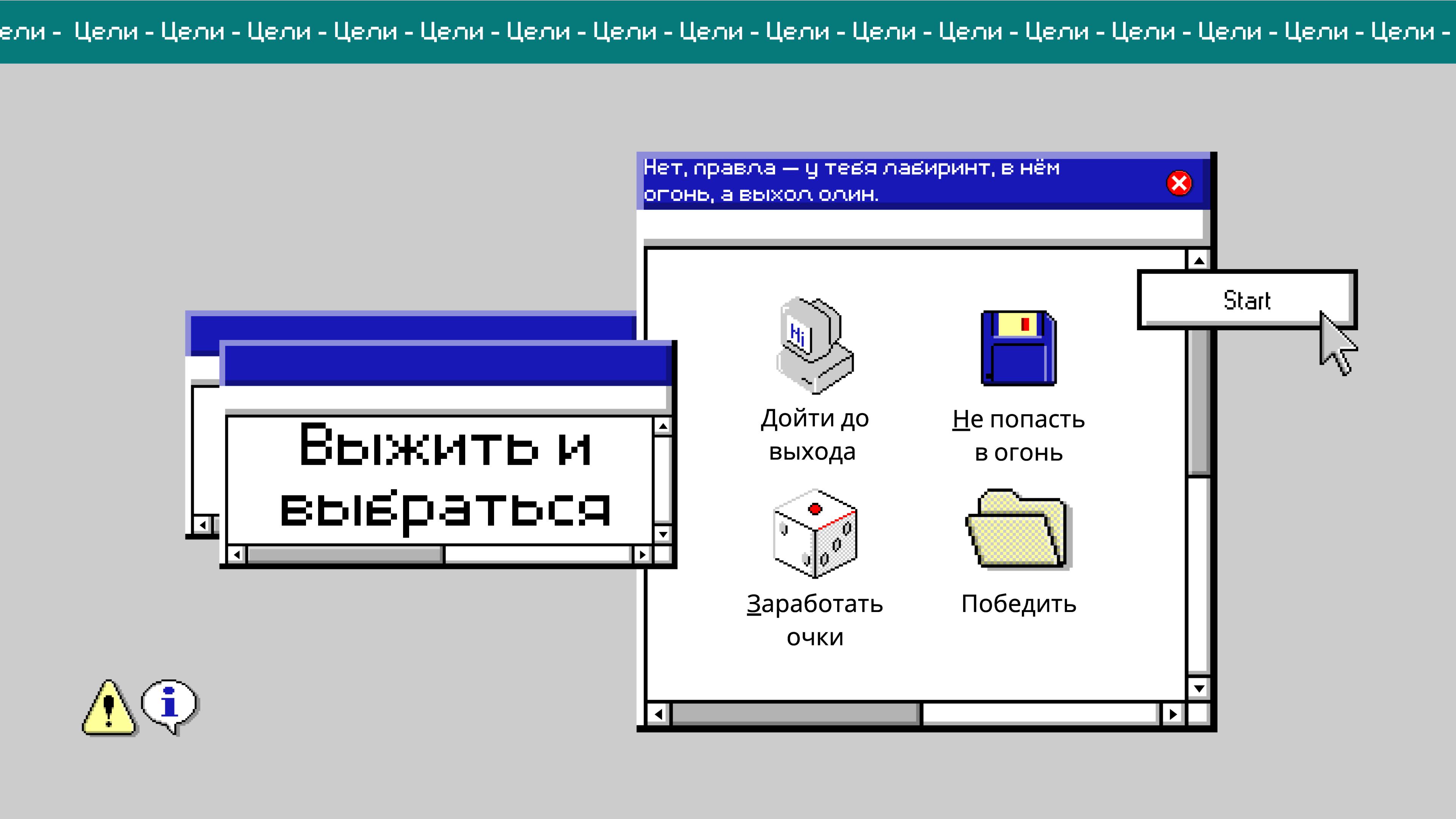
ame-mi/FIRE_GRAPH

Contribute to ame-mi/FIRE_GRAPH development by creating an account on GitHub.

[GitHub](#)



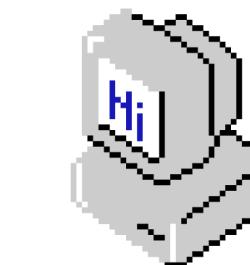
11:11PM



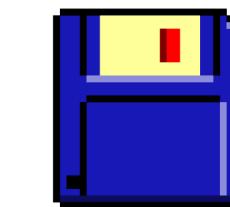
Нет, правда — у тебя лабиринт, в нём огонь, а выход олин.



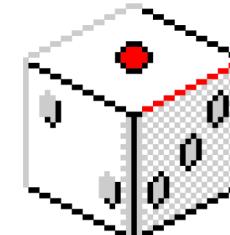
Start



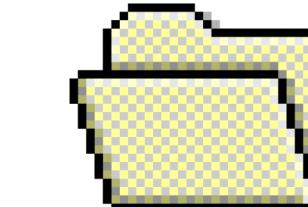
Дойти до выхода



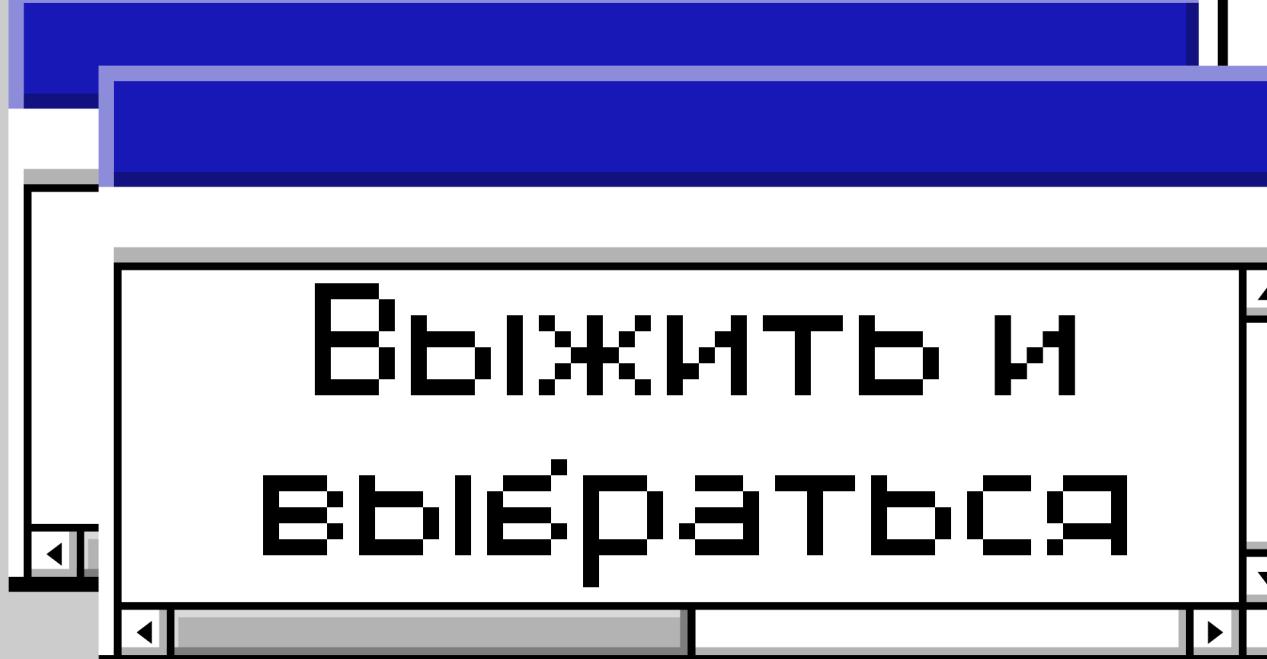
Не попасть в огонь



Заработать очки



Победить



Как играть???

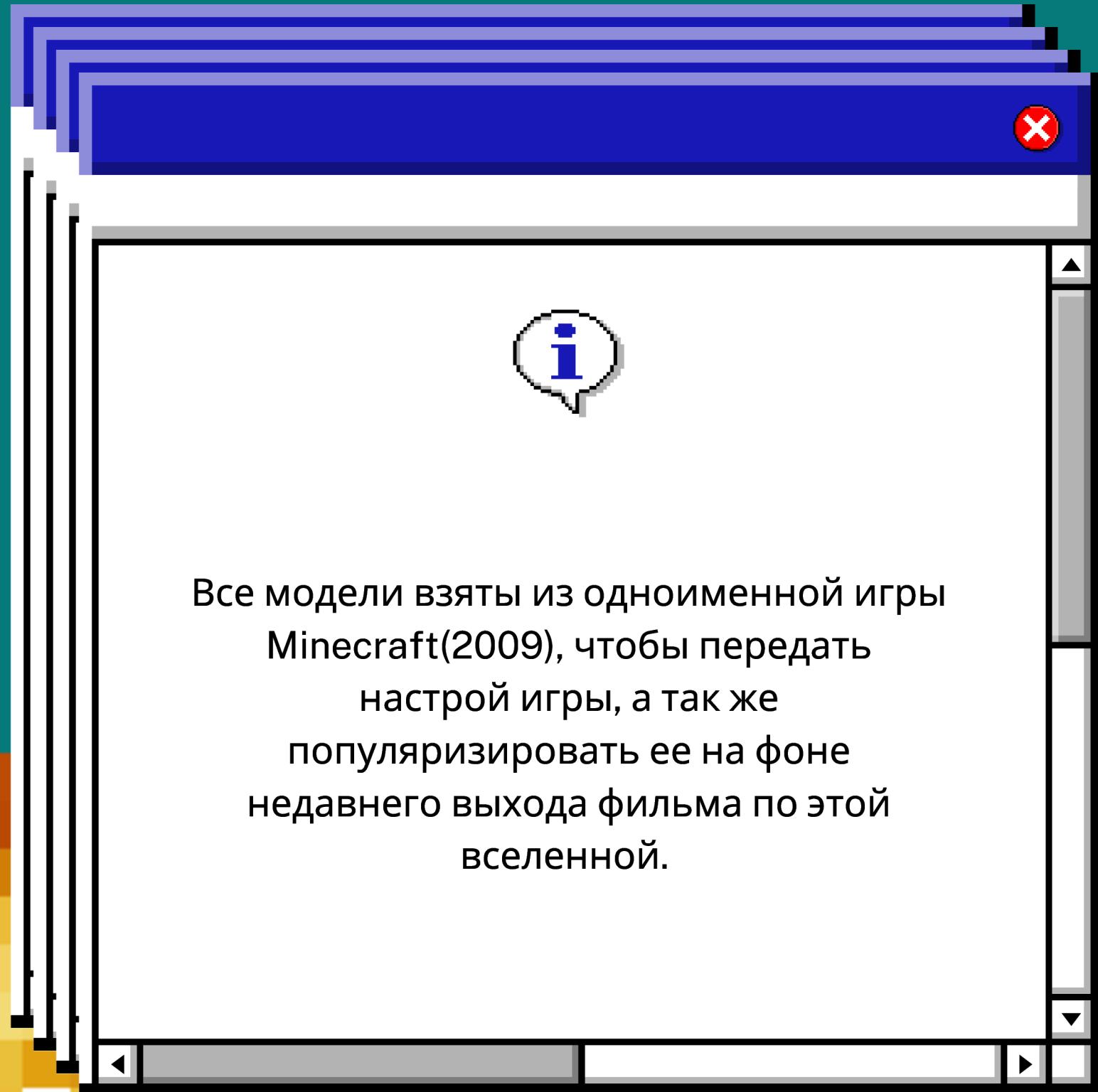
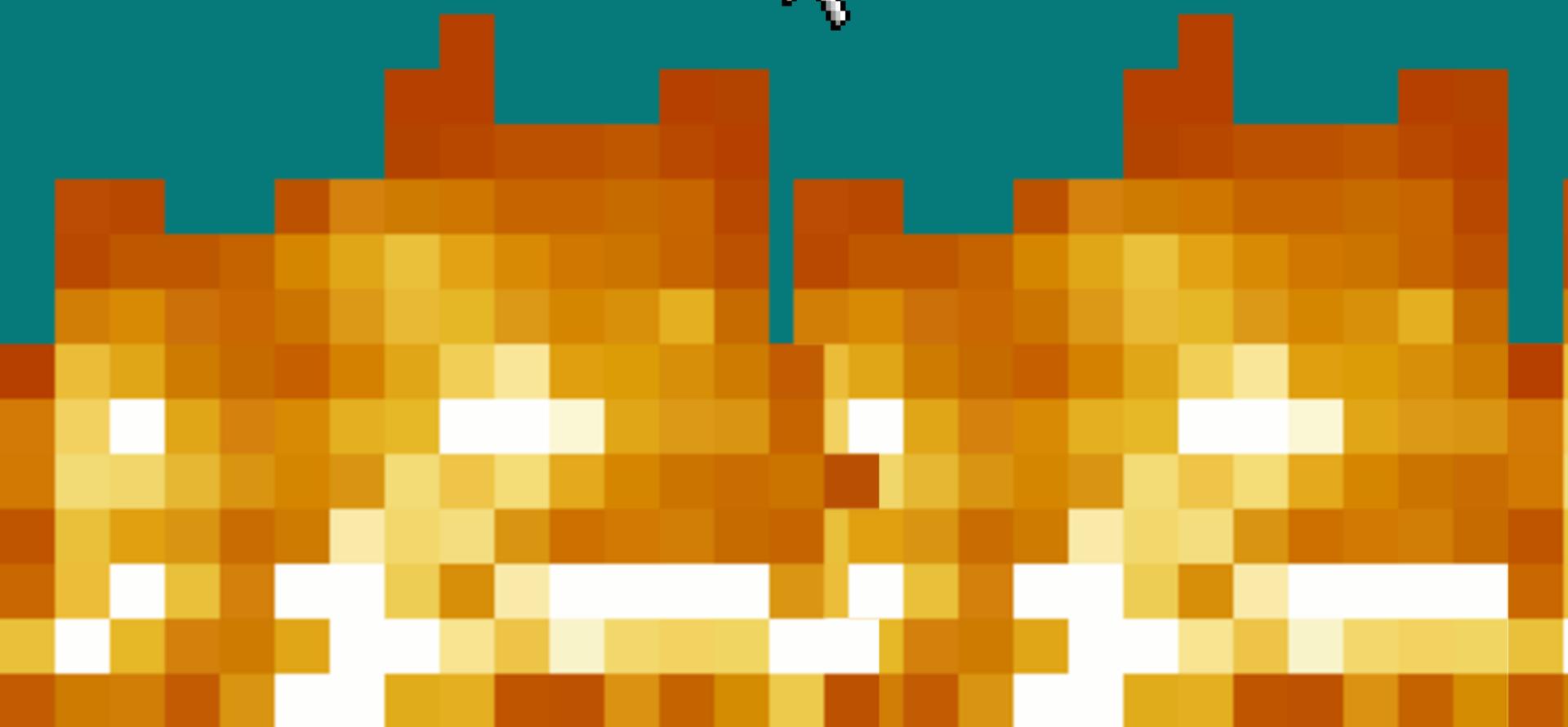
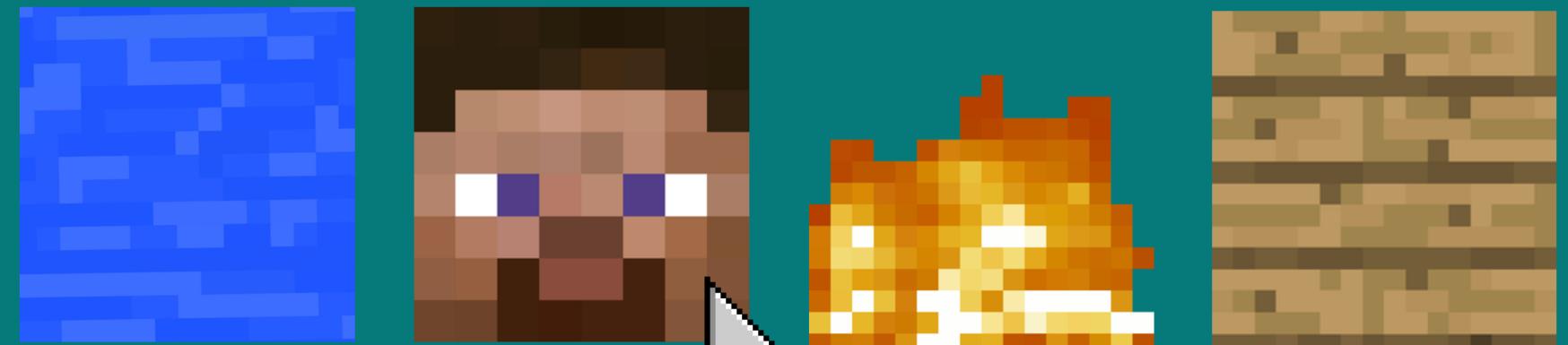
?

1. Кликни по пустой клетке, чтобы поставить человечка
2. Ляльше человечек начинает илти
3. С этого момента огонь тоже начинает распространяться
4. Лойли ло выхода (желательно живым)
5. Умер? Ну... сыграй снова 😊



[Выжить и выбраться](#)

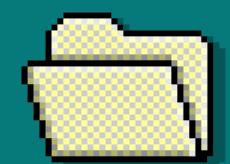
Модельки



[Выжить и выбраться](#)



Что можно ловить в будущем?



Плюшки

Ускорение, огнетушитель и
прочее



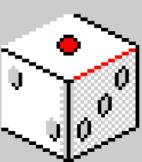
Опасности

Увеличить сложность игры
новыми препятствиями



Лидершип

Таблица лучших игроков в эту
игру

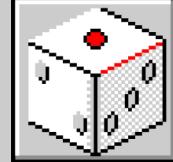
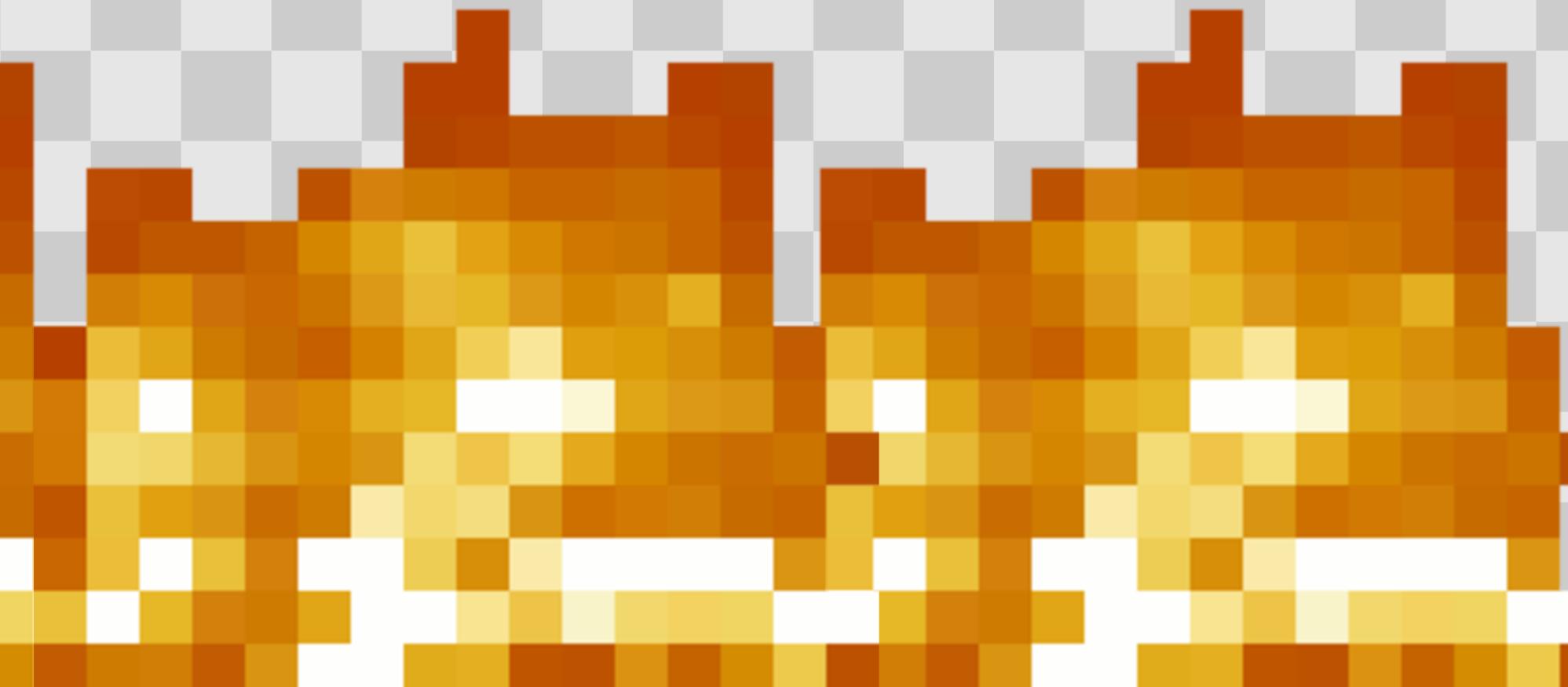


Выжить и выбраться

```
class class_FIRE:  
    def __init__(self, grid) -> None:  
        self.grid = grid  
  
    def Spread(self, grid) -> None:  
        New_Fire = []  
        for y in range(1, grid.shape[0] - 1):  
            for x in range(1, grid.shape[1] - 1):  
                if grid[y, x] == FIRE:  
                    for dx, dy in routes:  
                        nx, ny = x + dx, y + dy  
                        if grid[nx, ny] == EMPTY and random.random() < 0.06:  
                            New_Fire.append((nx, ny))  
        for x, y in New_Fire:  
            grid[y, x] = FIRE  
        return grid
```

Соседние пустые клетки могут загореться с вероятностью 6%. Горение – постепенное и случайное.

Механика огня

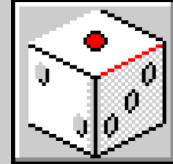


Выжить и выбраться

алго
ритм
зане
ре



1. Присваиваем каждой ячейке уникальное множество.
2. Проходим по строке слева направо:
 - а. Если решаем поставить правую стенку — просто ставим.
 - б. Если не ставим, но ячейки уже в одном множестве — ставим (во избежание циклов).
 - с. Иначе объединяем множества (справа → влево).
3. Обрабатываем нижние стенки:
4. Каждое множество должно иметь минимум 1 ячейку без нижней стены, чтобы проложить в следующую строку.
5. Переход к следующей строке:
 - а. Копируем текущую строку:
 - и. Удаляем правые и нижние стены.
 - ii. Ячейкам без связи вниз присваиваем новые множества.
6. Повторяем обработку правых и нижних стенок для каждой новой строки.
7. Завершающая строка:
 - а. Ставим нижние стены для всех ячеек.
 - б. Объединяем соседние ячейки с разными множествами (убираем стенку справа).
 - с. Все множества объединяются → елиный путь между всеми точками.



Выжить и выбраться

```
def GenLab(lines, clm):
    labirint = np.full((lines * 2 + 1, clm * 2 + 1), WALL)

    def Ri(x, y):
        labirint[y][x] = EMPTY
        Route_To = routes.copy()
        random.shuffle(Route_To)
        for dx, dy in Route_To:
            nx, ny = x + dx * 2, y + dy * 2
            if 0 <= nx < clm * 2 + 1 and 0 <= ny < lines * 2 + 1 and
            labirint[ny][nx] == WALL:
                labirint[y + dy][x + dx] = EMPTY
                Ri(nx, ny)

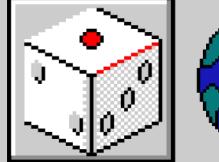
    Ri(1, 1)

    labirint[lines * 2 - 1][clm * 2 - 1] = EXT

    for _ in range(lines):
        x, y = random.randint(1, clm * 2 - 1), random.randint(1, lines *
        2 - 1)
        if labirint[y, x] == EMPTY:
            labirint[y, x] = FIRE

    return labirint
```

Строим лабиринт



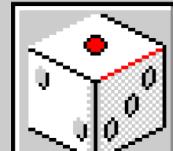
Выжить и выбраться

```
def grph(grid, strt):
    lines, clm = grid.shape
    last = set()
    queue = [strt]
    par = {}
    sx, sy = strt

    while queue:
        x, y = queue.pop(0)
        if grid[y, x] == EXT:
            Plan_route = []
            while (x, y) != (sx, sy):
                Plan_route.append((x, y))
                x, y = par[(x, y)]
            Plan_route.reverse()
            return Plan_route
```

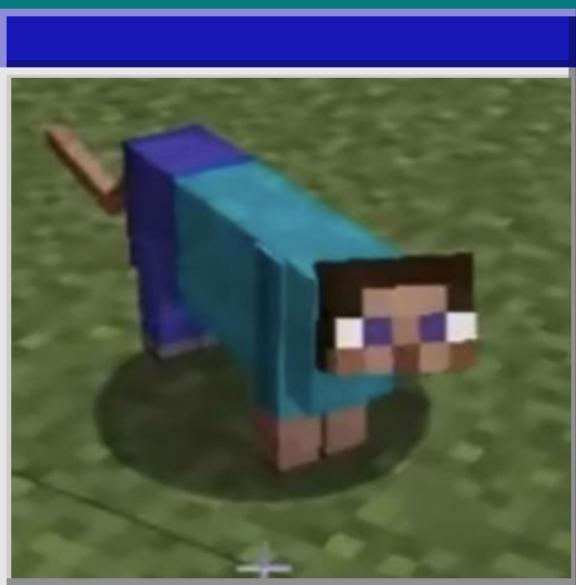
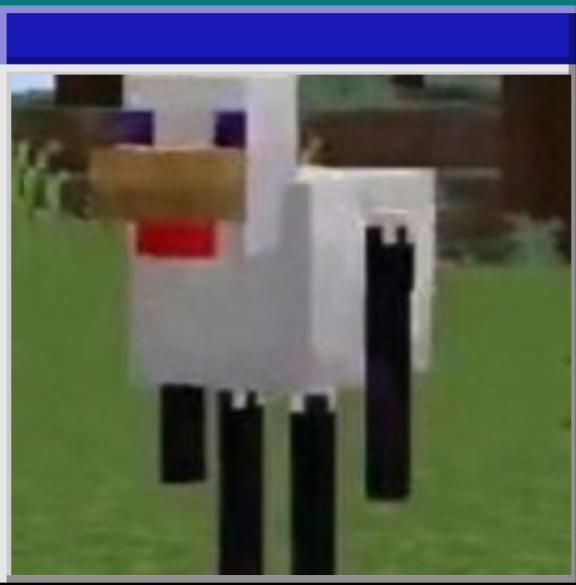
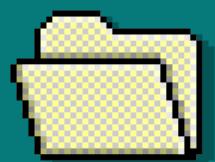
Путь от человека до выхода
ищется с помощью поиска в
ширину (BFS). Этот алгоритм
находит кратчайший путь,
учитывая огонь (начальный) и
стены.

Спасаем чужака



Выжить и выбраться

Лань Уважения Игре



Выжить и выбраться

КРИТ ЕРИИ

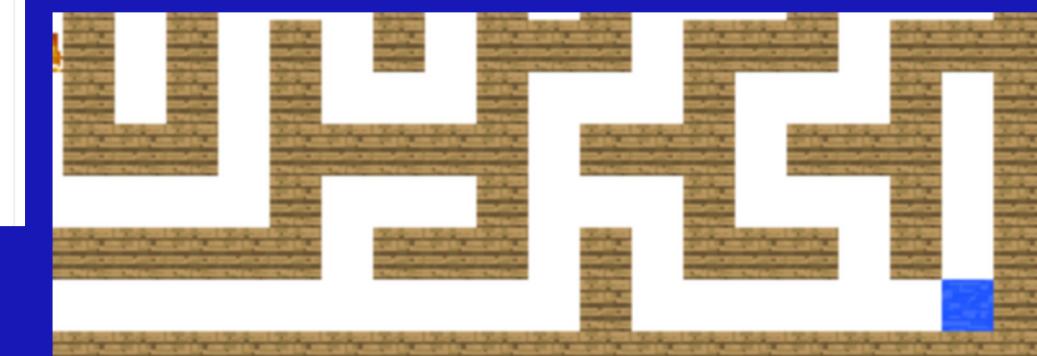
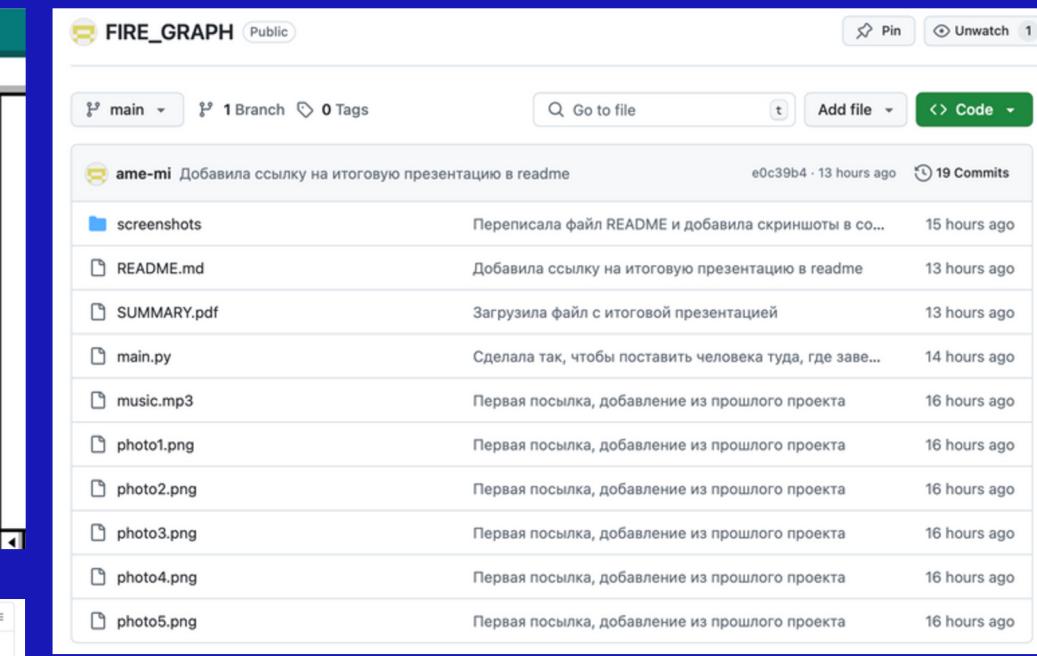
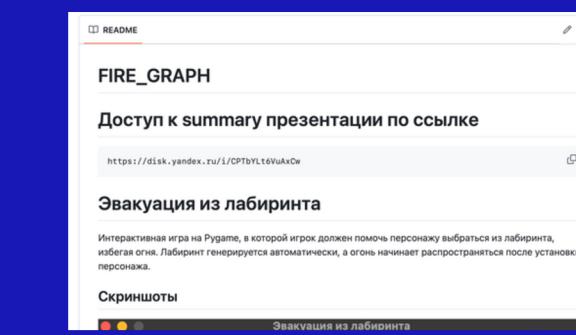
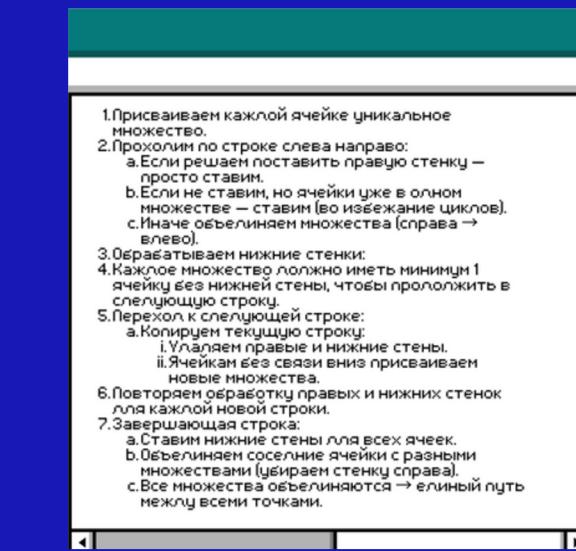
Само оценка



1. Самостоятельность выполнения работы
 - а. Проект выполнен полностью самостоятельно, без использования искусственного интеллекта для генерации кола. Допустимо использование материалов из интернета (например, реализация конкретного алгоритма), но с обязательной адаптацией под проект.
2. Структура программы и использование ООП
 - а. Кол хорошо структурирован, разделен на классы и методы. Все дополнительные файлы (изображения, звуки, конфигурации) вынесены в отдельные папки. Используются принципы ООП (наследование, инкапсуляция, полиморфизм).
3. Хранение кола в репозитории
 - а. Колложен в репозиторий (GitHub, GitLab или аналог). Репозиторий содержит README с описанием проекта, инструкцией по запуску и скриншотами.
4. Визуализация
 - а. Визуализация проекта выполнена на высоком уровне: используются спрайты, анимации, изображения, звуки. Интерфейс интуитивно понятен и эстетически приятен.
5. Сложность алгоритма и использование неизученных алгоритмов
 - а. Используются сложные алгоритмы, которые не изучались в курсе (например, A*, алгоритм Крускала, Прима и т.д.). Работа алгоритма рассказана на защите проекта.
6. Оформленный отчет
 - а. Отчет оформлен в виде HTML-страницы, выложен на хостинге (например, GitHub Pages, Tilda, Wix). Отчет содержит описание проекта, скриншоты, инструкцию по запуску и ссылку на репозиторий.
7. Дополнительные критерии
 - а. Проект содержит дополнительные функции, которые улучшают пользовательский опыт (например, меню настроек, сохранение прогресса, звуковое сопровождение, анимации).

```
class class_FIRE:  
    def __init__(self, grid) -> None:  
        self.grid = grid  
  
    def Spread(self, grid) -> None:  
        New_Fire = []  
        for y in range(1, grid.shape[0] - 1):  
            for x in range(1, grid.shape[1] - 1):  
                if grid[y, x] == FIRE:  
                    for dx, dy in routes:  
                        nx, ny = x + dx, y + dy  
                        if grid[nx, ny] == EMPTY and random.random() < 0.06:  
                            New_Fire.append((nx, ny))  
  
        for x, y in New_Fire:  
            grid[y, x] = FIRE  
        return grid
```

```
< FIRE_GRAPH ~/FIRE_GRAPH  
> photos_music  
> screenshots  
• Fire.pdf  
py main.py  
MD README.md
```



```
Grid_GRPH = GenLab(lines, clm)  
pygame.mixer.music.load("photos_music/music.mp3")
```

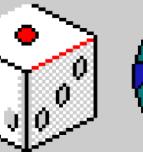
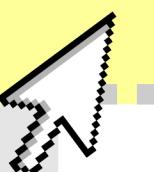


Выжить и выбраться



Google.com. (2025a). Available at: <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://habr.com/ru/articles/746916/&ved=2ahUKEwiK-NHY5saMAxXwSkEAHcG6ID0QFnoECBYQAQ&usg=A0vVaw0usuQ59JXDeh5c91qqC8Bm> [Accessed 7 Apr. 2025]. Google.com. (2025b). Available at: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://github.com/lpestl/Maze&ved=2ahUKEwiK-NHY5saMAxXwSkEAHcG6ID0QFnoECBcQAQ&usg=A0vVaw3LGGp_80k_nw4WWe7ZSqP [Accessed 7 Apr. 2025]. Google.com. (2025c). Available at: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://tproger.ru/articles/maze-generators&ved=2ahUKEwiK-NHY5saMAxXwSkEAHcG6ID0QFnoECBUQAQ&usg=A0vVaw3lT9VhowDfCqcbZ_Xnupriq [Accessed 7 Apr. 2025].

[Back to Agenda Page](#)



Выжить и выбраться