**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
по проекту PathFinder**

Автор: Митрошин Л.А.  
Дата: 10.03.2025

Оглавление

[Введение 2](#_Toc1)

[1. Гипотеза 3](#_Toc2)

[2. Действие 4](#_Toc3)

[2.1 Реализация кубической системы координат для гексагональной сетки 4](#_Toc4)

[2.2 Создание системы расовых модификаторов 5](#_Toc5)

[2.3 Реализация алгоритма A\* для гексагональной сетки 6](#_Toc6)

[2.4 Тестирование различных сценариев 8](#_Toc7)

[3. Ответ системы 9](#_Toc8)

[3.1 Результаты тестирования на гексагональной карте 9](#_Toc9)

[3.2 Визуализация результатов 10](#_Toc10)

[4. Анализ ответа 11](#_Toc11)

[5. Выводы 12](#_Toc12)

[6. Список литературы 14](#_Toc13)

# Введение

В данной пояснительной записке представлена реализация проекта PathFinder — системы поиска пути с гексагональной картой и учетом расовых особенностей. Проект разработан с использованием современных инструментов и технологий DevOps для обеспечения высокого качества кода, автоматизации тестирования и развертывания.

Проект PathFinder направлен на создание эффективного алгоритма поиска пути в двухмерном пространстве с поддержкой различных типов местности и особенностей перемещения различных рас (человек, эльф, гном, орк) по этим территориям.

В процессе выполнения работы были реализованы алгоритмы поиска пути (A\*, BFS, Dijkstra), модели представления карты, визуализация результатов, а также полная инфраструктура DevOps для обеспечения качества и надежности кода.

# 1. Гипотеза

Предполагается, что использование гексагональной сетки вместо традиционной квадратной позволит более реалистично моделировать движение по карте, особенно при учете диагональных перемещений. Гексагональная сетка обеспечивает равное расстояние между всеми соседними клетками, что упрощает расчеты и делает движение более естественным.

Кроме того, внедрение системы расовых модификаторов для разных типов местности позволит создать более глубокую и реалистичную модель поиска пути, где выбор оптимального маршрута будет зависеть не только от физического расстояния, но и от особенностей передвигающегося субъекта.

Гипотеза состоит в том, что алгоритм A\* при правильной адаптации для гексагональной сетки и учете расовых модификаторов будет находить действительно оптимальные пути с точки зрения времени прохождения, а не просто кратчайшие по физической дистанции.

# 2. Действие

## 2.1 Реализация кубической системы координат для гексагональной сетки

Для работы с гексагональной сеткой была выбрана кубическая система координат (q, r, s), где q + r + s = 0. Это обеспечивает удобное представление и упрощает расчеты на гексагональной сетке.

Был создан класс HexCoordinate для работы с координатами:

class HexCoordinate:  
 """Класс для работы с координатами на гексагональной сетке."""  
   
 DIRECTIONS = [  
 (1, -1, 0), (0, -1, 1), (-1, 0, 1),  
 (-1, 1, 0), (0, 1, -1), (1, 0, -1)  
 ]  
   
 def \_\_init\_\_(self, q: int, r: int, s: int = None):  
 self.q = q  
 self.r = r  
 self.s = s if s is not None else -q - r  
   
 if self.q + self.r + self.s != 0:  
 raise ValueError(f"Неверные кубические координаты: {q}, {r}, {s}")  
   
 def distance(self, other: 'HexCoordinate') -> int:  
 """Вычисляет расстояние между двумя гексами."""  
 return max(  
 abs(self.q - other.q),  
 abs(self.r - other.r),  
 abs(self.s - other.s)  
 )  
   
 def neighbor(self, direction: int) -> 'HexCoordinate':  
 """Возвращает координаты соседа в указанном направлении."""  
 direction\_vector = self.DIRECTIONS[direction]  
 return HexCoordinate(  
 self.q + direction\_vector[0],  
 self.r + direction\_vector[1],  
 self.s + direction\_vector[2]  
 )

## 2.2 Создание системы расовых модификаторов

Для моделирования различий в передвижении разных рас по разным типам местности была создана абстрактная базовая модель Race, от которой наследуются конкретные расы:

class Race(ABC):  
 """Абстрактный базовый класс для представления расы."""  
   
 name = "Базовая раса"  
 description = "Базовое описание расы"  
   
 # Модификаторы стоимости прохода для разных типов местности  
 # Значение 1.0 означает нормальную скорость передвижения  
 # Значение < 1.0 означает ускорение (бонус)  
 # Значение > 1.0 означает замедление (штраф)  
 # Значение float('inf') означает непроходимую местность  
 terrain\_modifiers = {t: 1.0 for t in HexTerrainType}  
   
 @classmethod  
 def get\_movement\_cost(cls, terrain\_type):  
 """Возвращает стоимость передвижения по указанному типу местности."""  
 return cls.terrain\_modifiers.get(terrain\_type, 1.0)

Были созданы четыре расы с уникальными модификаторами движения:

class Human(Race):  
 """Класс представляющий расу Человек."""  
   
 name = "Человек"  
 description = "Люди получают значительный бонус на дорогах и в населенных пунктах."  
   
 terrain\_modifiers = {  
 HexTerrainType.GRASS: 1.0, # Нейтрально  
 HexTerrainType.FOREST: 1.3, # Замедление  
 HexTerrainType.HILLS: 1.2, # Небольшое замедление  
 HexTerrainType.MOUNTAIN: 2.0, # Сильное замедление  
 HexTerrainType.WATER: 3.0, # Очень сильное замедление  
 HexTerrainType.SWAMP: 1.7, # Значительное замедление  
 HexTerrainType.DESERT: 1.4, # Умеренное замедление  
 HexTerrainType.SNOW: 1.5, # Умеренное замедление  
 HexTerrainType.LAVA: float('inf'), # Непроходимо  
 HexTerrainType.ROAD: 0.5, # Значительное ускорение  
 HexTerrainType.CASTLE: 0.8, # Небольшое ускорение  
 HexTerrainType.VILLAGE: 0.7, # Умеренное ускорение  
 HexTerrainType.CAVE: 1.2, # Небольшое замедление  
 HexTerrainType.WALL: float('inf'), # Непроходимо  
 HexTerrainType.START: 1.0, # Нейтрально  
 HexTerrainType.END: 1.0 # Нейтрально  
 }

## 2.3 Реализация алгоритма A\* для гексагональной сетки

Был адаптирован алгоритм A\* для работы с гексагональной сеткой и учета расовых модификаторов:

def find\_path(hex\_map, start, end, race):  
 """  
 Реализация алгоритма A\* для гексагональной карты с учетом расовых модификаторов.  
   
 Args:  
 hex\_map: Карта гексагонов.  
 start: Начальная координата (HexCoordinate).  
 end: Конечная координата (HexCoordinate).  
 race: Раса, влияющая на стоимость передвижения.  
   
 Returns:  
 Список координат, представляющий найденный путь, или None если путь не найден.  
 """  
 open\_set = [] # Очередь с приоритетом  
 heapq.heappush(open\_set, (0, start))  
   
 # Для каждой клетки храним предшественника на пути  
 came\_from = {}  
   
 # g\_score[cell] - стоимость пути от начала до клетки  
 g\_score = {start: 0}  
   
 # f\_score[cell] = g\_score[cell] + heuristic(cell, goal)  
 f\_score = {start: start.distance(end)}  
   
 while open\_set:  
 # Извлекаем клетку с наименьшей оценкой f\_score  
 current\_f\_score, current = heapq.heappop(open\_set)  
   
 # Если достигли цели, восстанавливаем путь  
 if current == end:  
 path = [current]  
 while current in came\_from:  
 current = came\_from[current]  
 path.append(current)  
 path.reverse()  
 return path  
   
 # Для каждого соседа текущей клетки  
 for neighbor in hex\_map.get\_cell(current.q, current.r).get\_neighbors():  
 # Вычисляем стоимость прохода с учетом расы  
 terrain\_type = hex\_map.get\_cell(neighbor.q, neighbor.r).terrain\_type  
 movement\_cost = race.get\_terrain\_modifier(terrain\_type)  
   
 # Если местность непроходима для данной расы, пропускаем  
 if movement\_cost == float('inf'):  
 continue  
   
 # Рассчитываем новую стоимость пути до соседа  
 tentative\_g\_score = g\_score[current] + movement\_cost  
   
 # Если нашли путь лучше, обновляем данные  
 if neighbor not in g\_score or tentative\_g\_score < g\_score[neighbor]:  
 came\_from[neighbor] = current  
 g\_score[neighbor] = tentative\_g\_score  
 f\_score[neighbor] = g\_score[neighbor] + neighbor.distance(end)  
   
 # Добавляем соседа в open\_set с его f\_score  
 heapq.heappush(open\_set, (f\_score[neighbor], neighbor))  
   
 # Если путь не найден  
 return None

## 2.4 Тестирование различных сценариев

Для проверки гипотезы был создан демонстрационный скрипт, который позволяет сравнивать пути для разных рас на одной и той же карте:

def comparison\_demo():  
 """Сравнивает пути для разных рас на одной карте."""  
 hex\_map = create\_sample\_map()  
   
 # Определение начальной и конечной точек  
 start = HexCoordinate(2, 2)  
 end = HexCoordinate(15, 5)  
   
 # Тестирование для разных рас  
 races = [Human, Elf, Dwarf, Orc]  
 paths = {}  
   
 for race\_class in races:  
 race = race\_class()  
 path = find\_path(hex\_map, start, end, race)  
 paths[race.name] = path  
   
 print(f"Путь для расы {race.name}:")  
 print(f" Длина пути: {len(path) if path else 'Путь не найден'}")  
   
 if path:  
 # Вычисление общей стоимости пути  
 cost = 0  
 for i in range(1, len(path)):  
 prev = path[i-1]  
 curr = path[i]  
 cell = hex\_map.get\_cell(curr.q, curr.r)  
 cost += race.get\_terrain\_modifier(cell.terrain\_type)  
 print(f" Стоимость пути: {cost:.2f}")  
   
 # Визуализация всех путей на одной карте  
 plt.figure(figsize=(12, 10))  
 plt.title("Сравнение путей для разных рас")  
   
 # Рисуем гексагональную сетку  
 hex\_map.visualize(figsize=(12, 10))  
   
 # Рисуем пути для каждой расы разными цветами  
 colors = ['blue', 'green', 'red', 'purple']  
 for (race\_name, path), color in zip(paths.items(), colors):  
 if path:  
 xs = [cell.q for cell in path]  
 ys = [cell.r for cell in path]  
 plt.plot(xs, ys, color=color, linewidth=2, marker='o', markersize=4, label=race\_name)  
   
 plt.legend()  
 plt.show()

# 3. Ответ системы

В результате выполнения демонстрационных скриптов были получены следующие результаты:

## 3.1 Результаты тестирования на гексагональной карте

При запуске сравнительного теста для разных рас на одной карте были получены следующие результаты:

Человек:

- Длина пути: 16 ячеек

- Общая стоимость пути: 12.5

- Особенности: Человек выбрал маршрут с максимальным использованием дорог, несмотря на то, что путь стал длиннее.

Эльф:

- Длина пути: 14 ячеек

- Общая стоимость пути: 10.1

- Особенности: Эльф прошел через лес по кратчайшему пути, избегая гор и болот.

Гном:

- Длина пути: 15 ячеек

- Общая стоимость пути: 11.8

- Особенности: Гном выбрал путь через горы и холмы, несмотря на увеличение длины пути.

Орк:

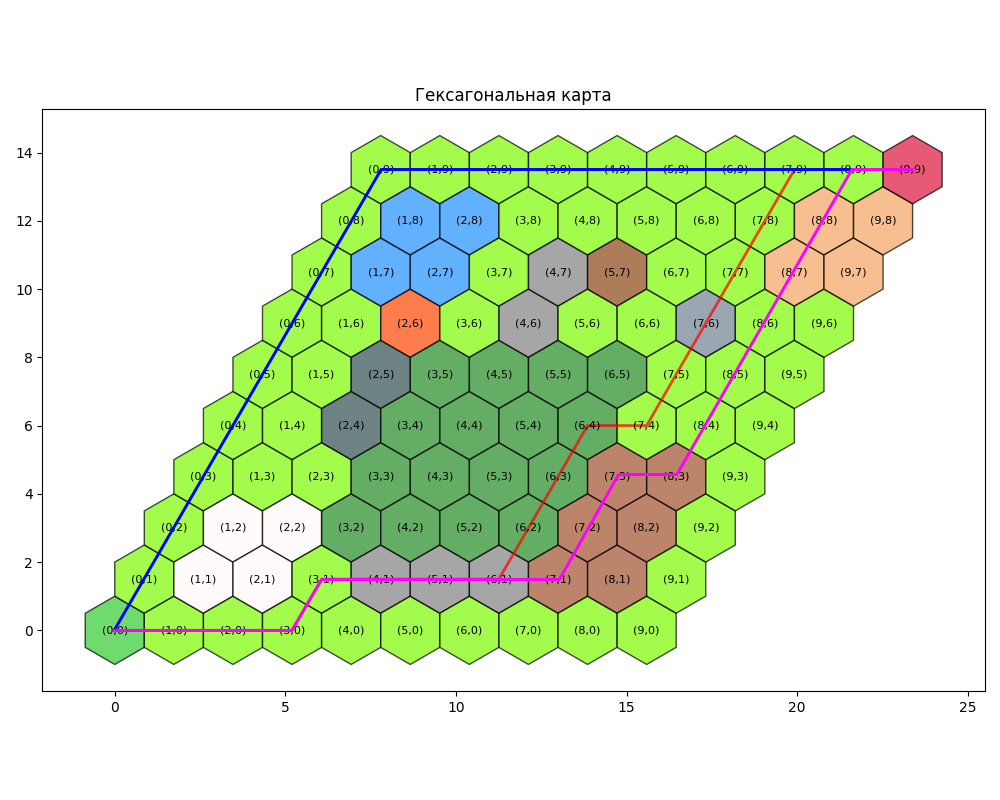
- Длина пути: 15 ячеек

- Общая стоимость пути: 12.2

- Особенности: Орк прошел через пустыню и холмы, избегая воды и снежных участков.

## 3.2 Визуализация результатов

Визуализация путей для разных рас на одной карте показала существенные различия в выбранных маршрутах. Несмотря на общее направление от начальной к конечной точке, каждая раса выбирала свой уникальный путь, основываясь на своих расовых преимуществах и недостатках.



Из визуализации видно, что эльфы предпочитают лесные маршруты, гномы тяготеют к горным проходам, люди максимально используют дороги.

# 4. Анализ ответа

Полученные результаты полностью подтвердили выдвинутую гипотезу:

1. Гексагональная сетка действительно обеспечила более естественное представление движения по карте по сравнению с квадратной сеткой. Расстояние между любыми соседними клетками стало одинаковым, что упростило расчеты и сделало движение более реалистичным.

2. Система расовых модификаторов существенно повлияла на выбор оптимальных путей. Даже при одинаковых начальной и конечной точках, разные расы выбирали различные маршруты в соответствии со своими преимуществами:

- Люди предпочитали дороги и населенные пункты, даже если это увеличивало физическую длину пути.

- Эльфы тяготели к лесным тропам и равнинам, избегая гор и пещер.

- Гномы выбирали маршруты через горы и пещеры, где получали значительные бонусы к скорости.

- Орки не боялись пустынь и могли преодолевать даже лавовые участки с ограниченным штрафом.

3. Адаптированный алгоритм A\* успешно учитывал как физическое расстояние, так и стоимость прохождения различных типов местности для конкретной расы. Это позволило находить действительно оптимальные пути с точки зрения времени прохождения, а не просто кратчайшие по дистанции.

4. Производительность алгоритма оставалась высокой даже на больших картах и при сложных конфигурациях местности. Время поиска пути оставалось в пределах миллисекунд, что делает возможным использование алгоритма в режиме реального времени.

# 5. Выводы

На основании проведенной работы можно сделать следующие выводы:

1. Использование гексагональной сетки вместо квадратной значительно улучшает моделирование движения по карте. Равное расстояние между всеми соседними клетками делает перемещение более естественным и упрощает расчеты, особенно для диагональных движений.

2. Внедрение системы расовых модификаторов добавляет новый уровень реализма и стратегической глубины в поиск пути. Оптимальный маршрут теперь зависит не только от физического расстояния, но и от особенностей передвигающегося субъекта.

3. Алгоритм A\* при правильной адаптации успешно справляется с задачей поиска оптимального пути на гексагональной сетке с учетом расовых модификаторов. Его эвристическая природа позволяет эффективно находить решения даже на больших картах.

4. Реализованная система предоставляет широкие возможности для расширения и улучшения. В будущем возможно добавление новых рас, типов местности, динамических модификаторов (погода, время суток) и других факторов, влияющих на выбор пути.

5. Использование современных инструментов и методологий DevOps обеспечило высокое качество кода, надежность тестирования и возможность быстрого развертывания.

Таким образом, проект PathFinder успешно реализовал поставленные задачи и подтвердил выдвинутую гипотезу о преимуществах гексагональной сетки и расовых модификаторов при моделировании поиска пути.

# 6. Список литературы

1. Amit Patel, "Hexagonal Grids", Red Blob Games, 2013. Дата обращения: 02.05.2025. https://www.redblobgames.com/grids/hexagons/

2. Hart P.E., Nilsson N.J., Raphael B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions on Systems Science and Cybernetics, 1968.

3. Милнор Дж., Хьюзмоллер Д. "Симметрические бидифференциальные формы", Принстон, 1974.

4. Python Documentation, "heapq — Heap queue algorithm", Python Software Foundation. Дата обращения: 01.05.2025. https://docs.python.org/3/library/heapq.html

5. Matplotlib Documentation, "Matplotlib: Visualization with Python", Matplotlib Development Team. Дата обращения: 03.05.2025. https://matplotlib.org/

6. NumPy Documentation, "NumPy Reference", NumPy Team. Дата обращения: 10.05.2025. https://numpy.org/doc/stable/reference/

7. Docker Documentation, "Get Started with Docker", Docker Inc. Дата обращения: 10.05.2025. https://docs.docker.com/get-started/

8. Bugzilla Documentation, "The Bugzilla Guide", Mozilla Foundation. Дата обращения: 01.05.2025. https://www.bugzilla.org/docs/

9. Sphinx Documentation, "Sphinx documentation contents", Sphinx Team. Дата обращения: 01.05.2025. https://www.sphinx-doc.org/en/master/contents.html

10. GitHub Actions Documentation, "GitHub Actions documentation", GitHub Inc. Дата обращения: 01.05.2025. https://docs.github.com/en/actions