Программный проект: "Стена - это ты

Целью данного проекта является реализация небольшой графической головоломки, основным структурным блоком которой является алгоритм поиска пути через лабиринт.

Игра "*Wal l Is You*

Wall Is You1 - это новая интерпретация классического сценария, в котором игрок берет на себя роль отважного искателя приключений и должен войти в зловещее подземелье, чтобы победить злых существ внутри. В *Wall Is You* роли поменялись местами: игрок берет на себя роль доброжелательного обитателя подземелья, который должен помочь особо глупому искателю приключений избавить его от населяющих его монстров.

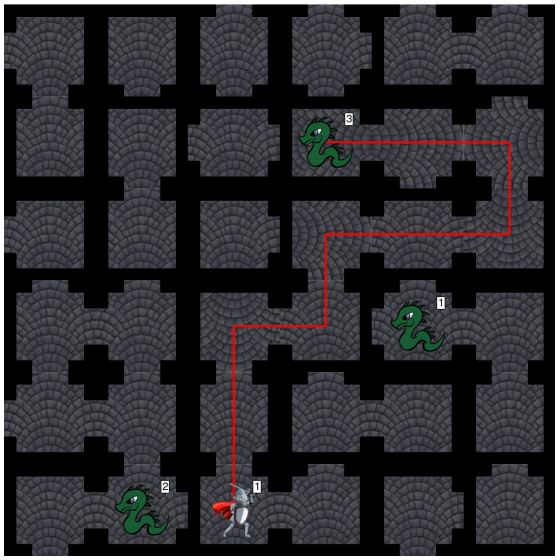


Рис. 1: Игра "*Wall Is You*" в процессе.

В игре чередуются раунды подземелий, управляемые игроком, и раунды приключений, управляемые программой:

Подземелье

- Подземелье представляет собой прямоугольную сетку *комнат*. В каждой комнате

может быть проход в каждом из четырех кардинальных направлений (вверх, вправо, вниз, влево).

[1] Название игры является отсылкой к игре-головоломке *Baba Is You*.

- В комнатах могут находиться *драконы*. У каждого дракона есть уровень, и все драконы имеют разные уровни.
- В свой ход игрок может нажать один или несколько раз на столько комнат, сколько он хочет, чтобы повернуть их. Каждый клик поворачивает комнату на 90° по часовой стрелке. Несколько нажатий подряд могут повернуть комнату на 180° или 270° или вернуть ее в исходное положение.

Рис. 2: Четыре возможных ориентации L-образной комнаты.

- Игрок указывает конец хода в подземелье, нажимая клавишу "Пробел".

Авантюрист

- *Авантюрист* находится в одной из комнат подземелья и начинает с уровня 1.
- Искатель приключений *горд* и всегда старается отправиться за самым сильным драконом, которого он может найти. В каждый момент времени искатель приключений обозначает свое *намерение* красной линией. Например, на рис. 1 намерение искателя приключений - сразиться с драконом 3-го уровня. Если дракон недоступен, у искателя приключений нет намерений.
- В свой ход искатель приключений перемещается из комнаты в комнату вдоль красной линии, пока не достигнет дракона. Если он встречает дракона равного или более низкого уровня, чем его собственный, он убивает его и получает один уровень. Если он встречает дракона более высокого уровня, то убивает его и *проигрывает* игру.
- Игра *выиграна*, когда все драконы убиты!

Реализация

Проект состоит из трех основных задач, все из которых являются обязательными. Необязательные улучшения отмечены количеством звезд (л), примерно соответствующим их сложности.

Задача 1: Разработка игрового движка

Первая задача проекта - запрограммировать внутреннюю логику игры (*модель*), то есть ту часть, которая позволяет представить состояние игры в соответствующих структурах данных, и модифицировать их для применения эффектов действий игрока и искателей приключений.

Эта очень большая задача делится на несколько подзадач.

Представление состояния игры

Здесь предложены структуры данных для представления состояния игры (подземелье, персонажи). Однако вы вольны изменить их, **если вы объясните и обоснуете свой выбор.**

Каждая **комната** в подземелье представлена **кортежем из 4 булевых чисел** (вверх, вправо, вниз, влево), указывающих, какие проходы возможны в комнате.

Подземелье представлено списком подземелий, состоящим из списков комнат. Размеры списков соответствуют размерам представленного подземелья. Значение `dungeon[i][j]` описывает комнату в строке **i** и столбце **j**.

Например, приведенное ниже подземелье представлено следующей структурой:

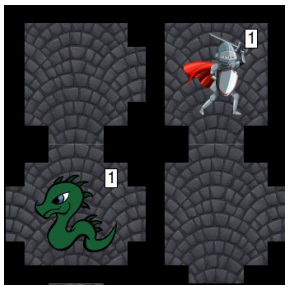


Рис. 3: Очень маленькое подземелье

```
dungeon = [[(False, True, True , False), (False, False, True,
           False)], [(True , True, False, True ), (True , True , True
           , True, False)]].
```

Таким образом, значение dungeon[1][0] представляет левую нижнюю комнату подземелья (ту, в которой находится дракон). Оно установлено в (True, True, False, True), чтобы указать, что комната позволяет проходить вверх, вправо и влево, но не вниз.

Каждый **персонаж** (искатель приключений или дракон) представлен словарем с ключами 'position' и 'level'. Позиция представлена парой координат (строка, столбец), а уровень - целым числом. Например, авантюрист на рис. 3 представлен следующим словарем:

```
авантюрист = {
                положение" : (0,1)
                уровень": 1
              }
```

Поскольку подземелье может содержать несколько драконов, словари, соответствующие каждому дракону, группируются в списке драконов.

Управление подземельем

Движок игры должен обеспечивать возможность вращения комнат подземелья и легко определять, связаны ли две комнаты между собой. Эти возможности будут обеспечиваться следующими двумя функциями:

- rotate(dungeon, position): функция изменяет список комнат подземелья, поворачивая комнату с координатами позиции на 90 градусов по часовой стрелке. Функция ничего не возвращает.
- connects(dungeon, position1, position2): функция возвращает True, если комнаты в списке подземелья в координатах position1 и position2 являются *смежными* и *соединенными* (проход из первой комнаты во вторую открыт, и наоборот), и False в противном случае.

Намерение авантюриста

Самой сложной частью игрового движка является вычисление *намерения* искателя приключений, т.е. пути, который он намерен пройти после завершения игры в подземелье. Для этого мы предлагаем написать функцию intention(dungeon, position, dragons), возвращающую путь (список последовательно соединенных позиций), ведущий искателя приключений к доступному дракону самого высокого уровня, или None, если ни один дракон не доступен.

Для написания такой функции можно использовать несколько подходов. Мы начнем с реализации наивного алгоритма поиска (называемого алгоритмом поиска с *обратным ходом*, или алгоритмом *глубокого поиска*), который, начиная с позиции в игре, будет перебирать все возможные связанные позиции и рекурсивно искать путь из каждой новой полученной позиции.

Для того чтобы алгоритм не вызывал бесконечных рекурсивных вызовов, мы убедимся, что каждая позиция посещается не более одного раза. Для этого мы будем записывать каждую посещенную позицию в набор visit и преждевременно прерывать рекурсивные вызовы на уже посещенных позициях.

Алгоритм работает следующим образом, начиная с позиции, заданной в качестве параметра, и изначально пустого множества visit :

1. Если позиция соответствует позиции дракона уровня, ответ ([position],level): найден тривиальный путь к дракону уровня.
2. Если позиция уже находится в посещении, ответьте Нет. Эта позиция уже рассматривалась, поиск прерывается.
3. Добавляется позиция в посещении.
4. Мы рекурсивно запускаем поиск на каждой целевой позиции, связанной с позицией, и восстанавливаем полученные таким образом пары (путь, уровень).
5. Если ни один поиск не дал пары (путь, уровень), мы делаем вывод, что нет дракона, доступного из данной позиции, и возвращаем None.
6. В противном случае мы получаем пару (путь, уровень) самого высокого уровня и возвращаем ([позиция] + путь, уровень): уровень - это уровень самого сильного доступного дракона, и путь [позиция] + путь к нему найден.

Башня искателей приключений

Во время своего хода авантюрист следует своему намерению и применяет последствия. Вот предложение функций для реализации:

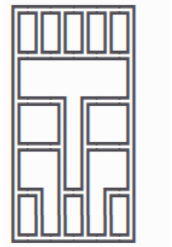
- encounter(adventurer, dragons): функция проверяет, находится ли авантюрист в одной позиции с драконом, и применяет последствия: если авантюрист встречает дракона равного или более низкого уровня, дракон убивается (и удаляется из списка), а авантюрист получает уровень. Если приключенец встречает дракона более высокого уровня, он погибает (решите, каким способом это изобразить).
- apply_path(adventurer, dragons, path): функция перемещает искателя приключений по пути и вызывает функцию встречи, когда это необходимо. Функция ничего не возвращает.
- end_game(adventurer, dragons): функция возвращает 1, если игра выиграна (все драконы убиты), -1, если игра проиграна (авантюрист убит), и 0, если игра продолжается.

Задача 2: Представление и загрузка подземелий

Цель этой задачи - позволить программе читать и загружать предопределенные подземелья, хранящиеся в файлах. Так, подземелье *Wall Is You* будет представлено текстовым файлом, состоящим из двух частей:

- Первая часть представляет собой план подземелья, нарисованный специальными символами
  ╨, ╟, ╔, ═, ╥ и их повороты. Например, символ ╔ обозначает комнату с проходами вправо и вниз, но не влево или вверх.
- Во второй части указаны координаты и уровни персонажей. Например, A 2 5 указывает, что искатель приключений находится в строке 2 и столбце 5, а D 2 0 2 указывает, что в комнате в строке 2 и столбце 0 находится дракон второго уровня. Уровень искателя приключений не указывается, так как он всегда начинает с уровня 1.

Пример файла с соответствующим подземельем приведен ниже:



```
A 0 2
D 1 1 1
D 2 0 2
D 2 5 3
```

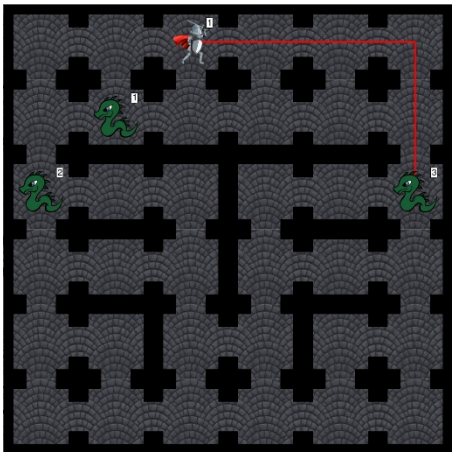


Рис. 4: Файл подземелья и соответствующее подземелье.

Ваша программа должна уметь читать файлы, записанные в указанном выше формате. В частности, вы напишите функцию load(file), которая принимает в качестве параметра путь к файлу и возвращает структуры данных, выбранные вами в задании 1, правильно заполненные информацией из файла. Функция вернет None, если предложенный файл отформатирован неправильно, например, если он содержит неожиданный символ.

Примечание: Сетка не обязательно должна быть квадратной, она может быть и прямоугольной.

Задача 3: Графический интерфейс

Третья задача - запрограммировать графический интерфейс игры (*вид*).

Ожидается эргономичный интерфейс. При запуске программы игрока должно встречать меню, позволяющее загрузить подземелье по своему выбору из набора доступных подземелий.

После запуска игры игрок может играть, нажимая на комнаты, которые нужно вращать. С каждым сделанным ходом программа будет обновлять намерения искателя приключений и соответствующим образом отображать их на экране. Затем игрок укажет конец своего хода, нажав клавишу "пробел", и программа воспроизведет и отобразит ход искателя приключений.

Игрок может перезапустить выбранное подземелье с самого начала клавишей R, а вернуться в меню для выбора другого подземелья клавишей Escape. В случае победы или поражения на экране должно появиться соответствующее сообщение, предлагающее игроку вернуться в меню.

Вся графическая часть проекта должна быть выполнена с использованием библиотеки fltk.

Дополнительные задачи

В этом разделе предлагаются улучшения проекта, к которым следует приступать только после **полного выполнения** всех обязательных задач.

(^^) Кратчайшие пути

Предложенный в задании 1 алгоритм вычисления намерений искателя приключений не обязательно приводит к кратчайшему пути к целевому дракону. Этот недостаток приводит к ситуациям, когда искатель приключений совершает непонятные обходные пути, как в приведенном ниже случае:

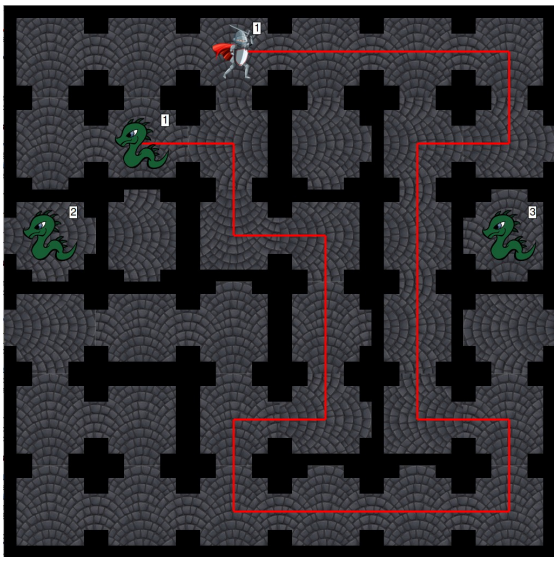


Рис. 5: Ненужное отвлечение внимания.

Предложите новый алгоритм для поиска кратчайшего пути к дракону высшего уровня *(подсказка: используйте очередь!)*

(^^) Сокровища

Представленная игра оставляет игроку очень мало выбора: нужно просто найти путь от искателя приключений к первому дракону, затем от первого дракона ко второму и т.д. Мы хотели бы обогатить игру, добавив *сокровища, руководствуясь* следующими правилами:

- В подземелье есть определенное количество сокровищ, установленное заранее.
- Во время своего хода подземелье может щелкнуть правой кнопкой мыши, чтобы поместить сокровище в незанятую комнату. В подземелье может быть только одно сокровище.
- Авантюрист жаден, его в первую очередь привлекают сокровища, даже если драконы доступны.
- Когда авантюрист находит сокровище, он забирает его (сокровище навсегда удаляется) и заканчивает свой ход.

Таким образом, сокровища являются инструментом для подземелья, позволяя ему заманить искателя приключений в выбранную им комнату и таким образом осуществить маршруты, которые в обычных условиях были бы невозможны.

В примере ниже игрок разместил сокровища, чтобы заманить искателя приключений в последнюю комнату в последнем ряду, а затем повернул эту комнату, чтобы заставить искателя сначала сразиться с драконом первого уровня. Легко понять, что это подземелье невозможно пройти без сокровищ.

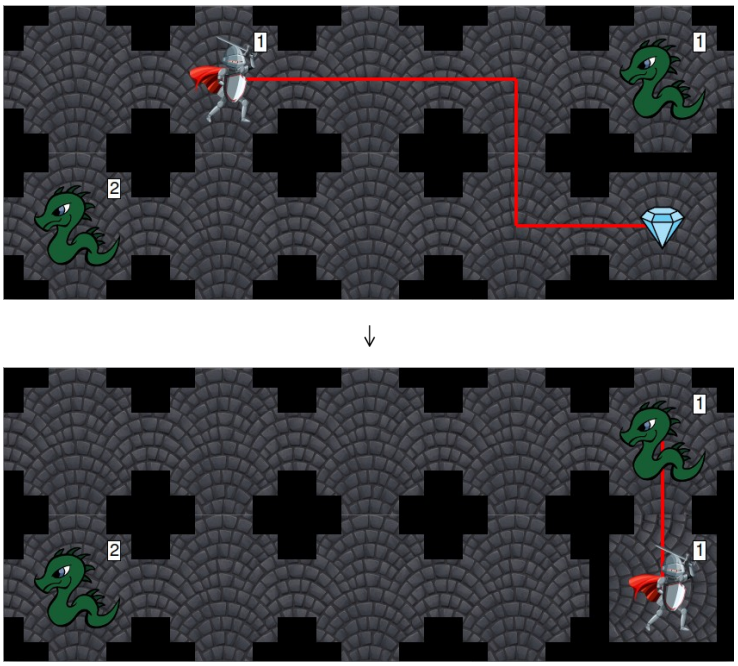


Рис. 6: Использование сокровища.

Это улучшение требует адаптации различных структур данных и предлагаемых алгоритмов, а также предоставления новых предопределенных подземелий с соответствующим количеством сокровищ.

Другие предлагаемые улучшения

Вот некоторые другие, менее подробные предложения по улучшению. Имейте в виду, что некоторые из них довольно легкие, а другие - довольно трудные. Не стесняйтесь спрашивать совета у своих учителей, прежде чем встать на какой-либо из этих путей.

- (*õ*) Позволяет игроку сохранять проходящую игру или загружать сохраненную игру. Запись должна быть сделана в файл, чтобы ее можно было извлечь в другой игровой сессии.
- (*õ*) Улучшить внешний вид игры: декорации меняются от одной комнаты к другой, убитые драконы оставляют труп на земле, искатель приключений меняет свой облик по мере повышения уровня...
- (*õõ)* Реализовать редактор подземелий, т.е. графический интерфейс, позволяющий пользователю выбрать размер подземелья, расположение комнат, и где разместить драконов и приключенцев. После завершения работы сетка будет сохранена в файл.
- (*õõ*) Реализовать режим игры "один ход": как только игрок прошел свой ход, он не может больше играть до конца игры, а авантюрист должен иметь возможность победить в одиночку. **Намерения авантюриста должны отображаться до конца игры, а не**
  только до первого встреченного дракона.

- (*xx*) Внедрить новые игровые механики, которые не привлекают искателя приключений и не прерывают его ход, но активируются, когда он проходит над ними. Например: рычаги для открытия или закрытия дверей, эликсир для победы над следующим драконом, даже если он более высокого уровня, и т.д. **Это улучшение требует выполнения задания "более короткие пути" и обеспечения подходящих и интересных подземелий.**

- (*õõ*) Выберите интересные правила передвижения для драконов и подходящий способ их визуального представления, так, как это задумано искателем приключений. **Это улучшение требует наличия подходящих подземелий и может быть отключено в любой момент.**
  томатически, когда игрок загружает "обычное" подземелье.

- (*õõõ*) Реализовать функцию "Подсказка", показывающую игроку возможный путь к следующему дракону, даже если комнаты в данный момент повернуты неправильно.

- (*õõõ*) Реализуйте генератор случайных уровней. Генерируемые уровни всегда должны иметь решение.

Любые другие улучшения возможны в соответствии с вашими идеями и желаниями, при условии, что вы предварительно обсудите их с одним из ваших преподавателей.