Создание игры Лабиринт на языке Python

Федотов Фёдор Алексеевич
9 класс, МАОУ СШ № 10 г. Павлово
Научный руководитель В.Г. Харитинов,
учитель информатики МАОУ СШ №10 г. Павлово

В данной работе рассматриваются способы создания игры Лабиринт на языке Python. Особенностью программного продукта является самогенерируемый лабиринт, который не повторяется от уровня к уровню. В написании программы, используется объектно-ориентированное программирование, создав класс Rooms, объект которого Room управляет всеми комнатами. Его методы это: генерация новой комнаты, переход на следующую комнату, на предыдущую и вывод комнаты в консоли для отладки.

Недавно я узнал о таком игровом жанре, как Roguelike, отличительными чертами которого является генерируемость и случайность уровней. Название данного жанра пошло от игры 1980 года «Rogue», особой темой которой является исследование подземелий. Играя в любимую игру, я задумался о проблеме: можно ли запрограммировать создание поля лабиринта на Python, чтобы это поле генерировалось случайным образом.

И я поставил перед собой цель: создать игру с бесконечно-генерируемыми комнатами с лабиринтами.

Актуальность и перспективы проектного продукта: в будущем возможно подключить к проекту нейросеть, которая бы могла управлять персонажем или окружением.

Желанием было сделать лабиринт, состоящий из «блоков», которые можно вписать в массив, и потом уже его выводить на экран, однако большинство разработанных алгоритмов для лабиринтов делали их из «палочек» (см рисунок 1), которые я не мог вписать в массив, не увеличив его во много раз

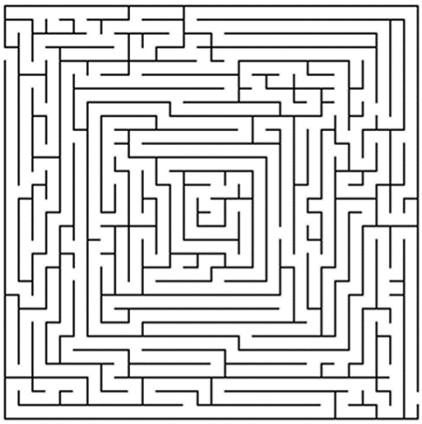

Рисунок 1 – пример тонкостенного лабиринта

В процессе долгих поисков, как построить то, что мне нужно, притом просто и понятно, я пришел к следующему выводу:

1. Заполнить массив (размеры должны быть нечетными, проверено путём проб и ошибок) блоками «стен»
2. Поставить в крайнем левом верхнем углу «каменщика», который построит стены.
3. Случайный ход на 2 клетки в одно из четырёх направлений
4. Если он выходит за границы лабиринта, то перейти к шагу 3, если на нужной клетке стена, то «пробить» эти две клетки, поставив там пустое пространство, если на нужной клетке пустая клетка, то перейти к ней.
5. Если все четные столбцы четных рядов являются пустыми клетками - конец.

Этот алгоритм имеет некоторые недочеты, но я сделал программу, используя его в чистом виде. На рисунке 2 один из вероятных лабиринтов.

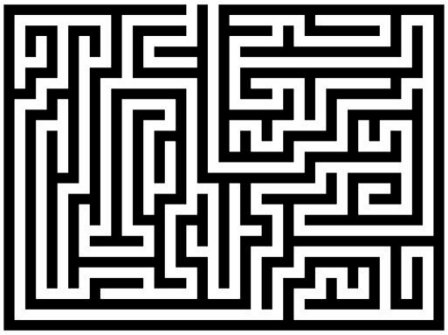

Рисунок 2 – пример толстостенного лабиринта по алгоритму

Одной из самых трудных для меня задач оказался вывод информации в текстовое поле. Модуль Tkinter в Python позволяет сделать программе интерфейс, используя такие виджеты как текст, текстовое поле, кнопка и т.д. Для программы больше всего подходит многострочное текстовое поле, в которое можно вписать построчно массив. Но сам массив вписать нельзя, т.к. выдаст не картинку, которую мы хотим, а целую кипу ненужных слоёв и символов. Но проблемой так же было правильно отобразить комнату, ведь символы обычно выводятся с конца. Для решения этой задачи я основательно продумал алгоритм, по которому символы выводились «задом на перед», и пришел к следующему его виду на Python:

```
def update_screen():
    main_screen.delete(1.0, t.END)
    for i in range(len(Room.active_room)):
        num = i + 1
        for j in range(len(Room.active_room[i])):
            num1 = j + 1
            main_screen.insert(1.0,
                               f'{Room.active_room[len(Room.active_room) - num] [len(Room.active_room[i]) - num1][0]}')
            if num1 == len(Room.active_room[i]):
                main_screen.insert(1.0, " ")
        main_screen.insert(1.0, f'\n')
```

(main_screen – объект класса Text, многострочного текстового поля, activeroom – переменная объекта Room, выводимая комната). Работает – безукоризненно, поэтому во всех версиях программы я его и использовал.

В написании программы, я использовал объектно-ориентированное программирование, создав класс Rooms, объект которого Room управлял бы всеми комнатами. Его методы это: генерация новой комнаты, переход на следующую комнату, на предыдущую и вывод комнаты в консоли, для отладки. Перемещение реализовано отдельной функцией Move, которая принимает объект и сторону, в которую его надо переместить. Так же существует механика, которая по мере прохождения увеличивает комнаты, и уменьшает количество *бонусов*. Что же за бонусы? Это «доллар», который дает +5 к счёту, и «сердце», увеличивающее время «жизни» игрока на 5 секунд. Переход в следующую комнату даёт +15 очков.

Одна из интересных механик игры это «Dash» - быстрое перемещение по прямому пути. Когда игрок зажимает одну из клавиш управления, то умение некоторое время «заряжается», поле чего персонаж очень быстро бежит в определённую сторону. Однако, для правильного использования этой *фичи* нужна некоторая практика, потому что надо уметь вписываться в определённые *тайминги*.

Для добавления бонусов в программу использовалось следующее правило: проверяется каждая вторая клетка на наличие стен около неё.

```
def Move(obj, side):
    global score
    y = obj.coords[0]
    x = obj.coords[1]
    moving = ((0, -1,),  # up
              (1, 0,),  # right
              (0, 1,),  # down
              (-1, 0,))  # left
    dx = moving[side][0]
    dy = moving[side][1]
    if TIME_BONUS in Room.active_room[y + dy][x + dx]:
        Room.active_room[y + dy][x + dx] = [FLOOR]
        Room.active_room[y + dy][x + dx].insert(0, Room.active_room[y][x].pop(0))
        obj.coords = [y + dy, x + dx]
        Clock.now += TIME_VALUE
        Clock.label['text'] = Clock.now

    elif SCORE_BONUS in Room.active_room[y + dy][x + dx]:
        Room.active_room[y + dy][x + dx] = [FLOOR]
        Room.active_room[y + dy][x + dx].insert(0, Room.active_room[y][x].pop(0))
        obj.coords = [y + dy, x + dx]
        score += SCORE_VALUE
        score_lbl['text'] = score

    elif PREV_DOOR_SPRITE in Room.active_room[y + dy][x + dx]:
        Room.prev_room()
        player.coords = [len(Room.active_room) - 2, len(Room.active_room[1]) - 2]

    elif NEXT_DOOR_SPRITE in Room.active_room[y + dy][x + dx]:
        if Room.active_room == Room.start_room:
            Clock.update_clock()
        if Room.how_many_rooms == len(Room.all_rooms) - 1:
            score += NROOM_VALUE
            score_lbl['text'] = score
        Room.next_room()
        player.coords = [1, 1]


    elif FLOOR_SPRITE in Room.active_room[y + dy][x + dx]:
        Room.active_room[y + dy][x + dx].insert(0, Room.active_room[y][x].pop(0))
        obj.coords = [y + dy, x + dx]

    Player.update_screen()
```

Рисунок 3 – фрагмент программы.

Если стен нет или всего одна стена, то с шансом *score_bonus_chance = количество_комнат//3* появляется бонус к очкам. Если стен больше, чем 1, то с шансом *time_bonus_chance = целое от(количество комнат/2)**2)* появляется бонус ко времени.

Во время долгого периода кодирования я долго «отлавливал» различные баги, например, неверный вывод массива мог привести к тому, что персонаж просто не показывался. Долго я не мог решить проблему «подбирания» бонусов: при попадании игрока на клетку с бонусом, игра уничтожала часть лабиринта, из-за чего программа аварийно завершала работу. А реализовать таймер было сложнее всего, потому что делал это в первый раз.

В процессе настройки баланса бонусов, чтобы их было не слишком много и не слишком мало, я анализировал обратную связь от одноклассников и знакомых, которым предлагал попробовать игру. На рисунках 4 и 5 показаны примеры работающей игры

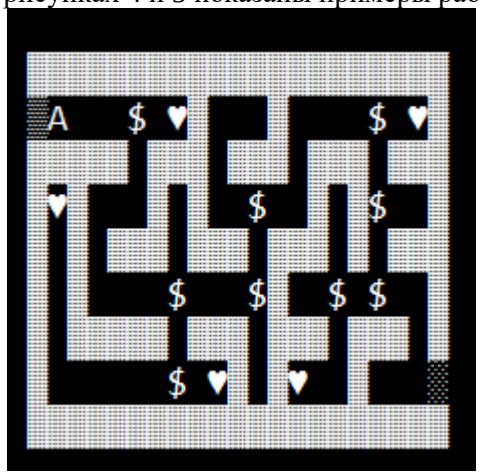

Рисунок 4 – Пример лабиринта и бонусы игры.

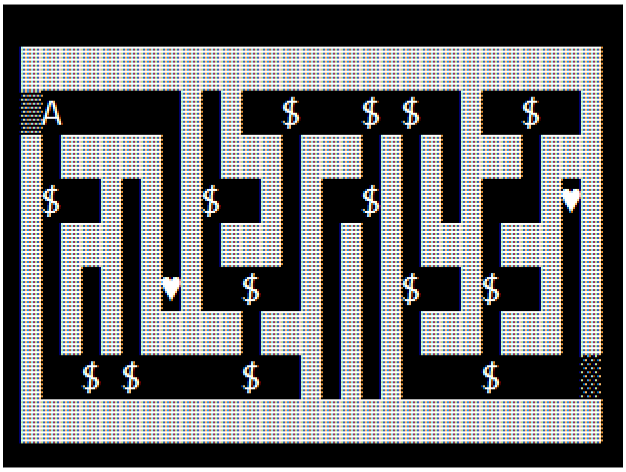

Рисунок 5 – Пример лабиринта более сложного уровня.

В ходе работы мы получили программный продукт, который самостоятельно случайным образом генерирует комнаты лабиринта. Данную работу можно в дальнейшем использовать для обучения нейросетей по поиску выхода из лабиринта.

Литература

Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с., ил. ISBN 978-5-93286-159-2

Tkinter - интерфейс Python к Tcl/Tk [Электронный ресурс] // Программирование на Python: сайт. URL: https://pythoner.name/documentation/library/tkinter (дата обращения: 14.03.2021)

Классы в Python [Электронный ресурс] // Python 3: сайт. URL: https://python-scripts.com/python-class (дата обращения: 14.03.2021)