Государственное учреждение образования

«Гимназия г. Наровли»

Проект

**Написание на языке программирования Python программы, наподобие игры Minecraft**

#### Направление «Программирование на языке Python»

Выполнил

Радкевич Станислав Васильевич,

учащийся XI класса

Наровля, 2023

**Оглавление**

Введение…………………………………………………………………………...3

Основная часть…………………………………………………………………….4

Заключение…………………..………………………………………………….....8

Список использованных источников………………………………………….....9

**Введение**

Python –  язык программирования с динамической типизацией и автоматическим управлением памятью. Язык является полностью объектно-ориентированным, а это значит, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Идея создания языка появилась в конце 1980-х годов, а разработка его реализации началась в 1989 году Гвидо ван Россумом.

По сравнению с другими языками Python намного проще в понимании и написании, так как функции в нём имеют названия от слов на английском. Это самый используемый язык в наше время. На нём пишут алгоритмы программ с машинным обучением и аналитические приложения. С помощью него обслуживают хранилища данных и облачные сервисы. Именно поэтому я выбрал этот язык для написания своего проекта.

Кто в наше время не слышал об игре Minecraft? До сих пор это самая продаваемая игра в мире. Она была разработана в 2009 году одним человеком – Маркусом Перссоном. В последствии игра обрела ошеломительный успех. В моём проекте я хотел повторить основу майнкрафта – возможность строить всё, что угодно.

**Основная часть**

Все игры песочницы начинаются с карты, поэтому первым файлом, который мы разберём, будет mapmanager.py. В нём, на основе файла .txt создаётся карта, с которой в дальнейшем будет взаимодействовать игрок.

def startNew(self):

    """создаёт основу для новой карты"""

    self.land = render.attachNewNode("Land") # узел, к которому привязаны все блоки карты

def clear(self):

      """обнуляет карту"""

      self.land.removeNode()

      self.startNew()

def addBlock(self, position):

      # создаём строительные блоки

      self.block = loader.loadModel(self.model)

      self.block.setTexture(loader.loadTexture(self.texture))

      self.block.setPos(position)

      self.color = self.getColor(int(position[2]))

      self.block.setColor(self.color)

      self.block.setTag("at", str(position))

      self.block.reparentTo(self.land)

def loadLand(self, filename):

      """создаёт карту земли из текстового файла, возвращает её размеры"""

      self.clear()

      with open(filename) as file:

          y = 0

          for line in file:

              x = 0

              line = line.split(' ')

              for z in line:

                  for z0 in range(int(z)+1):

                      block = self.addBlock((x, y, z0))

                  x += 1

              y += 1

      return x,y

Данные 4 функции создают всю карту. Первая функция создаёт узел рендера, к которому крепятся все блоки на карте. Вторая функция обнуляет карту после запуска программы, чтобы изменения на карте не сохранялись. Третья функция ставит блоки. Последняя функция генерирует карту, разбивая текстовый файл на список, по которому проходится и строит блоки. Высота блоков зависит от чисел в файле .txt. Чем больше число, тем выше блоки. Как и все отсчёты в Python, во время генерации карты высота блоков считается с нуля. Поэтому требуется к высоте добавлять 1.

def getColor(self, z):

      if z < len(self.colors):

          return self.colors[z]

      else:

          return self.colors[len(self.colors) - 1]

В данной функции происходит получение цвета в зависимости от высоты.

self.color = self.getColor(int(position[2]))

self.block.setColor(self.color)

Эти строки в def addBlock окрашивают блоки.

Далее рассмотрим файл hero.py.

key\_switch\_camera = 'c' # камера привязана к герою или нет

key\_switch\_mode = 'z' # можно проходить сквозь препятствия или нет

key\_forward = 'w'   # шаг вперёд (куда смотрит камера)

key\_back = 's'      # шаг назад

key\_left = 'a'      # шаг влево (вбок от камеры)

key\_right = 'd'     # шаг вправо

key\_up = 'e'      # шаг вверх

key\_down = 'q'     # шаг вниз

key\_turn\_left = 'n'     # поворот камеры направо (а мира - налево)

key\_turn\_right = 'm'    # поворот камеры налево (а мира - направо)

key\_build = 'b'     # построить блок перед собой

key\_destroy = 'v'   # разрушить блок перед собой

def accept\_events(self):

        base.accept(key\_turn\_left, self.turn\_left)

        base.accept(key\_turn\_left + '-repeat', self.turn\_left)

        base.accept(key\_turn\_right, self.turn\_right)

        base.accept(key\_turn\_right + '-repeat', self.turn\_right)

        base.accept(key\_forward, self.forward)

        base.accept(key\_forward + '-repeat', self.forward)

        base.accept(key\_back, self.back)

        base.accept(key\_back + '-repeat', self.back)

        base.accept(key\_left, self.left)

        base.accept(key\_left + '-repeat', self.left)

        base.accept(key\_right, self.right)

        base.accept(key\_right + '-repeat', self.right)

        base.accept(key\_switch\_camera, self.changeView)

        base.accept(key\_switch\_mode, self.changeMode)

        base.accept(key\_up, self.up)

        base.accept(key\_up + '-repeat', self.up)

        base.accept(key\_down, self.down)

        base.accept(key\_down + '-repeat', self.down)

        base.accept(key\_build, self.build)

        base.accept(key\_destroy, self.destroy)

В этих строках кнопки привязываются к действиям, таким как движение, поворот камеры, размещения блоков и смена режимов обзора и передвижения.

def cameraBind(self):

        base.disableMouse()

        base.camera.setH(180)

        base.camera.reparentTo(self.hero)

        base.camera.setPos(0, 0, 1.5)

        self.cameraOn = True

Эта функция привязывает камеру к управляемому герою.

def look\_at(self, angle):

        x\_from = round(self.hero.getX())

        y\_from = round(self.hero.getY())

        z\_from = round(self.hero.getZ())

        dx, dy = self.check\_dir(angle)

        x\_to = x\_from + dx

        y\_to = y\_from + dy

        return x\_to, y\_to, z\_from

def check\_dir(self,angle):

        if angle >= 0 and angle <= 20:

            return (0, -1)

        elif angle <= 65:

            return (1, -1)

        elif angle <= 110:

            return (1, 0)

        elif angle <= 155:

            return (1, 1)

        elif angle <= 200:

            return (0, 1)

        elif angle <= 245:

            return (-1, 1)

        elif angle <= 290:

            return (-1, 0)

        elif angle <= 335:

            return (-1, -1)

        else:

            return (0, -1)

Данные две функции определяют куда смотрит камера и решают в каком направлении двигаться или строить блоки.

Описание остальных функций находится в начале файла в комментариях. Названия функций там написаны вместе с key\_.

Последний рассматриваемый файл будет game.py

def \_\_init\_\_(self):

    ShowBase.\_\_init\_\_(self)

    self.land = Mapmanager()

    x,y = self.land.loadLand("land.txt")

    self.hero = Hero((x//2,y//2,2),self.land)

    base.camLens.setFov(90)

Это единственная функция в файле.

x,y = self.land.loadLand("land.txt")

В этой строке загружается карта из файла.

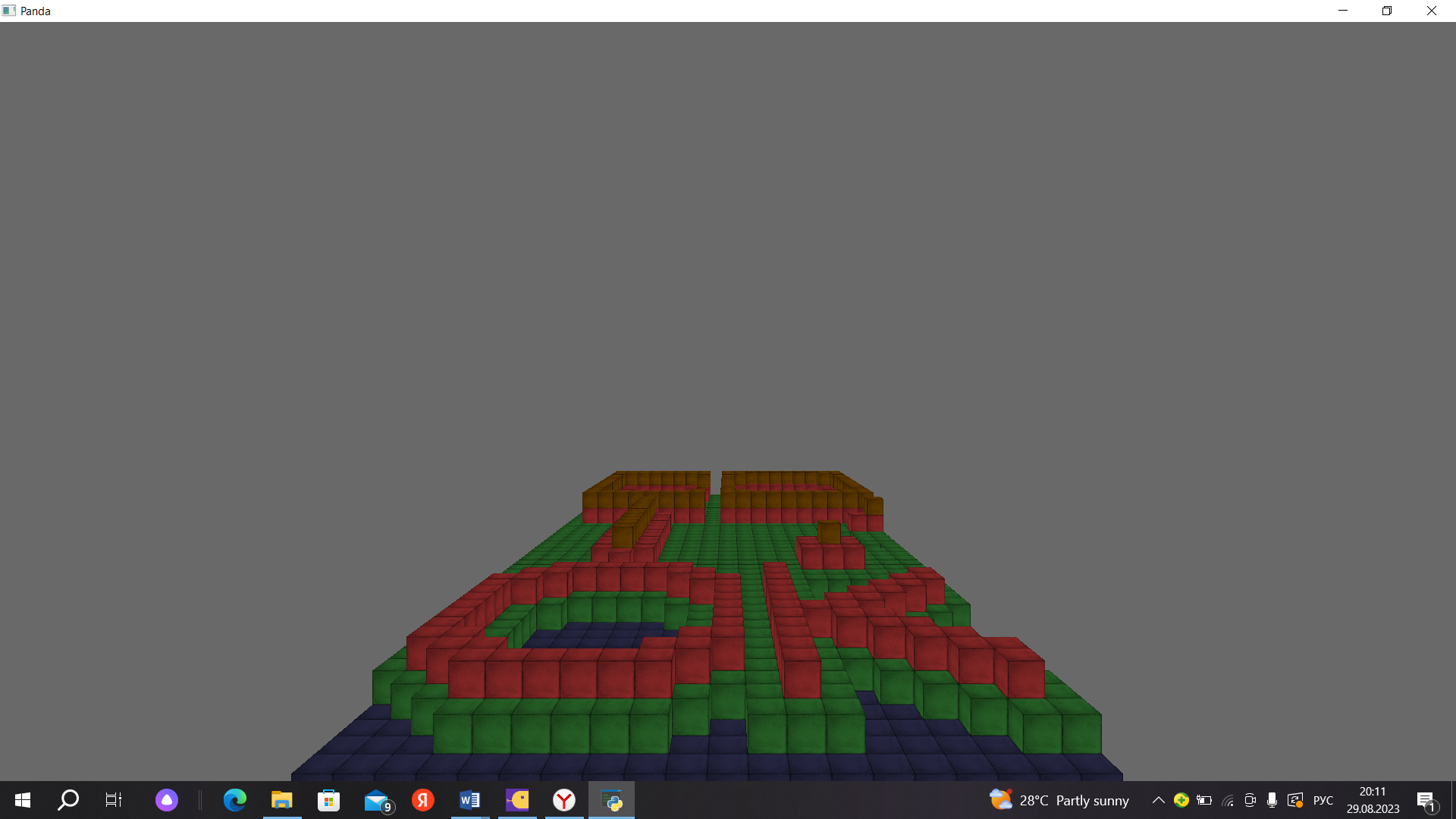
self.hero = Hero((x//2,y//2,2),self.land)

В этой строке устанавливается где появится игрок.

game = Game()

game.run()

Эти строки запускают всю игру.



Так выглядит финальный результат. При запуске game.py открывается окно с игрой.

**Заключение**

В моём проекте удалось создать игру, похожую по игровому процессу на Minecraft. Учитывая, что для создания игры требуется только идея и стремление, мой проект должен показать, что игру можно написать на любом языке программирования, и замотивировать людей быть уверенными и оригинальными.

**Список использованных источников**

1. Minecraft <https://ru.wikipedia.org/wiki/Minecraft> - Дата доступа 29.08.2023.
2. Образовательный курс «Алгоритмика».
3. Ответы на различные вопросы находились на сайте Stack OwerFlow <https://stackoverflow.com/questions> - Дата доступа 29.08.2023.