УДК 004

Создание 3D модели рельефа территории города Биробиджана на основе цифровых данных SRTM

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье приведён способ построения трёхмерной модели в blender с помощью языка программирования python и цифровых данных рельефа предоставляемой NASA. В результате исследования будут разобраны методы загрузки цифровой модели рельефа, построение полигональной решётки в blender с помощью python, создание визуально похожей и 3D модели высокой точности территории города Биробиджана.
Ключевые слова: blender, python, SRTM, api.

Creation of a 3D model of the terrain of the city of Birobidzhan based on digital SRTM data

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article provides a way to build a three-dimensional model in blender using the python programming language and digital terrain data provided by NASA. As a result of the research, the methods of loading a digital terrain model, building a polygonal grid in blender using python, creating a visually similar and high-precision 3D model of the territory of the city of Birobidzhan will be analyzed.
Keywords: blender, python, SRTM, api.

1 Введение

1.1 Актуальность

SRTM (Shuttle Radar Topography Mission) – это миссия, осуществленная NASA (Национальное управление по аэронавтике и исследованию космического пространства) в 2000 году с использованием шаттла Endeavour. Целью миссии было создание почти глобальной высокоточной цифровой модели рельефа Земли с использованием радиолокационной интерферометрии. Главный прибор являлся радиолокационный зонд, установленный на космическом шаттле.

SRTM собрала данные о рельефе поверхности Земли, используя радиоволны для измерения высоты точек на поверхности. Это позволило создать трехмерную модель рельефа, называемой цифровой моделью высот

(DEM). Данные включают всю территорию между 60° северной широты и 56° южной широты, что составляет около 80% всей суши Земли. DEM, полученная из SRTM, предоставляет информацию о высоте местности в различных точках и используется в различных областях, таких как география, геология, геодезия и геоинформационные системы. DEM от SRTM являются открытыми источниками данных, их можно использовать для многих приложений, включая картографирование, исследования окружающей среды, а также в различных отраслях, связанных с изучением земной поверхности.

1.2Обзор исследований

Е.В.Конопацкий, О.А.Чернышева, Я.А.Кокарева в своём исследовании смоделировали поверхность рельефа местности на основе спутниковых данных SRTM [1]. Ж.Эгамбердиев описал особенности геометрической и гидрологической коррекции глобальных цифровых моделей рельефа SRTM [2].]. А.А.Недоступ, А.О.Ражев исследовали имитационное моделирование работы радара рыболовного тренажерного комплекса с использованием цифровых моделей рельефа SRTM [3] М.Ю.Катаев, А.Г.Чугунов реализовали чтение, визуализацию и анализ трехмерной модели поверхности земли по данным SRTM [4]. А.В.Остроухов применил данные SRTM 4 при создании геоморфологической основы ландшафтно-инвентаризационных карт [5]. Цыдыпов Б.З., Б.З.Цыдыпов, И.А.Миронов, А.И.Куликов выявили опустыненные территории на основе комплексного анализа мультиспектральных и радарных спутниковых данных [6]

1.3Цель исследования

Цель исследования – используя цифровую модель рельефа SRTM построить высотную модель территории города Биробиджана в 3D редакторе blender используя только скрипт python.

2 Материалы и методы

Для реализации используются общедоступные файлы высот NASA, язык программирования python, среда построения трёхмерных моделей blender.

3 Результаты и обсуждение

Данные SRTM предоставляются в формате HGT (Height) файлов. Формат HGT представляют собой бинарные файлы, содержащие высотные данные для конкретного региона в виде высотных точек. Каждая высота представлена двумя байтами в формате знакового 16-битного целого числа. Значение высоты представляет собой количество метров над уровнем моря. Название файлов основано на координатах географической широты и долготы соответствующего региона.

Стандартное разрешение данных SRTM.1 составляет 1 дуговую секунду, что примерно равно 30 метрам. Это означает, что каждая точка сетки в HGT файле представляет собой значение высоты в пределах 30 метров. Таким образом, для области размером 1 градус широты и 1 градус долготы,

количество ячеек будет равно 3600x3600 (1 градус = 3600 дуговых секунд). Однако, чтобы учесть границы между ячейками и обеспечить полное покрытие области, добавляется еще одна строка и столбец, поэтому общий размер файла становится 3601x3601.

Посмотреть и загрузить визуальный ландшафт можно через web интерфейс геоинформационной системы NASA[9]. Территория города Биробиджана находиться в диапазоне координат 49 градусов северной широты и 132 градусов восточной долготы (рис. 1).

Рис. 1. Визуальное представление цифровых данных рельефа web сервиса NASA

Предоставляемые, таким способом, данные доступны как изображения в формате jpeg. (рис. 2).

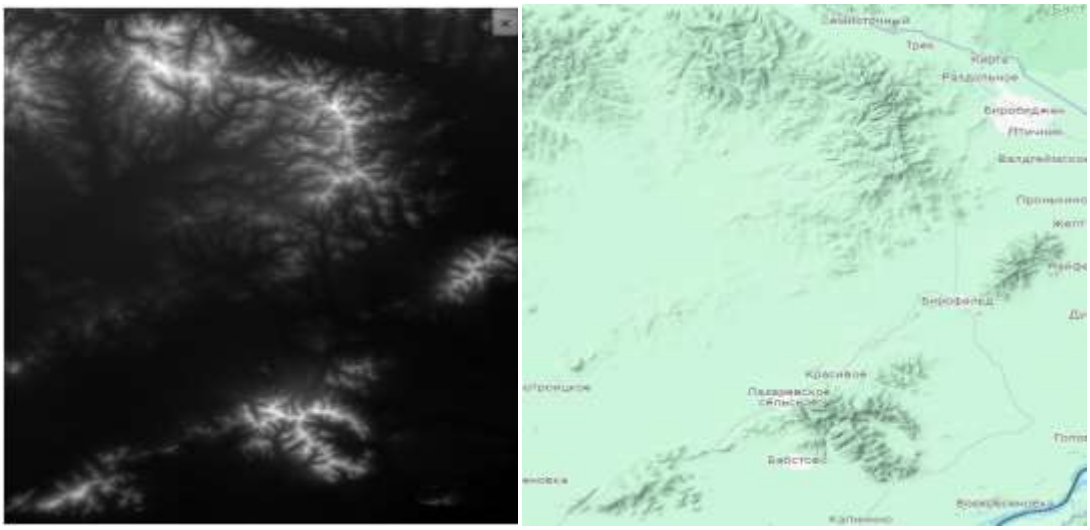

Рис. 2. Сравнение изображений рельефа SRTM и Google map

Такие данные, а также файлы HGT доступны в хранилище, называемом пул данных [10]. Пул данных обеспечивает прямой доступ к файлам продуктов данных через HTTPS. Искать нужный файл по координатам можно и вручную, но проще реализовать запрос с помощью языка программирования (рис. 3).

```python
import requests

# Функция для загрузки данных о высоте рельефа с помощью API SRTM NASA
def load_elevation_data(api_url):
    response = requests.get(api_url)
    elevation_data = response.content
    return elevation_data

# Координаты города Биробиджан
latitude = 48
longitude = 132

# URL API SRTM NASA для получения данных о высоте рельефа
api_url = f'https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/2000.02.11/N{latitude}E{longitude}.SRTMGL1.hgt.zip'

# Загрузка данных о высоте рельефа
elevation_data = load_elevation_data(api_url)

# Сохранение данных в файл
with open('elevation_data.zip', 'wb') as file:
    file.write(elevation_data)
```

Рис. 3. Загрузка высотных данных с помощью python

В арсенале языка python доступен модуль gdal, способный работать с большим количеством геоинформационных файлов, в том числе с файлом высот. (рис. 4).

```python
from osgeo import gdal
import matplotlib.pyplot as plt

# Открытие файла .hgt
dataset = gdal.Open('N48E132.hgt')

# Чтение данных
data = dataset.ReadAsArray()

# Построение графика
plt.imshow(data, cmap='terrain')
plt.colorbar()
plt.xlabel('X')
plt.ylabel('Y')
plt.title('График на основе файла .hgt')
plt.show()
```

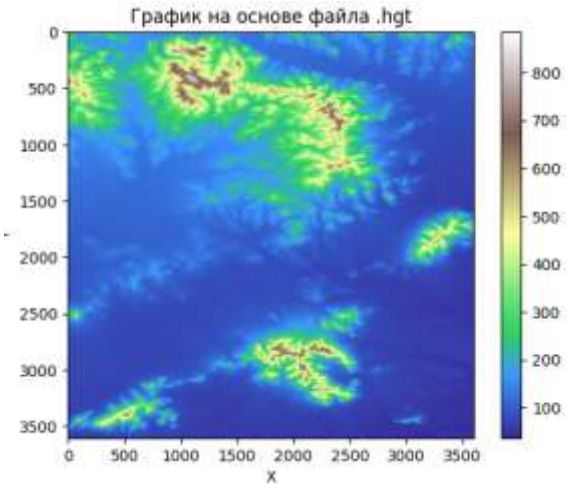

Рис. 4. Двумерный график, построенный на основе файла высот

Для построения трёхмерной модели территории города, нет необходимости брать все значения. Необходимо обрезать массив до приемлемых координат для визуальной видимости города (рис. 5).

```python
from osgeo import gdal
import matplotlib.pyplot as plt

# Открытие файла .hgt
dataset = gdal.Open('N48E132.hgt')

# Чтение данных
data = dataset.ReadAsArray()

# Вырезание данных
data = data[670:800, 3140:3280]

# Построение графика
plt.imshow(data, cmap='terrain')
plt.colorbar()
plt.xlabel('X')
plt.ylabel('Y')
plt.title('График на основе файла .hgt (вырезанный)')
plt.show()
```

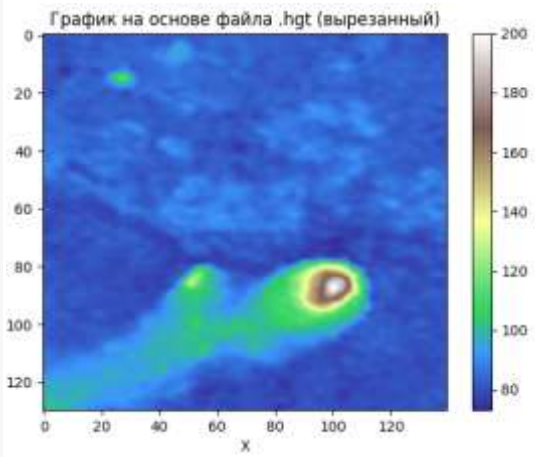

Рис. 5. Выделение данных рельефа территории города Биробиджана

Уже на этом этапе можно посмотреть, как будет выглядеть 3D модель. С помощью графических модулей python можно построить трёхмерную диаграмму. Так как каждая точка это значение в области 30 метров график получается зауженным (рис. 6).

```python
from osgeo import gdal
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Открытие файла .hgt
dataset = gdal.Open('N48E132.hgt')

# Чтение данных
data = dataset.ReadAsArray()

data = data[700:900, 3200:3450]
# Создание сетки координат
x = np.arange(0, data.shape[1])
y = np.arange(0, data.shape[0])
X, Y = np.meshgrid(x, y)

# Создание 3D-графика
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, data, cmap='terrain')

# Настройка осей
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Значение')

# Показать график
plt.show()
```

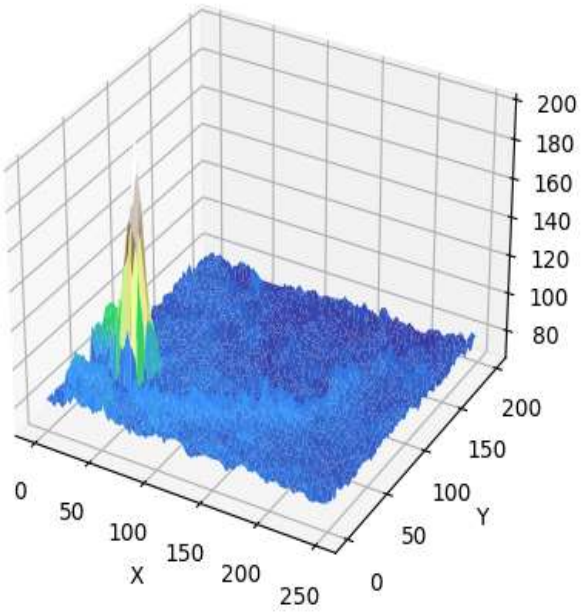

Рис. 6. Трехмерная диаграмма высоты рельефа

Для построения 3D модели в blender, помимо стандартных инструментов можно использовать python blender api. Этот интерфейс позволяет взаимодействовать с blender из своих собственных скриптов на языке Python. С помощью данного инструмента можно автоматизировать различные задачи, создавать и изменять объекты, применять материалы и текстуры, управлять камерами и источниками света, создавать анимации и многое другое.

Модуль bpy – это основной модуль python blender api, который предоставляет доступ к функциям и классам blender. С помощью данного модуля будет создан mash объекта.

В трехмерной компьютерной графике, mash – это трехмерная модель, состоящая из вершин (vertices), ребер (edges) и граней (faces). Мэш представляет собой полигональную решетку, которая определяет форму и структуру объекта.

Вершина (vertex) – это точка в трехмерном пространстве. Вершины определяют углы и узлы мэша. Они имеют позицию в пространстве и могут быть связаны с ребрами и гранями.

Ребро (edge) – это линия, соединяющая две вершины. Ребра определяют границы между вершинами и могут быть частью граней.

Грань (face) – это полигон, образованный тремя или более ребрами, которые замыкаются в цикл. Грани определяют поверхности объекта и могут иметь форму треугольника, четырехугольника или любого другого многоугольника.

Для начала необходимо подключить требуемые модули, загрузить и обработать данные. Написание программного кода можно осуществить только в среде blender, так как модуль bpy доступен только в ней, соответственно отладку программы можно осуществить только с доступными инструментами среды (рис. 7).

```
import bpy # Импорт основных модулей Blender Python API
from math import *
import numpy as np

filename = 'C:\\PATH\\N48E132.hgt'
data = np.fromfile(filename, np.dtype('>i2'))  # Загрузка данных как массив NumPy
data = data.reshape((3601, 3601))  # Изменение формы массива в соответствии с размерами файла .hgt

data = data[650:850, 3130:3400]

width,height=data.shape
```

Рис. 7. Подготовка модулей и данных

Далее нужно создать кортеж вершин и заполнить значениями высот по оси z. В качестве размера расстояния между точками выбран шаг 0,1 от местных координат blender. Грани будут строится по четырём точкам, это означает, что каждые четыре вершины будут объединены в плоскость четырёхугольника. Рёбра в данной реализации не понадобятся, на их место подставится пустой массив (рис. 8,9).

```
vertex = () # вершина
VERTEXES = () # кортеж вершин
# переменные, использующиеся для получения X и Y координат
x_start = 0
y_start = 0
x_end = width/10
y_end = height/10
x_step = 0.1
y_step = 0.1
# инициализация переменных - X/Y-координат регулярной полигональной решётки
x = x_start
y = y_start
while y < y_end:
    x = x_start # реинициализация переменной X, чтобы считать X-ряд заново
    while x < x_end:
        # Генерация X, Y, Z координат в виде трёхкомпонентного кортежа
        vertex = (x, y, data[int(x*10),int(y*10)]/10)
        # "наращиваем" кортеж новой вершиной
        VERTEXES += vertex, # обратите внимание на запятую, завершающую строку
        x += x_step # инкремент X-оси
    y += y_step # инкремент Y-оси     y += y_step # инкремент Y-оси
```

Рис. 8. Заполнение вершин

```
# Генерация "списка" четырёхугольных граней
#
face = () # кортеж индексов вершин одной грани
FACES = () # кортеж кортежей индексов всех граней
x_cnt = 0
y_cnt = 0
x_len = int((x_end - x_start) // x_step)
y_len = int((y_end - y_start) // y_step)
# Обход списка вершин
while y_cnt < len(VERTEXES) - x_len - 1: # цикл выбора индексов строк кроме
    # последней (самой верхней)
    x_cnt = 0 # реинициализация переменной x_cnt, чтобы считать X-ряд заново
    while x_cnt < x_len: # цикл выбора индексов строки кроме
        # крайнего правого ряда
        face = (x_cnt + y_cnt, # нижняя левая вершина
        x_cnt + y_cnt + 1, # нижняя правая вершина
        x_cnt + y_cnt + x_len + 2, # верхняя правая вершина
        x_cnt + y_cnt + x_len + 1) # верхняя левая вершина
        # "наращиваем" список граней кортежем индексов вершин грани
        FACES += face, # обратите внимание на запятую, завершающую строку
        x_cnt += 1 # инкремент индекса в пределах строки
    y_cnt += (x_len + 1) # инкремент индекса, кратный длине строки
```

Рис. 9. Заполнение граней

Теперь нужно создать сам объект. Для этого сначала нужно создать mash и привязать к новому объекту. Затем, полученный объект привязывается к сцене и заполняется расчётными данными (рис. 10).

```
# Создание mesh-объекта
#
# создаём полигональную решётку
mesh = bpy.data.meshes.new(name='NEW_MESH_OBJECT')
# создаём объект, связанный с созданной полигональной решёткой
mesh_obj = bpy.data.objects.new('NEW_MESH_OBJECT', mesh)
# связываем объект с текущей сценой, полученной из контекста
bpy.context.collection.objects.link(mesh_obj)
bpy.context.view_layer.objects.active = mesh_obj
# выделение объекта после его создания
bpy.context.active_object.select_set(state=True)
# Наполняем mesh-объект ранее сгенерированной геометрией
mesh.from_pydata(VERTEXES, [], FACES)
# обновляем mesh-объект согласно его новым данным
mesh.update()
```

Рис. 10. Создание объекта на основе полигональной решети

После обновления на сцене появится новый объект (рис. 11).

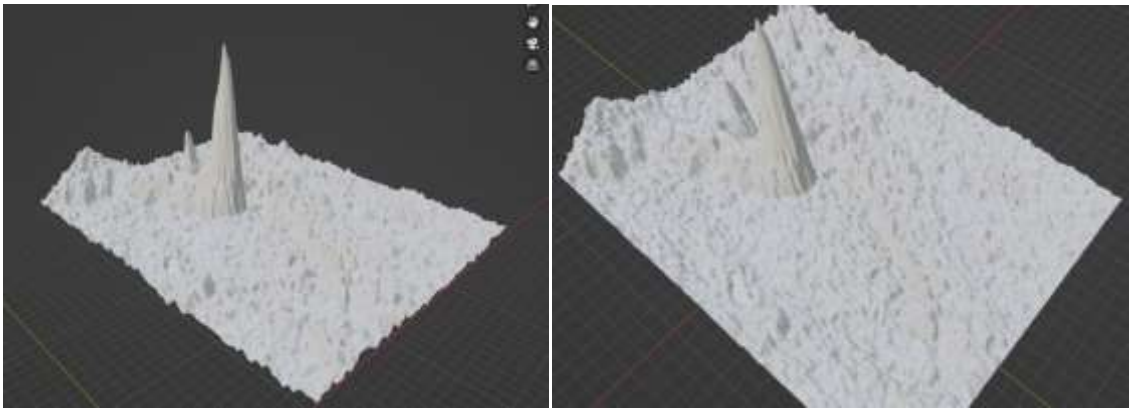

Рис. 11. 3D модель рельефа

В зависимости от требований можно изменить расстояние между точками по осям x и y для более правдоподобного результата. Однако, так как в данной местности незначительные перепады высот, расширение рекомендуется делать после создания объекта (рис. 12).

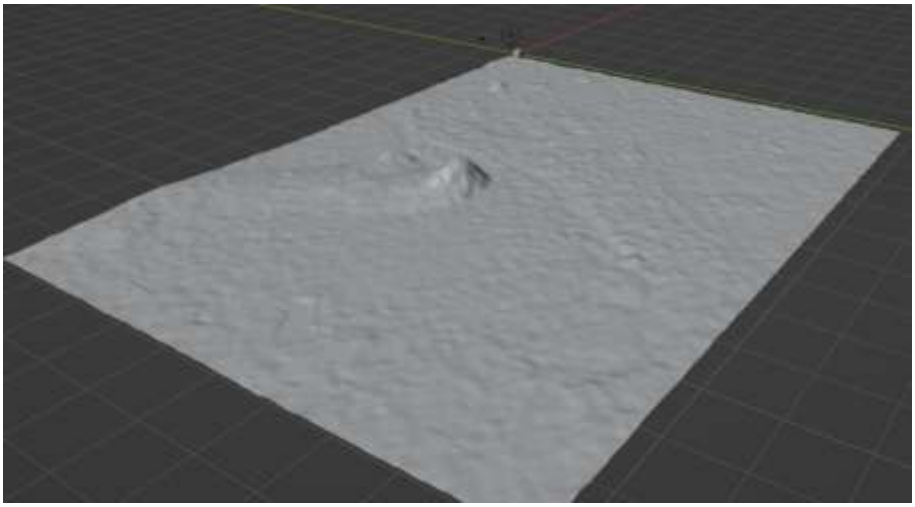

Рис. 12. 3D модель рельефа после расширения по осям x и y

Для полноты картины нахватает цветовой гамы. Используя python, создаётся новый материал, который генерирует цвет для каждой вершины по высоте (рис. 13,14).

```
import bpy

# Выбираем объект
obj = bpy.data.objects['NEW_MESH_OBJECT']

# Создаем новый материал
mat = bpy.data.materials.new(name="TM")
obj.data.materials.append(mat)

# Выбираем только что созданный материал
mat.use_nodes = True
nodes = mat.node_tree.nodes
links = mat.node_tree.links

# Создаем узлы шейдера
node_geo = nodes.new(type='ShaderNodeNewGeometry')
node_sep_xyz = nodes.new(type='ShaderNodeSeparateXYZ')
node_color_ramp = nodes.new(type='ShaderNodeValToRGB')
node_bsdf = nodes.get("Principled BSDF")

# Располагаем узлы
node_geo.location = (0, 300)
node_sep_xyz.location = (200, 300)
node_color_ramp.location = (400, 300)
node_bsdf.location = (600, 300)


# Устанавливаем цвета для ColorRamp
#node_color_ramp.color_ramp.elements.new(0.0)  # Добавляем первый уровень
node_color_ramp.color_ramp.elements[0].color = (0.0, 0.0, 1.0, 1.0)  # Синий

node_color_ramp.color_ramp.elements.new(0.25)  # Добавляем второй уровень
node_color_ramp.color_ramp.elements[1].color = (0.0, 1.0, 0.0, 1.0)  # Светло-зеленый

node_color_ramp.color_ramp.elements.new(0.4)  # Добавляем третий уровень
node_color_ramp.color_ramp.elements[2].color = (0.0, 0.75, 0.0, 1.0)  # Зеленый

#node_color_ramp.color_ramp.elements.new(0.75)  # Добавляем четвертый уровень
node_color_ramp.color_ramp.elements[3].color = (0.0, 0.5, 0.0, 1.0)  # Темно-зеленый

# Создаем связи между узлами
links.new(node_geo.outputs['Position'], node_sep_xyz.inputs[0])
links.new(node_sep_xyz.outputs['Z'], node_color_ramp.inputs['Fac'])
links.new(node_color_ramp.outputs['Color'], node_bsdf.inputs['Base Color'])
```

Рис. 13. Окраска вершин объекта по высоте

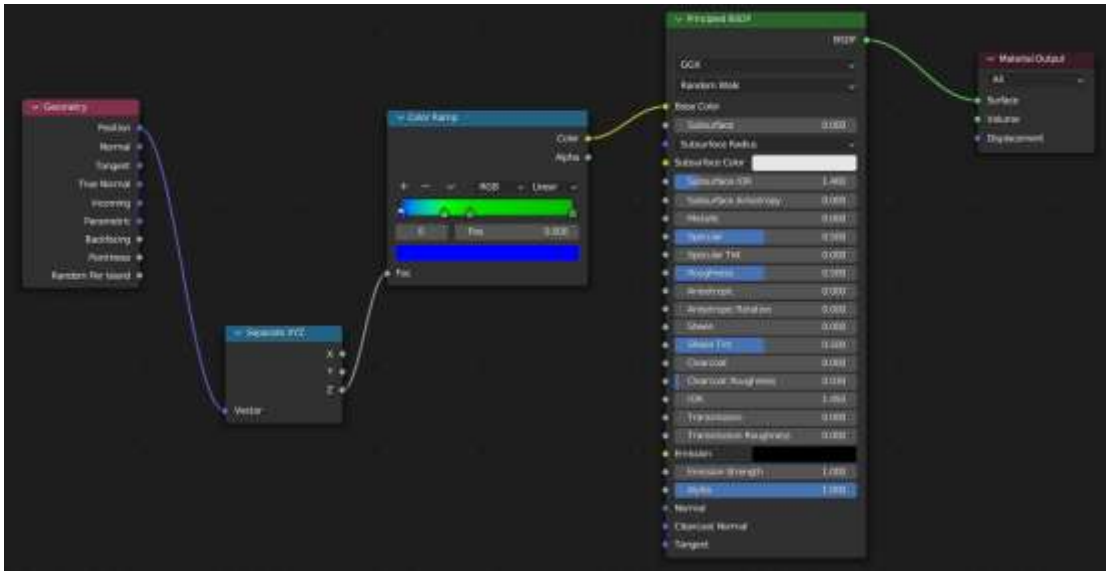

Рис. 14. Конструкция материала в окне «shader editor»

В результате получится отчётливая видимость участков суши и водоёмов (рис. 15,16).

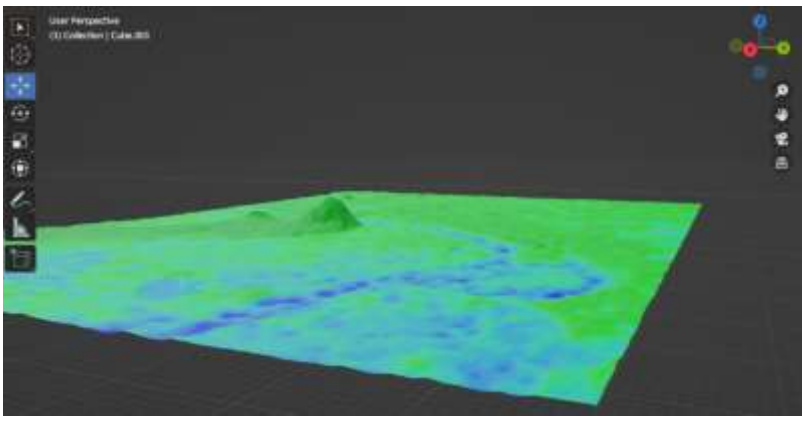

Рис. 15. Наложение материала на 3D модель рельефа, вид с боку

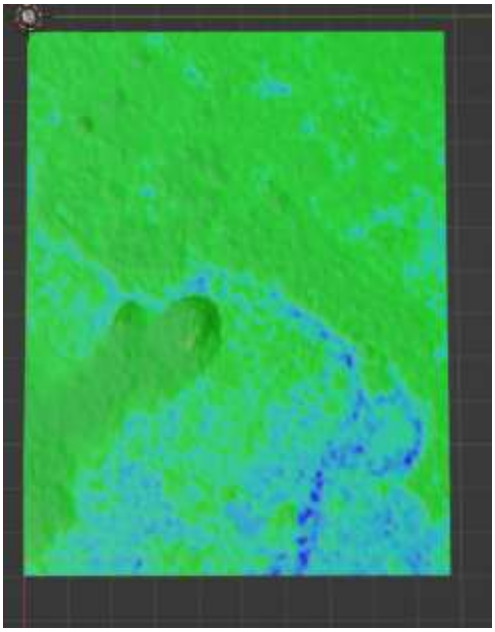

Рис. 16. Наложение материала на 3D модель рельефа, вид сверху

В данной статье было описано создание 3D модели в blender с помощью языка python. Рассмотрен способ загрузки и обработки данных высот. Построена полигональная решетка с использованием только языка python. Все приведённые методы можно объединить в одну программу для полной автоматизации построения моделей различных регионов.

Библиографический список

1. Конопацкий Е.В., Чернышева О.А., Кокарева Я.А. Моделирование поверхности рельефа местности на основе спутниковых данных SRTM// Вестник компьютерных и информационных технологий. 2019. № 6 (180). С. 23-31.
2. Эгамбердиев Ж. Особенности геометрической и гидрологической коррекции глобальных цифровых моделей рельефа SRTM // В сборнике: Избранные доклады 64-й университетской научно-технической конференции студентов и молодых ученых. сборник докладов. 2018. с. 906-907.

3. Недоступ А.А., Ражев А.О. Имитационное моделирование работы радара рыболовного тренажерного комплекса с использованием цифровых моделей рельефа SRTM // Морские интеллектуальные технологии. 2018. № 4-4 (42). С. 279-282.
4. Катаев М.Ю., Чугунов А.Г. Чтение, визуализация и анализ трехмерной модели поверхности земли по данным SRTM // Доклады Томского государственного университета систем управления и радиоэлектроники. 2006. № 6 (14). С. 38-41.
5. Остроухов А.В. Применение данных SRTM 4 при создании геоморфологической основы ландшафтно-инвентаризационных карт // В сборнике: Региональный отклик окружающей среды на глобальные изменения в Северо-Восточной и Центральной Азии. материалы Международной научной конференции. Российская академия наук. Сибирской отделение. Институт географии им. В.Б. Сочавы, Русское географическое общество. Восточно-Сибирское отделение. 2012. С. 168-170.
6. Цыдыпов Б.З., Миронов И.А., Куликов А.И. Выявление опустыненных территорий на основе комплексного анализа мультиспектральных (LANDSAT) и радарных (SRTM) спутниковых данных // Вестник Иркутского государственного технического университета. 2012. № 4 (63). С. 67-74.
7. Старостин Д.П. Метод восстановления отсутствующих данных SRTM // В сборнике: Наука молодых - будущее России. сборник научных статей 7-й Международной научной конференции перспективных разработок молодых ученых. Курск, 2022. С. 265-267.
8. Васильев И.Л., Стародумов О.И., Стародумов И.О. Решение кинематического уравнения одномерной модели стока реки с данными SRTM-топографии // Наука и новые технологии. 2010. № 2. С. 6-26.
9. Цифровая модель рельефа NASA // https://search.earthdata.nasa.gov/search (дата обращения: 28.12.2023)
10. Пул данных онлайн-хранилищ LP DAAC // https://lpdaac.usgs.gov/tools/data-pool/ (дата обращения: 28.12.2023)