“Visualization application development in virtual reality systems"

Andreas Theofilopoulos, Vasilis Mylonas

Bachelor's thesis

Supervisor: Fountos Ioannis

Ioannina, February 2022

DEPARTMENT OF ENGINEERING COMPUTER & INFORMATION UNIVERSITY OF IOANNINE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY OF IOANNINA

Thanks

For the successful completion of this work, we would like to thank him
our supervising professor, Mr. Ioannis Fundo, for his help and guidance throughout the duration of the work.

February 2022

Andreas Theofilopoulos, Vasilis Mylonas

Summary

It is a visualization application of the city of Ioannina and the surrounding areas. Specifically, the island of Ioannina is included, the same Ioannina, as well as the university of Ioannina, the village of Ligiades as well and the surroundings of the lake of Ioannina.

Various weather and environmental parameters from a specific file type (geotiff). Specifically: the vegetation indices (NDVI ARVI), percentage of water vapor (water vapour), probability of snowfall or snowy field (NDSI), altimetrics various and longitude and latitude in degrees - minutes - seconds. Still there is a compass orientation bar as well as a Mini map which helps to visualize the wider area in which each is located moment the user.

Finally, there are some functions which are activated with the pressing some predefined keys. Initially for each indicator the 3D visualization of an area around the user is provided so that he can see the information around him better. As an extension of this function we have implement an extension in which when the user acts on its function 3D visualization can select and see a percentage of the information that exists around it, both for maximum and minimum values.

Keywords: Ioannina, Ioannina Island, Ligiades, geotiff, Virtual Reality, NDVI, ARVI, NDSI, water vapour.

Abstract

Our bachelor's thesis is about a virtual visualization of the city of Ioannina and the surrounding areas. More specifically the areas of the small island of the lake, the lake itself, the mountainous village "Ligkiades" and the small villages surrounding Ioannina and the lake.

For the areas mentioned above, we draw various weather and environmental parameters from a specific file type (geotiff). More specifically: vegetation indices (NDVI, ARVI), water vapor percentage (water vapor), probability of snow and snow coverage (NDSI), elevation and latitude/longitude in decimal degrees. Moreover, there is compass bar as well as a mini map that helps in the visualization of the surrounding area of the user.

Finally, there are functions triggered by hotkeys, starting with those that provide us with a 3D visualization of the geotiff indices mentioned above.

Keywords: Ioannina, Ioannina Lake, Island of Ioannina Lake, Ligkiades, geotiff, Virtual Reality, NDVI, ARVI, NDSI, water vapor.

Contents

Chapter 1. Introduction

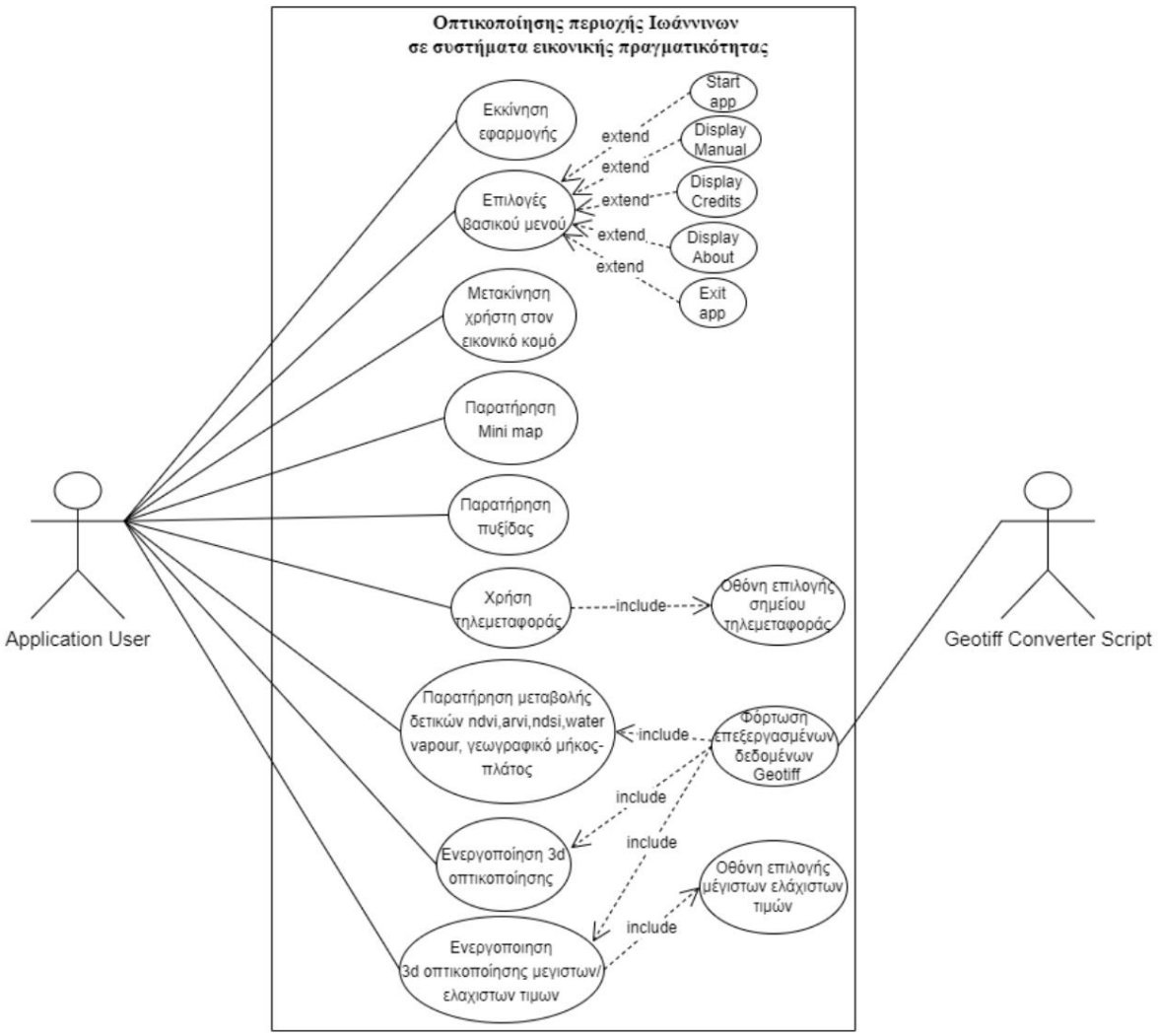
1.1 Target

This work aims to render, as realistically as possible,
a 3D representation of the area of Ioannina and the surrounding areas, as well
and the representation of weather and environmental data. The data
relate to actual measurements and properties for a specific moment in time
day. For this reason, the data is extracted from a specific
file type called Geotiff.

For the best representation of the area of Ioannina and the surrounding area
regions we will use Unity. In addition, a tool was created
(Asset) which vegetation database can create a forest which
can be placed in any area.

Geottif files are managed using the Python language.
The choice of language is not accidental since it is suitable for fast
retrieval of the data as well as its easy connection with the language that
uses Unity (c#). With Python we create an appropriate type
file (format) so that when starting the application we can
load, in the best possible way, onto the application the multitude of
data that are necessary for its operation.

We have created a Uml use case diagram for better understanding
of the application's goals (Figure Uml1)



Uml Image1. Uml use case diagram.

1.2 Application Capabilities

1.2.1 Interactive tour of the Ioannina area

It is an interactive tour where the user can move freely in an open area (open world). In this process the user has the ability to choose its starting point. There are 3 starting points the city of Ioannina, the islet of Ioannina and Ligiades.

After this selection the user has the opportunity to watch his single a 3D representation of all of the above, incl meaning all the local and physical elements around it, such as 3d rendering standing of water, trees, mountains, buildings and boats.

In addition, a special 3D visualization of the sky has been added, which makes the result even more realistic and relaxing for the user's eye thanks to special sky effects which include a different lighting that it makes every piece on the area more realistic.

For easy orientation, a compass has been added to help him useful at any time to be able to direct and find his destination easy and fast. A small illustration of the surroundings has also been created area of the user which is called Mini map. Mini map still helps more user oriented so they can still understand more easily located at any time.

1.2.2 3D Visualization of data

After selecting the location the user as he moves has it ability to see the data provided by the application in tokens bars in the area of Ioannina. As an extension of this we created 3D visualization of these data in 2 modes.

At any time the user has the option with a predefined button to activate the first 3d visualization of the data concerning 400 data in its surrounding area (20x20 Square Grid around the user).

Then, while the first mode is active, it has the option with pressing another button will activate the second Mode. The second Mode gives the user the possibility to choose for these 400 data a percentage of these, either of the larger or smaller values and to represent them another form so that they are clearly visible in the background.

During any Mode that is open is enabled switching from one to another, as well as the ability to change the type of data displayed at any time, for example to change the visualization of vegetation in visualization of water vapor (Chapter **3.2.2).**

Chapter 2. Theoretical background

and

development tools

2.1 Introduction to 3D graphics

By the term 3D (three-dimensional) graphics, we refer to graphics that they use a three-space representation of geometric data dimensions (often Cartesian). Virtual reality uses e computers to create and simulate the real and the non-existent environments that the user has the illusion of being surrounded by and in which can move freely, interacting with the objects which include as it would in the real world.

2.2 Unity Engine

Unity is one of the most well-known game engines [1]. First time, was released in 2005 by the company Apple Inc, which then collaborated with Unity Technologies [2]. Later, it was expanded and now it is available for 25 different platforms including Windows, Linux, Android Windows, Steam VR [3]. Unity can be used for 2D as well as 3D games and simulations as well as for virtual reality applications. Unity can use C# scripts, which are processed easily due to Visual Studio integration [4]. Her work environment Unity is shown below in Figure 1.

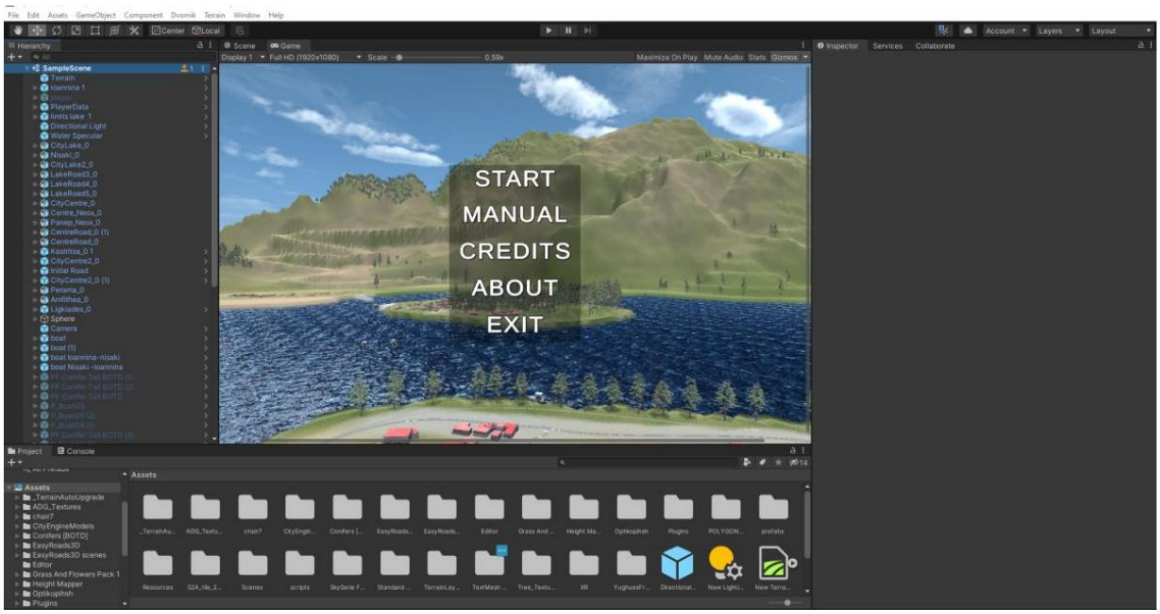


Figure 1. The Unity interface version 2020.3.0f1.

2.2.1 Asset Store

The Unity Asset Store (Figure 2) is a library built into the platform of Unity and first presented in 2010 [5]. The asset store offers various assets, which are data and models used in a Unity project and were created by Unity Technologies and also by users of the application [6].

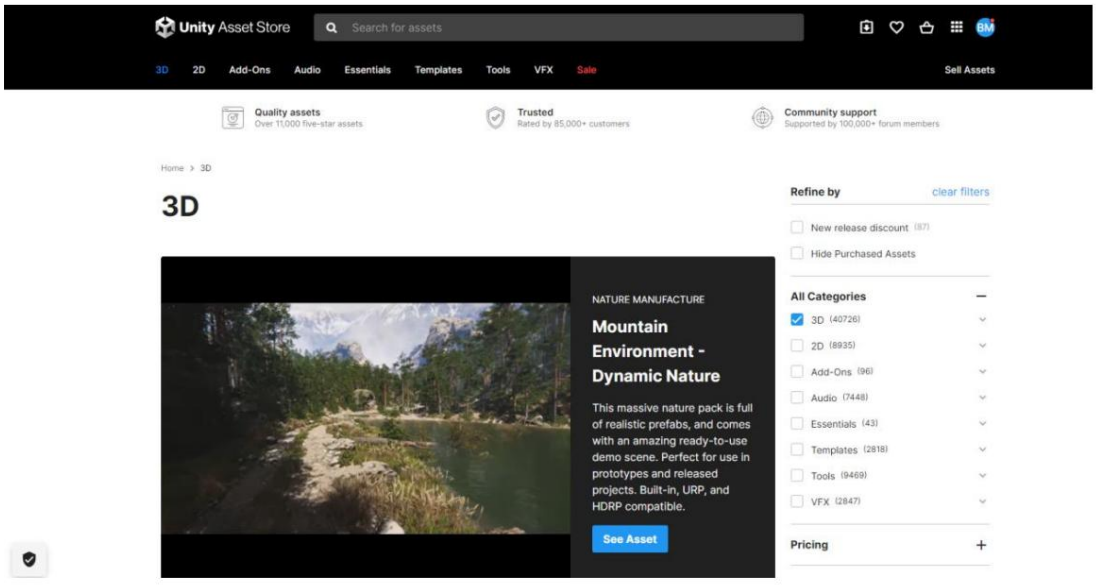


Figure 2. Asset Store home page.

2.2.2 Introduction to the basics of Unity

In this chapter we will break down the basics of Unity. Specifically we will analyze the Unity object, Inspector, and Unity Standard Asset.

2.2.2.1 Unity Game Object

A Unity Game Object is any object that exists in a project. By itself it has no functionality but the feature is offered adding components to it [7]. A component is a tool that enables various operations on a Game Object such as conversion (resizing, scaling, and rotation), adding sound and many more [8]. A component it can also be a Script (c# file) which we insert into the Object to perform an operation (Figure 3).

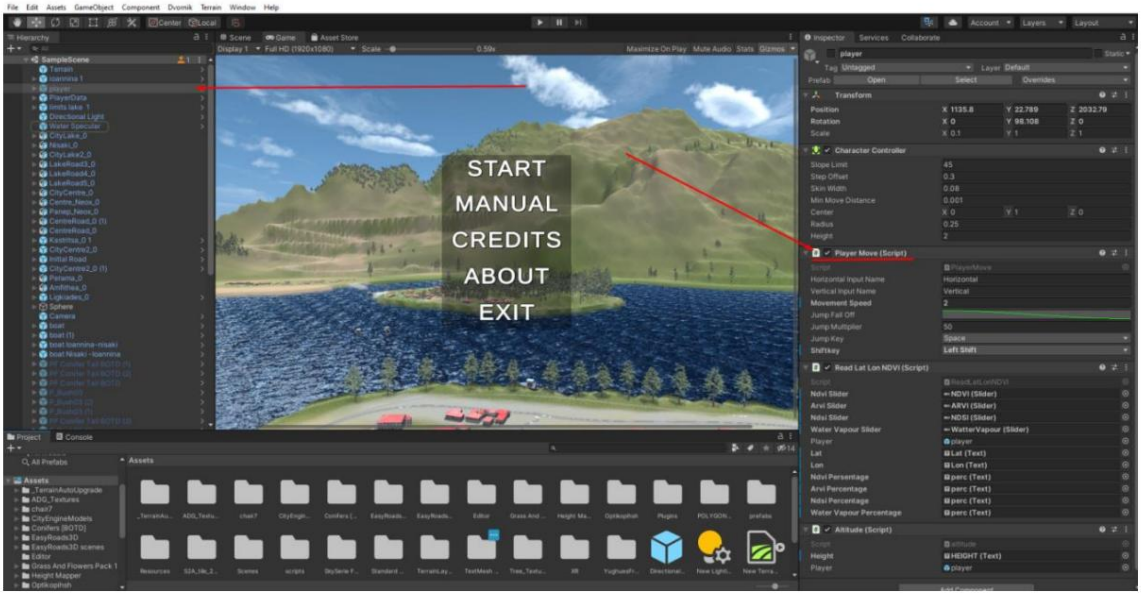


Figure 3. Example of an Object (player) containing a component (player move c# Script).

2.2.2.2 Inspector

The Inspector displays detailed information about the current selected GameObject, including all attached items and its properties. It gives the possibility to modify its functionality GameObjects on stage. Any property displayed in the Inspector can to be modified directly [9].

For example (Figure 4) one of the Object Player components is the Script Player Move. Observing the Inspector we can distinguish 7 fields which are modifiable, such as changing the key of the player's bounce, his speed, even the way in which the his bounce. Specifically for Script components any field we declare public it will appear in the Inspector of unity for this reason the public fields are much more limited than in other languages programming.

Figure 4. Example of Inspector operation.

2.2.2.3 Unity Standard Assets

The Unity platform includes various assets which are used more by users. Some of them are the camera, effects, environment and various functionalities. Standard assets are available free from the unity asset store [10].

Terrain (Height mapper)

For the creation of the area (from now on we will refer to this as terrain) we used an asset called height mapper. The specific Asset giving it as Input a Geotiff file creates us one terrain exactly as it actually is in the area we chose (Janina). The third person will be placed on this area later Character object [11].

First Person Character

In a third person game, the user interacts with the application by using a character/avatar. This is usually done using the asset Third Person Character offered in Standard Assets. Includes sounds for the character's steps, a 3D model, the animation clips according to the which are made the movements of the avatar and the scripts necessary for various functions. After downloading a third person character we changed basic things in the code like for example we added the feature pressing shift increases its speed. For this Object we did not maintain some folder in the thesis files since it wasn't essential.

Natural elements in Terrain

To import trees into Terrain we imported the asset Conifers[BOTD] [12]. For the import of various flowers or bushes the Grass and Flowers asset Pack 1 [13]. To import the lake we found a ready-made Object called Water Specular, offered for free by a member on the Unity forum.

Terrain layers

For the most correct representation of the terrain as well as to be visually more beautiful we used the Asset Terrain sample asset pack which adds to the terrain layers [14] ie special textures that make the result more realistic[15].

3d visualization of buildings

For the 3d Visualization of the buildings of Ioannina and the surrounding areas we used the CityEngine program. CityEngine is advanced lo 3D modeling software for creating massive, interactive and immersive urban environments in less time than traditional ones modeling techniques.

The buildings we created using CityEngine are based on real world GIS data. CityEngine made it possible for us to create the city of Ioannina directly without having to model each building separately. The program is paid but due to student status we used the free version provided from the company.

Transport boats to the island of Ioannina

There could not be missing the ships that carry people for it islet of Ioannina. For this reason we added an asset that gave us the 3d model which we processed appropriately to resemble the familiar ones boats going to and from the island of Ioannina (Figure 5).

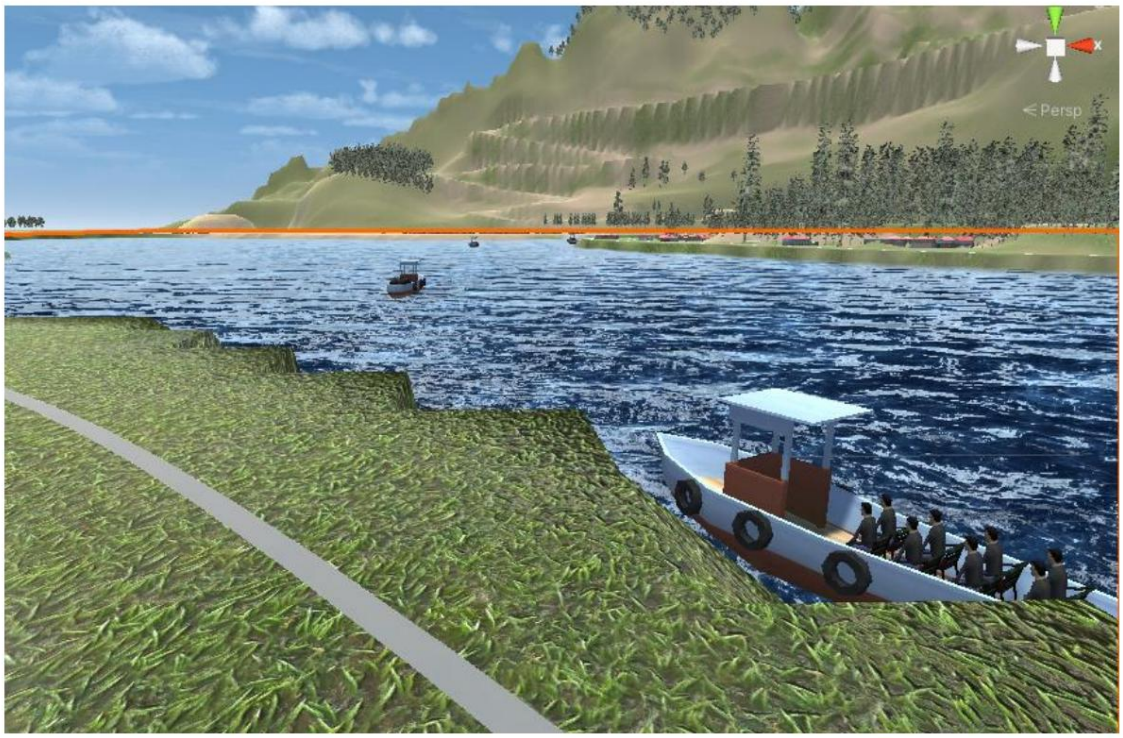


Figure 5. Transport ships on the island of Ioannina.

Roads

To add the roads we initially used an asset which called easyRoads3D combined with the roads created for us by City Engine mentioned above [16].

Heaven

Unity by default has some options for the sky which it doesn't they make a realistic effect as well for the lighting, which for the user it's the sun For this reason we used an asset to change it lighting and the sky called Sky freebie which makes the total Visualization more realistic [17].

2.3 Geotiff data

A GeoTIFF file extension contains geographic metadata that describe the actual location in space that each pixel represents in a image. When creating a GeoTIFF file, the spatial information are included in the file. tif [18] as embedded tags, which can include raster image metadata such as:

- horizontal and vertical data
- spatial extent, i.e. the area covered by the data set
- the coordinate reference system (CRS) used for the data storage
- spatial resolution, measured in the number of independent pixel values per unit of length
- the number of layers in the .tif file
- ellipsoid and geoid - estimated models of the shape of the Earth
- mathematical rules for map projection to convert data for one 3D space on a 2D screen

.TIFF files retain their full quality when compressed for transport and then rebuilt into an application. For things like aerial surveys with geospatial data that must remain intact no matter how many times they have been compressed, copied, edited and uploaded in different applications, .TIFFs are the perfect container. Unfortunately, this it also means that .TIFFs create extremely large files, making them less than ideal for display on a web page or for transfer via email [18].

2.3.1 Types of Geotiff

The GeoTIFF file format is widely used worldwide. The DAACs of NASA provide data in GeoTIFF format as do other providers of NASA Earth science data. There is strong software support in the form of the open source library libgeotiff and the Geospatial package Data Abstraction Library (GDAL). Many commercial GIS software products and spatial data analysis support data reading and writing GeoTIFF.

The two main types of Geotiff data are raster and vector data. Raster data is stored as a grid of rendered values on a map as pixels. Each pixel value represents an area in surface of the Earth.

Raster data is any pixelated (or gridded) data (Figure 6) where each pixel is associated with a specific geographic location. The price of one pixel can be continuous (e.g. elevation) or categorical (e.g. land use). He is the way we represent any digital image. A Geotiff raster only differs from a digital photo in that it comes with spatial information that links the data to a specific location. This includes the extent and size of the raster cell, the number of rows and columns and its coordinate reference system (or CRS) [19].

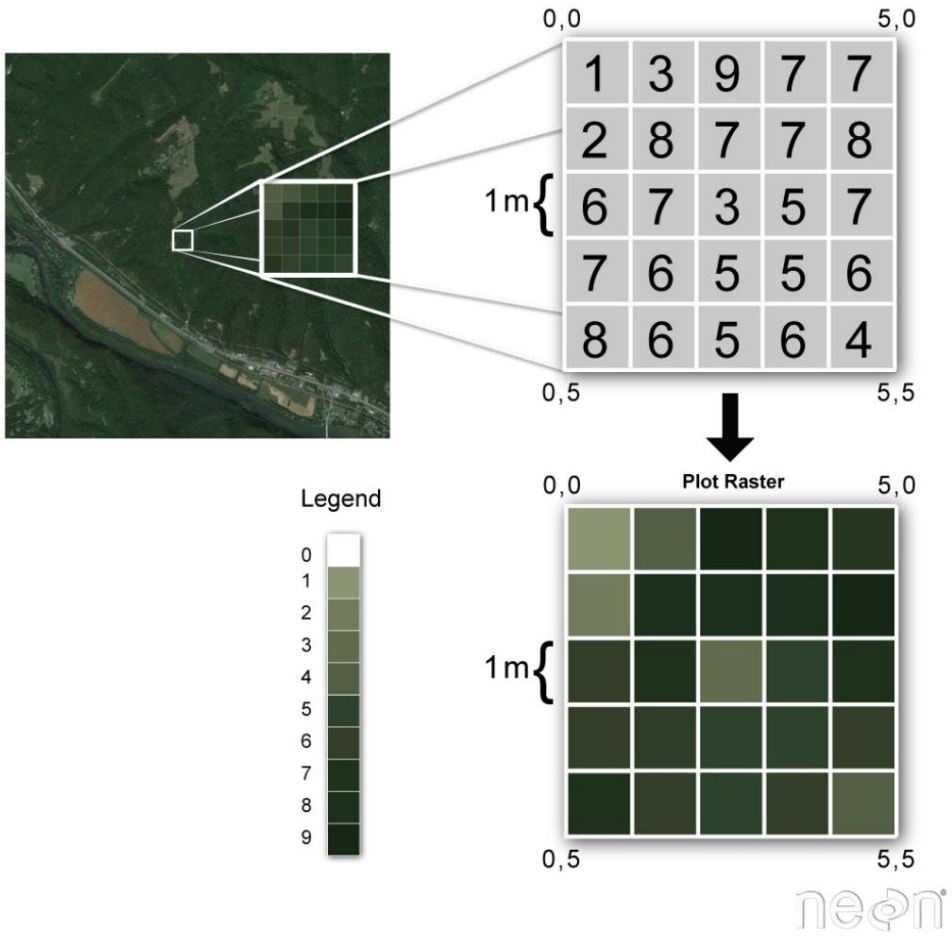


Figure 6. Geotiff raster grid (pixels) [19].

2.3.2 Advantages and Disadvantages

Raster data has some important advantages [19]:

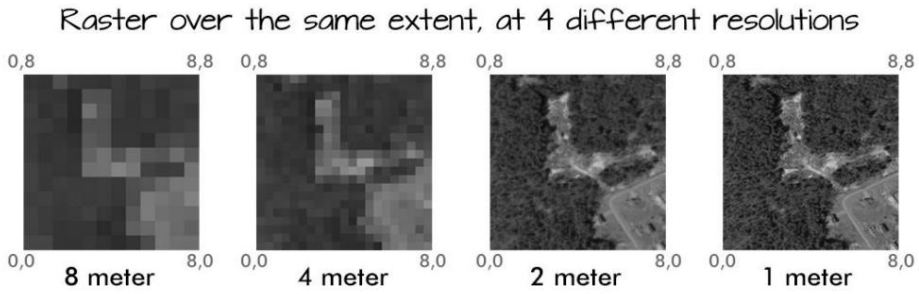
- Representation of continuous surfaces.
- Potentially very high levels of detail.
- The data is "unweighted" over its entire extent - the geometry
  it does not indirectly highlight the features.
- Cell-to-cell calculations can be very fast and
  effective.

The disadvantages of raster data are [19]:

- Very large file sizes as the cell size decreases.
- Currently popular formats do not integrate well
  metadata .
- May find it difficult to represent complex information.

2.3.3 Geotiff analysis

A raster resolution represents the area on the ground that covers every pixel of the raster (Figure 7).



`Figure 7. The figure shows the effect of changes in the resolution [19] .

2.3.3.1 Raster data format

Raster data can come in many different formats.
We use the GeoTIFF format which has the extension 'tif.'. A ".tif" file stores metadata or attributes about the file as embedded tifs tags. For example, the camera can store a tag that describes the make and model of the camera or the date it was taken photo when saving a .tif. GeoTIFF is a standard '.tif' format image with additional spatial (georeferencing) information embedded in the file as tags [19].

2.3.3.2 Multi-band Raster Data

A raster can contain one or more bands. A guy Multi-band raster data that is familiar is a color image. A basic color image consists of three bands: red, green and blue. Each band represents light reflected from red, green or blue parts of the electromagnetic spectrum. The pixel brightness for each zone, when combined, it creates the colors we see in an image (Figure 8) [19].

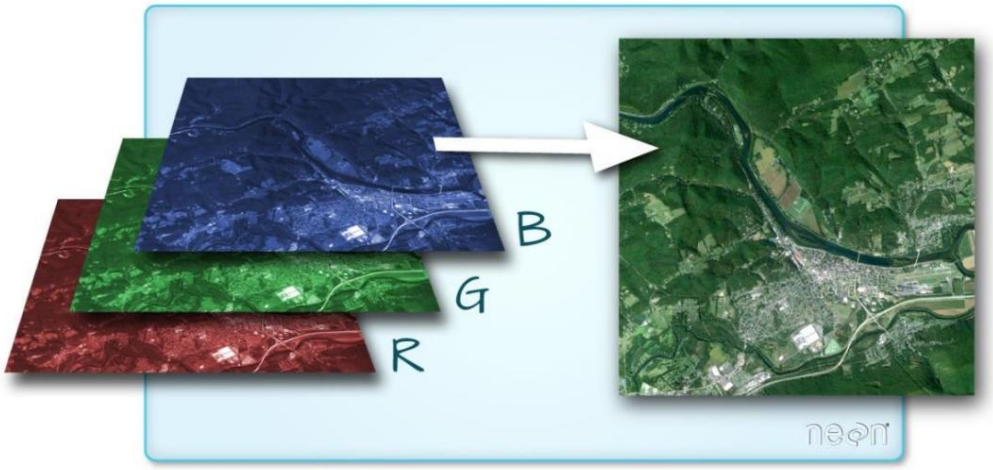


Figure 8. The image shows the composition of three Bands [19].

We can plot each zone of a multi-zone image separately (Figure 9) .

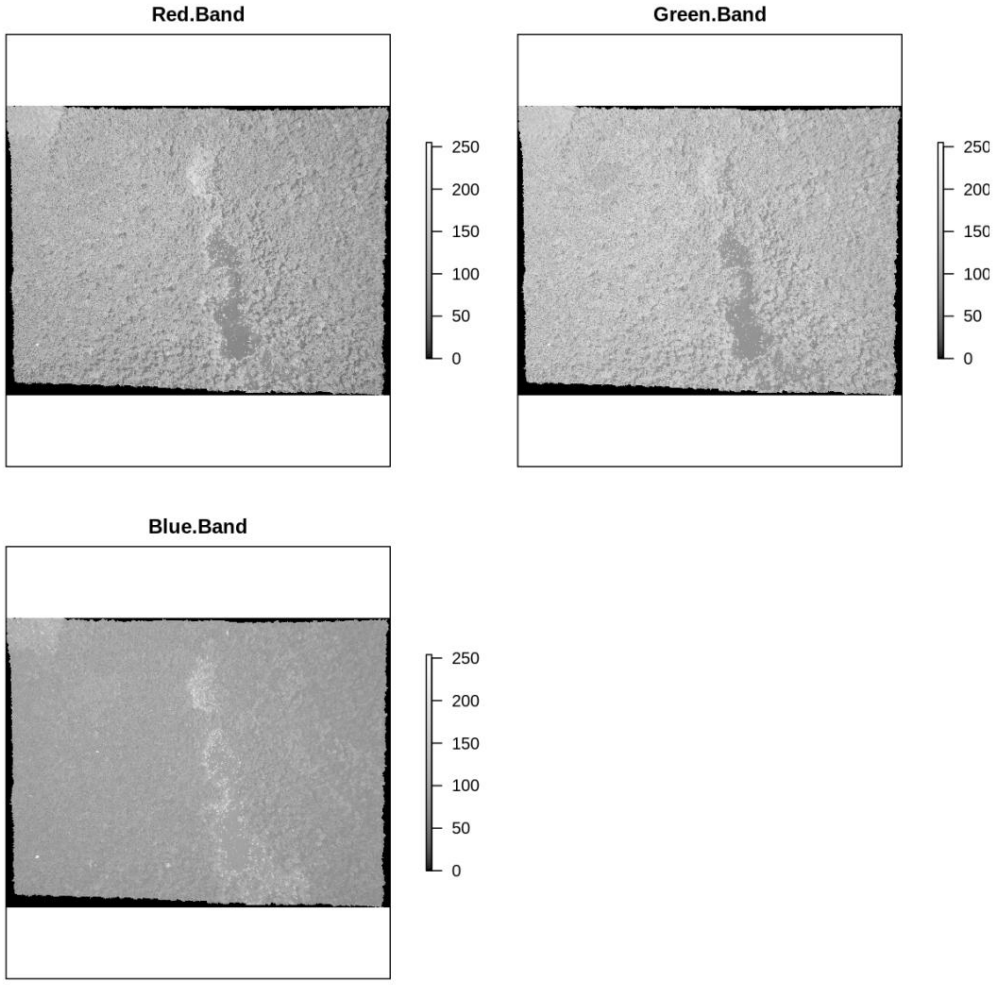


Figure 9. The image shows the three separate Bands red blue green [19].

Or we can combine all three zones together to create a color image (Figure 10) .

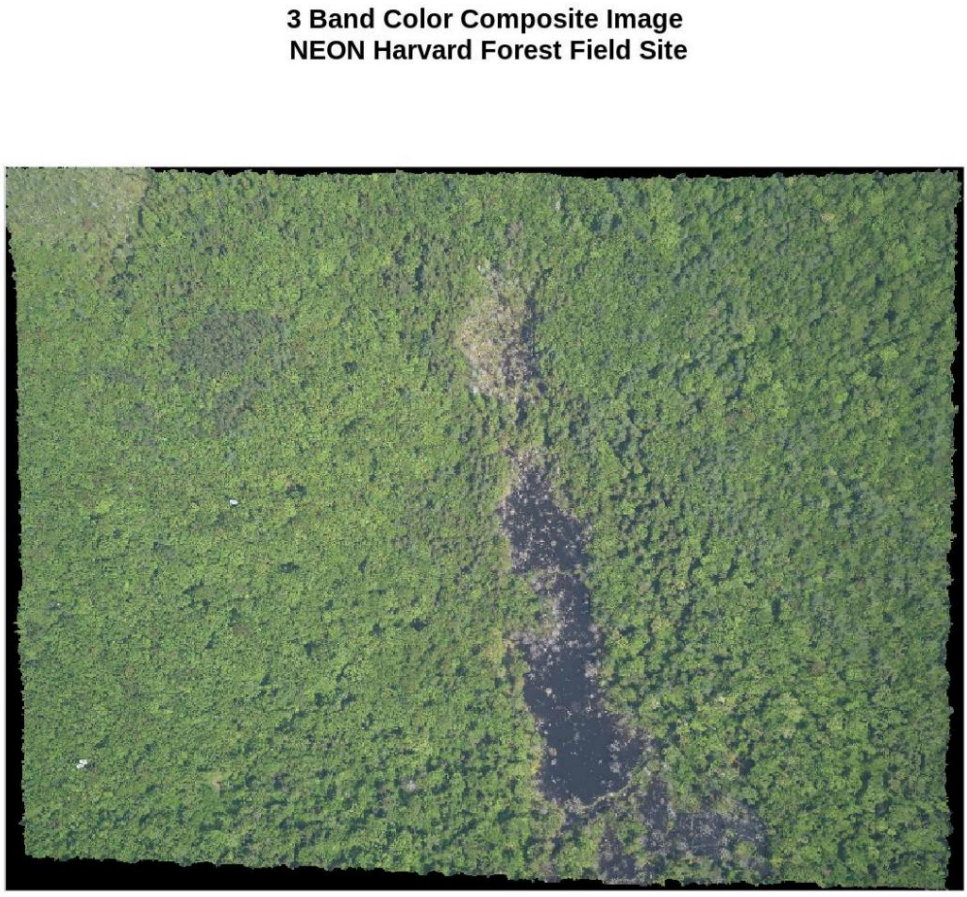


Figure 10. The figure shows the composition of the three Bands of Figure 9 [19].

2.3.4 Raster Bands

Some rasters have a single band (one meter of a single characteristic) of data, while others have multiple bands. A belt represented by a single array of cell values and a raster with multiple bands contains multiple spatially contiguous arrays of cell values representing the same spatial area. An example of a set A single-zone raster is a digital elevation model (DEM). Each cell in one DEM contains only one value representing the elevation of the surface. You can also have a single band orthophoto, which sometimes it is called a panchromatic or grayscale image. Most satellite images have multiple bands, which usually contain values within of a region or band of the electromagnetic spectrum [20].

There are three main ways to display (render) datasets single zone raster (Figure 11):

- Using two colors: In a binary image, each cell has a value of 0 or 1 and often shown using black and white. This guy display is often used to display scanned maps with simple line work such as parcel maps.

- Grayscale: In a grayscale image, each cell has a value from 0 to another number such as 255 or 65535. These often used for black and white aerial photography.

- Color Map: A way to represent colors in a image is with a color map. A set of values is coded for to match a specified set of red, green, and blue values (RGB). For more information, see Basic concepts of raster dataset colormaps.

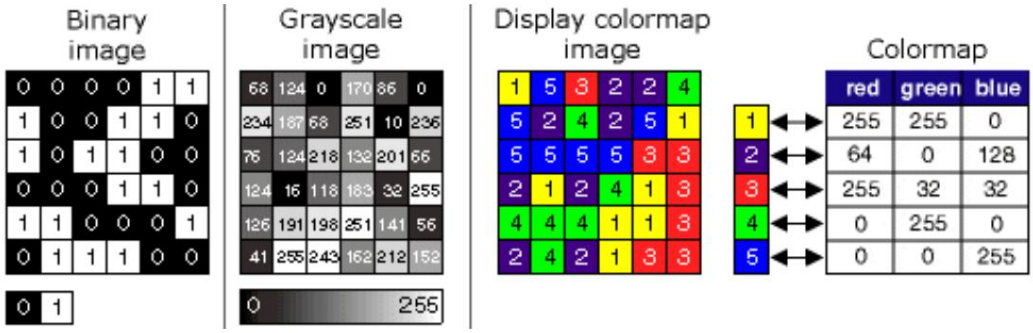


Figure 11. The three main ways of displaying one-band raster datasets [20].

A band contains a table of values. This table depends on number of pixels we have chosen to download and process. With the suitable acts between the Bands we can get specific information about an area. For example we could take information on the percentage of vegetation of an area by performing the action (B8A B04)/(B8A+B04) where B8A BO4 are the names of the bands corresponding to a area.

There are different ways to calculate an information through tif files, doing different operations between bands, each way calculation of an information depends on the area we want to let's process. The way we will choose to extract a piece of information, like for example vegetation, depends on whether there is water around area, if there are mountains nearby and various other parameters that us determine which way of calculating an information is more efficient for it specific area .

Due to the fact that the different Bands vary in pixel sizes, so the size of the tables they contain is different, the following problem appears. A transformation must be done between arrays to perform the addition. This our problem is solved by the site land viewer [21] which performs these actions and gives us the final result in the form of another Band. It informs us about the information contained in each band And with the help of Python we can get any information we want from any area.

2.3.5 Rasterio (Python)

Rasterio is a Python library based on GDAL and Numpy which is designed to make working with geospatial raster data more productive. Rasterio is a very useful library for raster processing which we can use to read and write many of different raster formats in Python. Rasterio is based on GDAL and the Python automatically registers all known GDAL drivers for the reading supported formats when importing the module. GDAL is a translator library for raster and vector formats geospatial_data. The use of Rasterio and the way we manage it the data will be seen in more detail in the next chapter [22].

Chapter 3. Application development

3.1 Edit Geotiff

In the previous chapter it was analyzed what a Geotiff file is and what it is Bands raster. At this point we will analyze the Bands we processed.
As we mentioned different acts between Band give us a specific information about the area we have chosen. Through Landviewer we didn't have to perform any act between the Bands.
For example, the vegetation calculation with the NDVI method is done with operation (B8A-B04)/(B8A+B04) between the corresponding bands. Land's website viewer gives us the possibility to download a band as a result of this deed.

The information we have chosen to deal with is vegetation (Ndvi,Arvi), water vapor (Water vapour), determination information of snow or snowfall (Ndsi), longitude and latitude (Longitude, Latitude).

The Normalized Differential Vegetation Index (Normalized Differential Vegetation Index) is often used to monitor drought, the monitoring and forecasting of agricultural production, forecasting fire hazard zones and desert encroachment mapping. THE NDVI is a standardized vegetation index that allows us to create an image showing the relative biomass.

The Arvi (Atmospherically Resistant Vegetation Index) index is one enhanced NDVI, used to correct its influence atmosphere. It is most useful in areas with a high content of atmospheric aerosol, including tropical regions that have contaminated with soot.

The NDSI (Normalized Difference Snow Index) is one reason
spectral band exploiting the spectral differences of snow at
infrared and visible shortwave spectral bands for its determination
snow versus other features in a scene. In visible wavelengths h
snow cover is just as bright as the clouds, and therefore it is difficult
to be distinguished by cloud cover. However, at 1.6 meters its coverage
snow absorbs sunlight and therefore appears much darker than
clouds. This allows effective discrimination between snow cover and
of clouds

For each piece of information we mentioned above there is a corresponding file band which is stored in the diploma file. All Bands
except for the calculation of Latitude Longitude they were processed in the same way.

For better understanding we have created 2 diagrams, data flow diagram and information flow diagram which are shown in images 1D, 2D

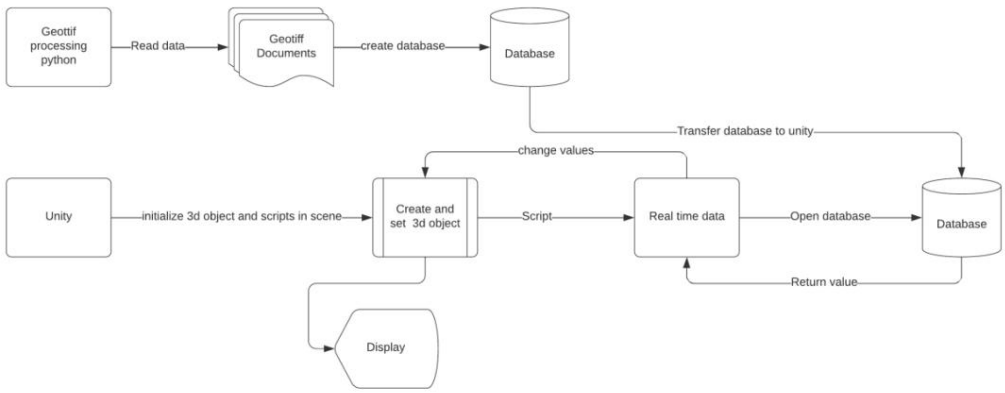
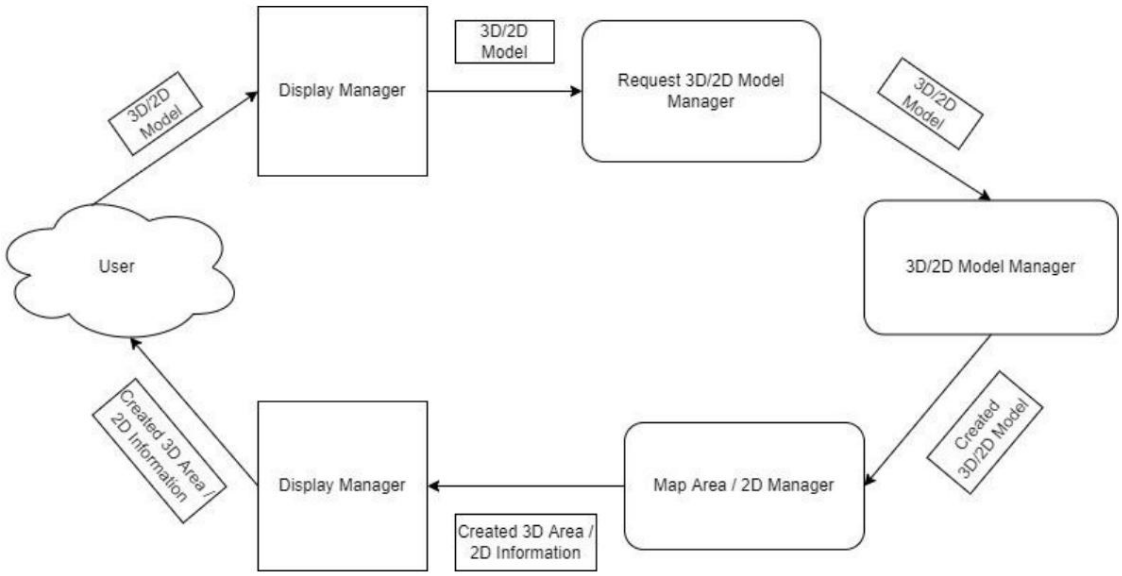


Figure 1D. Data flow diagram.



2D image. Information flow diagram.

3.1.1 Calculation of Latitude-Longitude

To calculate the Latitude-Longitude we were helped by the Gdal library. The Gdal library enables us to access common information concerning the bands of an area. For example as many different bands and to take for a certain area we will always have 5 pairs Latitude –Longitude values for this area. Specifically these couples concerns the 4 corners (Top left-Bottom left-Top right-Bottom right) and the centre.

The process is very simple as it does not require anyone to write algorithm. We just need to install the Gdal library and run the gdalinfo command "Input filename" > "Output filename".

Whenever with the command (gdalinfo S2A_tile_20210915_34SDJ_0_R10B03.TIF> ReportGdalInfo.txt) we will store this common information for all Bands in a file named ReportGdalInfo.txt. This information can be seen in Figure 12.

ReportGdalInfo.txt - Σημειωματάριο

Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια

```
Driver: GTiff/GeoTIFF
Files: BandAOT.TIF
Size is 1333, 1252
Coordinate System is:
PROJCRS["WGS 84 / UTM zone 34N",
    BASEGEOGCRS["WGS 84",
        DATUM["World Geodetic System 1984",
            ELLIPSOID["WGS 84",6378137,298.257223563,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4326]],
    CONVERSION["UTM zone 34N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",21,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["unknown"],
        AREA["World - N hemisphere - 18°E to 24°E - by country"],
        BBOX[0,18,84,24]],
    ID["EPSG",32634]]
Data axis to CRS axis mapping: 1,2
Origin = (482760.000000000000000,4396720.000000000000000)
Pixel Size = (10.000000000000000,-10.000000000000000)
Metadata:
  AREA_OR_POINT=Area
Image Structure Metadata:
  COMPRESSION=LZW
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  (  482760.000, 4396720.000) ( 20d47'55.86"E, 39d43'12.65"N)
Lower Left  (  482760.000, 4384200.000) ( 20d47'57.04"E, 39d36'26.54"N)
Upper Right (  496090.000, 4396720.000) ( 20d57'15.77"E, 39d43'13.24"N)
Lower Right (  496090.000, 4384200.000) ( 20d57'16.03"E, 39d36'27.13"N)
Center      (  489425.000, 4390460.000) ( 20d52'36.17"E, 39d39'49.98"N)
Band 1 Block=512x512 Type=UInt16, ColorInterp=Gray
  NoData Value=0
```

Figure 12. Common information shared by all Bands of a region.

We notice in the Corner coordinates that there are 2 pairs of values. The second pair in Degrees, Minutes and Seconds (DMS). The pair (DMS) values it shows us which number is Latitude and which Longitude. The letters E (East), N (North) show us which is Latitude and which is Longitude. The geographic latitude (Latitude) characterizes the North (North N) part, while the longitude (Longitude) is designated East E of a point. We choose to isolate the values in the form of Decimal Degrees (DD) because then they are easier to process.

In the S2A folder we will find the first algorithm in Python that will explain. The ReadGeotiffScript.py file is responsible for processing the geotiff and the creation of specific structures so that they are quickly accessible by Unity to make the overall operation more optimal. So for his part Latitude Longitude we have implemented a function (calculatelatlot) which accepts a DMS value and returns us a value in DD format. Let's emphasize that it is not we made this piece, of the conversion, we but found it available at internet.

The aforementioned command (gdalinfo) you use in the algorithm whenever running the file will create an output file. It will then load, will select the DMS values, send them to the function (calculatelatlot) and to finally a final file will be created containing the 5 coordinates in DMS format, as shown in Figure 13.

To calculate DD->DMS initially for degrees we use integer part of the decimal. For the minutes we multiply it decimal remainder times 60. We use the integer part of the answer as minutes. For seconds we multiply the new remaining decimal by 60

Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια

Coordinate	Latitude	Longitude
Upper left	39.72017976316562	20.798849880686713
Lower left	39.60753579769413	20.798849880686713
Upper Right	39.72017976316562	20.954453520696205
Lower Right	39.60753579769413	20.954453520696205
Center	39.66388421078756	20.876714853433327

Figure 13. Final file with Latitude Longitude coordinates.

In the continuation of this chapter we will refer to the way that the application manages these values.

3.1.2 Ndvi-Arvi-Ndsi-WaterVapour

Regarding Vegetation information (Ndvi-Arvi), chance of snow (Ndsi), and water vapor (Water Vapor), the corresponding bands are managed with the same way. Next we will analyze how to process one of

that's all.

First we follow the same process in all 4 of these files, so we have create a read function that needs as input the file that will and edit a file name for the output where it will save it information in an appropriate format.

The read function through the rasterio library opens a Band, and in then converts the information to a float. As we said a Band contains one array of values, so we convert each value from it to a float. Then in a new file we will need to save the size of the table i.e. 20x30 information occupying the first 2 lines and in continue to record the price table.

Finally the new file has the format of Figure 14, and with this way we can load any file we want by replacing it

already existing input file with a new one with exactly the same name.

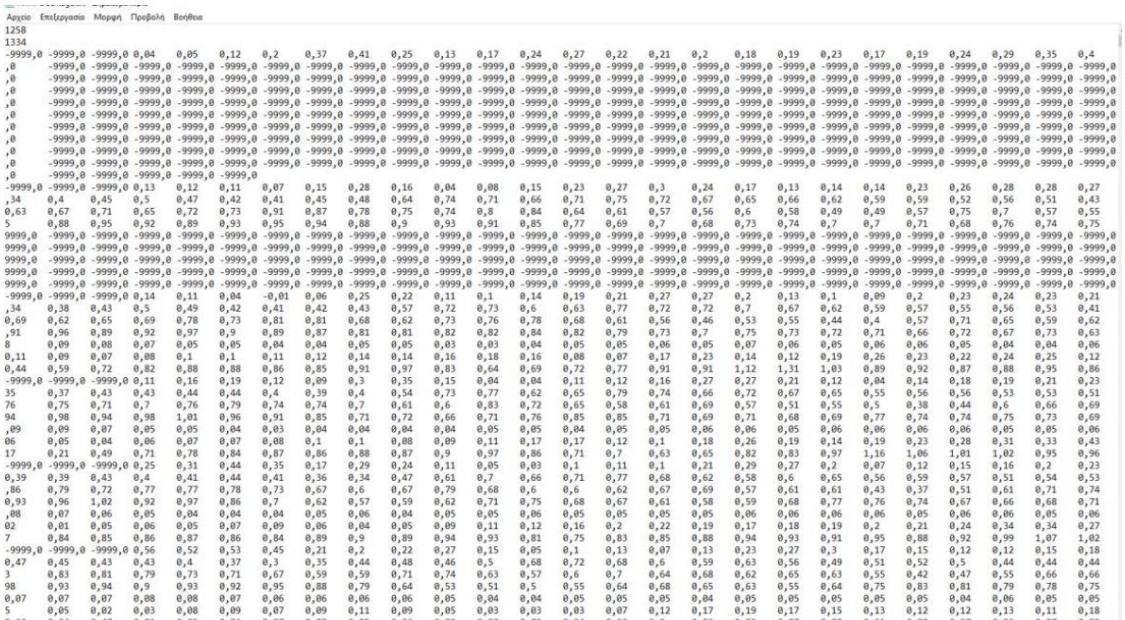


Figure 14. Final file with the table size and the information table of a band itself.

3.2 Creating a virtual world in Unity

3.2.1 Terrain

To create a virtual world is almost impossible to build a realistic and faithful model of an area by hand. On this point the height mapper unties our hands. Height mapper is a tool in unity which accepts various types of files as input and creates one for us terrain based on the input it takes. Height mapper can also accept one Geotiff image, but not Bands as we saw above. The difference is that a lot bands together as we said create the final image.

After importing the Height mapper we need to download a file Geotiff image of the area we want. There are many sites that give us this possibility. We used terrain.party which gives us the ability to select an area with a grid adjust the size and the extent of the file (Figure 15).

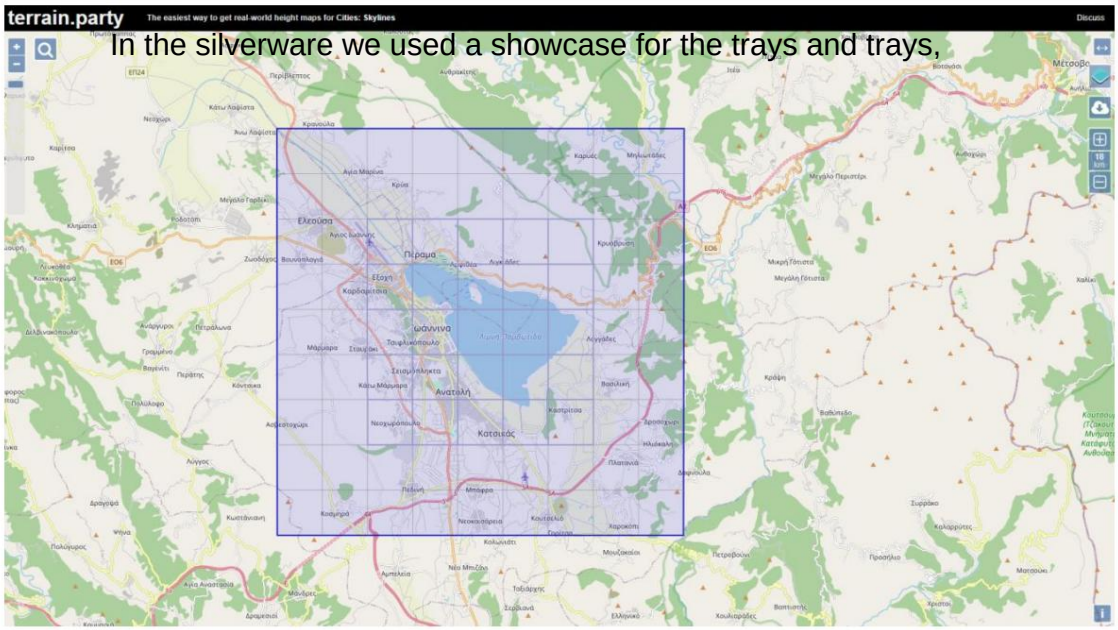


Figure 15. Terrain Party.

After downloading a Geotiff image we can use it Height mapper for Terrain creation. Giving as input the file that downloaded we are given the possibility to edit some elements of it terrain. From the terrain party we know exactly the values to enter in height, width and length so that the result is realistic. The prices these are in the settings we select during download. During import of the geotiff file will open a tab like Figure 16.

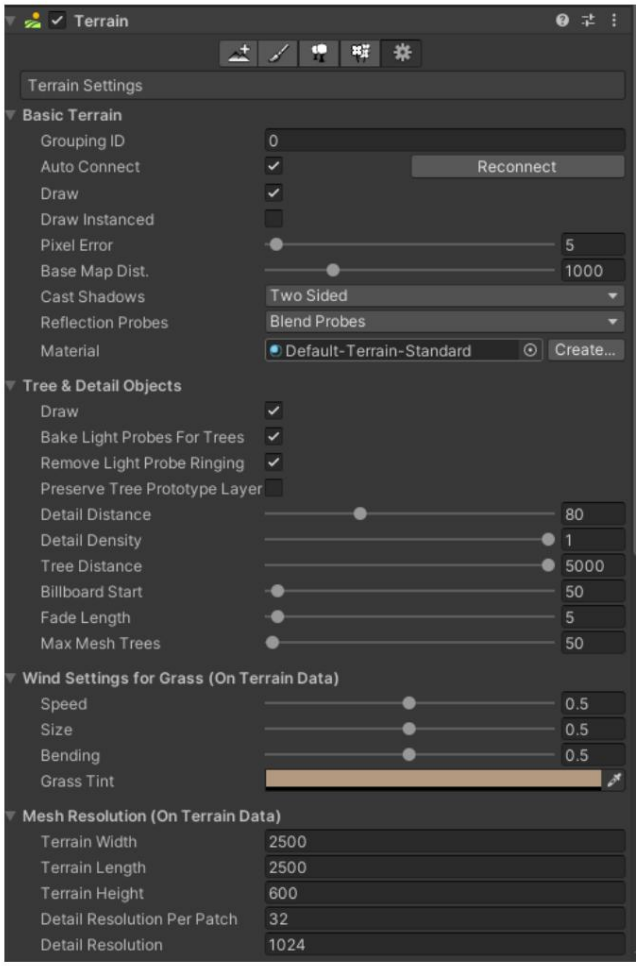


Figure 16. Terrain Settings.

Saving these changes will create the terrain that is the virtual world of the Ioannina area (Figure 17).

Figure 17. Terrain object.

Our new Object which is the terrain has additional settings. Specifically has a settings bar that is strictly for Object terrain only. Who change made base of this bar is an integral part of the Object ie if the terrain is deleted they are also lost (Figure 18).

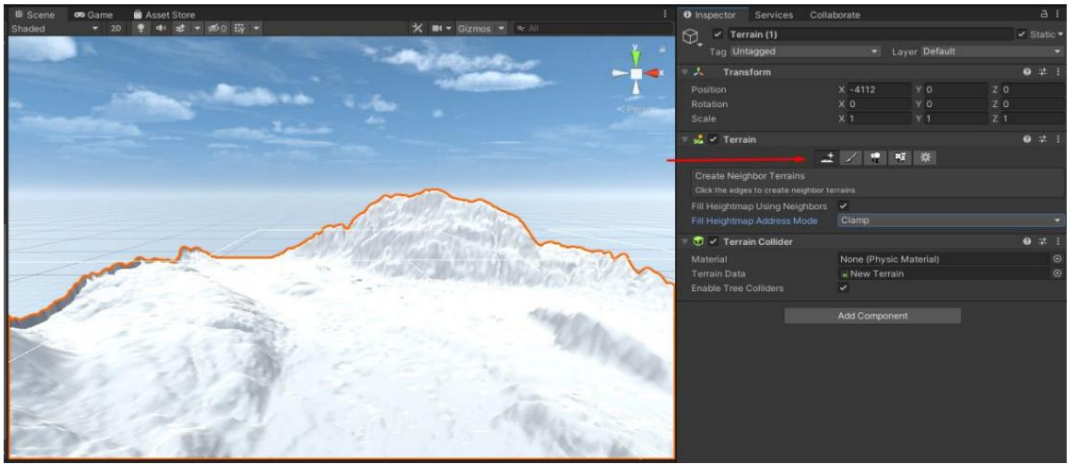


Figure 18. Terrain bar settings.

A very important setting is Layers. The layers are any texture are placed evenly over the terrain and gives a more realistic perspective to Object (Figure 19).

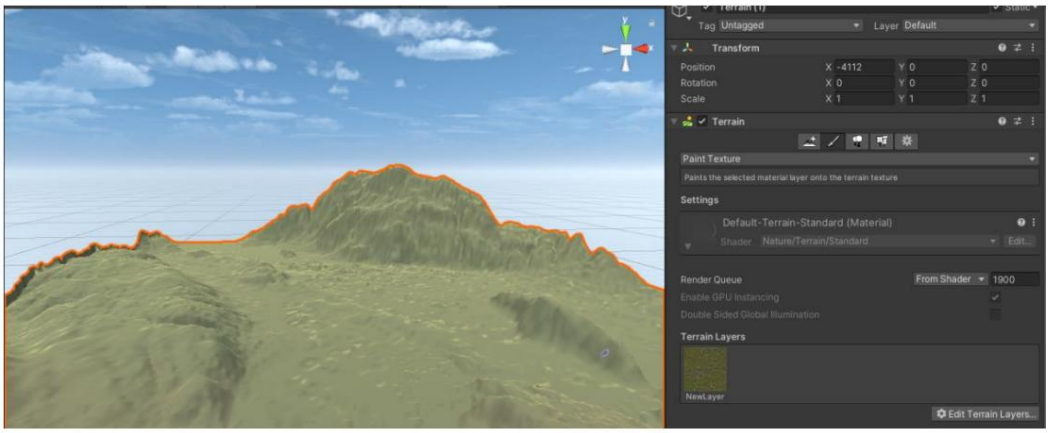


Figure 19. Placement of Terrain layer.

Geotiff files and Height maps have several errors. We mean error the wrong values generated in the file due to various factors. In this the point we can see that there are areas we should have loot surface. To solve this problem we can smooth the terrain to some points so that we can work more easily with the tools that provides us with the terrain itself.

It gives us the possibility to adjust the height in an area and with specific brushes (brushes) to implement the smoothing. The brushes these have their own settings such as size and how visible the these changes (Figure 20). At this point we choose very carefully appropriate height and we do the smoothing in such a way that it does not we spoil the height system of the terrain which we will need later.

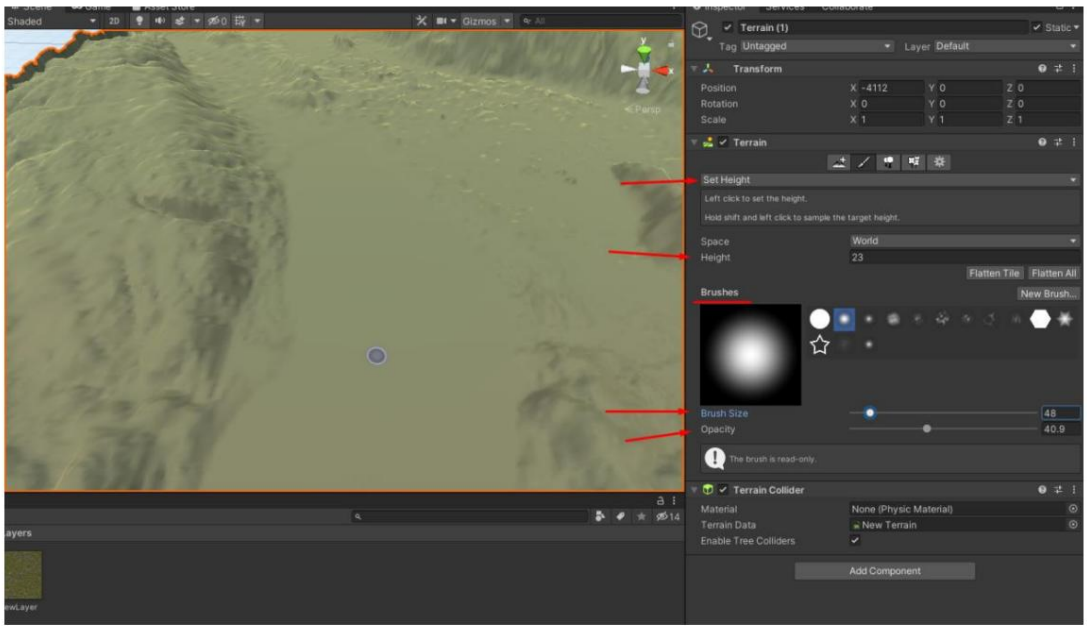


Figure 20. Ground smoothing application.

For inserting natural elements such as for example trees for still once the terrain object offers us the solution. By downloading the assets that we mentioned in other chapters, we have the possibility of importing trees. We add a tree Prefab and then in the same way we did smoothing we can add trees to the area. They are provided to us and additional settings for importing trees such as their density of trees or their height (Figure 21).

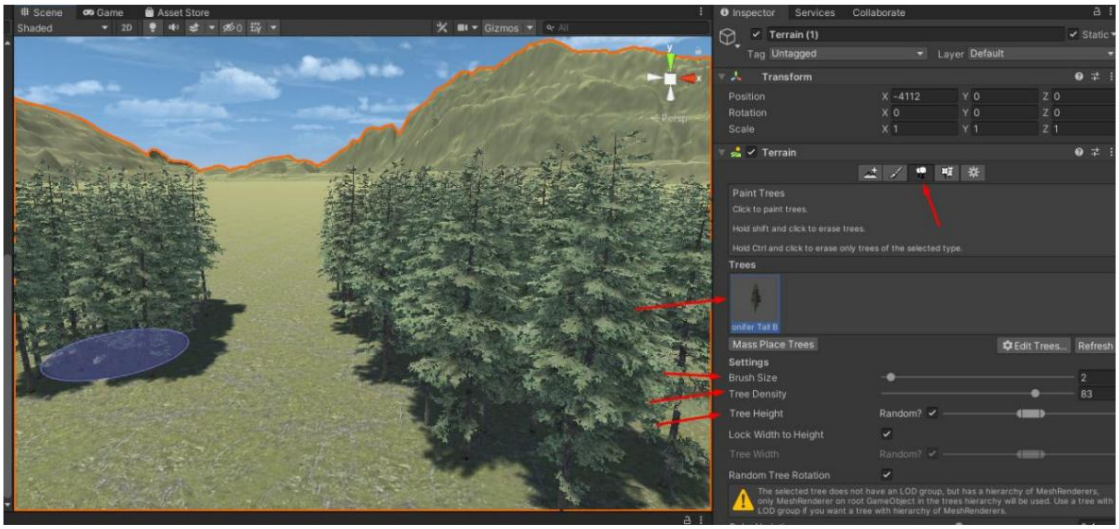


Figure 21. Creating a forest manually.

A very important part of unity Objects are Prefabs. A prefab this is also an Object with above positive elements. Prefab has it ability to hold the position and any other characteristic one had Object at the time we created it.

Its creation is simple, we drag and drop an Object from it scene in the asset folder and the prefab is created. For example if we prefab the terrain, we will be able to reproduce it as many times as we want terrain with exactly the same settings. It is also a way of doubling backup. In the event that for some reason an important object does not work or is lost we can drag-drop on the scene to create the same Object as the same settings.

3.2.2 First Person Character-Lake-CityEngine-Sky

3.2.2.1 First Person Character Introduction

First Person Character was introduced by a member of his community Unity who provided it for free on the forum. Generally there is an abundance of asset choices of the Unity store. We chose this particular one because it was easily modifiable camera algorithm.

This Object contains 1 script for moving the character at any time and for moving the camera in any direction (Player move) In fact, the only modification made to the Player Move script is the creating another button to speed up the player's movement.

The character settings are shown in Figure 22. In a large terrain constant speed is not enough, the ability to be fast is necessary movement. For this reason we edited the already existing script so that it changes speed each time the shift key is pressed. His statement button is done in the inspector while the algorithm needs its declaration field as Public to make it visible to the inspector.

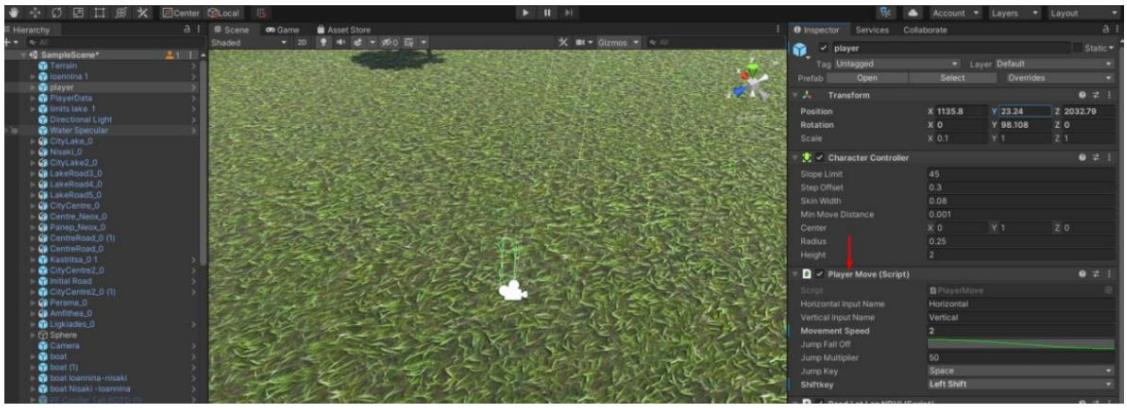


Figure 22. First Person Character Settings.

3.2.2.2 Lake

For the pond placement we imported a Water Prefab from the Unity assets. We used the Prefab as is without any modifications to its script for the effects water movement. Picture 23 shows the presentation of the lake. As long as consider the average level of the terrain which was calculated at the height of 22. At then we consider this as level 1 and add the lake with a height of 21 this has the result that all areas that are above 21 are practically above the water and all areas below it to be flooded with water (Fig 23).

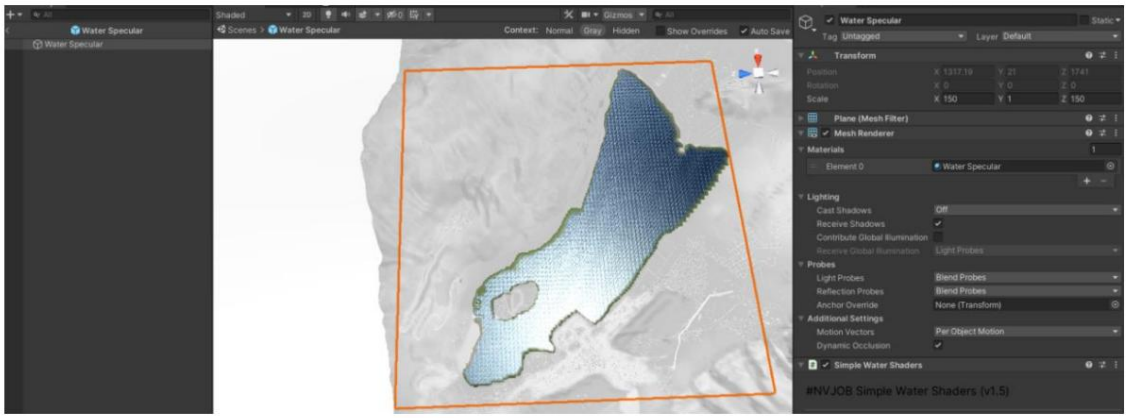


Figure 23. Create a lake, level 0 is water and any colored piece, level 1 all gray areas.

The movement of the player is done freely but special limits have been placed in the lake so he can't get in. Access to the island of Ioannina it can be done at any time through the teleport menu that we will analyze in next chapter. These boundaries are rectangular surfaces which we made and we placed perimeters around the lake that the collider played with collides with him from the rectangular grids and does not allow him to enter in the lake (Figure 24).

Collider is nothing more than a component that is used for object collision. Regarding the appearance of these rectangles grids we have removed any visual element on them and they are invisible surfaces that the user cannot somehow penetrate outside of it teleport map.

An Object can be visually visible from a component called Mesh renderer. The Mesh renderer has a field called Material which specifies the visual representation of the Object. If we remove the specific component then we create an invisible Object for the user however can retain any property and interaction with the user.

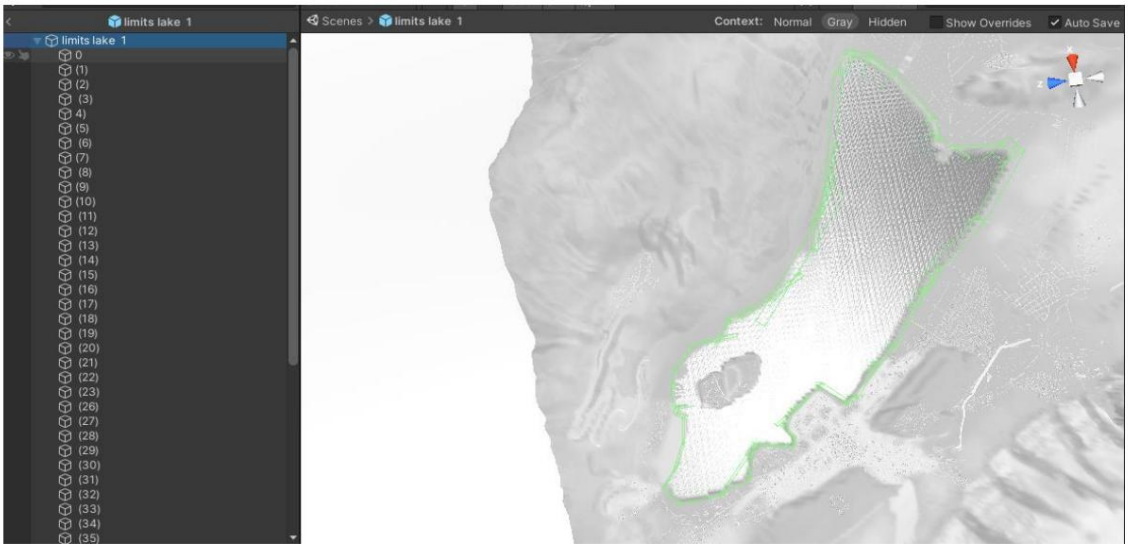


Figure 24. Boundaries of the lake, Rectangular pieces placed around the perimeter.

3.2.2.3 City Engine

We chose the city engine to implement a more realistic representation of it city of Ioannina. The city engine is a program that allows us to create 3d models of buildings along with the streets of the area easily and quickly. We navigate to the area we want to render in Unity, select an area (Figure 25), and save the file (.fbx) which is a prefab of the area we selected.

Figure 25. Selection of buildings of a Random area.

We should follow certain steps while saving it file since if proper storage settings are not made there will be problem modeling in Unity. First select file -> export models In the option that comes up, select Autodesk Fbx option. Then we choose Models with Shape Fallback (Figure 26).

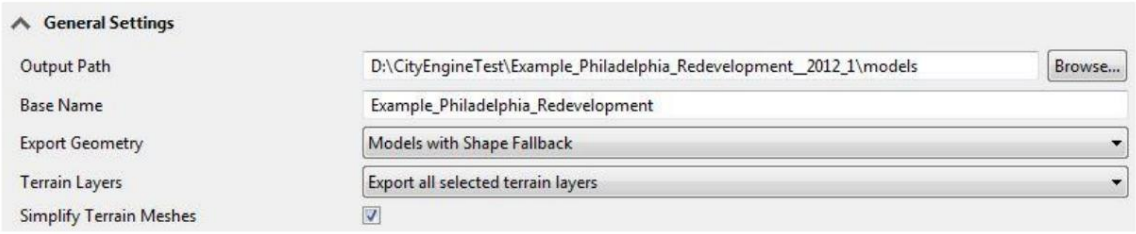


Figure 26. Storage settings

Then in the Geometry Settings tab in the vertex normal field we choose Write vertex normals, in the Normals Indexing field we choose Allow shared normals, in the Texture Coordinates field we select Write all UV layers and finally in the Global Offset field we click on the center to set the Offsets (Figure 27).

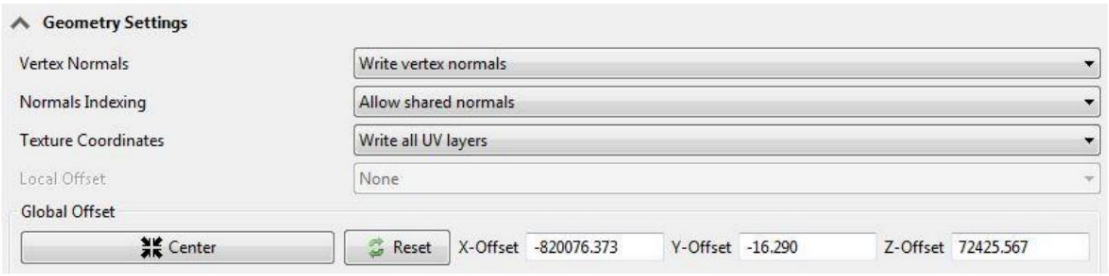


Figure 27. Storage settings.

In the Material Settings tab, select the Include Materials check box.
In the Texture Settings tab, select the Collect Texture check box.
In the Advanced Settings tab, select the Embed Textures check box (Figure 28).
Without these settings it is not possible to recognize the prefab from the Unity.

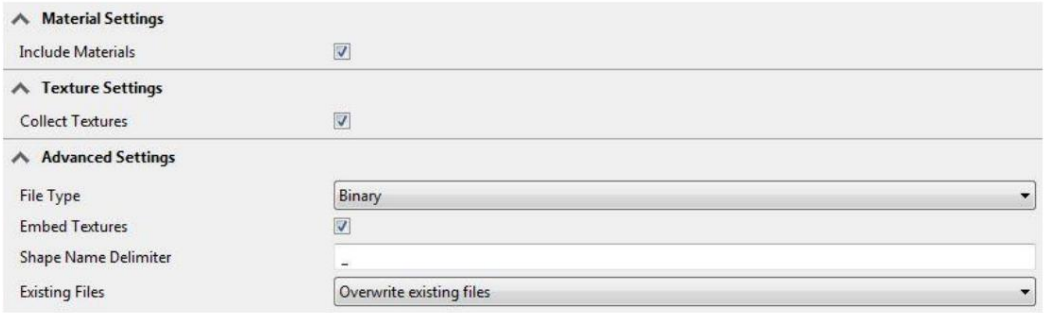


Figure 28. Storage settings.

After we have selected all the buildings we want, we do Import unity by going to Assets -> import new asset and select the folder we have save the Prefabs. When we complete this process, we have Unity asset a folder with our 3d models (Figure 29).

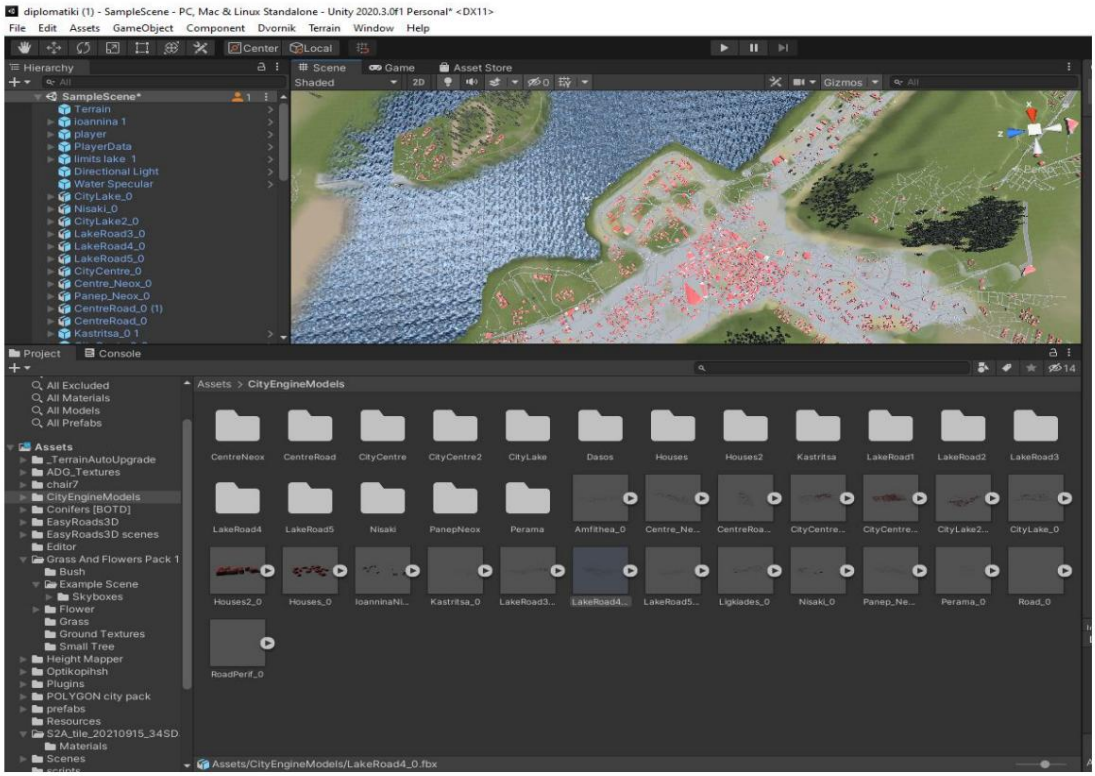


Figure 28. City engine folder.

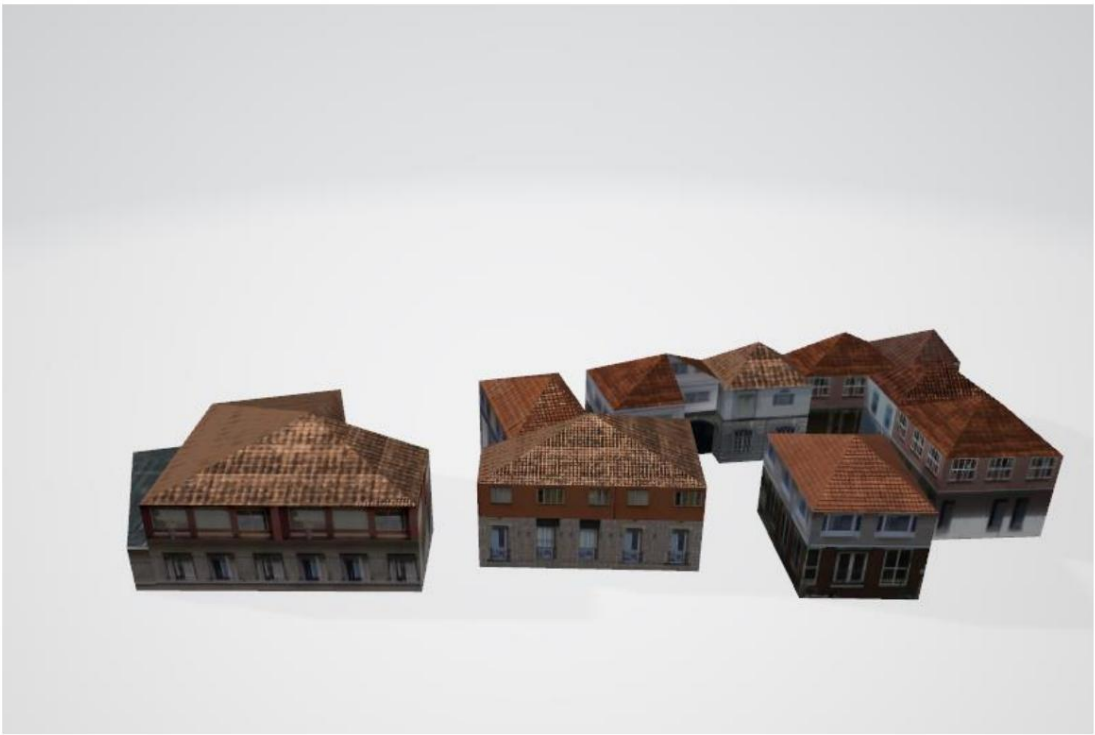


Figure 30. 3d representation of a group of buildings.

With drag-drop on our scene we can now visualize any part of the city we want (Figure 31). A problem for which we will analyze in the next chapter is the problem of Performance. Because every single piece building is a new Object in Unity this creates a problem in performance if we consider that a house needs about 10 objects together with the his materials.

Figure 31. 3d representation of the city of Ioannina.

3.2.2.4 SKY

There are many different light sources, as well as many different skybox material alternatives in the Unity asset store. We chose a skybox with some clouds to make the effect more realistic. For import process in unity first we go window->Rendering->Lighting. In the window that will open we can choose to change the source light. The source of light to the user is the sun. Going to Environment will find the skybox material field. As we mentioned earlier, the material concerns the visual part of any Object in Unity same for sky. In at this point we experimented a lot with different skyboxes, since they are clear the whole process is a matter of aesthetics (Figure 32).

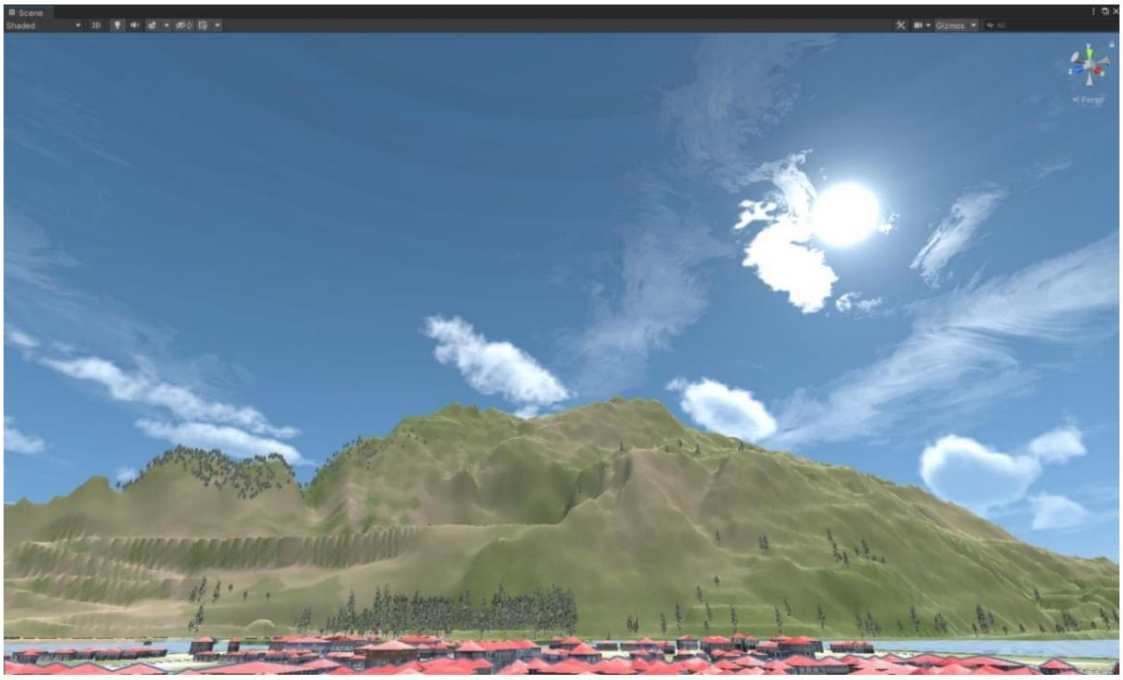


Figure 31. Sky representation.

3.2.2.5 Boats

To create the ships we used the free asset as is. We placed it in a suitable position and using the boat script we made the boat move. In the algorithm we accept as an argument the boat itself and shift its position along the xz axis (in unity oy axis is height).

3.3 Visualization

To visualize the user's screen we used an object what do you call canvas? That object we place inside it is visible in 2d Form to the user. In essence the canvas is an Object whatever it is invisible and has the ability to be in front of the camera every time and to adapt to the resolution of each screen so that o user to see the same result on each screen. So placing any 2d object in a position above the boundaries on the canvas becomes visible to the user.

Placing a script in an Object is simple either with drag drop process or by inserting component script into it. It demands many times to give as an argument some Object which the we also place with a drag drop process.

The application at the start of the program consists of the main menu. The main menu consists of an object with a photo which constitutes the Background, 1 more object which it is identical and it is also this background, but it is not contains photo just a gray-black soft color to give a beautiful effect on Buttons and 4 Buttons menu.

Each of these changes the main menu by showing it its content. This change in the script main menu is done with the enable and disable the Father Object. The Objects father is essentially an Object that contains the soft background for the effect in various sizes depending on the text to be cover up The start Button generally disables the Menu and us activates the teleport map (Figure 32).

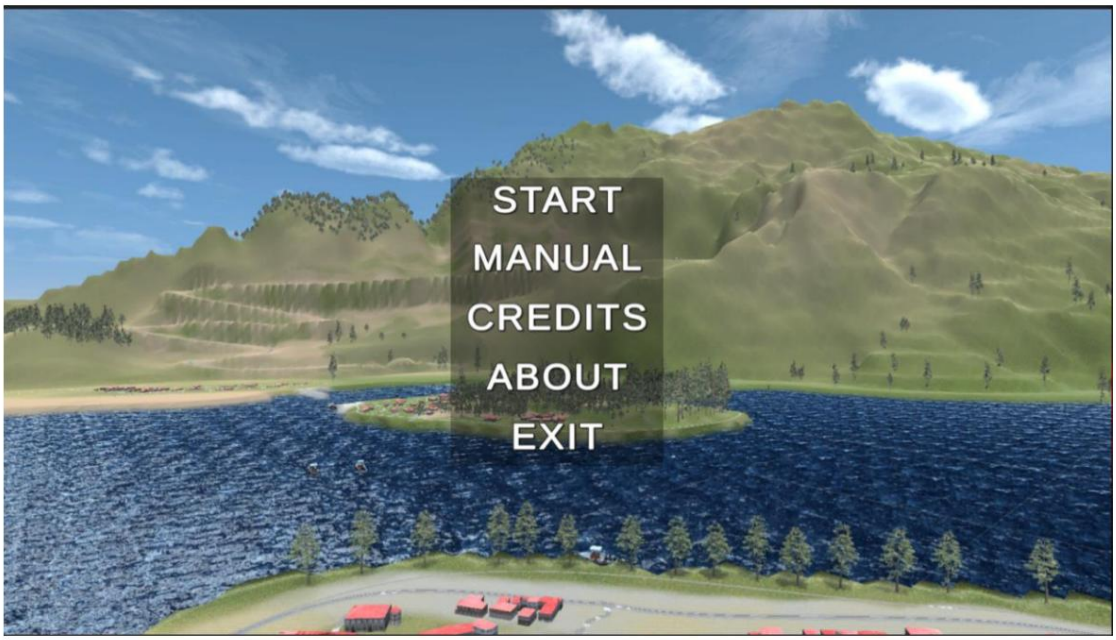


Figure 32. Main menu.

The teleport map is made in a similar way where they exist buttons without text but photos (pins) and pressing them pushpins the character is transported to that point. THE transfer to the code is done via the teleport script which in substance adjusts the camera coordinates to three positions according to which Button will be pressed.

At startup and after selecting the teleport map o
user has the possibility to activate the Main menu with exactly
the same functions with the resume button having taken the start position
button (Figure 33).

Figure 33. Main menu in run time.

3.3.1 User Screen

This chapter will analyze the user's home screen during the launch the game. Note that the application does not start directly with this screen but it is the final stage where the user is now allowed the its interactive interaction with its surrounding environment. Initially there are 4 analysis points, the compass (1), the data bars (2), the mini map and the latitude longitude and altitude information (3) (Figure34).

Figure 34. Home screen of the user.

3.3.1.1 Compass (1)

To create the compass, we needed a photo that it will have numbered degrees and the indications S, W, N, E. The compass during rotation of the camera with the mouse, changes its direction. For this reason, photography had to be done in a repetitive manner as shown in the Image 35. Then we create an Object to which we add the component image. We modify the compass photo in the texture type field sprite (2d and ui) (Figure 36) and press apply.

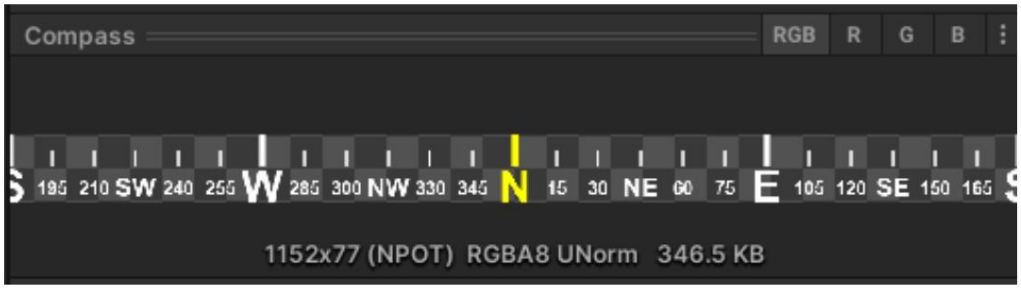


Figure 35. Image of a compass.

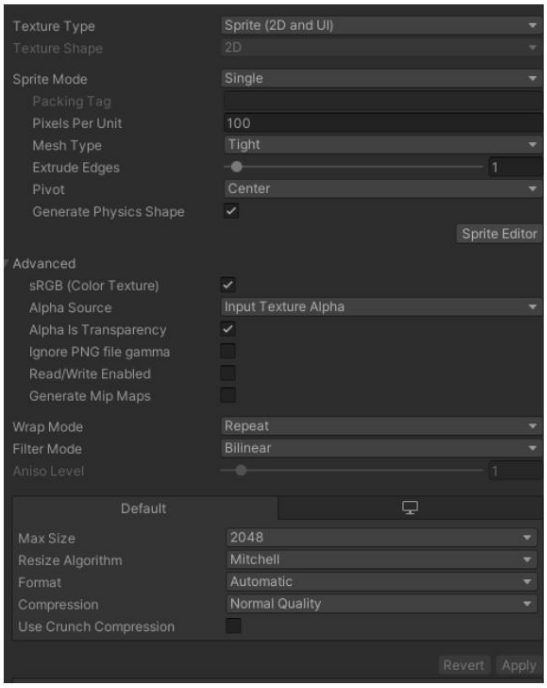


Figure 36. Modifying compass settings.

Important detail here is to pay attention to the settings field wrap mode is repeat for one we have the circular one repeated effect. If we want we can add one background color by duplicating the Object and adding for material, whatever color we want. We didn't change the image either we just teased the transparency to give a gray feel to the back of the screen. We place these two objects with drag drop inside another object that we call father who will have the script of the compass (Figure 37).

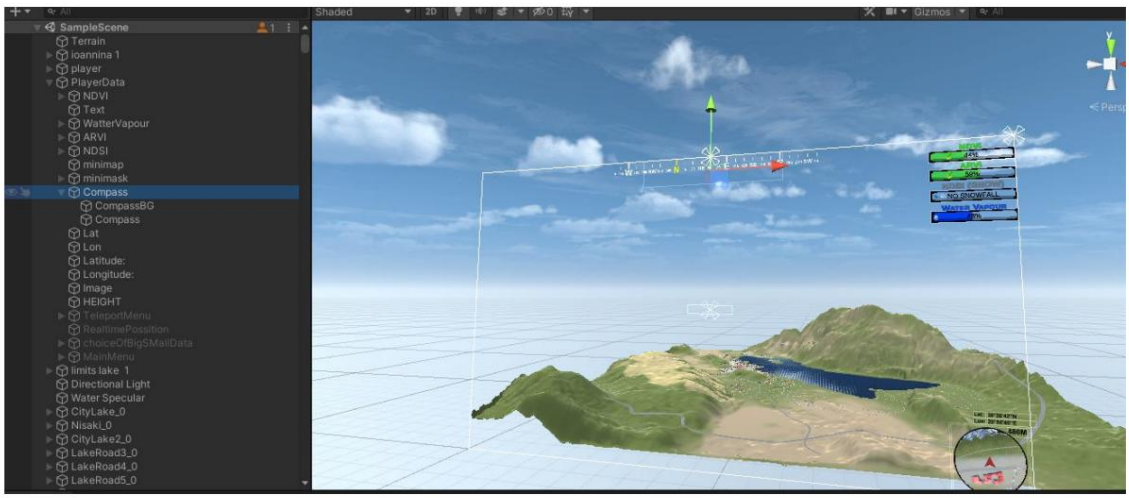


Figure 37. Father compass object with 2 child Objects. Canva display example.

The algorithm we put on the compass simply controls y axis rotation of the character and each time places it on the compass so that this is done in every frame.

Update is a Unity function that executes any code stream every frame in the Unity project. And this is her its main difference with Start which will execute a stream of code only once when starting the unity project.

The placement of the object is done inside another parent that we call it canva. That Object we place inside it is visible to 2d Format to the user. In essence the canvas is an Object which it is invisible and has the property of always being in front of camera and to adapt to the resolution of each screen so that o user to see the same result on each screen.

The placement of the scripts as well as the Objects that must we enter as fields in the script shown in Figure 38.

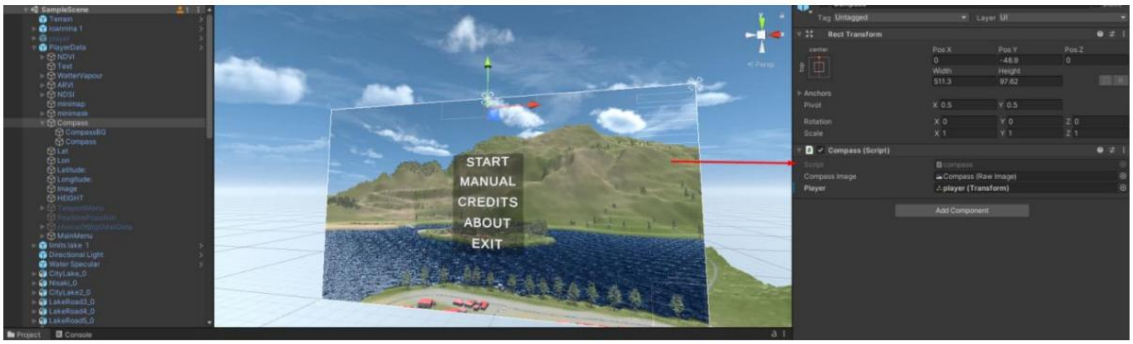


Figure 38. Placement of script and Object fields.

3.3.1.2 Data bars (2)

3.3.1.2.1 Bar Visualization

First we will explain how these bars are made. At first we create a parent object. Inside it we will need an Object that we call it Lifebar and it is the filling that the bar does, we add to it a component image and then we give it the color of our liking. Another Object that we will need is the Border which is the bar outline. We also insert in this object a component image and here we place the photo of the bar that we have found from the internet.

Then we will need another object with a component photo that we will give as an argument of the photo some element that will tells us what the bar is in front of us for example a card for the vegetation. An object which will be a textfield and will show us the amount in % it's the bar. In the order we referred to Object. In this order they must exist within the father we first created. The reason for her order is because the object above is covered by the one below and exists hierarchy (Figure 39).

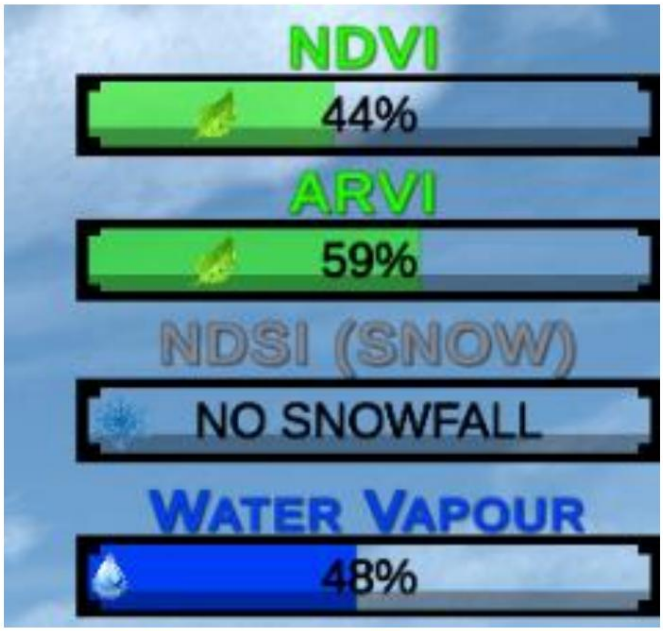


Figure 39. Urbanization of bars.

3.3.1.2.2 Loading Data

For each of these bars, a script has been placed on Canva which implements the loading of the data and their correspondence with the terrain. At first we will analyze the correspondence of a table stored in the files that we have already mentioned with the unity terrain.

Unity consists of a grid of squares that define the his workspace. This grid the more we enlarge it the more it also zooms in showing new squares (Figure 37).

Then we will need another object with component One photo that we will give as an argument of the photo some element that will tells us what the bar is in front of us for example a card for the vegetation. An object which will be a textfield and will show us the amount in % it's the bar. In the order we referred to Object. In this order they must exist within the father we first created. The reason for her order is because the object above is covered by the one below and exists hierarchy (Figure 40).

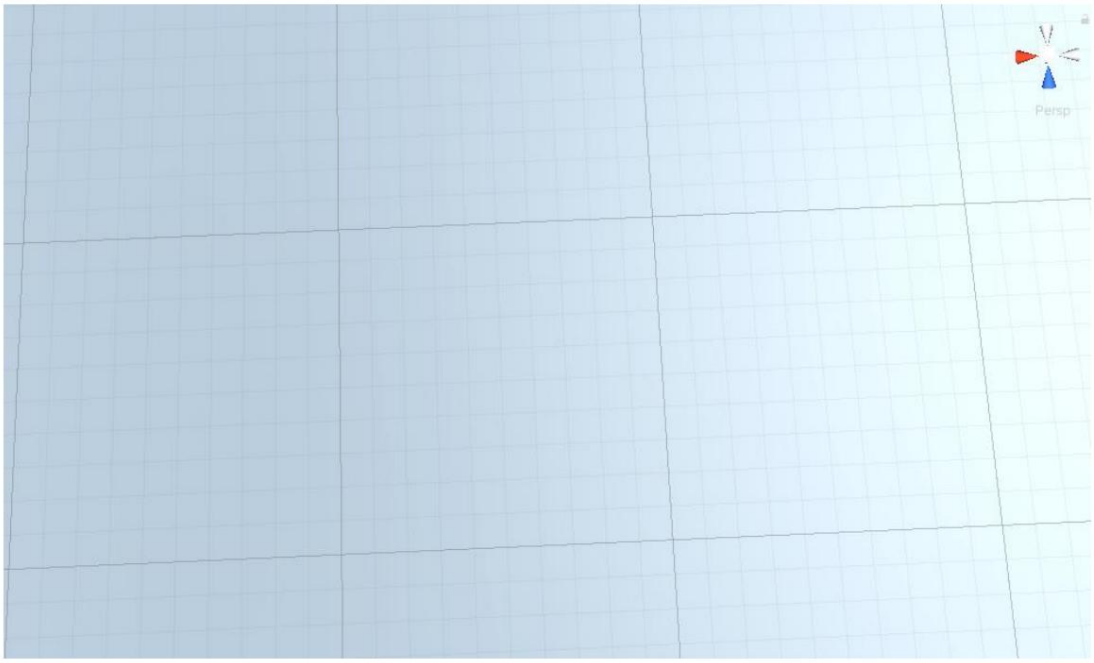


Figure 40. Unity grid.

At this point we can understand that an Object in space occupies any of these boxes. Specifically, the terrain occupies 2500x2500. In our files, in addition to the data table, we also store the board dimensions. So for example if we have a 1000x1000 table we need to make a match with the terrain. Most of the time a geotiff file will not have a larger panel than 2500x2500 whenever it should a piece of information to be assigned to more than 1 box in the terrain.

In this case we work with coordinates and with exact decimal numbers. The terrain is placed at position 0.0.0 (xyz) on coordinate axis of unity. This means that the terrain takes up space in the width from 0-2500 and length from 0-2500 so a net number of 2500.

The algorithm we have implemented for loading the data uses the logic of correspondence. It calculates a step which it tells us that the position we are now x, y, z corresponds to the information in the array [x, y]. O our character when he is on the terrain also has coordinates x, y, z. It can move freely in coordinates 0 - 2500 for x and z axis. THE logic is that we pass the position of the character each time and find it match the data in the file. We will need to see the terrain as a 2d object (square) and find 2 steps one for the length and one for the width.

These steps will help us calculate where the character is and which we need to search inside the file. We will give an example to better understand the problem.

Example:

We have a terrain of size (2500, y, 2500) (x, y, z) the y axis does not is interested because it is the height, we have a file with a table size of 1000x1000. THE camera is positioned at (500, y, 500). How do we know in which position? are we in the data table to print the corresponding value ?

Solution:

First, as we said, we have to find 2 steps for width and length.

The formula is step=file length (width)/terrain length (width).

Stepx=1000/2500=0.4

Stepz=Stepx (in this case it happens, we generally repeat the type)

Next we have to calculate the actual position of the character in the archive. This is done with the type playerX=player location at that moment (in terrain) for X * StepX same for Z.

PlayerX=500*0.4=200=PlayerZ (only in this case).

So we have 2 numbers [200, 200] this is also the cell that should retrieve from the file and print the information.

In the algorithm because this procedure is called for each of the data we have made a readfile function that does this job it finds the information, isolates it and temporarily stores it in memory.

Next we've made a Setsliders function that takes this information and gives it to Bars. Specifically it updates the exact price for example 50% and correspondingly the bar fills up to 50%. Because each file has a different type of information this function also accepts a variable choice so we know which file it's called from. Every file wants different management for the view whenever we implement it this way algorithm to be more automatic.

3.3.1.2.3 Mini map Latitude Longitude (3)

3.3.1.2.3.1 Minimap

To create a Mini map we will need a circular Border. In
then we will need another camera which will be placed above him
user at a sufficiently high altitude and inclined to look towards him
character. unity does not allow 2 cameras to work for the same one at the same time
scene. For this reason we change the camera path so that it points to one
material-texture (Figure 41).

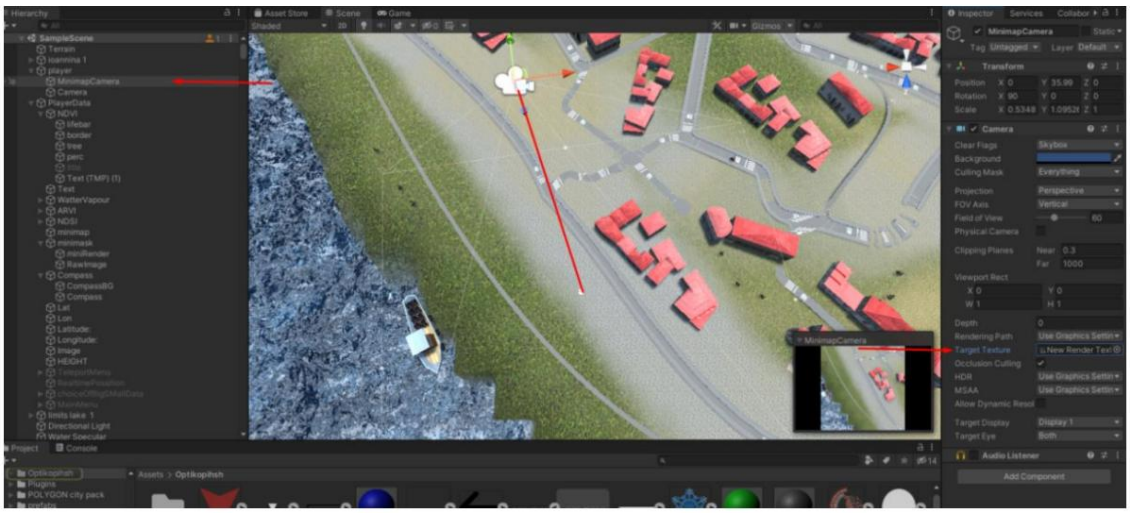


Figure 41. Editing a second Unity camera.

Finally we will need one more Object Minimask which will be the
father. The children of the father will be the MiniRender Object which will get like
component a raw image to which we will give it as its Input render texture
camera we imported so that the camera now points to this
Object. Finally we place in the center of the circle a triangular photo that
indicates the direction of the character (Figure 42).

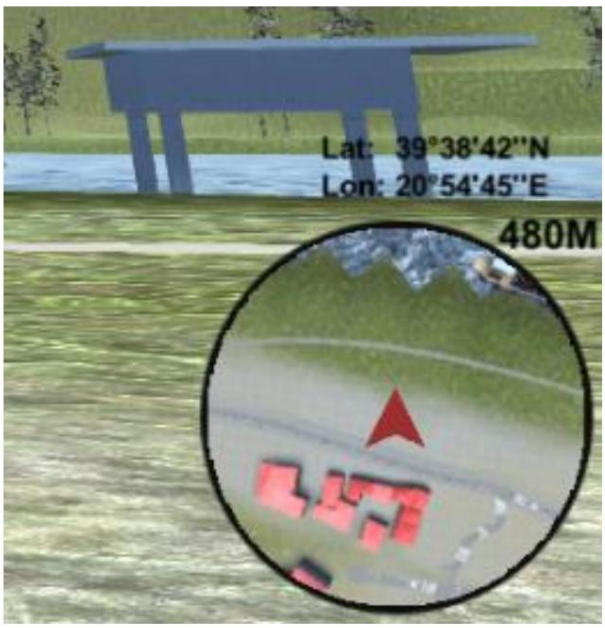


Figure 42. Unity Mini map visualization.

3.3.1.2.3.2 Latitude Longitude

Longitude and latitude are calculated with the same logic correspondence. First we remember that we saved them in DD format so that we can manage them as numerical values and not mess with degrees and minutes. What changes is we have 4 corners with Latitude coordinates longitude. The top or bottom corners (left right) have the same x but different z while the 2 corners on the same side (top bottom left or right) have different x. So we can find a value for X, Z and let it use as "table size" and work with the steps as in geotiff data. As far as their appearance to the user is concerned, they are simply 2 text objects which we update each time per frame.

They are displayed in the form of degrees, minutes, seconds, DMS as we have explained in a previous chapter. For the DD->DMS calculation initially for degrees we use the integer numerical part of the decimal. For the minutes we multiply the remaining decimal by 60. We use it integer part of the answer as minutes. For the seconds we multiply it new remaining decimal by 60. The result is updated in every frame.

3.3.2 3D Data Visualization.

In this chapter, we will analyze about 3d data visualization. Besides the use of bars to display the data to the user we give the possibility for the user to see the data in 3D format in the space around him in 3d Object format.

The logic behind this is that we should no longer access the file for one value, but for a group of values around the user. So to her case we create an array of values just like the one in the file in smaller scale. This array is called a buffer.

The buffer is refreshed every time the user gets close to its limits. THE camera position is the center of this panel. For example if at camera position corresponds to cell [50, 50] then we need to create an array that will start before row 50 and column 50 and end after line 50 and column 50. So if we define Buffer to be 20 in size then we will have a buffer array starting at cell [40, 40] and will ends in cell [50, 50].

The script responsible for implementing this logic is the CamAreaSpawn.cs. Initially the process is the same for each of the 4 species data that we have loaded whenever the same technique is followed with the same ones functions that we will analyze. The first thing the algorithm applies is to disable the data bar for which we want to create a 3d representation so that the user understands which visualization sees. The remaining bars remain active.

In this script we use the functions readfile and setText which they do the same job as before. readfile opens the file for loading and the setText instead of editing the bars now edits the prefabs ie 3d models that the user will see. It specifically defines the urbanized price of % for each Object.

Initially for this part of the visualization you will need to
let's build 2 Objects, one translucent and one visible, whatever you want
use to create clones, which we will
modify the data they represent. During her
application of the object the invisible Object will be placed in a specific position with
100% of its size while the colored Object will vary accordingly
data it represents. Then we place exactly in the same position
in order to overlay the translucent. That's how we manage to
let's create a rectangular container where depending on the value that
we have at the specific point we will have less or more full.
An example of this model is shown in Figure 43-44.

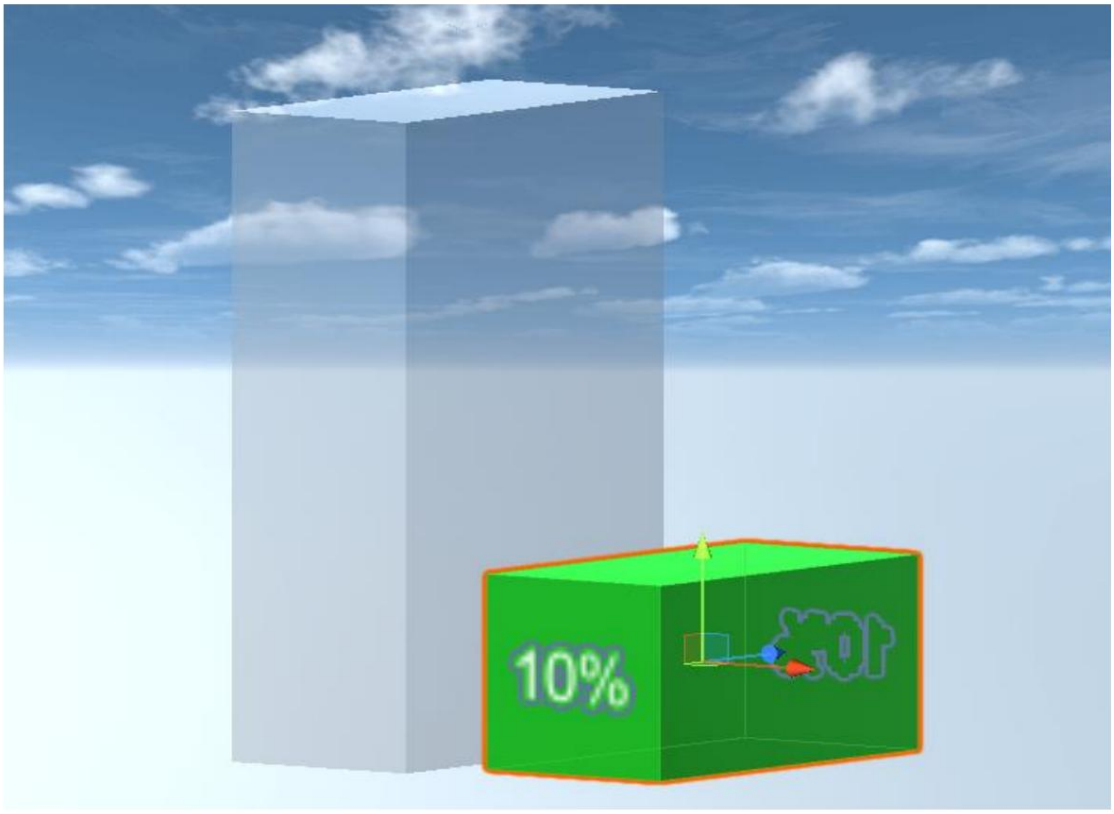


Figure 43. Presentation of 3d models for Data Visualization.

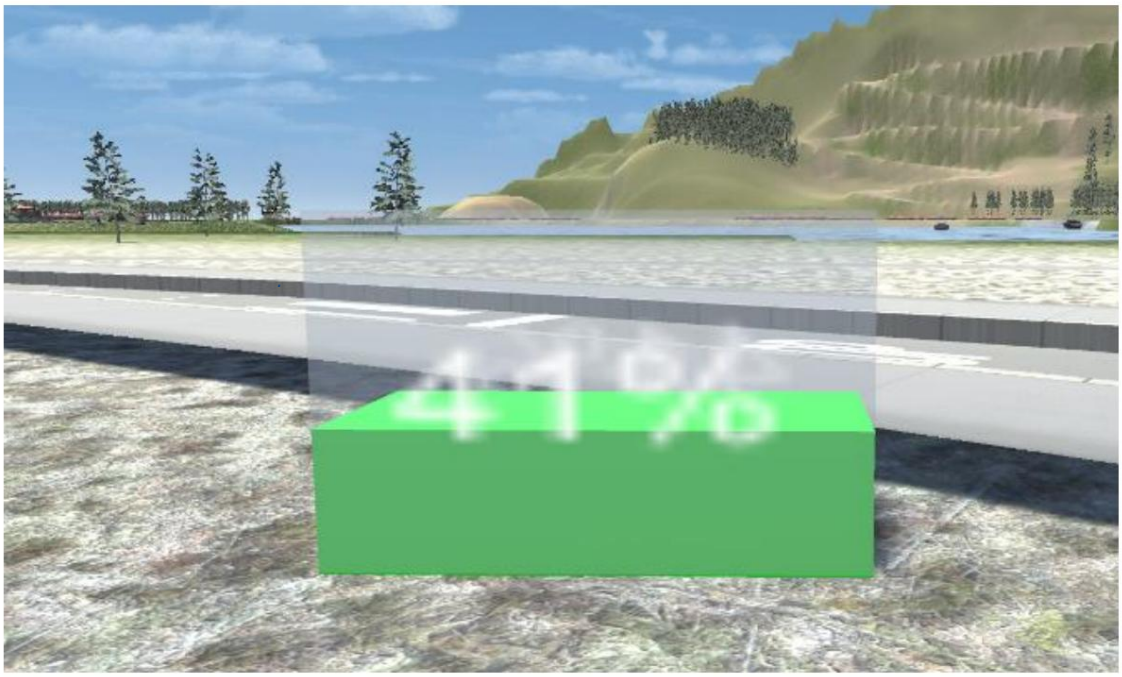


Figure 44. Final result for the 3d data visualization.

Next we will use the createArea function. This n function creates the boundaries of the Buffer ie from the most cell of the file will starts and where it will stop like for example from the cell [40, 40] –[60, 60]. We will use these limits below to start the creation clones for the area around the user.

Our core function is AutoSpawn. This function checks initially for which data we are going to create 3d object. We use the data loading logic we saw in chapter **3.3.1.2.2** (Refer to example). We modify the logic, since we now want a table and not only one item. The size of the Buffer is declared at the beginning of the program.

First we calculate the current position of the character at which position in the file corresponds. Here we have some very important checks that are worth it we report on the correctness of the algorithm.

We will give an example for better understanding.

Example

Suppose the Buffer has size 20x20, the array in the file has size 1000x1000 and the camera position in the file corresponds to position [5, 5]. Who is it the buffer table after all?

Solution:

In this case the buffer must start from position [0, 0] to the position [20, 20]. In this case the camera is not our center and is the exception to the rule for logic we mentioned above. There is one more exception, the corresponding case if we are at the end of the array for example at position [1000, 1000].

So the checks we use check that the Buffer will be found always within the bounds of the table and we will not try to insert elements beyond the size of the data array .

Then we go to the central logic of the algorithm, which is creating a 3d object for every single data we have. The creation of buffer is not done with the logic that we create the buffer table afterwards we create the 3d objects. Basically when creating the 3d object the Buffer fills up at the same time.

What we need to check here again is from which point in the file will start reading and where it will end, and for each a value of this will we call the Object creation function (callthespawner). Accordingly you must check here and calculate the position on the terrain where it will be placed first given.

The call the spawner function is responsible for creating the Object and its placement on the terrain. At this point let us say that the calculation of the position is done before calling this function. The calculation of the position is as follows we find the step that states the size that must pass each time, i.e. if we start at position [5, y, 5] where is the our initial datum we should advance one [5+step, y, 5] to place our next datum.

Next, we find where to start placing Object in terrain, and then after adding each Object we increment the initial position by step. When the row of the table essentially ends, we restore the original one position. This is done in terms of width and correspondingly in terms of height (it should always we observe the area as a 2d object square or rectangular depending on Buffer size) (Figure 45).

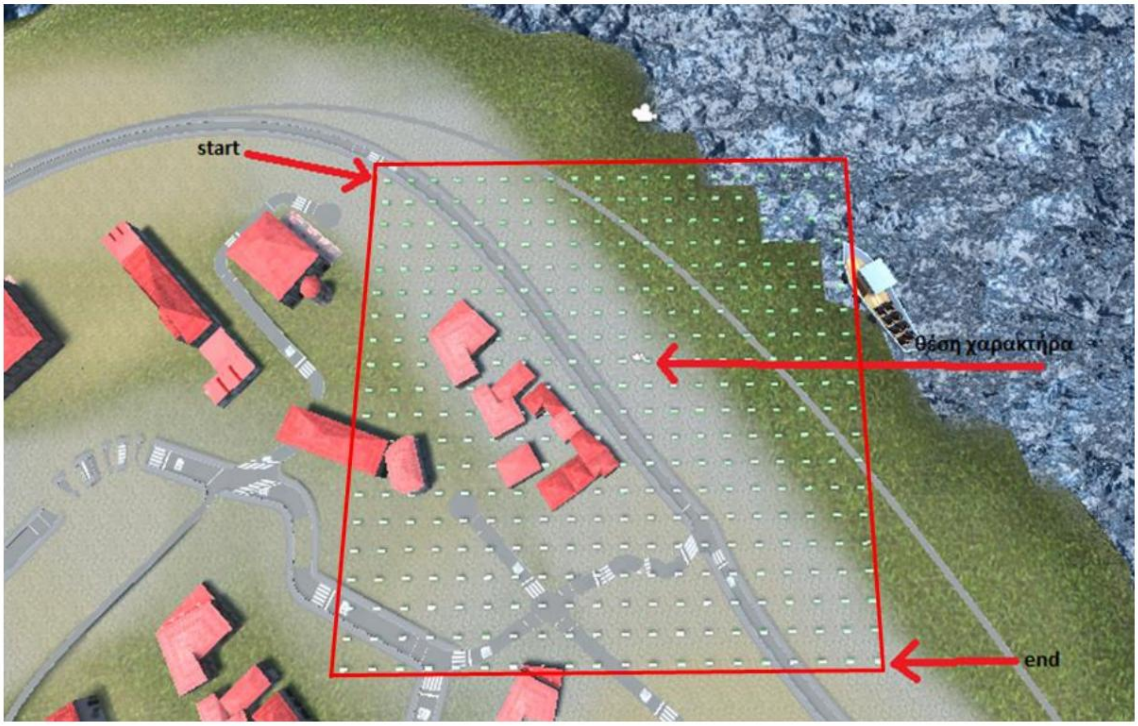


Figure 45. Start and end position of object creation. The square shown is the buffer size.

Then, after we know all the necessary information, the call the spawner. This in turn calls SetScale to process it size of the translucent Object and the colored Object. Then she is called setMaterialcolor during which the appropriate color is set to the colored Object, for example green for vegetation. And then the spawn object is whatever creates the Object and places it in the precomputed position.

At this point let's say that when calling the spawnObject it is not done simply the placement of the objects, but the creation of the Objects is also done at the same time each Object created goes into a list. So we have access to each created Object.

At this point the analysis of the 3d object creation for each data and the result is shown in Figure 46.

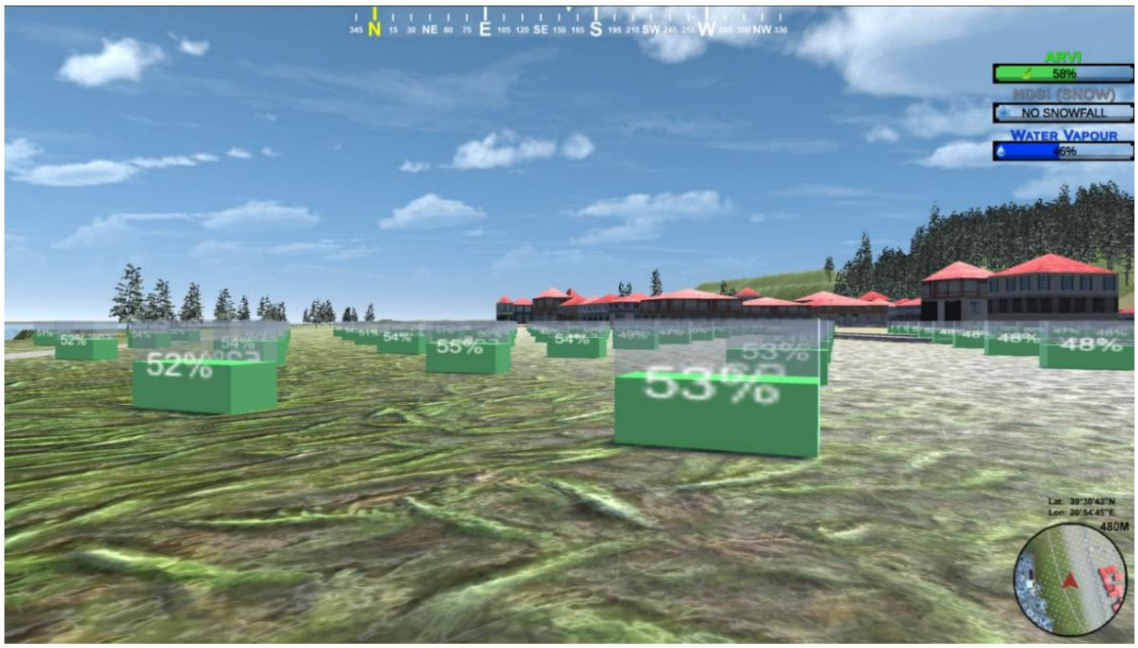


Figure 46. Final algorithm result. Hide NDVI bar.

We can observe the problem that is created. In depth it is almost impossible to distinguish the prices and understand what prices there are. So this Mode was built to display 400 data in 3d format where the user can move freely and observe. In case the user exits this area where they are placed data then the Mode is automatically disabled and the user has the ability to activate it.

To solve the aforementioned problem we implemented a extension of 3d visualization in which the user has the ability to choose whether to see the highest or lowest values for a percentage of data displayed to it. A necessary condition is that it already has activated the first mode that shows 400 3d objects and in can then activate the second Mode as an extension of this (Figure 47).

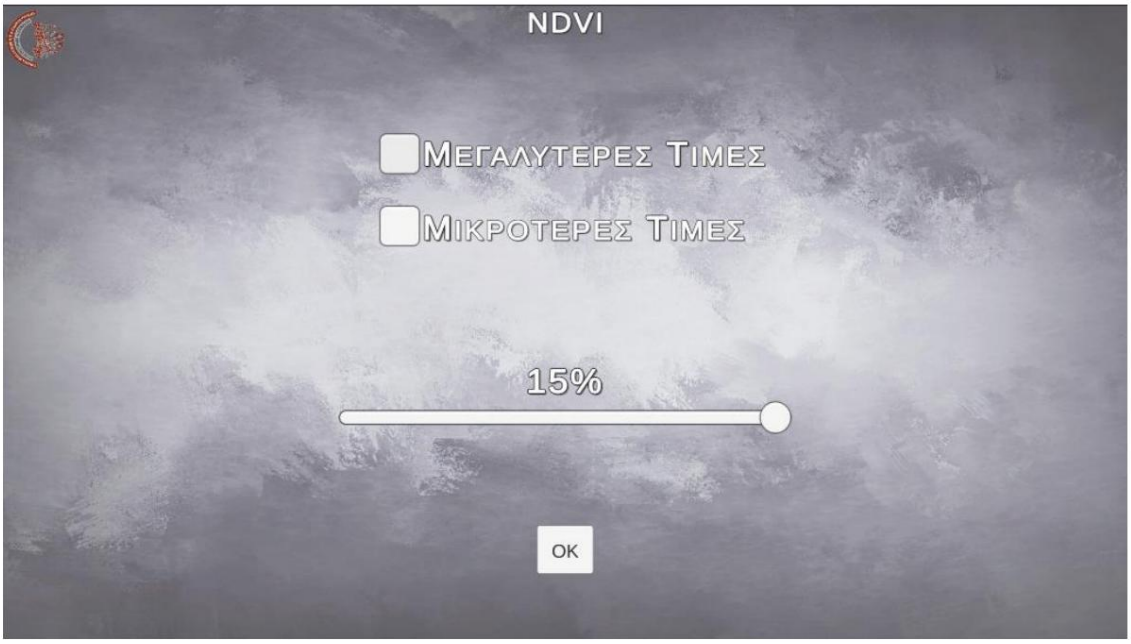


Figure 47. Initial mode extension screen. Selection of minimum or maximum Prices.

The screen that the user sees when the second Mode is activated consists of an Object With component photo, 2 objects with component check box button an Object that makes up the slider a textfield and an Object that is the Ok button.

Regarding the code, we connect the OK button with the function that we want it to run. So after this screen we have the representation of a percentage of the data by maximum or minimum values. This is what helps us list we made containing each Object that has been made. This list is sorted by the value of the data it represents. Depending on him number of data we want to display, we display x first or x last depending on whether we want maximum or minimum values (Figure 48).

Figure 48. Connection of Ok button with the execution function.

We will need a cylindrical Object with the exact same logic that we made the rectangles. Because the list of Objects Contains exactly the objects with exactly all their settings. So we have sorted Object in the list with their position so we can modify the new cylindrical object without reading the whole file again and doing the whole process that we did at the beginning, saving time and performance for the application.

The final result of the expansion of the first Mode is shown in the Image 49. Now we also have a better picture of the prices in the background.

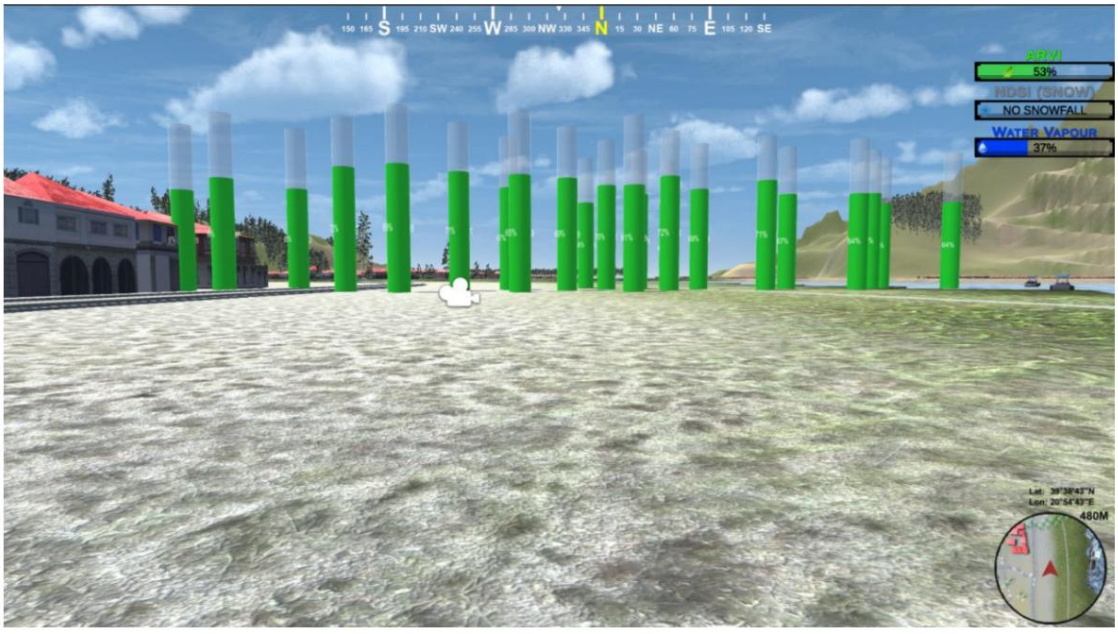


Figure 49. Visualization result of expansion of the first Mode.

3.4 Scripts and Functionality

3.4.1 Scripts

3.4.1.1 Analysis of Scripts

In order for the application to work, we must have given all the scripts the appropriate arguments from Object. A detailed analysis of each will not be made explain about a basic script and then we will give with pictures for the rest.

Initially scripts can enter any Object even if this it has nothing to do with the overall Project. We have separated those that is for the screen useful with those of the camera in 2 different Objects.

We will analyze the camera spawn script found in the object player data and it is also the largest in area (Figure 50).

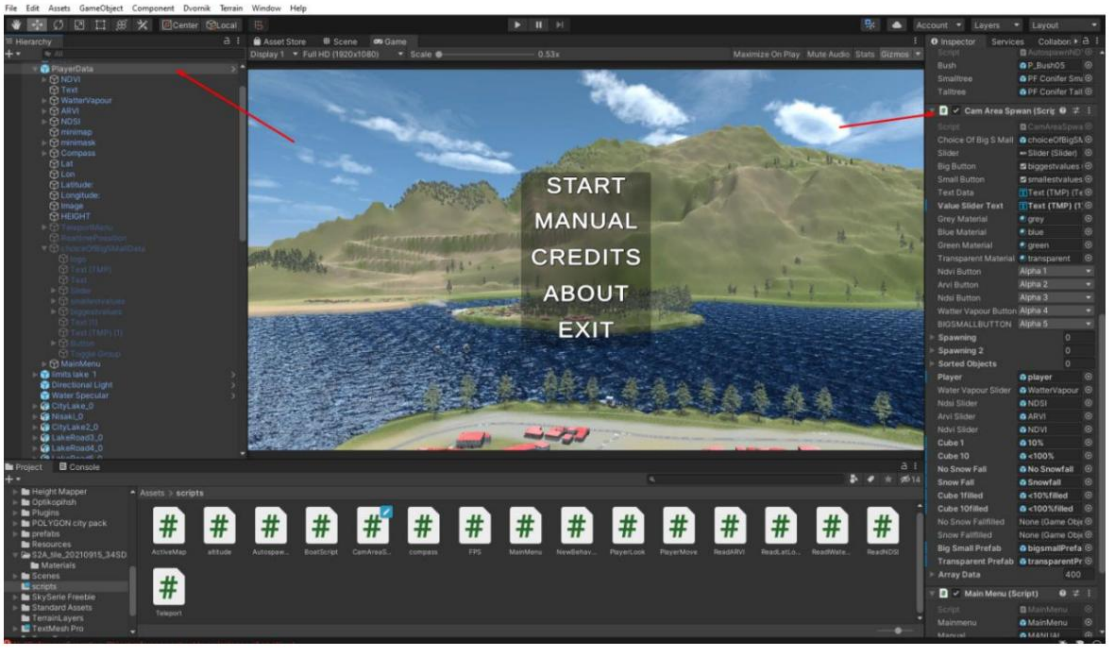


Figure 50. Script settings.

In the Inspector we can see all the available fields. First you have to go to the folder inside the assets that you call Prefab, And to the visualization folder. The prefab folder contains all the Objects we used and have been made prefab and in the visualization folder all the photos used like material.

We have named the fields with the same name as the Object to make it easy matching. For the script when we want to join a field with an Object we drag and drop in the field from the scene (sample scene) we will need it field choice of big small data, which is the object choice of big small data whatever is in the Player data, in the slider field the slider in the choice of big small data, for the field Big button and small button the Object Biggest values and smallest values, for text data and values slider text text (tmp) and text (tmp) (1), drag and drop as shown in Figure 51.

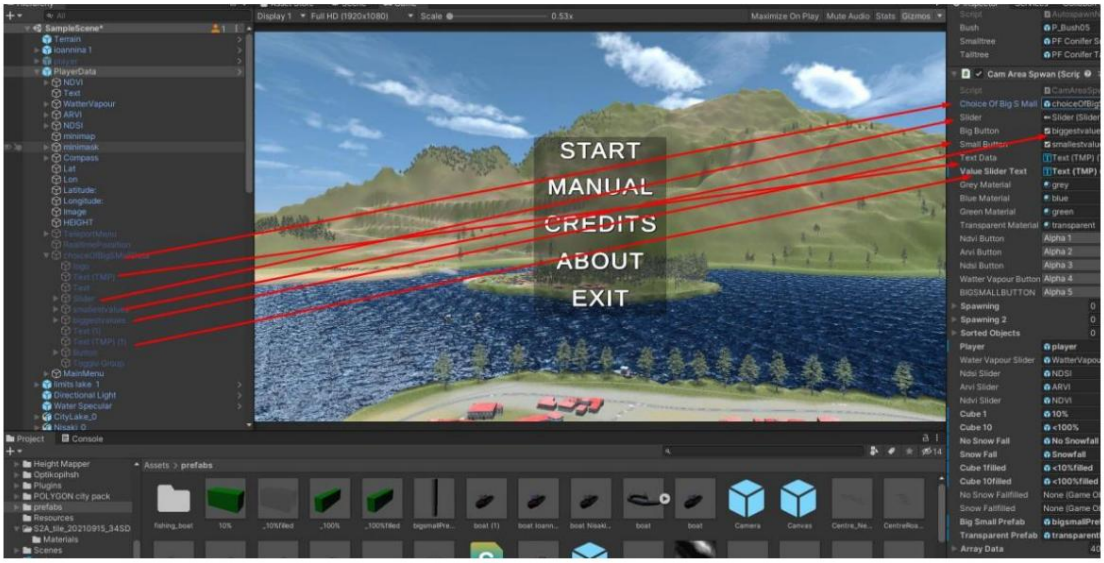


Figure 51. Script settings.

Accordingly, from the visualization folder we will take the materials as shown in Figure 52. We continue matching until all objects are be replaced. We do the same in all scripts.

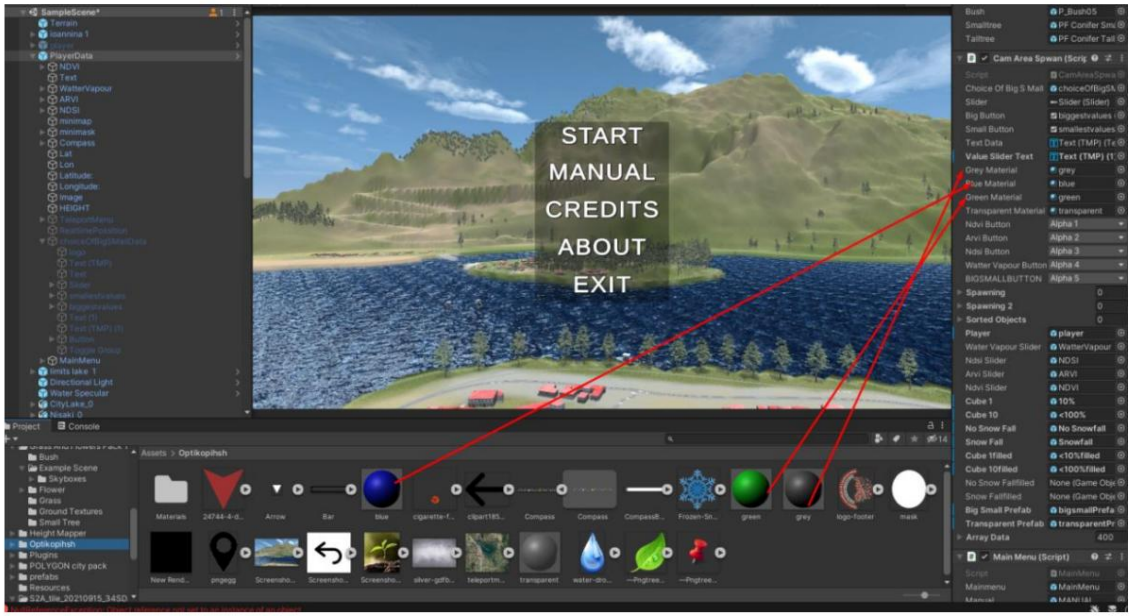


Figure 51. Script settings.

Here's a series of photos of all the scripts and objects you need to we enter .

Script in Object playerData:

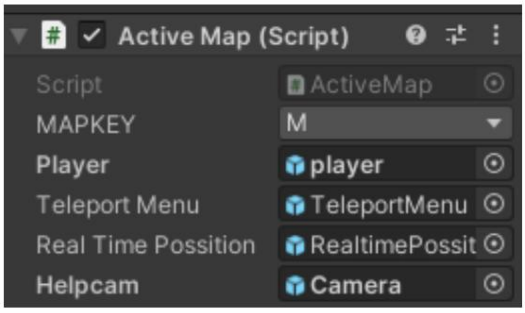


Figure 52. Script settings.

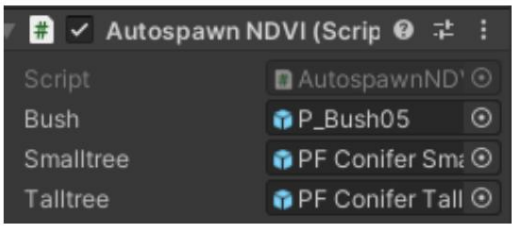


Figure 53. Script settings.

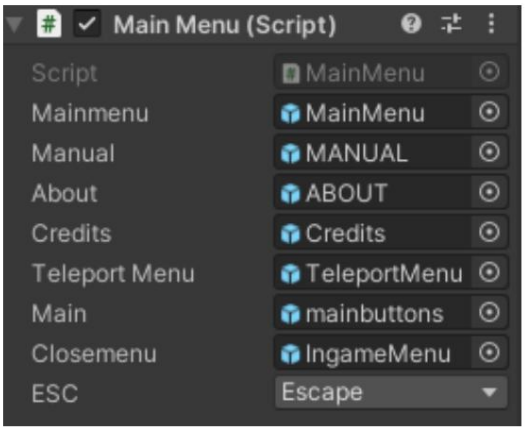


Figure 54. Script settings.

Script in the Player Object

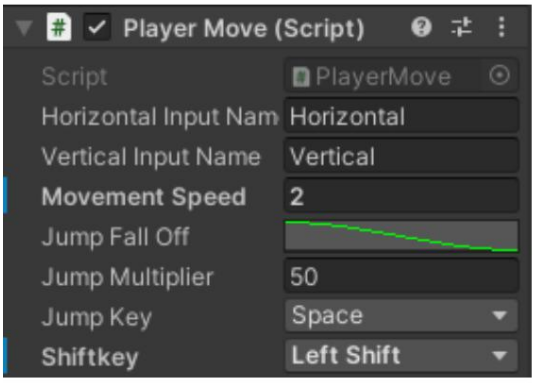


Figure 54. Script settings.

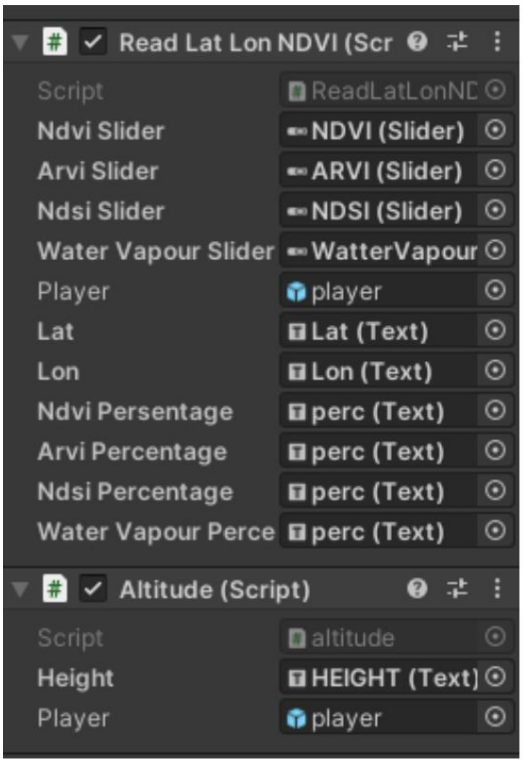


Figure 55. Script settings.

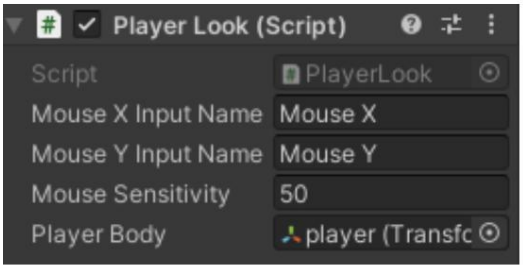


Figure 56. Script settings.

3.4.2.1 Creating a Forest Asset

One script we didn't mention at all is the autospwan script. The specific script needs only 3 inputs 2 Prefab trees and 1 bush. The script needs 4 coordinates that form any closed shape to create an area.

For this area for each coordinate it contains the file is opened vegetation, and finds the values that are matched just like we explained for 3D visualization. A random spawning pattern is then followed. Initially the area created by the 4 coordinates is divided into smaller regions, each of these regions contains a single value for an information like for example about Ndvi.

The pattern we follow is the following:

For 0% to 10% (0 to 0.1 value for Ndvi) no Object will be displayed

For 10% to 20% (0.1 to 0.2 value for Ndvi) it will appear in the area corresponding to a piece of information, a bush.

For 20% to 30% (0.2 to 0.3 value for Ndvi) it will appear in the area which corresponds to one piece of information, two bushes.

For 30% to 40% (0.3 to 0.4 value for Ndvi) it will appear in the area corresponding to a piece of information, four bushes.

For 40% to 50% (0.4 to 0.5 value for Ndvi) it will appear in the area which corresponds to a piece of information, four bushes and a small tree.

For 50% to 60% (0.5 to 0.6 value for Ndvi) it will appear in the area corresponding to one piece of information, four bushes and two small trees.

For 60% to 70% (0.6 to 0.7 value for Ndvi) it will appear in the area corresponding to one piece of information, eight bushes and four small trees.

For 70% to 80% (0.7 to 0.8 value for Ndvi) it will appear in the area that corresponds to one piece of information, eight bushes and one large tree.

For 80% to 90% (0.8 to 0.9 value for Ndvi) it will appear in the area corresponding to one piece of information, eight bushes and two large trees.

For 90% to 100% (0.9 to 1 value for Ndvi) it will appear in the area that corresponds to one piece of information, eight bushes and four large trees.

The way spawning is done is random, i.e. a random one is chosen number x, y, z within a limit of the region corresponding to a piece of information and the object is created there. Still, we do checks to avoid some millimeters so that they do not fall on top of each other and look more realistic. This script is easily modifiable so that one can change either the data file or replace the prefabs with others of your choice, even to make his own pattern or way of spawning.

3.4.2 Functionality

3.4.2.1 Basic First Screens

When starting the application the user is asked to choose between Menu (Figure 57).

Figure 57. Main menu.

The menu consists of the options:

Start

Manual

Credits

About

Exit

With the exit option the application terminates.

With the about option, the menu in image 58 appears.

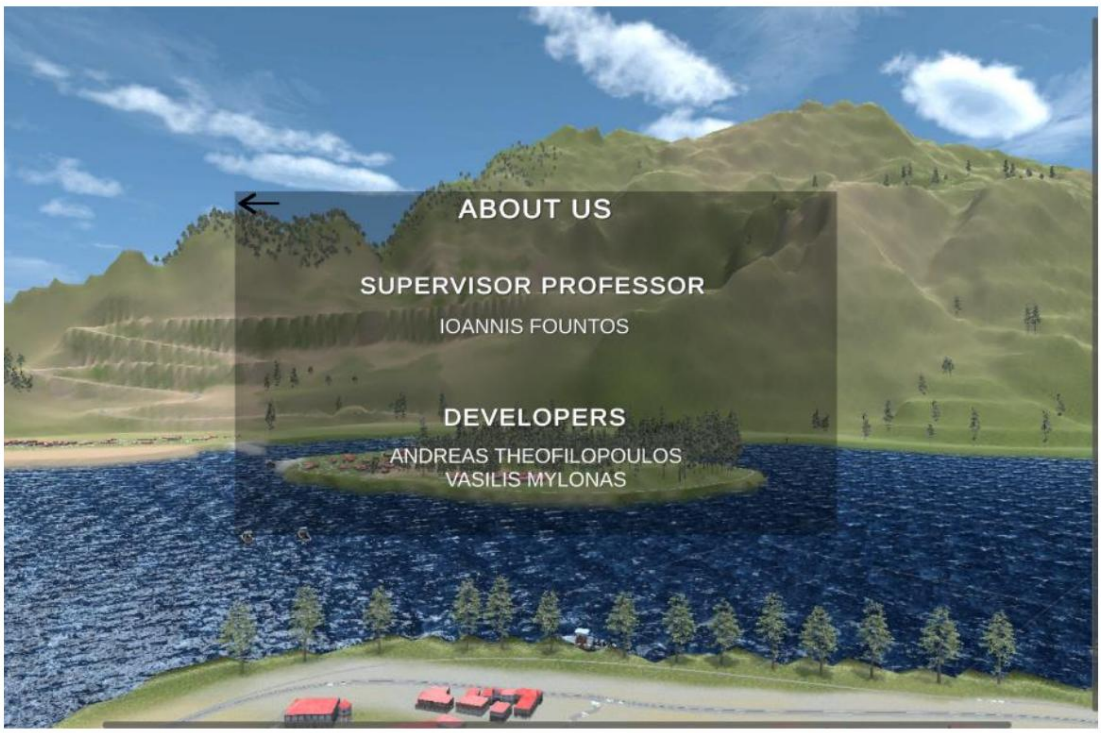


Figure 58. Main menu/ About.

With the credits option, the menu in image 59 appears.

Figure 59. Main menu/ Credits.

Selecting Manual displays the menu in image 60.

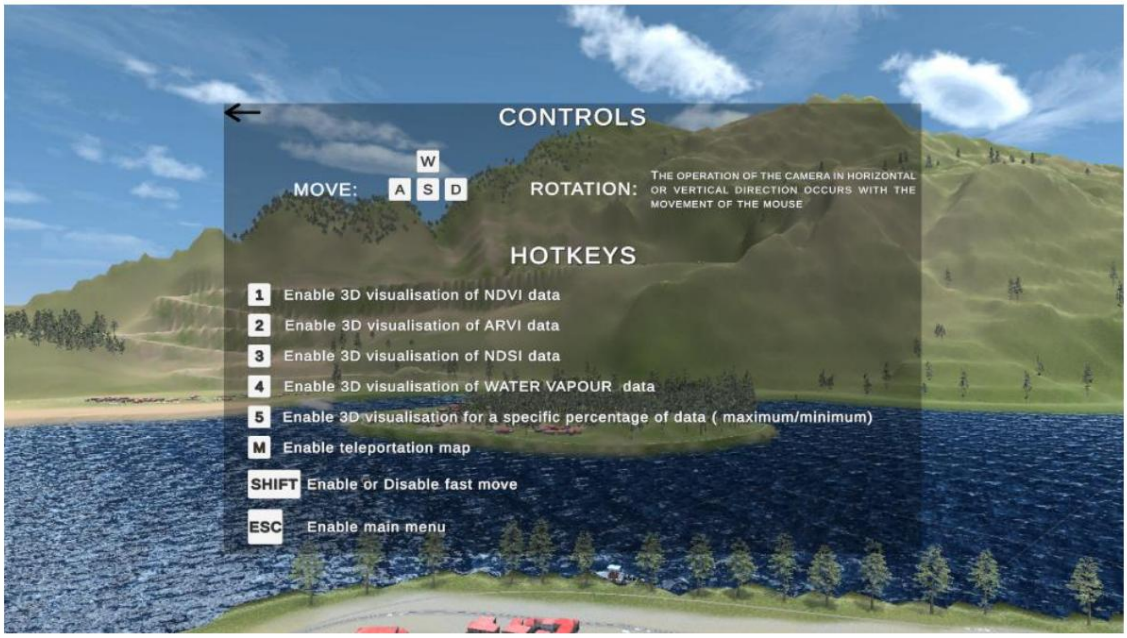


Figure 60. Main menu/ Manual.

With the start option, the application is launched. The first user's ability is to select a starting location on the teleport map whichever is available at any time with the M key (Figure 61).

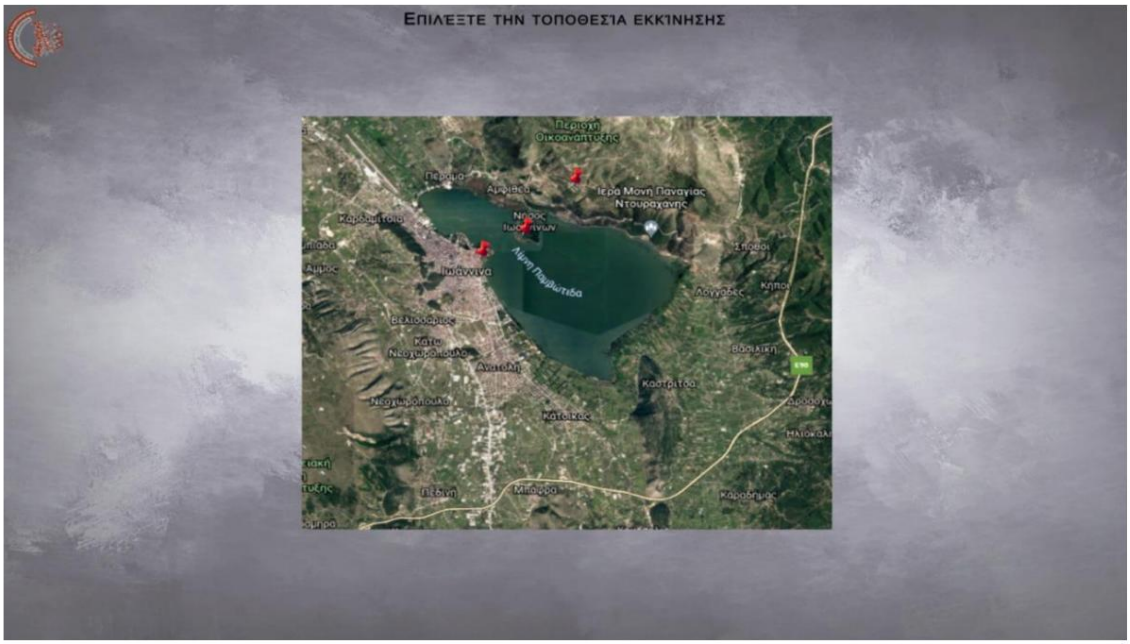


Figure 60. Teleport map.

3.4.2.2 Capabilities during interactive interaction.

After selecting a launch location the user is in the main part of the application that is its interactive interaction with the environment that has been created. Now the screen in front of him looks like picture 61.

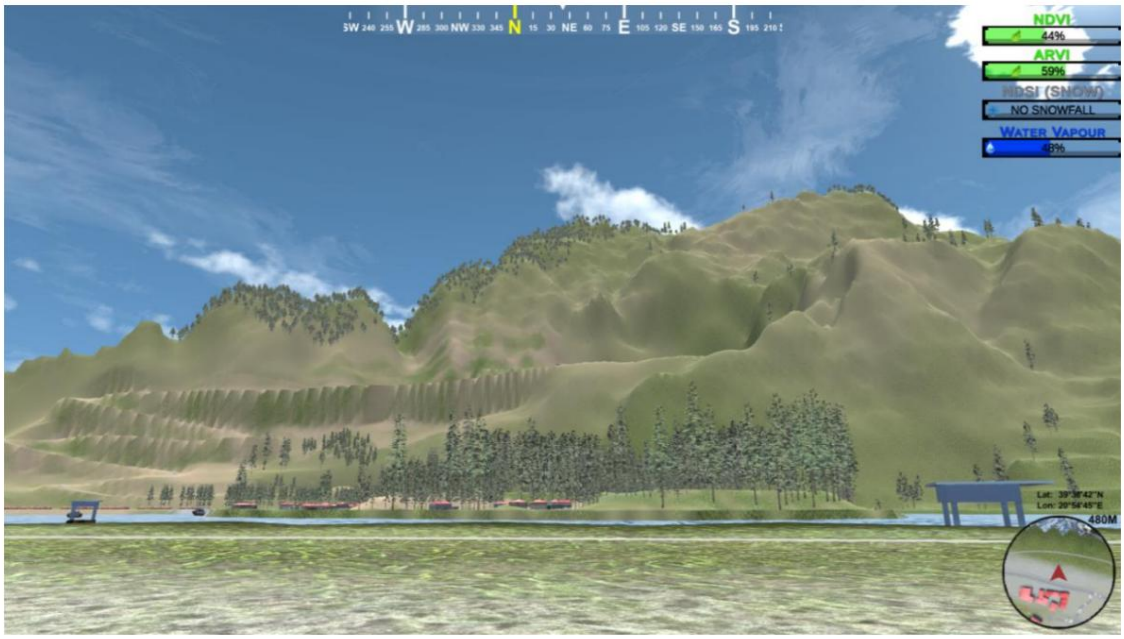


Figure 61. Main screen in game.

In the central and upper part we have the compass for orientation of the user. The data bars at the top right. The Mini map at the bottom right part along with the altitude and longitude information and latitudinal.

From this point onwards the user is given the possibility to activate the first Mode that concerns the 3d visualization of the data. This can activate it for any of our 4 data Ndvi Arvi Ndsi Water vapor with the keys 1-2-3-4 (1 for Ndvi 2 for Arvi 3 for Ndsi 4 for Water vapour).

After activating the Mode the user's screen will look like this picture 62.

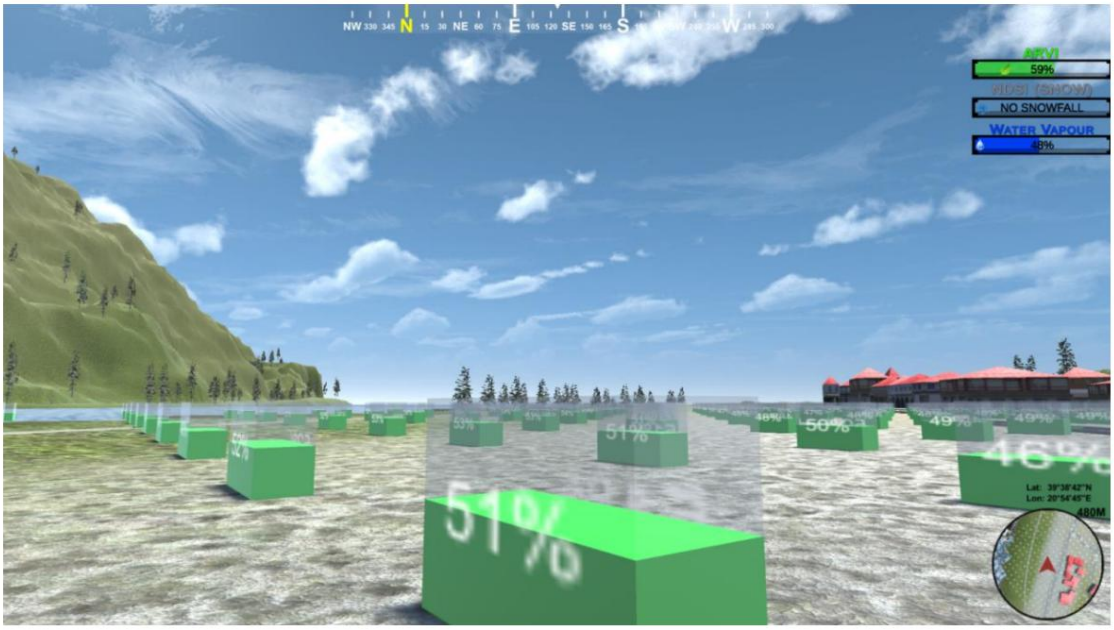


Figure 62. Enable 3d visualization (1 Ndvi key).

As long as the 3d visualization Mode remains active, ie the user remains within the area where the 3d objects are located, it is given the option to enable Mode expansion for the maxima and minima of one percentage of the data it is already seeing. This is done with key 5. After the activation of the extension the user's screen will look like image 63.

Figure 62. Enable 3d visualization extension (1 Ndvi key).

After saving his choices the user presses OK button will now see the new 3d models for the data (Figure 63).

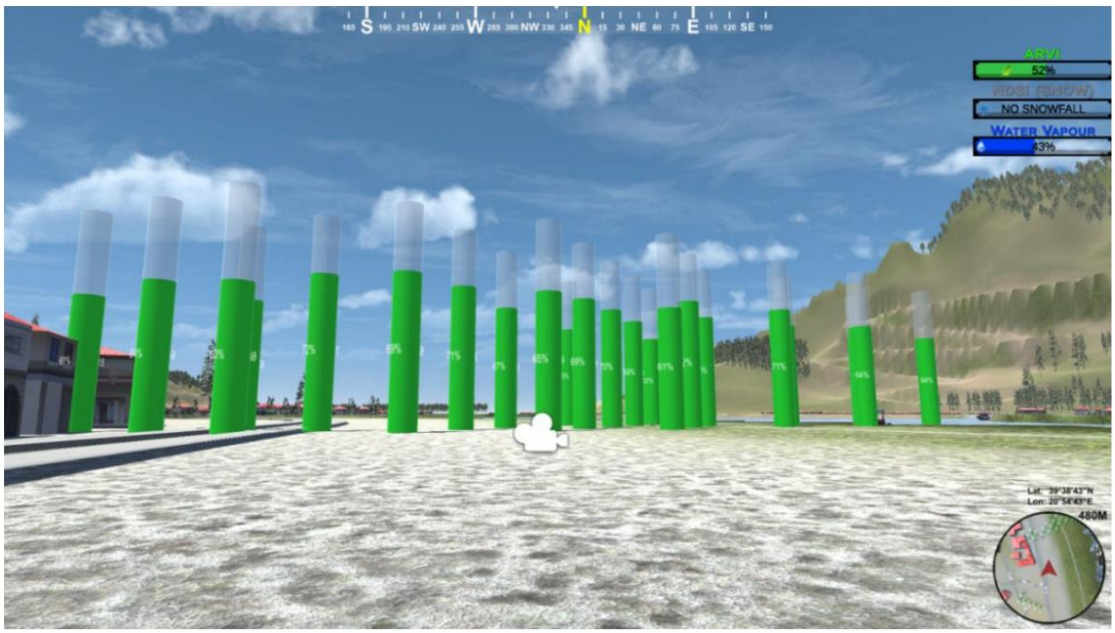


Figure 63. Result of enabling extension of 3d Visualization (key 1 Ndvi) .

Finally the user can move freely and interact with it world created along with the functions offered. The forest automatically generated based on Ndvi vegetation is shown in Images 64-65.

Figure 64. Ndvi database forest visualization.

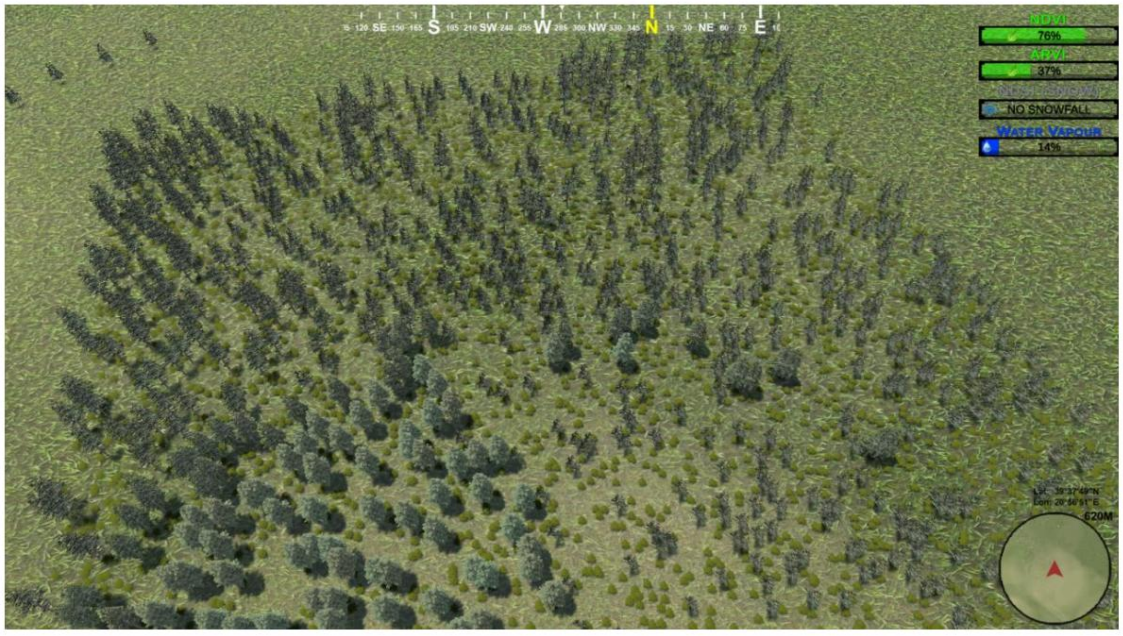


Figure 65. Panorama download Forest Visualization Ndvi database.

3.5 Performance

Unity renders new frames continuously. Usually 30 frames per second FPS in short is the minimum to aim for and 60FPS is ideal. Those numbers appear often because many devices have a screen refresh rate 60 hertz. We can't draw frames faster than that without turn off VSync, which will cause image distortion. Unless can achieve a consistent 60 FPS, then the next best rate is 30 FPS, which is once every two screen refreshes. A step lower would be 15 FPS, which is insufficient.

3.5.1 Application window statistics

The application window has a *Statistics* overlay panel that can be activated via the Stats toolbar button . It displays the measurements made for the last rendered frame. He doesn't tell us many, but it is the simplest tool we can use for to have an indication of what is going on. While in edit mode application windows usually update only sporadically, after something change. Refreshes every frame while in playback mode.

The statistics are for the graph with the torus function and the analysis to 100, using the default built-in performance pipeline, which we will refer to as BRP from now on. We have it enabled VSync for the game window, so refreshes are synced with the screen at 60 Hz (Figure 66-67).

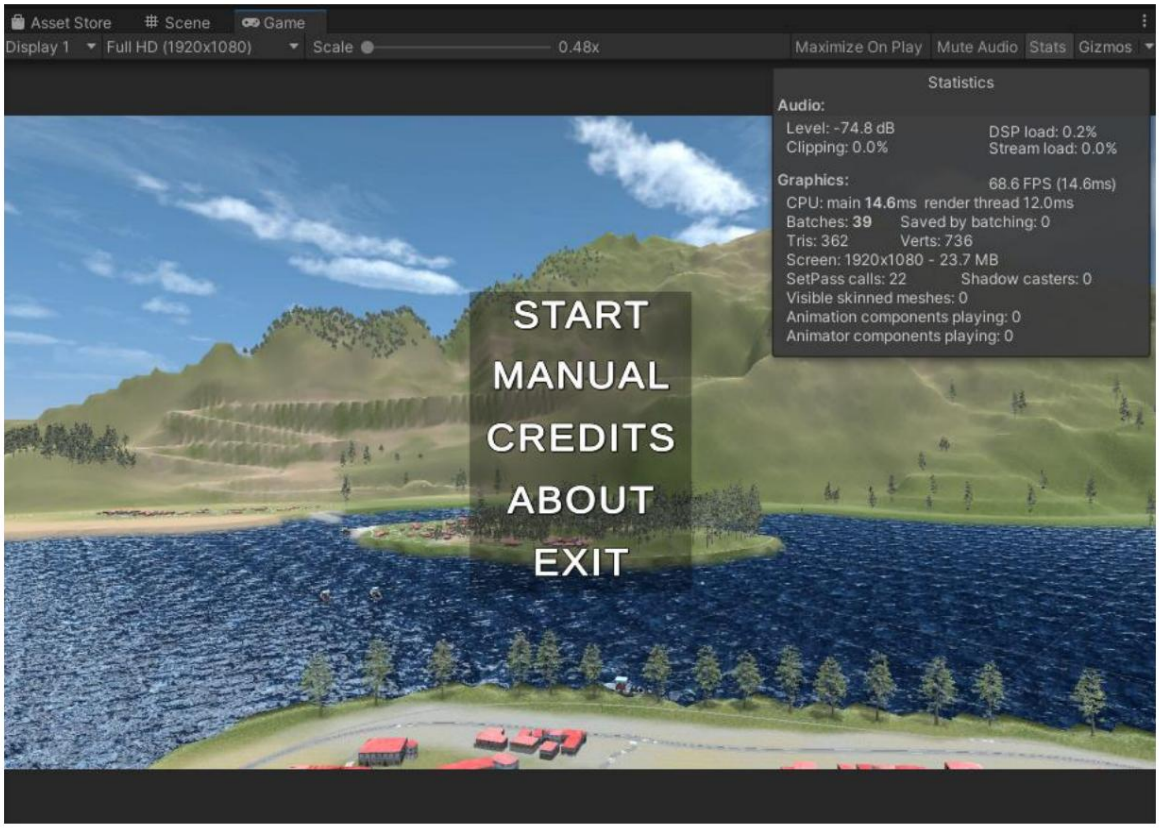


Figure 66. Presentation of menu statistics.

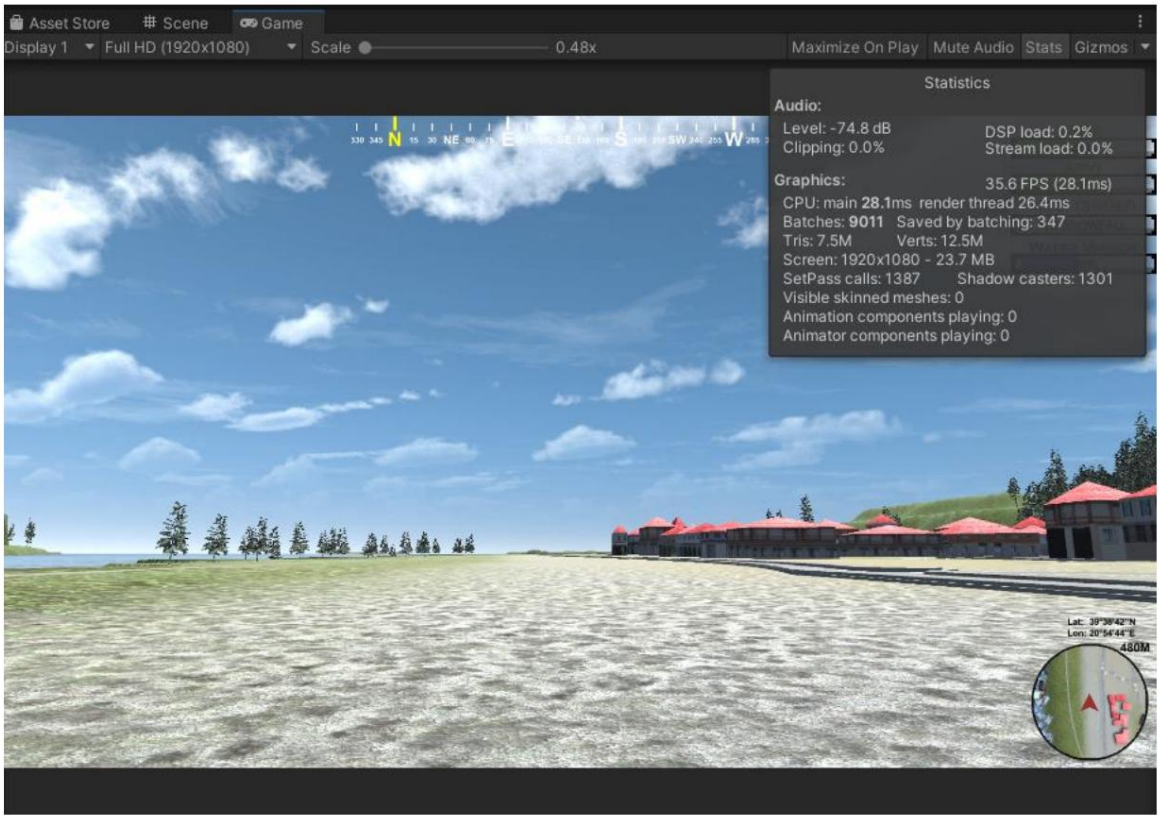


Figure 67. Presentation of in-app statistics.

The statistics show a frame in which the main CPU thread took 28.1ms and render thread 26.4ms. (Figure 67) In our case, indicates that the entire frame took 54.5ms to render, but the stat table reported 35.6 FPS, corresponding to CPU time. THE FPS indicator seems to take the worst time and assumes it matches with the frame rate. This is an oversimplification that only takes into account the side of the CPU, ignoring the GPU and the display. The actual frame rate is probably lower.

In addition to the durations and the FPS indicator, the statistics table shows also various details about what was rendered. There were 9011 batches, and 347 saved in batches. These are design commands that are sent to the GPU. Our chart contains 1301 points, so it looks like each point was scored three times. There were also 1387 set-pass calls, the which can be thought of as the GPU being reconfigured to render with different way, like with different material.

These values are not available to the user. For an average user these the prices are indifferent and there is a good chance that he does not know what he is seeing. But something that is interesting to be able to check is fps and MS. MS number expresses the duration of the frame, the duration of the FPS. For this one that's why we gave the user the option, if he wishes, by pressing key 6 to he can see his fps and Ms as he navigates the world (Figure 68).

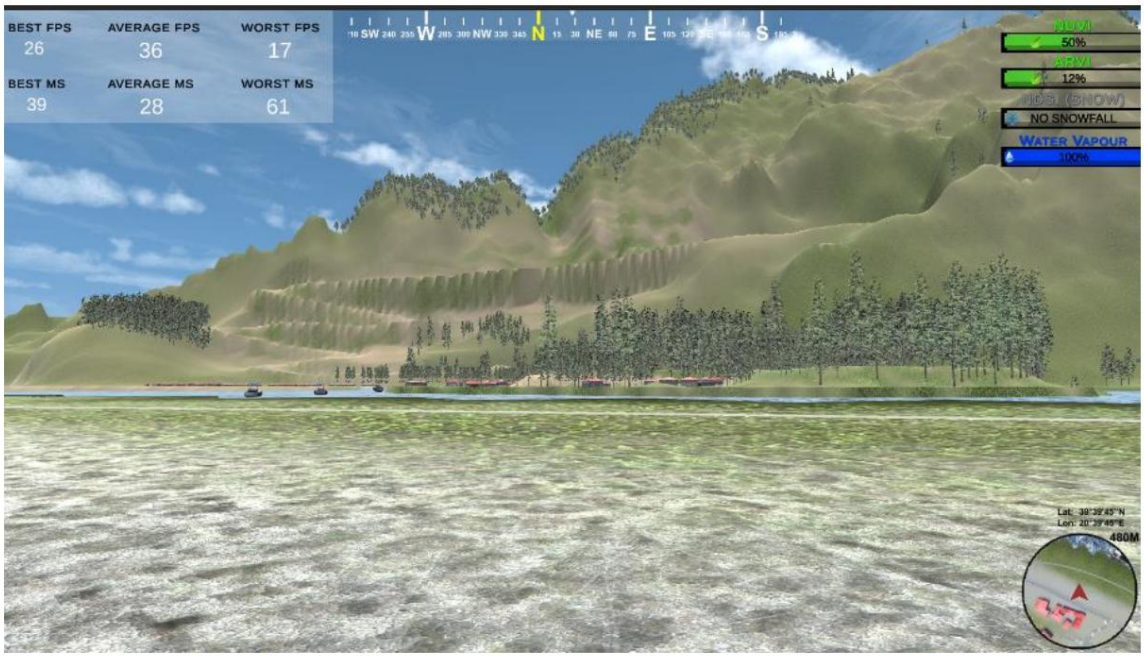


Figure 68. Presentation of in-app statistics.

Chapter 4. Problems and Dealing with them

As in any work and especially when tools are used it is difficulties and problems are expected to arise. Especially ours case, something that further complicated the situation was the appearance of the virus Sars-CoV-2 and specifically the quarantine period implemented because of it its spread. During this period we had to lodge from Ioannina and let's move. ISPs made it difficult for us even more since we didn't have internet in our homes for 2 months since we chose not to live with our parents. In total we lost a semester where we could work. Fortunately, with the help of the supervisor our teacher, Mr. Fundo Ioannis, as well as the secretariat we managed to we are extending our thesis by one semester after one was lost without being able to do anything about it. Of course, in addition to we also faced problems of a special nature, regarding its content work.

4.1 The problem of loading files

The files we mentioned above are files with excessive information. For loading these files we tried different techniques.

First we tried to load the whole file into memory until the end solution. This was delaying the program about 10-12 minutes on startup of. Apart from that this makes the application now heavy, in the sense that it won't it could run on weaker systems with limited ram. also this time is long and would tire the user in waiting.

The next most optimized solution was to create a Buffer around it by the user. However, its regular renewal was an inevitable part for her maintaining data accuracy. This reduced our waiting time to 3 minutes. But then it created us lag spikes that created a bad experience for the user since after a certain number of steps o user could not move and had to wait for Buffer to refresh for the data.

Finally we arrived at the most efficient technique. Now for each position user we refresh a value for each of our data by opening each file and finding the exact position in the table we want to load. This reduced the waiting time to 0 seconds. Basically the user now can move continuously and view data without lag. But everything has a loss. In this particular case, this loss it is the performance. In the chapter with the analysis of Performance we see that the cpu processing takes several seconds. This affects the low fps when moving the user on the map. It does not render the application unplayable, but not the best result that could be achieved.

We will analyze the best version as a solution to the problem in capital of the improvements.

4.2 Completion of the terrain

Terrain is one of the main pillars of our work.
Initially, we based it on the most correct implementation in order to have the accuracy of them data. This resulted in a giant difficult terrain being created technically separable. The cityEngine program doesn't exactly give us that the entire area that we wanted to present as a city. In many places, due to incomplete information, we are not given the possibility to export file (buildings). This results in everything having to be done manually with duplicates object.

Initially we tried to fill as many areas as possible. In then we started to understand the implications of multiple objects of their own on the terrain. In particular, we once again had a drop in fps. The solution to this was to introduce the camera render scale. Because of the huge however, urbanization was deteriorating. Now the user in the background does not he could distinguish neither mountains nor lake nor islet from a reasonable distance. Whenever we did not adopt this solution. We left the terrain as much as possible full was done in terms of buildings so as not to affect the overall performance.

Another problem was the introduction of trees. The script that we created could be used for this problem as a solution. But this script selects an area and creates a forest. In so if we selected the entire terrain it would not be able to stand out areas with houses or the lake. Whenever our script solution here is not good. Even if we manually select different areas it is not a good solution. Unlike the terrain feature, which we analyzed in the terrain chapter, terrain allows us to add trees and vegetation without it to have an impact on performance.

The terrain does not create a new Object for each tree, but the embeds within it (Trees+terrain=one Object). This is the main reason why performance remains stable using terrain settings. She it is also the main difference with the script we created. This script for each a tree or bush creates an Object for it. For a 20x20 area it will need to create about 8000 Objects. The precision and the best urbanization always affects performance.

4.3 Understanding geotiff

The biggest problem we were called to face was theoretical. Understanding about a record that is not like the others consumed us most of the time we spent in total. At first we couldn't to understand exactly what information such a file contains.

We had to take a step back and read about her comprehensively geotiff concept. Little by little we managed to build a satisfactory background so that to understand exactly what a geotiff file is. What information it may contain and how we will manage them. After understanding the theoretical part h solution to our problem was simpler than we thought at first.

Through Python we were able to load the files and save with any structure we wanted. But one piece we couldn't find to understand is whence the effect of actions arises between the Bands of a geotiff.

No matter how hard we searched we could never find and understand the logic. Because for example the actions for ndvi between Bands are these that were mentioned and how we understand that they will bring us such result. This lack of information resulted in us not being able to we do our trial operations to calculate a file type of our own which not offered online on any site. We believe that the analysis of geotiff it is a science in itself in the part of meteorologists.

Chapter 5. Conclusions and

Improvements

Extensions

5.1 Extensions and improvements

When we started developing the app our goal was an app that which is amenable to extensions as well as improvements. For this reason well structured functions were used so that a 3rd party could do them edit and modify it accordingly. Extensions and improvements though they do not exist only in the part of the script but also in the whole of the diplomacy in the unity environment.

5.1.1 Extensions

Something that would be very interesting, is the filling of the terrain with manual mode (mainly for Performance issues). This includes her planting trees in all the necessary areas, exporting new buildings for the expansion of cities. The city piece is a major extension of one there are already created blocks based on the roads that have been placed which could be filled with new different buildings for the better visualization of the city.

One more piece would be adding new data to his screen user and their 3d visualization in space. So the user will have the ability to have access to even more information than it has as now. Which makes the permeability with the space around it still better.

It could still be added from the Unity asset HDRP (High Definition Render Pipeline). The High Definition Render Pipeline (HDRP) is a Scriptable High-fidelity Render Pipeline built by Unity to targets modern (Compute Shader compatible) platforms.

HDRP uses Natural Lighting techniques, linear lighting, HDR lighting and a configurable hybrid delay/forward architecture Tile/Cluster lighting. It provides the tools needed to create them applications such as games, technical demonstrations and animations at high graphic templates.

HDRP is based on the principle of physical performance which helps to achieving a realistic result. This effect is achieved due to of the fact that all objects in the scene receive the same lighting and the material interacts properly with lighting created by any source.

A good example of applying HDRP tools is a reconstruction VR of the Great Synagogue of Slonim developed by Exposit. Mr the goal of this project was to achieve the most realistic demonstration of it architectural monument while maintaining high quality graphics.

The achieved realism of the developed solution and the impressive image quality contribute to the creation of a completely new interactive visualization experience. This means that it makes a significant upgrade for the better visualization of our application as well as its better experience user

5.1.2 Improvements

In terms of the application, there is a lot of room for improvement. One of the most important improvements would be creating an api to which it will be loaded the data in table format. With request the application receives the appropriate data immediately and quickly. This will increase fps and the application's system requirements.

Another improvement would be creating custom 3d models for the projection of data in 3d format in space. In this way the user's struggle with these data it would be visually more beautiful.

Improvement could also exist in the terrain with different Layers either again custom or paid. Since the terrain is the main pillar will it would dramatically change the whole effect with more premium Layers that would do it terrain to look more realistic.

While we searched, the representation of the city with cityEngine seemed to us the most good case. Based on what we searched a Premium version is provided for full hd models the replacement of the existing ones with the ones we have certainly will he made the difference regarding the urbanization of the city.

Finally, given that all of the above has been implemented its transition application in a vr environment would be a good idea. The changes wouldn't be that many many since the main changes concern the camera and are not so related with the rest of the Objects inside Unity.

5.2 Conclusions

In closing this paper, we would like to mention that it is a informative application where the user can see environmental differences conditions but also to have fun exploring the area that has been created for this purpose. This makes the app more familiar even for the little ones ages since it looks quite similar to a modern computer game. Applying the improvements mentioned above could constitute a distinct concept for presenting environmental elements in various websites and weather services.

5.3 Software Evaluation

To evaluate the software, we explained to users the capabilities of the software and then let them use it and then they were asked to answer a questionnaire. Average 70% answered positively to the majority of questions with a score of 9-10/10 while 25% responded almost positively with scores above 5/10 and just 5% negative with scores below 5/10. A series of responses is heard users (Figures 69-87).

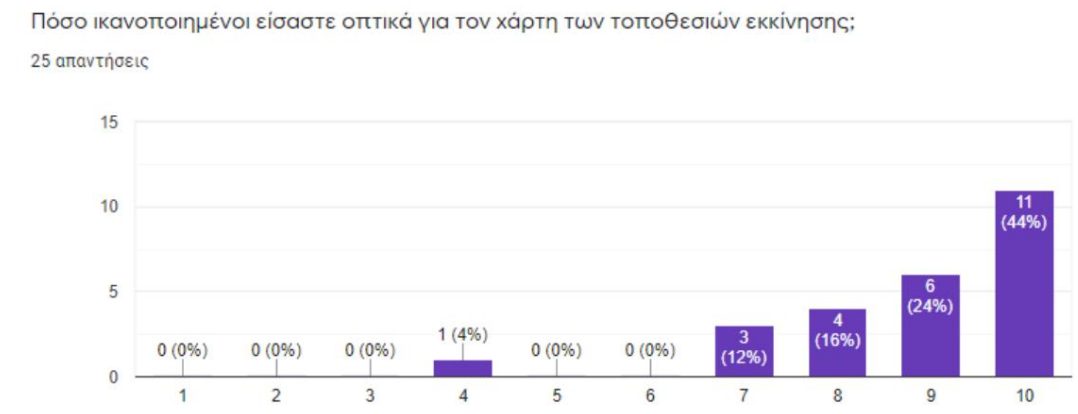


Figure 69. Software evaluation.

Figure 70. Software evaluation.

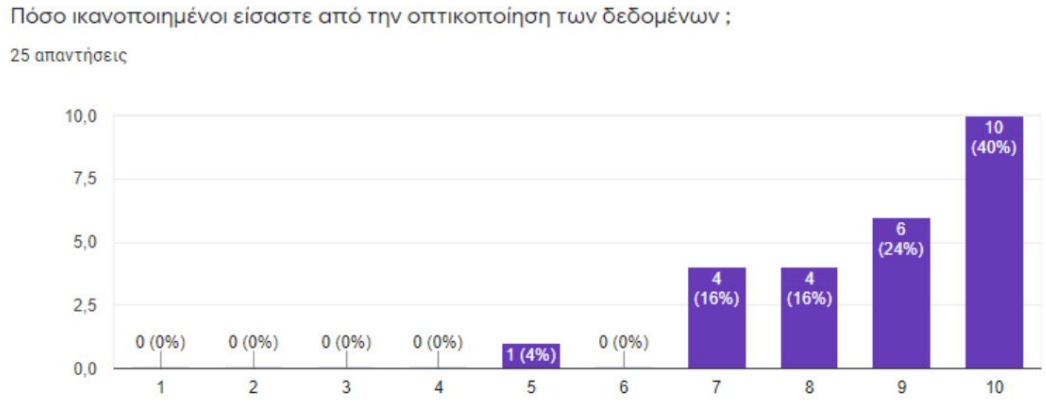


Figure 71. Software evaluation.

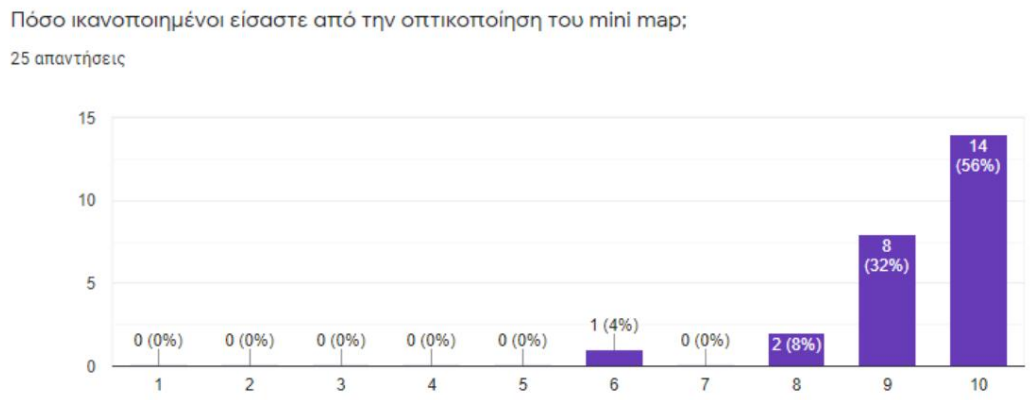


Figure 72. Software evaluation.

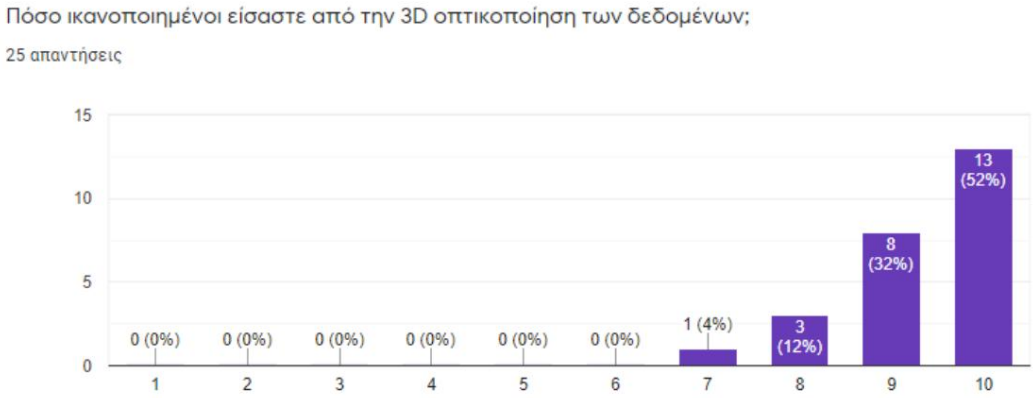


Figure 73. Software evaluation.

Figure 74. Software evaluation.

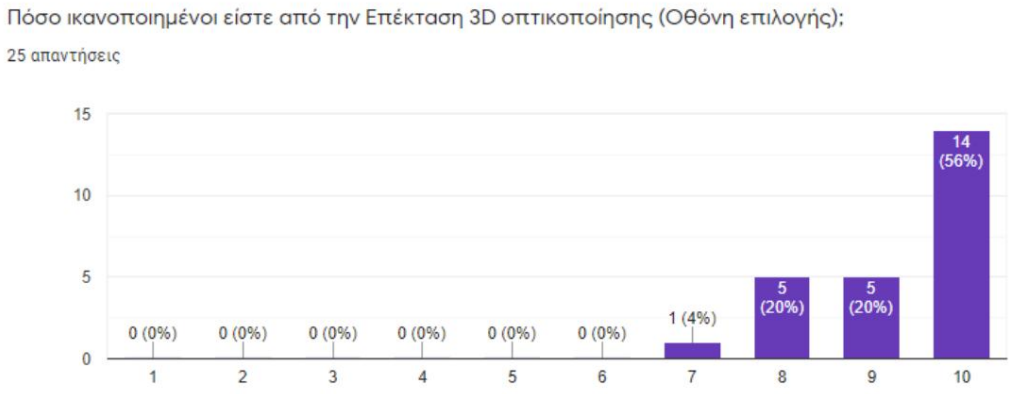


Figure 75. Software evaluation.

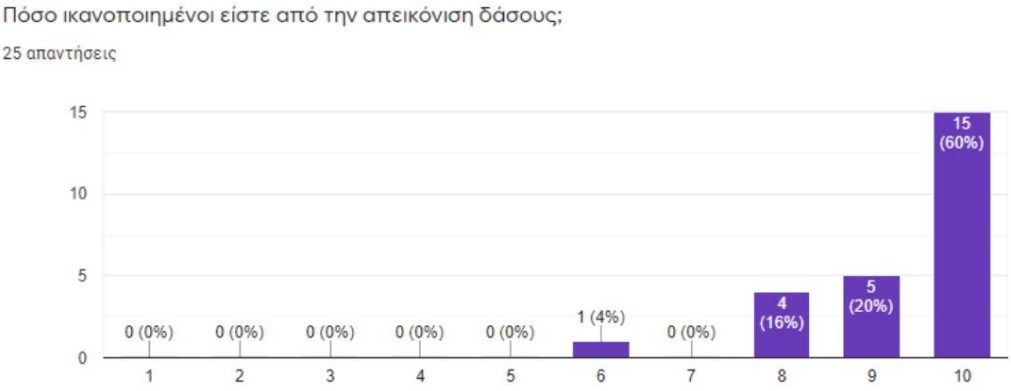


Figure 76. Software evaluation.

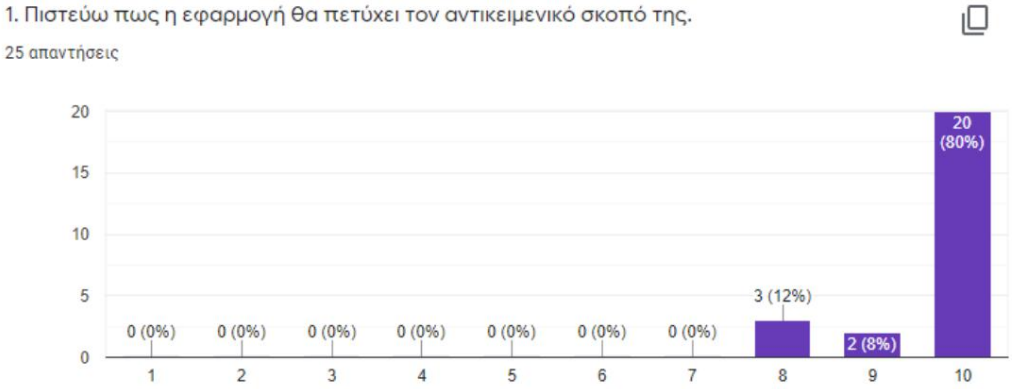


Figure 77. Software evaluation.

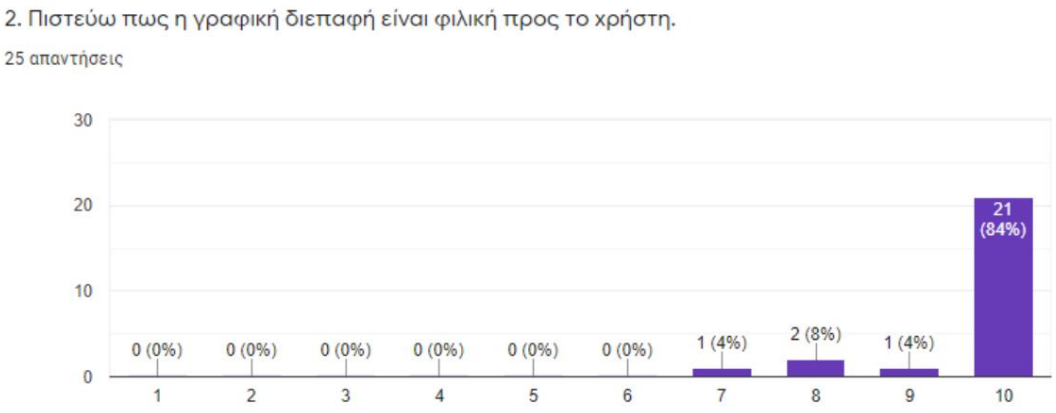


Figure 78. Software evaluation.

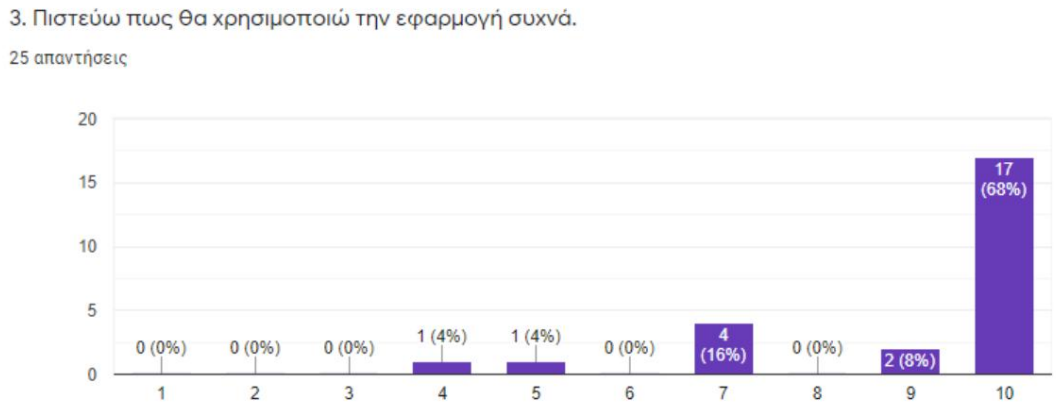


Figure 79. Software evaluation.

Figure 80. Software evaluation.

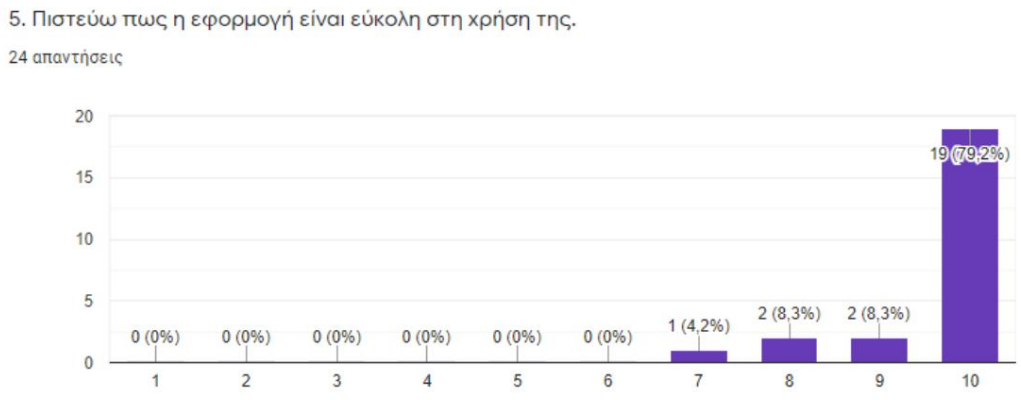


Figure 81. Software evaluation.

Figure 82. Software evaluation.

Figure 83. Software evaluation.

8. Πιστεύω ότι υπάρχουν ασυνέπειες σε αυτά που προσφέρει η εφαρμογή.

25 απαντήσεις

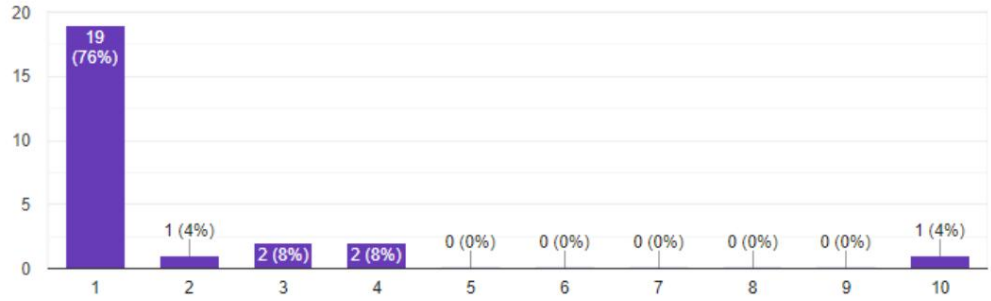


Figure 84. Software evaluation.

9. Πιστεύω ότι η εφαρμογή απευθύνεται σε ευρύ κοινό (από παιδιά εώς ηλικιωμένους και Α.Μ.Ε.Α).

24 απαντήσεις

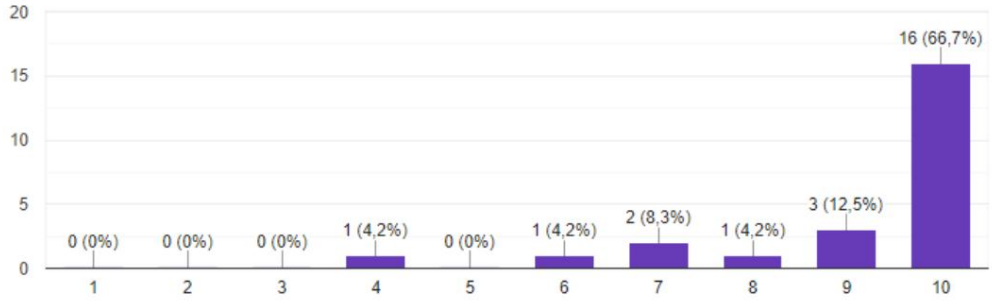


Figure 85. Software evaluation.

10. Πιστεύω ότι οι περισσότεροι χρήστες δεν θα δυσκολευτούν να μάθουν να χρησιμοποιούν την εφαρμογή.

25 απαντήσεις

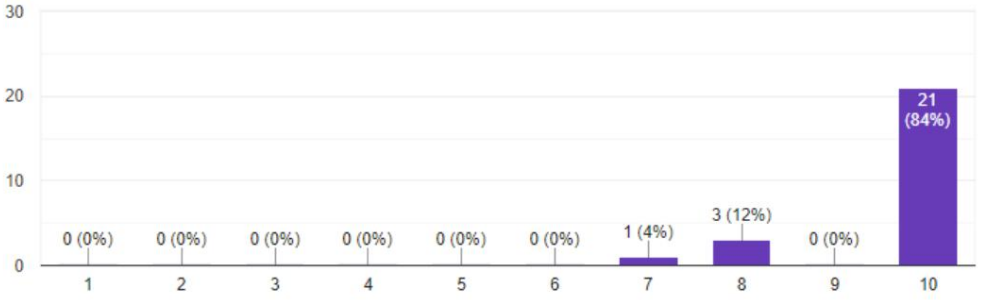


Figure 86. Software evaluation.

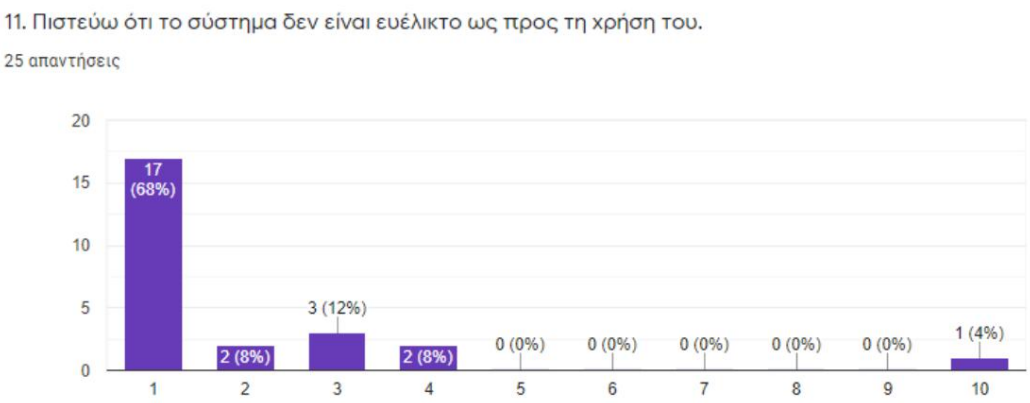


Figure 87. Software evaluation.

Bibliography

[1] Josh Petty, *What is Unity 3D & What is it Used For?:* https://conceptartempire.com/what-is-unity/

[2] Wikipedia, *Unity Technologies:* https://en.wikipedia.org/wiki/Unity_Technologies

Wikipedia, *Unity (game engine),:*
[3] https://en.wikipedia.org/wiki/Unity_(game_engine)

Josh Petty, *What is Unity 3D & What is it Used For?:* **[4]** https://conceptartempire.com/what-is-unity/

[5] Wikipedia, *Unity (game engine):* https://en.wikipedia.org/wiki/Unity_(game_engine)

[6] Unity Manual, *Using the Asset Store:* https://docs.unity3d.com/2018.3/Documentation/Manual/AssetStore.html

[7] Study Tonight, *GameObject elements in unity3D:* https://www.studytonight.com/3d-game-engineering-with-unity/elements-of-unity3d

[8] Unity Manual, *Introduction to components:*

https://docs.unity3d.com/560/Documentation/Manual/Components.html

[9] Unity Inspector:

https://docs.unity3d.com/510/Documentation/Manual/Inspector.html

[10] Unity Manual, *Standard Assets.*
https://docs.unity3d.com/560/Documentation/Manual/HOWTO InstallStandardAssets.html

[11] Height Mapper:

https://docs.unity3d.com/Manual/terrain-Heightmaps.html

[12] Trees

https://assetstore.unity.com/packages/3d/vegetation/trees/conifers-botd-142076

[13] Grass and Flowers Pack1:
https://assetstore.unity.com/packages/2d/textures-materials/nature/grass-and-flowers-pack-1-17100

[14] Terrain layers:
https://docs.unity3d.com/Manual/class-TerrainLayer.html

[15] Terrain sample asset pack:
https://assetstore.unity.com/packages/3d/environments/landscapes/terrain-sample-asset-pack-145808

[16] Easy Roads 3D:
https://docs.blender.org/manual/en/latest/modeling/meshes/structure.html,

[17] Sky freebie:
https://assetstore.unity.com/packages/2d/textures-materials/sky/skybox-series-free-103633

[18] What is Geotiff:
https://www.omnisci.com/technical-glossary/geotiff

[19] Introduction to Raster Data:
https://datacarpentry.org/organization-geospatial/01-intro-raster-data/index.html

[20] Raster bands:
https://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/raster-bands.htm

[21] Landviewer:

https://eos.com/landviewer/

[22] Rasterio:

https://automating-gis-processes.github.io/CSC18/lessons/L6/reading-raster.html