Visualizing Fluvial Erosion with Badlands

Kilian W. Liss

***Abstract*—We present data produced by fluvial erosion simulations with *Badlands* [1], and discuss various methods of visualizing the data for various types of audiences including researchers in computational geology, biodiversity or computer graphics. We provide various techniques for visualizing the data in *pyvista* or *blender*, and discuss their strengths and weaknesses with regards to the target audience. We discuss the symbolic representation for one of our visualizations using the semiology of graphics, and provide an alternative visualizations based on the symbolic representation.**

I. Introduction

Fluvial erosion and tectonic uplift are significant forces that determine the evolution of landscape dynamics over geological timescales. Although the physics of landscape dynamics is well understood, simulating it directly is extremely computationally demanding, and so alternative algorithms have been proposed. *Badlands* [1] is an open source python based framework that simulates sediment transport and mountain growth from tectonic uplift over geological timescales. It makes use of the *stream power equation* [4], which is the most widely accepted model to simulate erosion processes over geological scales without being too computationally demanding. It states that the rate of change in height for a point $\mathbf{p}$ is given by:

$$\frac{dh(\mathbf{p})}{dt} = u(\mathbf{p}) - kf(Q_s)A(\mathbf{p})^m s(\mathbf{p})^n \tag{1}$$

where $h(\mathbf{p})$ is the height of a point $\mathbf{p}$ on our landscape, $u(\mathbf{p})$ defines the tectonic uplift, $A(\mathbf{p})$ defines the drainage area, $s(\mathbf{p})$ is the local slope and $f(Q_s)$ represents various plausible models for the dependence of river incision rate on sedimentary flux. k, m, and n are constants

To test and verify various erosion models, the data produced by such simulations need to be visualized so that geologists can confirm the plausibility and validity of such models. Researchers in the field of biodiversity might also be interested in this type of data, since the habitability of various plant and animal species depend on geological factors such as mountain height and slope direction, which influence the temperature and sunlight flux of local ecosystems. Additionally, developers in computer graphics might also be interested in landscape data, as it provides means for procedural generation of realistic terrain maps that can be used in video games and/or movies. The data can also be animated for educational purposes, as it provides a very intuitive visualizations of how mountains grow.

II. Generating the Data

To generate our data, the transport limited fluvial erosion algorithm provided by *Badlands* [1] was used, where the river incision rate on sedimentary flux $f(Q_s)$ is defined to be almost parabolic. The simulation ran for 10 million years, on a grid with dimensions of 100×100 km^2, with a resolution of 100×100 m^2. The initial landscape and a constant (over time) tectonic uplift map was produced by the blender add-on *A.N.T Landscape*. The precipitation was defined to be 1 metre per year over the entire simulation, and other parameters such as soil erodibility were left as a constant over height.

Whilst running the simulation, a *.hdf5* data file was written every 20 000 years, leading to a total of 500 files. Our data contains the following attributes:

- **XYZ coordinates** - These 3 attributes specify the spacial location of each point of our terrain in a cartesian coordinate system. Cartesian coordinates can be ordered, their differences can be calculated, but since they do not have a true zero (negative values are allowed), they are considered to be of interval type. The cartesian coordinates of our terrain data could be plotted as a point cloud, however a 3D mesh produced by Delaunay triangulation of the points is more suitable at depicting digital terrain maps, since it gives the terrain a surface, making the visualization more similar to how the human visual system perceives terrains in nature.
- **Basin** - This attribute categorizes points on the terrain together whose water will end up flowing into the same basin. Since basins can not be ordered, this attribute is considered to be of nominal type and is best depicted by coloring faces on the terrain mesh.
- **Discharge** - Indicates the volume of water that is flowing through a given point on the terrain. It is of ratio type as it can be ordered, differences can be calculated and has a true zero when no water is flowing through a given point. It's value is close to zero for most points on the terrain, except for points where a river is flowing. It could be depicted using color, although a glyph of cylinders (or other shapes) with size and color modulated by the discharge value is also appropriate, as this leaves mesh color available for representing other attributes. The cluttering of cylinders hindering our view of the terrain is not an issue with this attribute (see figure 2a).
- **Flow Density** - Indicates the speed at which the water is running through a given point. It can be ordered, differences can be calculated, has a true zero and is therefore a ratio type attribute. It is best depicted by coloring faces on the terrain mesh.
- **Sediment Load** - Indicates the volume of soil being transported through a given point. It can be ordered, differences can be calculated, has a true zero and is therefore a ratio type attribute. Both color or glyphs

are suitable for visualizing this attribute, for the same argument as provided for *discharge*.

- **Chi** - In simple terms, this variable approximates how much soil is being eroded or removed from a particular point. A greater mathematical derivation of this attribute is provided by *Simon M. Mudd et al.* [6]. It can be ordered, differences can be calculated, has a true zero and is therefore a ratio type attribute. This variable is best visualized by coloring the surface of the terrain, and a glyph representation is not suitable as it would clutter the plot greatly.
- **Initial Height Map (XYZ)** - This attribute contains the cartesian coordinates of the initial height-map. Since it contains XYZ coordinates, each coordinate is an interval type attribute with the same argument as provided for *XYZ coordinates*.
- **Cumulative Tectonic Uplift** - An attribute that defines the total increase in elevation due to tectonic uplift. It is provided to badlands as an input variable and is best visualized by a mesh where the cartesian coordinates are set to $[X, Y, U]$, where X and Y are coordinates from the initial height-map, and U is the cumulative uplift. This value can be negative and is therefore a interval type attribute.

III. Methods for Visualizing the Data

A. 3D Mesh

Digital terrain maps are typically depicted using a mesh. We will use the python library *pyvista* [2] for scientific visualizations of the data, and *blender* [3] for more artistic renderings of the data.

A mesh in *pyvista* uses a list of Cartesian (XYZ) coordinates to represent vertices on the mesh. Edges and faces are represented by storing the indices of vertices that bound each edge or face. Other properties of the mesh such as color and opacity are stored in separate lists of same length as that of the vertices. We use delaunay triangulation to create the edges and faces from our data. *Blender* stores its mesh data in a similar way.

The human visual system has evolved to see terrains as a 3D surface, therefore it makes sense to use the 3D mesh representation for our terrain data. Alternatively, we could use a 2D heat-map of our coordinates, where heights are represented by color, however this is less intuitive and prohibits us from using color to represent other attributes in our data.

A potential downside of representing our terrain using a 3D mesh is that certain faces of our mesh might be hidden behind other faces, and if heights are not color coded explicitly, they could be misinterpreted. Both of these problems can be resolved by making the 3D plot interactive and allowing the user to pan and move the view-port camera. This is a feature that most 3D software offer, however can not be provided in the PDF format that this report is written in.

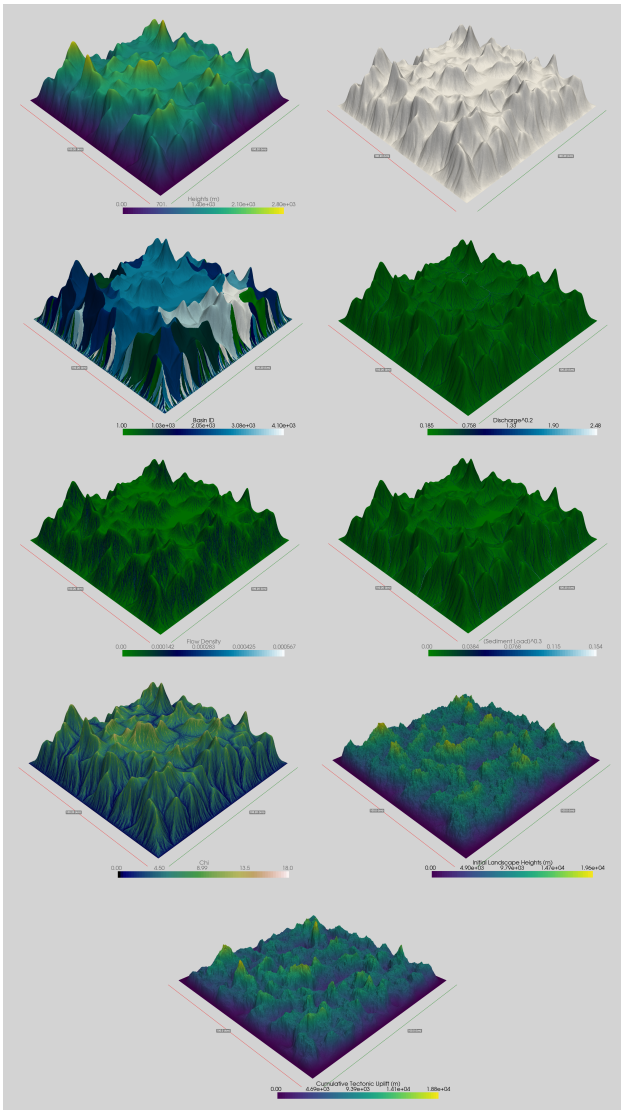


Fig. 1: Various visualizations of all our attributes

B. Color

Since our spatial coordinates are already being used by the topography of our mesh, we can make use of color to represent other attributes. The data type represented by color can be either nominal, ordinal, interval or ratio in nature, and so color can be used to represent any of our remaining attributes, including basin, discharge, or chi. In figure 1 we have used color to represent various attributes in our data, and color bar is provided to help interpret the value that each color represents.

A potential problem with color coding the mesh is that the rendering engine might distort colors in the process of shading, and thus some colors might appear brighter or darker compared to their true value. This problem can be minimized by choosing an appropriate shader and coloring scheme, which *pyvista* does quite well. The opposite of this issue is also true, the use of color can give an illusion of depth and therefore distorts the perceived topography of the landscape, which is

particularly apparent with our *chi* color coded meshes.

In the context of graphics art for entertainment purposes, the target audience may not necessarily care about the true underlying data values that the mesh visualization represents, and instead they may care more about how cool or realistic the visualization looks. In this context, the distortion of colors or perceived topography may be used intentionally, as it can result in more natural looking landscapes.

C. Shaders

When rendering 3D meshes, the interaction of how light interacts with the scene is calculated through a process called *ray tracing*. Different methods for controlling the input light and light interactions with the surface of a mesh are referred to as shaders. For scientific data visualization a shader that distorts colors the least is probably desirable, as it interferes less with the color coded attribute representation. Therefore a constant white lighting with a diffuse (non directional) light source might be desirable, such as the default *pyvista* shader.

Blender provides powerful and versatile tools for customizing shaders, which have been used to create figures 2a, 3c and 4. Here we used a combination of directional sunlight, and High Dynamic Range Imaging (HDRI) as input light. HDRI is a common technique in the field of computer graphics, where a 360 degree image is used to represent the input light from various angles in the scene, thus lighting the rendered object with appropriate environmental lighting and colors. The HDRI used in our visualizations was downloaded from [5].

A material for the surface of our mesh was created by using our *chi* attribute from our data as input, which was then passed through some RGB curves, a Hue/Saturation filter, and then passed into the base color input of a principled BSDF (Bidirectional Scattering Distribution Function) shader.

D. Glyphs

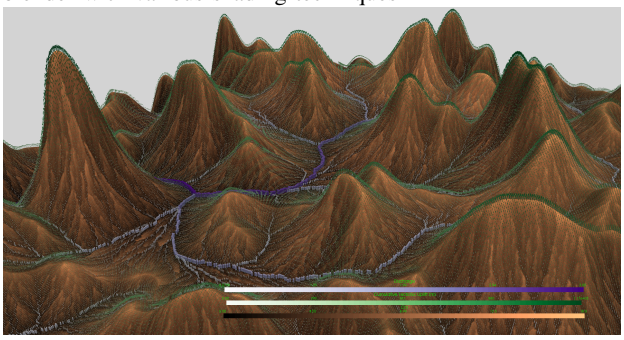
In the context of *pyvista* and the entirety of this report, a glyph is made by creating a template mesh object, and placing it in multiple locations given by a list of coordinates. The size and color of each instance of the template mesh can be scaled based on additional attributes that we wish to visualise. By placing glyphs on our landscape and modulating the size and color using our attributes, they can be an effective tool for visualising additional variables.

In figure 2b, we have used vertically pointing cylinders as our template mesh, which are placed slightly above our landscape. The size and darkness of each cylinder is modulated by discharge (purple) and tectonic uplift (green). The color coding has been chosen such that each color is as far away as possible in the Munsell color system. There are $2 \times 400 \times 400$ instances of cylinders whose XY coordinates are placed on a rectangular grid.

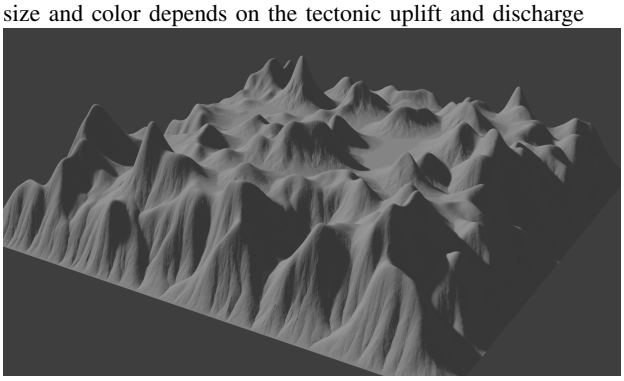
Potential problems with using glyphs in this way is that the cylinders may appear cluttered in some areas from our camera's point of view, and so they may hide features of our plot that are hidden behind them. It can also be difficult to interpret the actual size of each cylinder, since the ones that are further away appear to be smaller. This is less of a problem for representing features such as discharge, where most vertices have a discharge value close to zero, except for locations that have a river running through it. This allows us to increase the average size of our template mesh without resulting in cluttering, and the volume of water passing a given point becomes much more visible.

In figure 2a, we have placed template discs to represent discharge, however we have filtered out coordinates that correspond to discharge values that are smaller than a threshold value. This allows us to clearly see the river network, whilst avoiding cluttering. A separate material shader was assigned to the river network by using the *Glass BSDF* shader and setting the base color to blue, although a more advanced water shader could also be used.

(a) Visualization suitable for computer graphics, produced in blender with various shading techniques

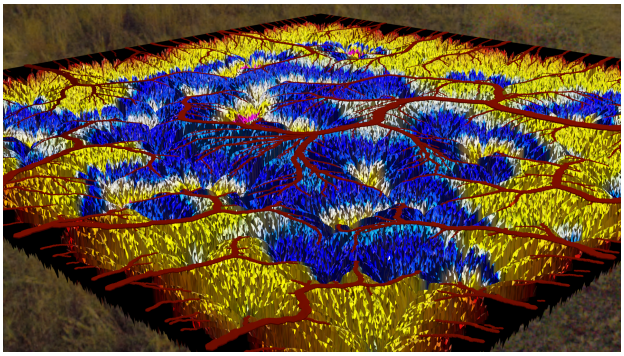


(b) Height map with surface colored by chi, and cylinders whose size and color depends on the tectonic uplift and discharge

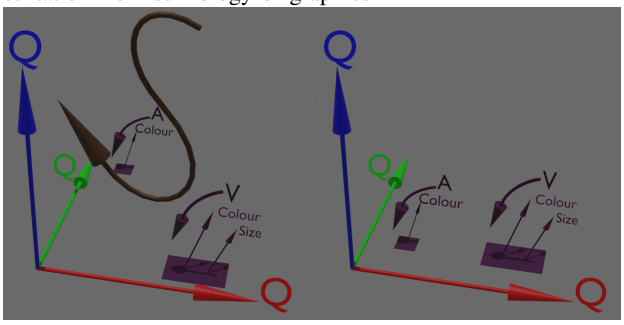


(c) Height map with no color but using sunlight shading

Fig. 2: More visualizations produced in blender [3]



(a) Alternative visualization derived from the symbolic representation from semiology of graphics



(b) 3D Symbolic Representation from Semiology of Graphics

(c) Another alternative visualization

Fig. 3: More visualizations

E. Animation

Since most of our plots were produced procedurally through code, and we have the data file of the landscape at 500 different time steps of the simulation, it is not too difficult for us to animate our data. This would allow us to visualize an additional attribute of time. Although I have produced some animations of the landscape evolving over time, I did not include it in this report, as the PDF format does not support movies. Alternatively, we could visualise the landscape over time by providing a series of images at various steps of the simulation, or we could provide an interactive slider that allows the user to specify the time.

IV. Visualizations Suitable for Different Kinds of Audiences

Depending on the target audience for visualizations, different methods might be more suitable. A geologists studying the plausibility of various erosion models might care more about the final topography of the landscape and locating where the soil is being eroded. For the purpose of analyzing topography, no coloring may be desirable, since this makes spatial features displayed by the mesh more visible. Meanwhile separate plots for other parameters should be provided separately.

An ecologist might care more about which parts of the landscape is habitable by different kinds of species, and whether a particular specie can traverse from one habitable area to another. Since the habitability of an area for a particular specie depends on temperature, which depend on height and sunlight flux at a given point on the landscape, color coding the plot by height and applying sunlight shaders on a separate plot (figure 2c might be desirable. The rate of erosion (chi attribute) might also be in their interest, since certain species may not grow on soil that is too dynamic.

For the purpose of computer graphics, where terrain data visualizations are produced for entertainment purposes, we are not necessarily worried about how various shaders might provide misleading interpretations of our color coded data or mesh topographies. Since *pyvista* does not provide much options for defining shaders, *blender* is more capable for artistic visualizations. The visualisation provided in figure 2a could be used as a good starting point for producing maps for video games. A digital artist might also be interested in the *chi* attribute, since low *chi* values tend to appear in the valleys of the landscape, and could provide useful for procedural spawning of plant meshes based on a probability that is modulated by *chi*.

A simple game map was produced in blender by placing objects manually into the scene. The assets were downloaded from [7], a website providing free game assets and were made by an anonymous internet user who calls himself *Quaternius*.

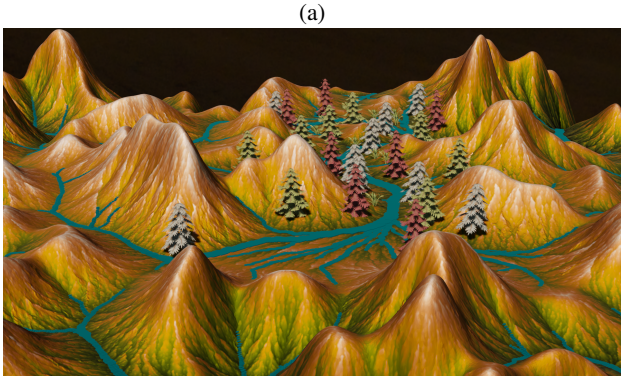
V. Semiology of Graphics

In figure 3b, we provide two symbolic representations of our visualization using the semiology of graphics in figure 2a. To the left of the image is our main symbolic representation, meanwhile on the right is our alternative representation. The symbols were produced in blender not necessarily because I think it looks better in 3D, but because it gives me an opportunity to practice my poor 3D modelling skills.

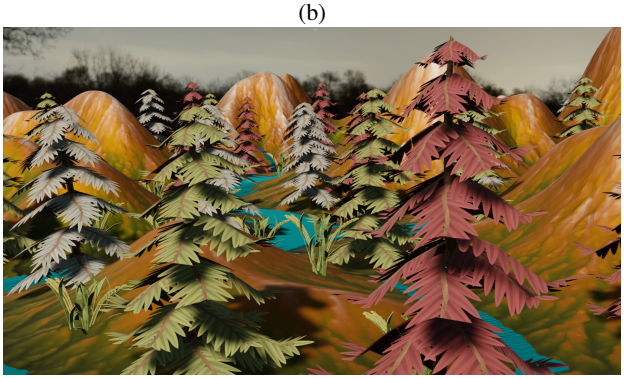
A. Our Primary Symbolic Representation

The vertices of our mesh are placed in a 3 dimensional space based on the 3 quantities (Q) of X, Y, and Z coordinates of our terrain data, so we used 3 orthogonally placed arrows labelled Q to represent this. The set of vertices and edges together form a network, which is within the Cartesian space,

(a)



(b)



(c)

Fig. 4: Terrain map produced in blender using our data with some scenery objects placed into the scene. Objects were downloaded from [7].

and so the S shaped arrow (representing networks) has been placed within our orthogonal arrows. Since the faces of the mesh are of area type and are part of the edge and vertex network, a curved shaped arrow labelled with A has been placed inside the S shaped arrow. Since the color of the mesh is a retinal variable that is also on our vertex/edge network, an arrow with a small face underneath has also been added inside the S shaped arrow.

The discharge in figure 2a is represented by a glyph of discs, which are not a part of the main terrain mesh and so the symbols representing it are placed outside of the S shaped arrow. Since the discs is a volumetric type representation, we have used a curved arrow labelled by V to represent this, and since each disc is modulated in size and color, we have used two retinal symbols labelled *Colour* and *Size*.

B. Equivalent but Different Symbolic Representation

Since only the faces of the mesh are actually visible in the final rendering of our terrain, we could argue that our visualization is not using a network at all, since neither vertices or edges are shown in our final visualization. In this case, the faces representing the terrain are still an area type quantity with a colour retinal variable associated to it, but the two symbols representing the faces no longer need to be inside the S shaped arrow, and we can remove the S shaped arrow entirely from our symbolic representation.

C. Alternative Visualization Derived from the Symbolic Representation

Within the context of our data, it is quite difficult to derive an alternative visualization from our symbolic representation whilst still being useful for the purpose of data analytics. Therefore our alternative visualization is designed for computer graphics and entertainment purposes only, and is provided in figure 3a.

Here the data was imported into blender in much the same way, but instead of using heights for the Z coordinate of our vertices, we have used negative *chi* values. Additionally, instead of using *chi* as the input of our mesh material shader (color), we have used the Z coordinate. We then tweaked some parameters in our shaders until it looked pretty damn cool. Essentially we have only swapped two input attributes in our code that import the data to blender, and since the fundamental method for producing the visualization has remained the same, the symbolic representation for our alternative graphic remains the same.

There are plenty of other attributes that we could have swapped to derive an alternative visualization, however the majority of them would result in a really boring graphic at best, or a completely broken visualization at worst.

D. Alternative Spherical Representation

Sometimes terrain data may span the entire globe of the earth, and so a spherical representation of the terrain data might be appropriate. Representing it in 2D or flattened out would result in distortion of distances. Although our specific terrain data was not intended for spherical representation, we have provided an example of it in figure 3c anyway.

Although the semiology of graphics provides symbols for circular representations, we could argue that our spherical terrain is still just a network of vertices and edges, with faces being attributes on the network of area type, and so the symbolic representation in figure 3b is still valid for our spherical terrain.

VI. Conclusion

We have discussed various methods of how our data set could be visualized, although there is still a lot that could be done to improve the simulation for producing the data, and there are a lot of further steps that could be taken to implement the simulated data into a game map.

We defined our tectonic uplift map in such a way so that vertices are only being pushed up vertically, however *badlands* has the capability of defining the uplift in terms of vectors, which opens the possibility to visualize the folding of mountains due to tectonic processes. To generate such an uplift map, we could use additional perlin noise to modulate the X and Y contributions of the uplift map.

The soil erodibility in our simulation was defined to be constant everywhere, which results in a constant maximum slope steepness in our final terrain. By modulating the soil erodibility as a function of height with perlin noise, we could have achieved more diverse slope gradient, such as those seen in most escarpments. Since running a single simulation took about 10 hours to complete, I did not bother further messing around with tectonic uplift, soil erodibility or precipitation parameters.

Additionally, for the purpose of generating a video game map, the *chi* values in our data could have been used to modulate probabilities of spawning various plants or objects into the map. Low *chi* values correspond to low soil movement, which tends to happen in valleys where trees and grass species would grow. Meanwhile high *chi* values correspond to high soil movements, which tends to happen on mountainous terrains, and could therefore be used to spawn rocks into the terrain. Since all of this (including specifying initial parameters for the simulation) is all done in code, further work could be undertaken to create a procedural map generator.

Badlands is a new software to me which I have not been familiar with before commencing this assignment. However I will be combining it with code from my current research project, and so I thought I'd take advantage of this assignment to get familiar with it. My current project involves simulating plate tectonics on a global scale, and the uplift data produced by my code will be fed into *badlands* to generate the finer details of the earth.

REFERENCES

[1] Salles, T., X. Ding and G. Brocard: *pyBadlands: A framework to simulate sediment transport, landscape dynamics and basin stratigraphic evolution through space and time*, PloS One, 13(4):e0195557, doi:10.1371/journal.pone.0195557, 2018.

[2] Sullivan et al., (2019). *PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK).* Journal of Open Source Software, 4(37), 1450, https://doi.org/10.21105/joss.01450

[3] Community, B. O. (2021). *Blender* - a 3D modelling and rendering package. Stichting Blender Foundation, Amsterdam. Retrieved from http://www.blender.org

[4] Sklar L, Dietrich WE. In: *River Longitudinal Profiles and Bedrock Incision Models: Stream Power and the Influence of Sediment Supply*. American Geophysical Union; 1998. p. 237–260.

[5] Greg Zaal (2020), *Dikhololo Night* - HDRIHaven, Retrieved from https://hdrihaven.com/hdri/?h=dikhololo_night visited in 2021/03/29

[6] SM Mudd, M Attal, DT Milodowski, SWD Grieve, DA Valters *A statistical framework to quantify spatial variation in channel gradients using the integral method of channel profile analysis* 2014 Journal of Geophysical Research: Earth Surface https://doi.org/10.1002/2013JF002981

[7] Quaternius, *Quaternius Game Assets* - Visited on 2021/04/01 - https://quaternius.com/index.html