School of Computing and Digital Technologies

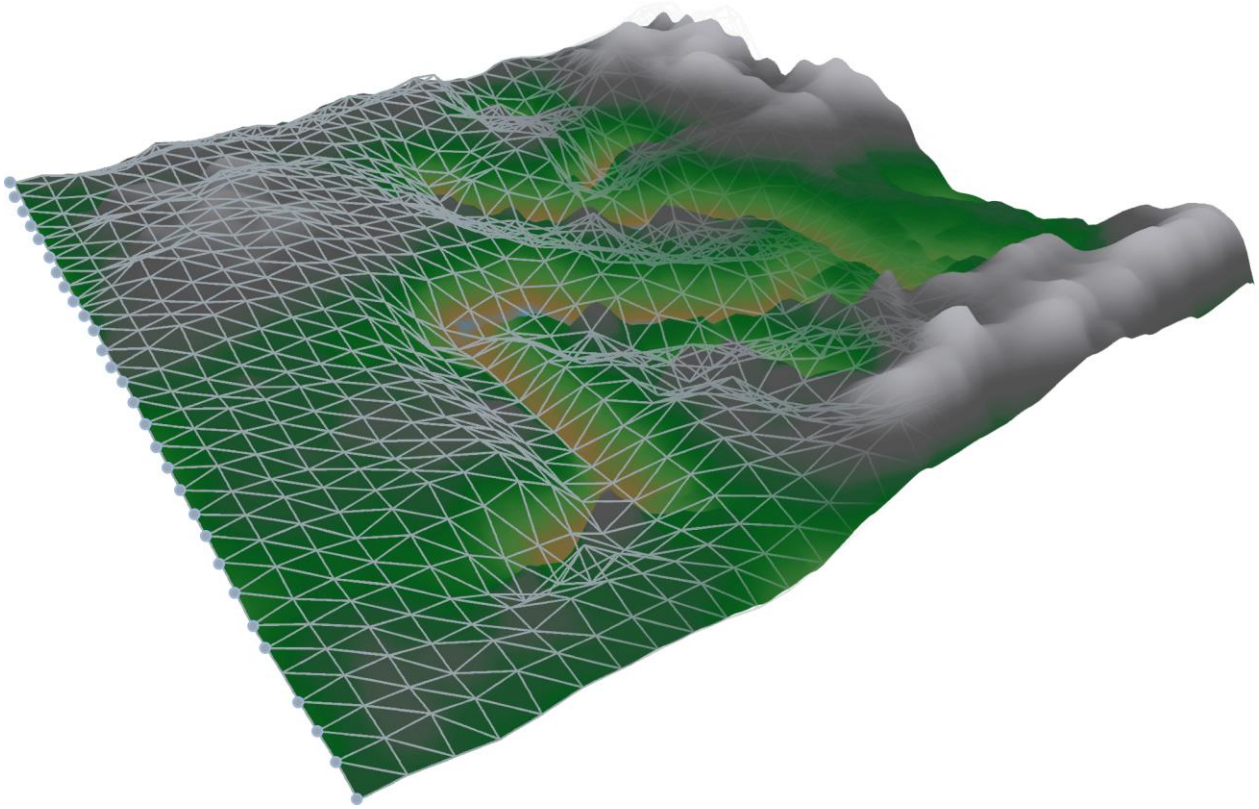
Teesside University

Middlesbrough TS1 3BA

Development of a machine learning based interactive virtual terrain authoring pipeline for videogames and 3D Media

Reflective Report

Submitted in partial requirements for the degree of MA 3D Games Art



Date: 14th August 2020

Francisco Múrias Mira Cabral Pinto

Supervisor: Dr Paul Noble

Abstract

The creation of virtual terrains that represent the user's artistic intent while also maintaining correct geomorphological characteristics has been a focal area of research in the past few decades. Recent advancements in the field of machine learning have presented new opportunities to tackle this problem. This thesis will explore a pipeline to train and develop a Conditional Generative Adversarial Network model from automatically generated real world terrain data. The pipeline can generate any number of realistic virtual terrains using the learned features from the training data, without requiring a manual definition of its procedural rules or parameter properties.

A custom tool was developed that provides users with the ability to sketch peak and valley lines that define their artistic intention and the algorithm automatically generates the resulting terrain from the user input in real time. A custom 3D previewer allows rapid iteration and visualization of the result. A simple erosion simulation is conducted and resulting terrains are then ready to be imported to a game engine and used as the basis for a 3D Landscape.

The system was tested by a sample of users with varying degrees of experience in 3D Art who were able to design various complex terrains in a short amount of time. The generated terrains are amplified and eroded and then demonstrated inside a game engine.

Contents

List of Illustrations

1. Introduction

As the techniques used for content creation for videogames and 3D media increasingly focus on procedural and algorithmic workflows, one major area of research where they have always been a central focus is the authoring of virtual terrains. Realistic terrain generation poses a particularly difficult problem as real terrains are subjected to a wide variety of geomorphological processes that influence their shape and features, such as thousands of years of exposure to various erosion agents, different ranges of temperatures, vegetation, wind and water sources that are hard to algorithmically define as usable editing tools. As a consequence of this, workflows for designing terrains are particularly challenging as the balance between allowing the intuitive expression of the designer's intent while still maintaining realistic and naturally convincing results is not an easy one to define using the currently used techniques. In the past three decades, researchers have investigated a variety of approaches to generate terrains, all with varying advantages and disadvantages.

Recent advancements in the area of Machine learning (ML), the study of computer algorithms that improve automatically through experience, have presented solutions to many problems in various fields of computer graphics such as content synthesis. The main principle of these algorithms is to generate a mathematical model based on given data, known as "training data". After the training is completed the resulting model can make predictions or decisions without being explicitly programmed to do so.

This thesis will examine the applicability of the latest machine learning algorithms as a basis for a responsive user-friendly artistic tool to solve the problem of authored, geologically believable, terrain generation. To this end, a specific type of machine learning approach called Generative Adversarial Network (GAN) (Goodfellow et al., 2014) and one of its extensions, conditional GAN (cGAN) was chosen as part of the image translation algorithm pix2pix (Isola et al., 2017) used. A complete workflow for training, testing, and using this process to synthesize realistic terrains was identified, developed, and tested. The resulting algorithm is oblivious to the underlying geological phenomena of the terrains and instead derives its generative capabilities from a provided bank of data automatically taken from the real world, that it uses to define its own generative model.

The resulting method allows for an intuitive and fluid interactive authoring process, while providing results that are close to the geological nature of the real terrains, without the need to ever specify the model used to simulate their underlying geomorphologic characteristics, as all of these properties are derived from the trained example datasets. An interactive tool was developed to deploy the algorithm in a real-world use case scenario, in which any user, despite experience, could author a usable terrain in seconds (Figure 1). The generated terrains were amplified and showcased in a game engine.

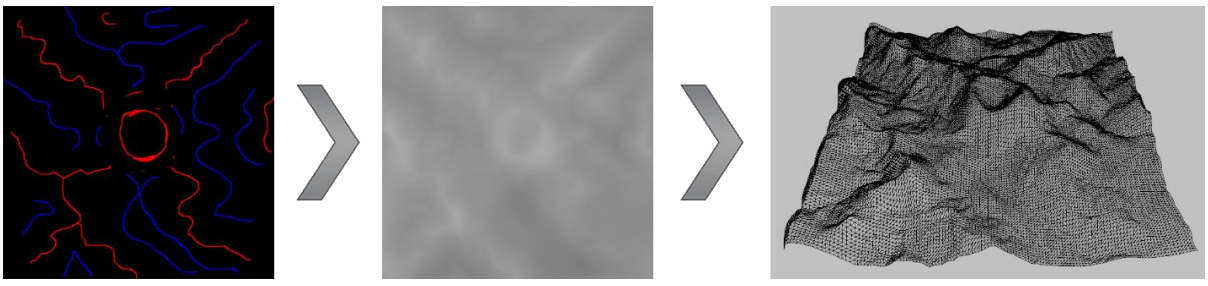


Figure 1. Example result from a terrain authoring session.

2. Research and Related Work

2.1 Overview of current terrain generation approaches

Virtual Terrain Generation has been a central area of research in computer graphics for decades, and different approaches to the problem have been considered, all with differing advantages and disadvantages. They can roughly be classified in the following categories:

Procedurally-based – These prove useful for terrains with strong fractal features and are usually defined by using mathematical models such as fractal noise and sinusoidal waves (Fournier et al., 1982). While these methods tend to be very fast, they usually lack control over the resulting terrain characteristics, as they seed from random number generators which result in unpredictable variations (Hnaidi et al., 2010; Génevaux et al., 2013). Additionally, these models tend to fail in correctly modulating erosion and other geological factors, as real terrains rarely conform to purely fractal definitions.

Simulation-based – Methods such as erosion (Musgrave et al., 1989) and hydrology-based algorithms (Chiba et al., 1998; Nagashima, 1998) have the advantage of providing physically correct results. However, they tend to be limited in user control and are computationally demanding, not allowing for quick iteration and refinement as part of the workflow.

Sketch-based – These methods rely on user input to modulate existing terrains or sample from existing terrains to create new ones (Hnaidi et al., 2010; Tasse et al., 2012). They provide the highest degree of user control, although they fail in producing geologically correct results and editing terrains of a larger size is tiresome and takes considerable time.

Example-based - Current example-based methods use small terrain samples and automatically replicate them over large terrains (Peytavie et al., 2009). These usually result in limited possibilities for the final result, with noticeable repeated features occurring as they fail to generate new elements. These limitations become more prevalent the larger the terrain being generated. A notable advancement with variations of this approach is the work by Zhou et al. (2007) that use patches from sample terrain guided by user sketched feature maps to modulate new terrains.

The proposed method in this thesis can be classified as an example-based technique, however it diverges from other used methodologies as it abstracts the examples from the actual resulting model used to produce the results. While one can argue that the results will only be as good as the provided data, the usage of machine learning to generate the model instead of the result itself, leads to completely new and unexpected possibilities spontaneously occurring in the synthesized terrains. A notable example of previous usage of machine learning for terrain synthesis is the work of Guérin (2017) that inspired the work within this research project. The proposed approach of this exploration expands upon from the referred article as it formulates a broad workflow for generating the training datasets, and employs a broad use-case algorithm (Pix2Pix) to determine the usability of the workflow across a wide range of purposes, and tests the resulting models inside a modern game engine.

2.2 Machine learning methodologies

Machine learning models generate a mapping function that resolves a set input into a useful output (Figure 3). Notable developments in this field, specifically Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) work by simultaneously training two competing networks (Figure 2):

- A generator network (G) - Attempts to create new examples by processing the input data and defining relationships between image pairs.
- A discriminator network (D) – Tasked with discriminating between the corresponding real dataset and the examples generated by the G Network.

This two-fold process improves as the Generator learns to produce better results, and the discriminator learns to better distinguish the generated images from the provided training set. As the training process progresses, we follow the evolution of their loss functions and determine if we are successful if the generator is ultimately able to produce results that can fool the discriminator. This technique has previously been used to solve problems relating to digital image generation and completion (Mirza and Osindero, 2014; Pathak et al., 2016).

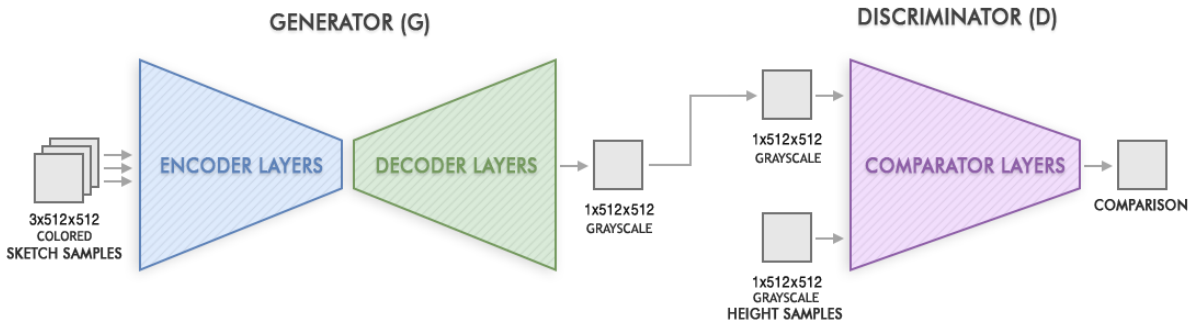


Figure 2. The model consists of 2 competing networks: a Generator and a Discriminator.

The method chosen for this thesis made use of a modified version of Pix2Pix (Isola et al., 2017), an implementation of conditional adversarial networks as a general-purpose solution to image-to-image translation problems. This was used to train various terrain synthesizers from automatically generated input datasets built from real-world examples. This algorithm automatically calculates a loss function for each network, and thus provides the possibility of applying the same generic approach to problems that traditionally would require very different loss formulations.

This algorithm has proven useful for a variety of different applications, such as photo synthesis from map labels, object reconstruction from edge maps, image colorization, and many others. Artists keep demonstrating inventive uses for the algorithm since its release.

Figure 3. Machine learning models consist of a mapping function that takes in a set input and outputs a result (Nordin, 2018).

3. Design and Implementation

3.1 Overview

The developed pipeline can be described as sequential process consisting of a training phase and an interactive authoring phase. This will be outlined in the next sections and is summarized in Figure 4.

The first part of the process is to obtain the training data (section 3.2). For this an automated pipeline was tested using Houdini PDG (2017) that automatically downloads height map information from USGS, the United States Geological Survey. The Houdini pipeline is capable of downloading and decompressing the data from the USGS database and processing it by dividing it into smaller manageable sections, using an HDA to extract features such as river and peak lines (section 3.3). This produces our training dataset of corresponding peak and valley maps and height maps.

The generated dataset of the heightmap data and the matching Peak and Valley maps are the input to train the conditional Generative Adversarial Network model with pix2pix (section 3.4). This essentially gives us an inverse mapping of what we generated in Houdini. The resulting model is a neural network that learned to map the peak and valley lines to its corresponding height and geomorphological information.

Figure 4. A full overview of the developed pipeline.

After the generative model is obtained from the training process, we can start the interactive generation phase (section 3.5).

A custom python graphical tool was programmed that loads the trained model and allows users to interactively and responsively sketch their own peak and valley lines and instantaneously synthesizes the resulting heightmap. The authoring process allows the user to draw any valleys or peak lines to define the features of the terrain as he wishes, being able to go back and erase and make changes in real time. The user can select from any model that has been previously trained and export the resulting heightmaps automatically to be used in external applications. Furthermore, a 3D viewer was developed in ModernGL (2020), allowing to user to preview, zoom and rotate the resulting terrain as a 3D mesh while in the editing process. The resulting heightmaps are then amplified and an erosion simulation is performed in an external software package (section 3.6). The result is tested inside a real use case 3D application (Unreal Engine 4) as a basis for a videogame landscape (section 3.7).

3.2 Gathering terrain datasets

In order to train the cGAN algorithm, a large data set is required. Generating this training data manually would not be feasible, fortunately the USGS 3D Elevation Program (2011) contains elevation data for roughly the entirety of the United States landmass.

Key locations featuring distinct geomorphological traits were identified to be used as the basis for different generator models: the rocky mountains, due to its distinct elevation patterns and clearly defined ridges and valleys, and the grand canyon range, due to its unique ravine formations. The exact same extraction and training process was used for these two distinct datasets as to ascertain the effectiveness and repeatability of the proposed pipeline.

SideFX Houdini and its automation pipeline PDG (2017) were used to automatically download, extract, and filter this data (Figure 5). The downloaded files from USGS contain height information in sections of approximately 1 degree latitude and longitude radius. The PDG system allows us to extract this data by inputting the coordinates into Houdini and the selected range of height information is then downloaded and processed. This information is loaded as a Houdini terrain and automatically split into smaller more manageable sections. A total of 8 heightmap patches of one square degree were extracted from USGS and used to create the resulting databases. The Total number of images generated for training was 560 for the Rockies dataset, and 436 for the Grand Canyon.

Figure 5. Dataset Generation PDG.

3.3 Peaks and Valleys Map Generation

Continuing down the PDG pipeline (Figure 6), the extracted smaller sections of height data are processed using an HDA based on the machine learning data preparation example provided of SideFX (2019) in order to compute the Peak (highest points) and Valley (lowest points) and map this information into a corresponding image that matches the heightmap tiles. The peaks will be marked in red and the valleys blue. This will form the basis of the intended user sketches during the authoring process.

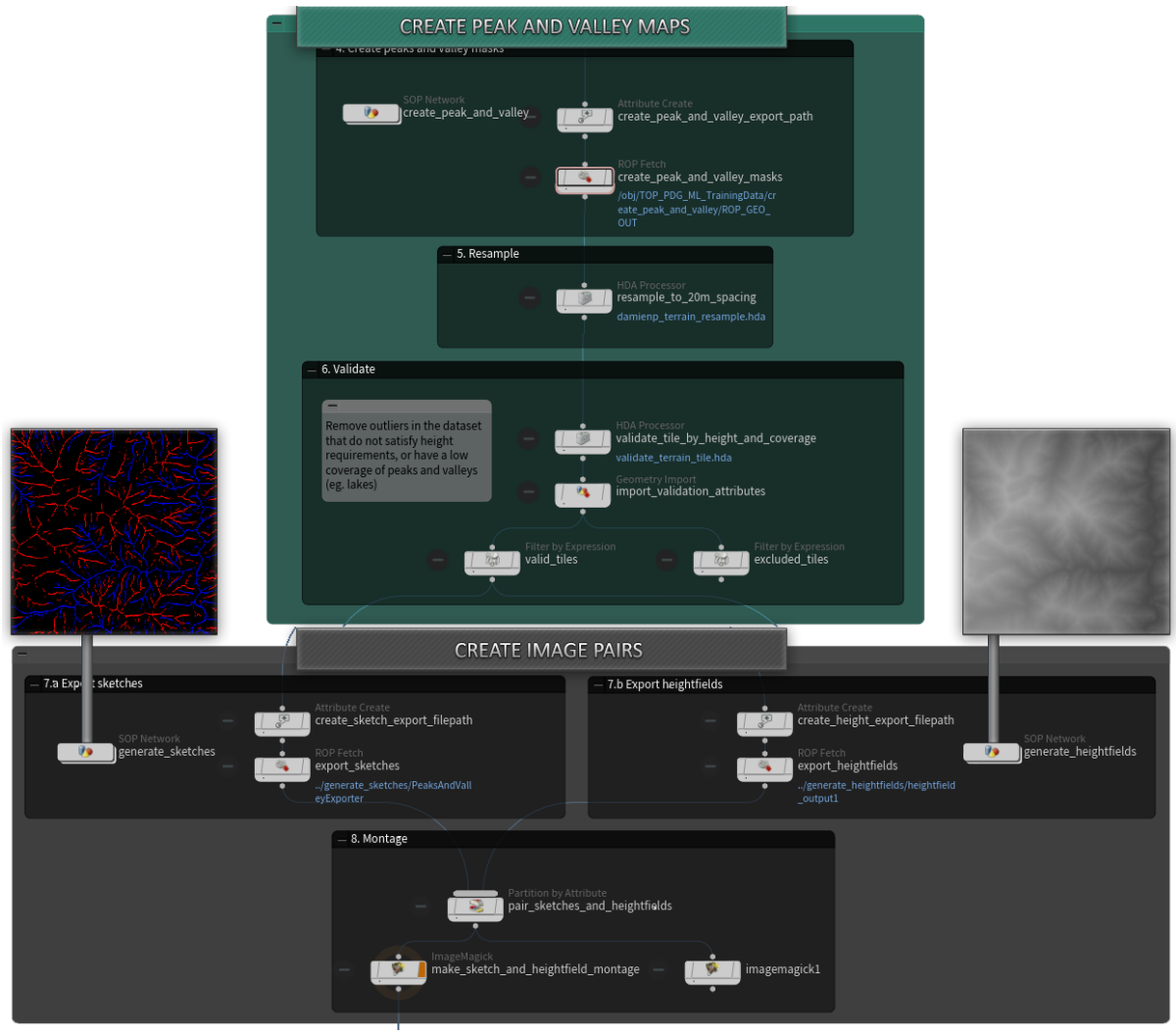


Figure 6. Peak and Valley Data Processing PDG.

The result of this process is a full dataset of matching image pairs ready to use for training our cGAN Models.

3.4 Training the Neural Network models

After obtaining the dataset with Houdini, we have all the data needed to train the model using pix2pix so it can learn the mappings from our image pairs of peak and valley maps to real terrain heightmaps. The training phase only needs to be performed once per model, however each training session can take considerable time depending on hardware and amount of data. To maximize the training data, a process of data augmentation is programmed into the algorithm so that it reads the images and then flips them, both horizontally and vertically, and reads them again, effectively tripling our sample count.

The algorithm chosen for this project was based on the pix2pix implementation by Affinelayer (2019) and several changes needed to be made to this model. A property of cGANs is that they are resolution dependent, meaning they can only produce images in the resolution that the network layers are configured for. The chosen implementation was designed to input and output 256x256 sized images. Some initial training tests were performed at this resolution with promising results, however, to allow for greater detail in the generated heightmaps, the model was altered to generate at size of 512x512. This required the addition of one extra network layer to both the generator encoder and decoder (Figure 7). Another significant modification was the necessity to output images with 16 bit depth, as this was the intended format for unreal heightmaps. To achieve this, the model was altered based on the changes by Guérin (2018). Both these alterations made the model considerably larger in size, as the parameter count (total number "neurons" in the network) was 90750660 after the modifications.

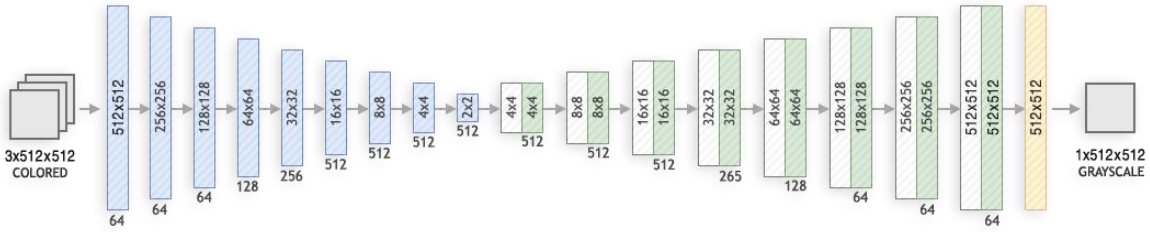


Figure 7. Structure of the Discriminator Model in the Conditional Generative Adversarial Network.

Training was conducted and several models were generated with both input datasets. The evolution of the loss functions was closely monitored as training developed and the algorithm was run for a total of 100 epochs for both datasets. The resulting models were exported and tested and can process input in 0.14 seconds (depending on hardware). Models were verified against the input data (Figure 8) until the results were convincing and determined ready to be used with the authoring tool. Further discussion on results will be discussed in section 4.

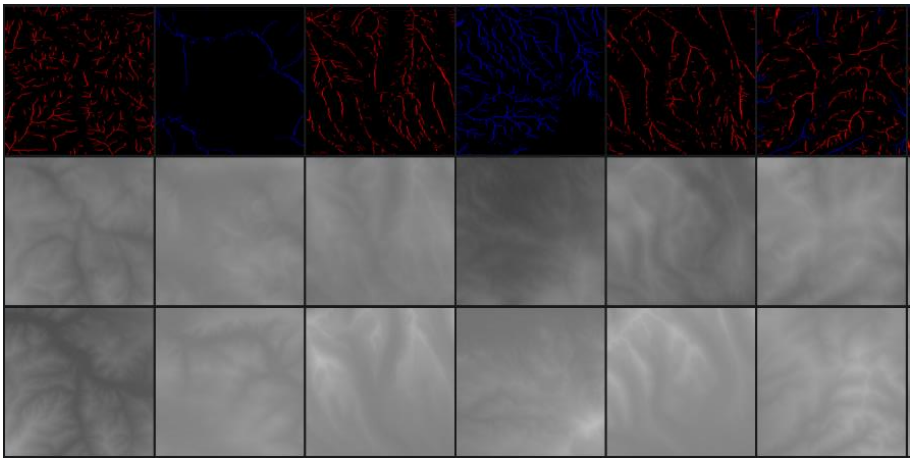


Figure 8. Some examples of the test output (Second Row) of the model when compared to the ground truth (Third Row) – More examples in appendix 2.

3.5 Creating and using the authoring tool

An authoring tool was developed using Python (Figure 9) alongside a real time 3D preview to visualise the generated terrains using ModernGL (2020). The intention was to match the quality of real-world tech art tools developed for game studios to solve custom problems. The authoring approach allows the user to interactively sketch peak and valley lines and erase and make changes as needed (Figure 10). As soon as a new line is drawn the system automatically runs the selected model and shows the resulting heightmap as well as a 3D preview on the bottom of the interface.

The tool is loaded with all the trained models for the rocky mountains and the grand canyon sets, and can be expanded to fit any custom model the user wishes to train, by simply adding any new models to the "models" folder and selecting them in the tool. The code for the authoring tool can be found at Appendix 1.

A notable feature of this application is its editing speed: we can quickly make changes to our input and the model takes only a few seconds to update the resulting terrain, giving us direct feedback on our editing process. Similarly, we can select any model we previously trained and compare the resulting heightmap with same input.

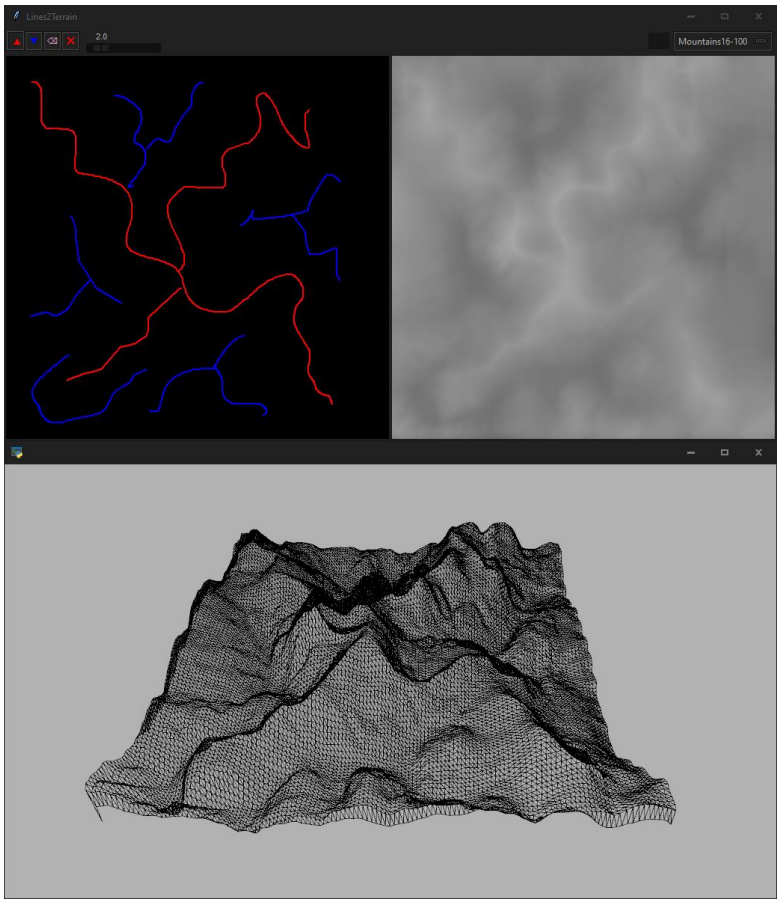


Figure 9. The interface of the authoring tool.

The user can interactively draw rough peak and valley lines of the intended terrain and the generator immediately computes the resulting terrain.

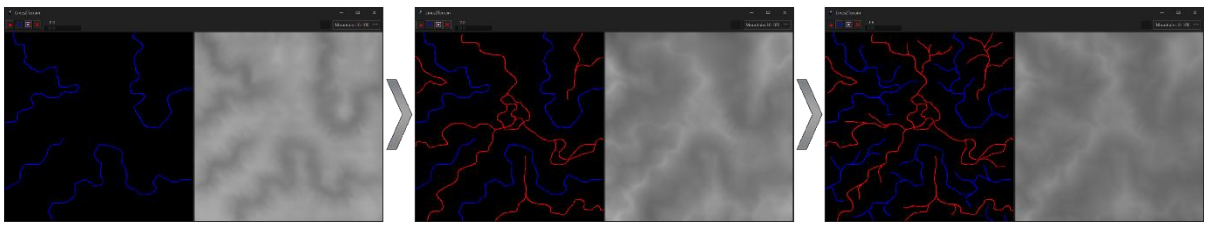


Figure 10. An example usage of the developed tool: The user starts by drawing valleys to define the river networks, then defines the peaks and continues iterating as the network generates a real time terrain in each change. – More Process screenshots can be seen in Appendix 3.

3.6 Amplification and Erosion Simulation

The result from the authoring tool is a usable 512x512 heightmap ready to be imported to any game engine, however, it can be further enhanced with an amplification and erosion simulation pass. There are a multitude of tools that can do this for us, and erosion algorithms were not the focus of this thesis. The process chosen used QuadSpinner's Gaea (2019) to achieve the erosion sim, as its node-based structure proved simple to use, and a custom project was created for each of the models (Figure 11). The heightmap is loaded into the program and erosion parameters can be tweaked depending on the desired erosion duration, sedimentation, and rock softness, amongst others. The result is then processed and allows us to output images that are 1k (1024) up to 16k (16384) in size. The resulting terrains were tested at a size of 1k.

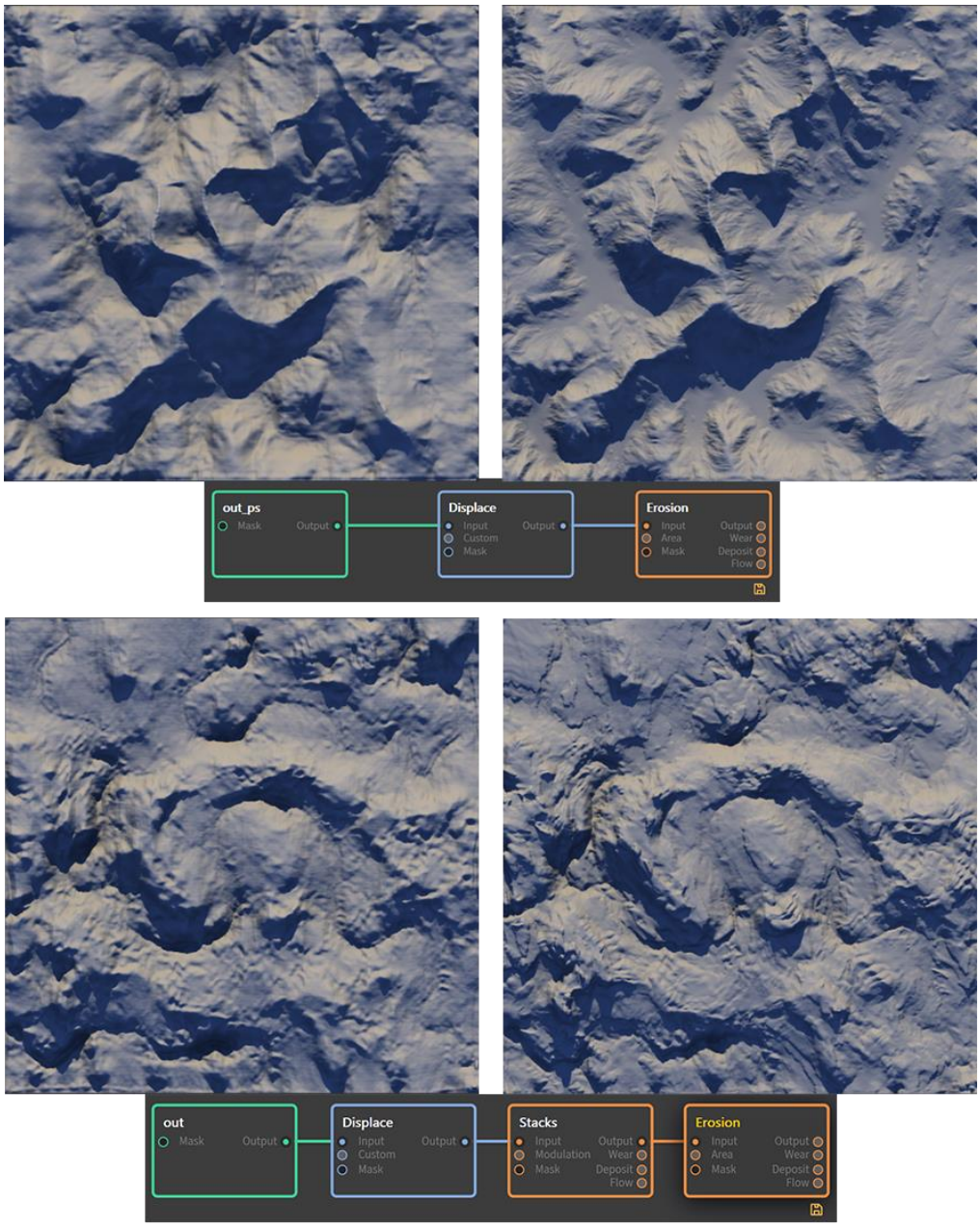


Figure 11. Custom erosion setups for Mountain and canyon models.

In future, a possible second neural network could be trained on the mappings between eroded and non-eroded terrains to verify if this process can also be automated with a machine learning model with convincing results.

3.7 Game Engine Result

The final step in the pipeline is to import the generated terrain into a game engine and determine if the result is usable in a 3D Game Art scenario. Unreal Engine (Epic Games) was chosen for its extensive landscape and material tools as it not only allows for remarkable landscape rendering, it also features a non-destructive landscape layer system that allows our heightmap to be imported and used as the base layer for a terrain, while retaining the possibility to edit and build upon it and adapt to any gameplay necessities such as structures, roads and scenery objects.

Before we import, however, we must adhere to Unreal Recommended Landscape sizes (Epic Games, 2015) and as a result, crop our 1024px sized images to 1009px in size. We do this with a simple photoshop action. But this could be automated in other ways. We can now import our heightmap (Figure 12).

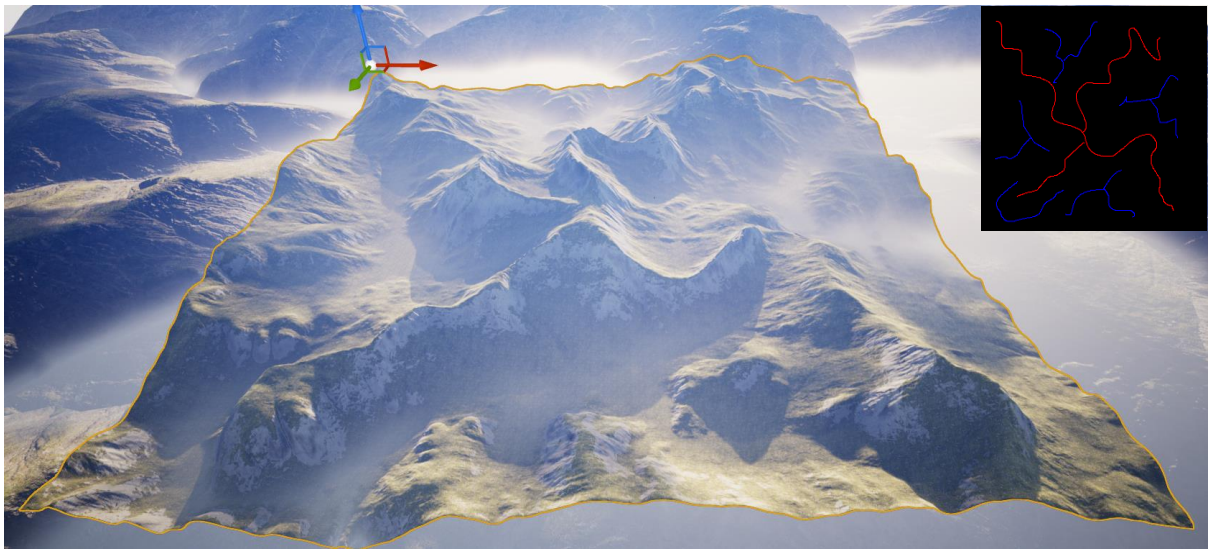


Figure 12. The imported mountains landscape in unreal engine.

A landscape master material based on MAWI United GmbH (2020) is applied to the terrain, that determines what kind of textures to use in the landscape depending on the normal direction of the terrain, and seamlessly blends between the different kinds of textures in world space, essentially providing an accurate initial pass on the geological terrain texture. We can then directly paint any other material directly on the terrain or import any flow maps from Gaea (or any other software) that automate this process for us.

A simple preview test scene was composed in unreal engine that allows for a preview of what a possible game ready solution could look like. Dynamic Lighting, depth fog and other unreal parameters were adjusted, and a custom procedural volumetric cloud blueprint based on the implementation by Emelianov (2019) was used in the scene for more believable results. Furthermore, a low poly background mesh was included in the scene for a better result when viewing the scene in a first-person perspective. In a real game, the bounds of the terrain would probably be better occluded with a different method.

Some showcases of obtained results are shown here in Figures 13 and 14 – More Images of the results can be seen in appendix 4.

Figure 13. Example in engine result from the Rocky Mountains model.

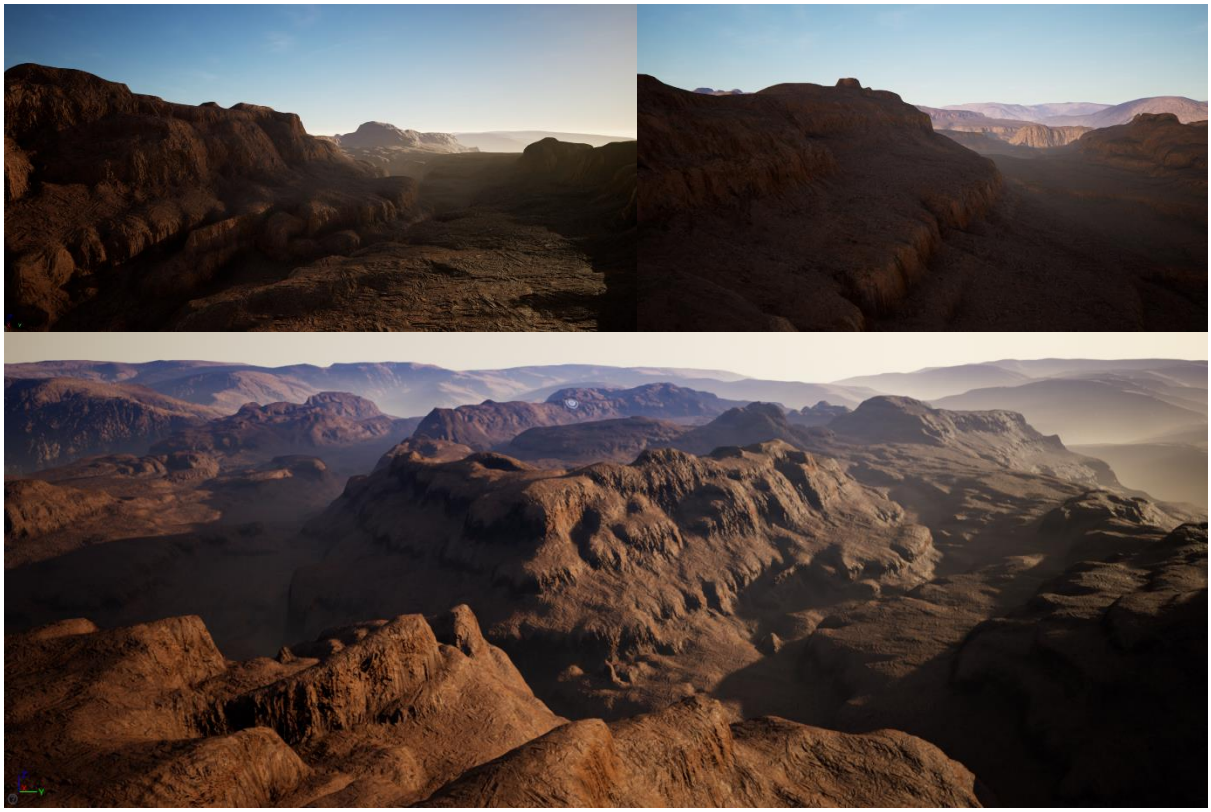


Figure 14. Example in engine result from the Grand Canyon model.

4. Results and Discussion

4.1 Evolution of training models

Several models were iteratively trained with different results. Even with the same dataset, the algorithm can initialize with different initial inputs and learn to make different predictions as the training progresses (Nordin, 2018). A random seed can be specified, and different seeds were tested with the same data to different results. The most promising models were trained for a total of 100 epochs.

To evaluate if the generator network is learning the features of the input data, and determine if the training is successful, we can analyse the evolution of the loss functions (Figure 15 and 16). In a perfect scenario, the generator and discriminator should always be in balance with each other. If one of them "wins" over the other, then the generator cannot improve. We then analyse the L1 loss function, which denotes the sum of the absolute difference between the ground truth examples and the resulting generated terrains. The closer this value is to zero, the better our model is at generating the desired result.

Mountains Model:

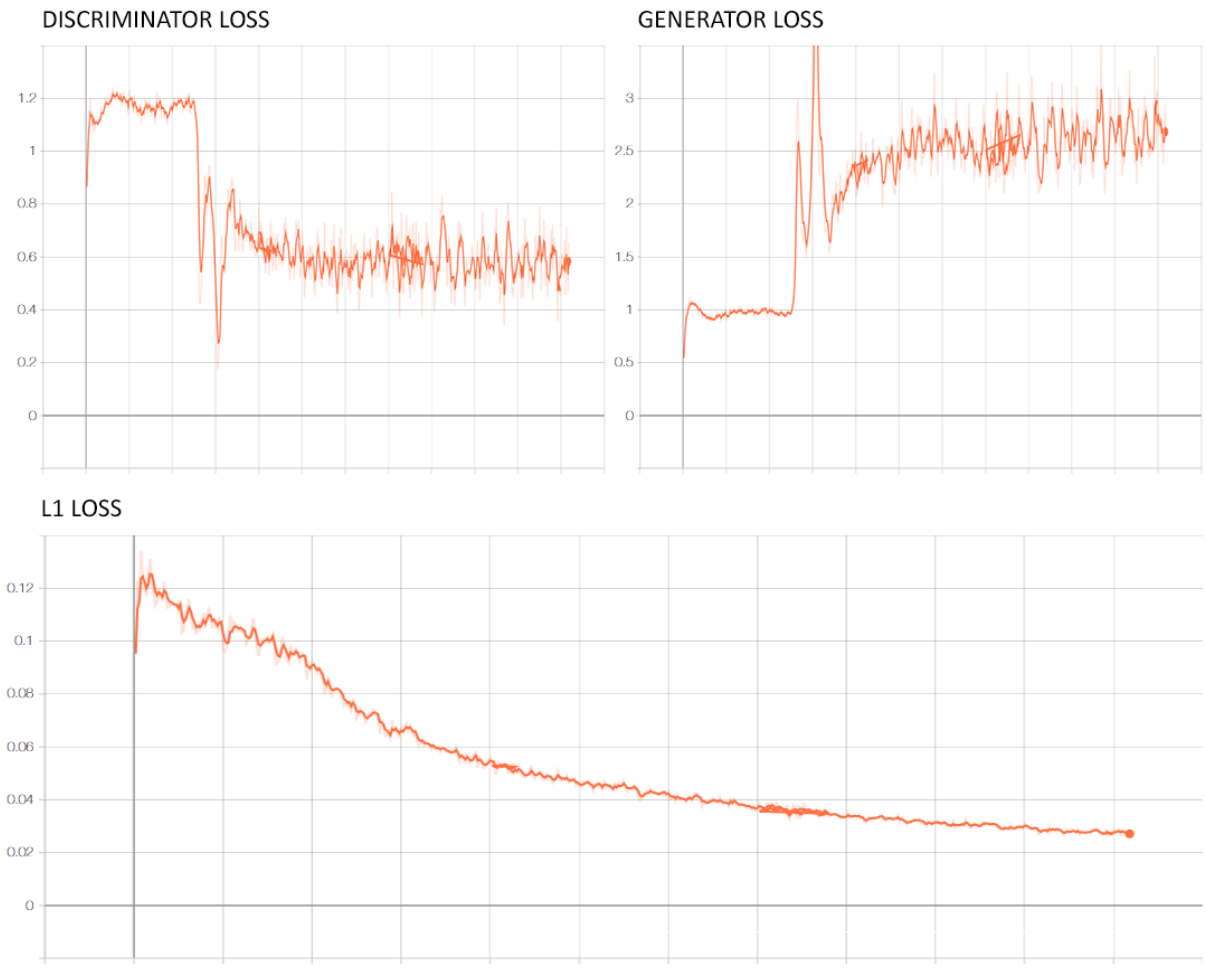


Figure 15. Evolution of Loss values of Mountains Model.

Canyons Model:

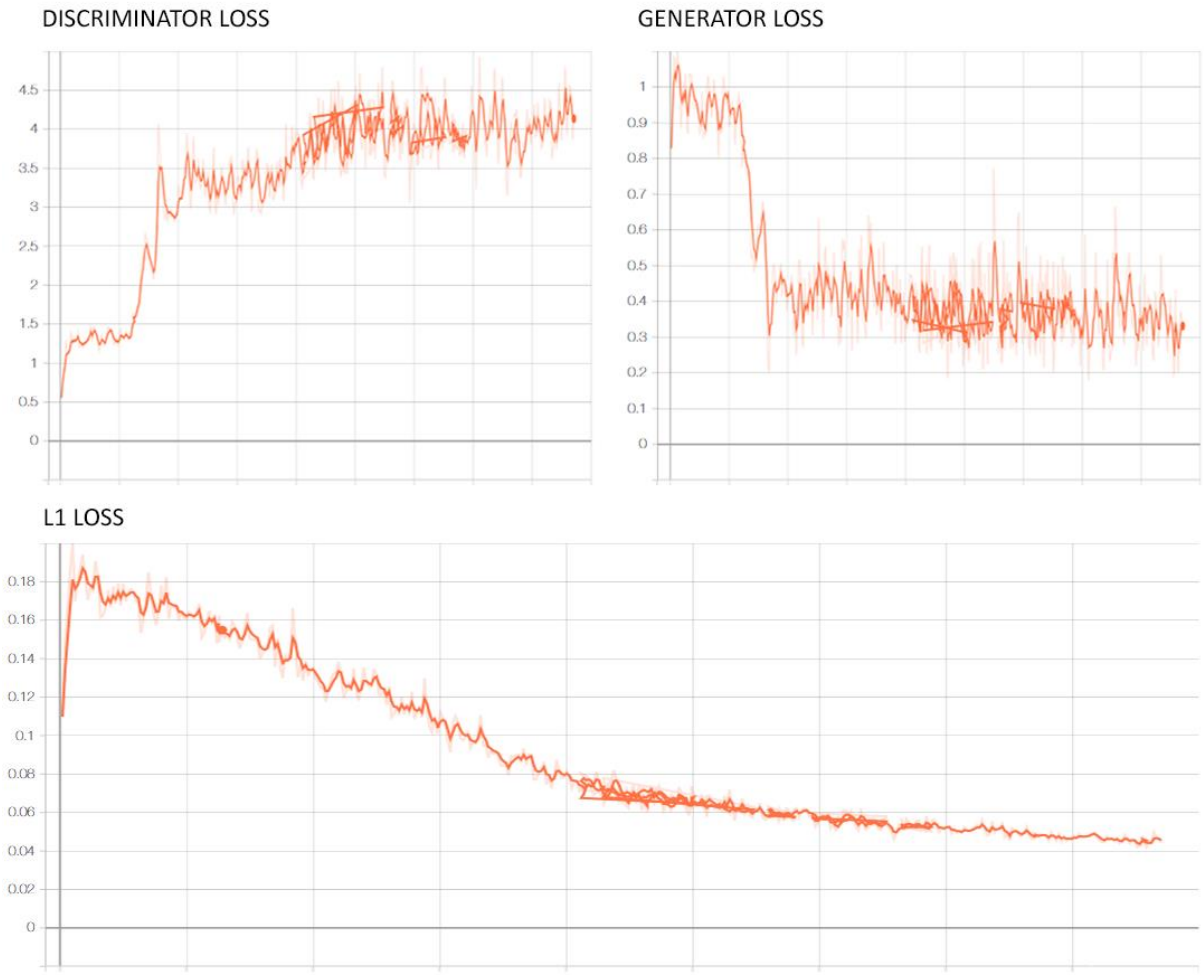


Figure 16. Evolution of Loss values of Canyon Model.

When analyzing the best resulting model based on both datasets, we can observe the discriminator and generator loss functions reaching a balance in both models, meaning one was never clearly "winning" over of the other. This means our models were able to continuously learn, and this can be verified by observing the L1 function, as it shows a clear reduction in value as training progresses. This translates to a clear improvement in the produced results of the model.

Training was stopped at 100 epochs, but both models were successful and as such, further training could be conducted, that would result in even better predictive models.

Building on the success of this training technique, other types of generators could be considered in future, such as ones that map heightmaps to different types of input sketches.

4.2 Usability Testing

To assess the usability of the pipeline, the tool was given to a select number of industry peers with differing levels of experience in 3D Art. The 8 participating users were explained how to use the tool and were asked to design terrains according to their artistic intention. After they experimented with creating their terrains, the participants were shown the end results of their preferred sketch in-engine. Following the showcase, they were asked to answer a small qualitative survey to gage their experience and evaluate the performance of the proposed pipeline on a scale of 1-5 (Figure 17).

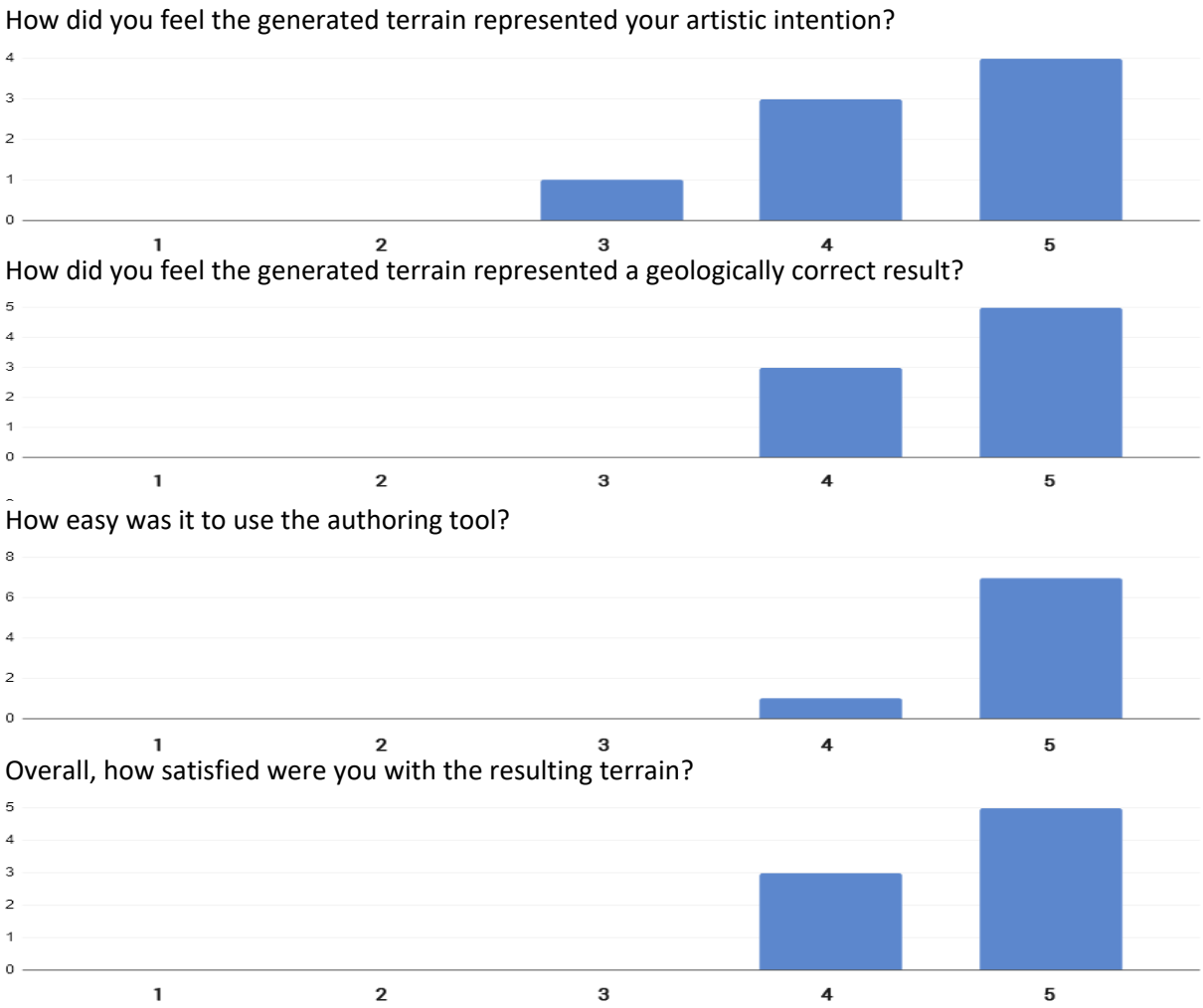


Figure 17. Usability test survey results.

Overall, most of the participants claimed they could design their intended terrain and found the tool easy to use. It is also notable that most participants described the result as geologically believable. It can be concluded that this approach is intuitive and simple and allows users to design terrains with little effort.

4.3 Limitations and constraints

The main drawback of this approach is that each generator is trained on a specific dataset. If there is a need to generate terrains based on new datasets, a new generator must be trained with this new data and added to the tool. This can be a lengthy process, however, we should note that training only needs to be performed once.

There were also some rare instances where the algorithm has more trouble in producing usable results, such as when the input sketch is very sparse, as if there is no input data for the algorithm to draw from, it cannot make an accurate guess of what the final result should look like. This can easily be alleviated by adding more sketches. We also counteract this by generating some intentionally sparse sketch maps during training process to prepare the algorithm to better deal with these cases.

Lastly, due to the nature of how cGANs work, the current implementation has an intentional size limitation of 512x512 as this was defined during training. Larger terrains can be generated by changing the parameters with pix2pix, however this will require retraining. A solution for this problem is to resample the generated terrains with an amplification procedure as described in section 3.6.

5. Critical Reflection & Conclusions

The main goal of this research project was to explore the potential for machine learning based solutions to content generation approaches, in the context of terrain generation. This thesis has provided a deeper insight into the potential of this emerging field, by showing that an implementation of an efficient and flexible framework for terrain generation can be driven by a general-purpose machine learning algorithm. This approach can achieve believable results that mimic complex natural processes, such as erosion, in much less time than conventional methods and can provide users with a good base to start from when designing terrains for games and 3D applications.

A fully automated robust pipeline for gathering and processing real world terrain data, that allowed for the generation of sketch maps that defined elevation and geomorphological features was tested and produced good results for future training models.

The development and implementation of the tech art authoring tool proved to be an illustrative example of real-world tech art tool development and allowed the exploration of several new systems and techniques. The result was an interactive application capable of running the trained model to synthesize terrains in seconds. Based on a qualitative test of the pipeline conducted on users with differing levels of experience, it can be concluded that a practical and efficient method was achieved. A quantitative analysis of the loss function evolution also confirms the success of the trained models as the loss value approached very low values. The exploration of erosion simulations also proved invaluable for the usability of the end results. The generated terrains were tested inside a game engine and reinforced the potential of this pipeline in achieving game ready results in a short amount of time.

An intended goal for future development of this project is to better integrate the tool with specific game engines in the form of a plug in, so that artists can better make use of the generative tools provided and preview the result directly in its intended use. Another promising possibility for research is to generalize the terrain generation of different features in the same network and to differentiate between them with different types of sketches in the authoring phase.

References

Affinelayer. (2019) Pix2pix-tensorflow. Available at: https://github.com/affinelayer/pix2pix-tensorflow (Accessed: 05 July 2020).

Chiba, N., Muraoka, K. and Fujita, K. (1998) 'An erosion model based on velocity fields for the visual simulation of mountain scenery.' *The Journal of Visualization and Computer Animation*, 9(4), pp.185-194.

Emelianov, H. (2019) 'Volumtric Clouds PT4', *Artstation,* 16 April. Available at: https://www.artstation.com/debug/blog/QB2N/volumtric-clouds-pt4 (Accessed: 22 June 2020).

Epic Games. (2015) 'Landscape Technical Guide', *Unreal Engine*. Available at: https://docs.unrealengine.com/en-US/Engine/Landscape/TechnicalGuide/index.html (Accessed: 13 June 2020).

Forselv, E. (2019) ModernGL. Available at: https://github.com/moderngl/moderngl (Accessed: 15 July 2020)

Fournier, A., Fussell, D. and Carpenter, L. (1982) 'Technical correspondence: comment on computer rendering of fractal stochastic models.' author's reply. *Communications of the ACM*, *25*(8), pp.583-584.

Gaea Bleeding Edge. (2019). Quadspinner.

Génevaux, J.D., Galin, E., Guérin, E., Peytavie, A. and Benes, B. (2013) 'Terrain generation using procedural models based on hydrology.' *ACM Transactions on Graphics (TOG)*, *32*(4), pp.1-13.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014) 'Generative adversarial nets.' In *Advances in neural information processing systems* (pp. 2672-2680).

Guérin, E., Digne, J., Galin, É., Peytavie, A., Wolf, C., Benes, B. and Martinez, B. (2017) 'Interactive example-based terrain authoring with conditional generative adversarial networks.' *Acm Transactions on Graphics (TOG)*, *36*(6), pp.1-13.

Guérin, E. (2018) 16 bits png support. Available at: https://github.com/affinelayer/pix2pix-tensorflow/pull/97/files (Accessed: 02 August 2020).

Hnaidi, H., Guérin, E., Akkouche, S., Peytavie, A. and Galin, E. (2010) 'Feature based terrain generation using diffusion equation.' In *Computer Graphics Forum* (Vol. 29, No. 7, pp. 2179-2186). Oxford, UK: Blackwell Publishing Ltd.

Houdini: Procedural Dependency Graph (PDG). (2017) SideFX.

Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A. (2017) 'Image-to-image translation with conditional adversarial networks.' In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125-1134).

MAWI United GmbH. (2020) 'MW Landscape Material', *Unreal Engine,* 6 February. Available at: https://www.unrealengine.com/marketplace/en-US/product/mw-landscape-material (Accessed: 15 June 2020).

Mirza, M. and Osindero, S. (2014) 'Conditional generative adversarial nets.' *arXiv preprint arXiv:1411.1784*.

Musgrave, F.K., Kolb, C.E. and Mace, R.S. (1989) 'The synthesis and rendering of eroded fractal terrains.' *ACM Siggraph Computer Graphics*, 23(3), pp.41-50.

Nagashima, K. (1997) 'Computer generation of eroded valley and mountain terrains.' *The Visual Computer*, 9(13), pp.456-464.

Nordin, M. (2018) 'Deep Learning: Beyond the Hype.' Game Developers Conference, 19 March, San Francisco, California.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T. and Efros, A.A. (2016) 'Context encoders: Feature learning by inpainting.' In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2536-2544).

Peytavie, A., Galin, E., Grosjean, J. and Mérillou, S. (2009) 'Arches: a framework for modeling complex terrains.' In *Computer Graphics Forum* (Vol. 28, No. 2, pp. 457-467). Oxford, UK: Blackwell Publishing Ltd.

SideFX. (2019) 'Machine Learning Data Preparation', SideFX*,* 09 April. Available at: https://www.sidefx.com/tutorials/machine-learning-data-preparation/ (Accessed: 05 March 2020).

Tasse, F.P., Emilien, A., Cani, M.P., Hahmann, S. and Bernhardt, A. (2014) 'First person sketch-based terrain editing. '

Unreal Engine 4. (2019). Epic Games.

USGS. (2011) '3D Elevation Program' Available at: https://www.usgs.gov/core-science-systems/ngp/3dep (Accessed: 14 May 2020)

Zhou, H., Sun, J., Turk, G. and Rehg, J.M. (2007) 'Terrain synthesis from digital elevation models.' *IEEE transactions on visualization and computer graphics*, *13*(4), pp.834-848.

Appendices List