h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Simplification of Terrain Generation Authoring using Conditional Generative Adversarial Networks

Christian Lott, 757263 | Semester 7 | Winter Term 2020/2021

+animation game
B.A. ANIMATION AND GAME
AG.MEDIENCAMPUS.H-DA.DE
M.A. ANIMATION AND GAME DIRECTION
AGD.MEDIENCAMPUS.H-DA.DE

ABSTRACT

High-level terrain generation authoring is a task for which there is no simple solution to date. While procedural terrain generation algorithms generate highly realistic fractal terrains, it is not easy to control the creation process since they are generated by random numbers.

↓

The introduction of neural networks into various production workflows to improve quality and speed or introduce new capabilities is frequently carried out as machine learning research is rapidly advancing. In the field of procedural terrain generation, generative adversarial networks can be leveraged to allow the user to easily author the terrain generation process, meaning that a user without technical knowledge of procedural terrain generation could add a mountain range by simply drawing a line or remove a river by simply deleting the river from a height map representation of the terrain. Presented is a method for generating terrain height maps from scratch by drawing simple lines and points on a small canvas.

RELATED WORKS

For procedural terrain generation , three primary methods are existing. Subdivision schemes, Faulting, and Noise based procedural generation. The last one is the most commonly used method for procedural terrain generation. All of these methods are based on random procedures, reducing artistic control.

Local Procedural Terrain Generation was introduced to reintroduce artistic authoring back into this process. This is done using terrain sketching methods or by introducing high-level parameters etc.; this, however, requires much work to implement, and for really enabling high-level authoring, much work has to be invested into the terrain.

Generative Adversarial Networks (GAN) are a form of machine learning where two networks are trained adversarially to each other with a min-max objective of the loss. A Discriminator network is trained to distinguish between real and fake images, and a Generator network is trained to fool this Discriminator. Once both networks are trained, the Generator network can be leveraged to generate images from a descriptive input in real-time.

REFERENCES

[1] Ian J. Goodfellow et al.Generative Adversarial Networks.2014. arXiv:1406.2661[stat.ML].

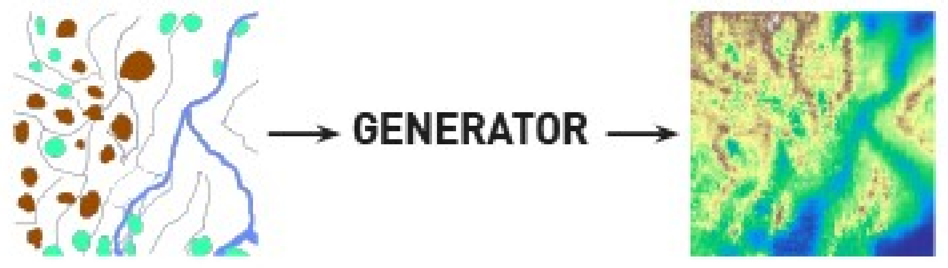
[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox.U-Net: Convolutional Net-works for Biomedical Image Segmentation.2015. arXiv:1505.04597 [cs.CV].

[3] Chuan Li and Michael Wand.Precomputed Real-Time Texture Synthesis with Marko-vian Generative Adversarial Networks.2016. arXiv:1604.04382 [cs.CV].

[4] Nasa. "NASA Images." In: (2020). url:https://visibleearth.nasa.gov/images/73934/topography(visited /15/2021).

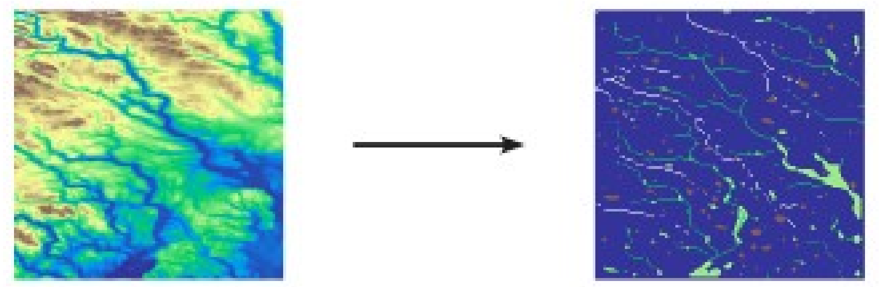
METHODOLOGY

The Idea of the approach is to take sketches and convert them into height maps by synthesizing them with a conditional Generator network trained inside of a GAN.

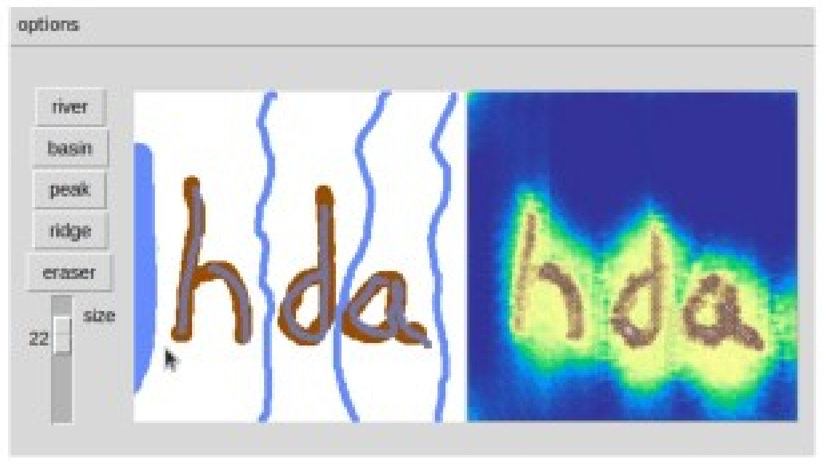


For the generation of the height maps from an input sketch, a GAN is implemented that uses a similar network architecture to the one used for pix2pix [1], for this being a proven network for image synthesis out of descriptive input. Both the Generator and the Discriminator are fully convolutional neural networks, with the Generator uses the U-Net architecture [2] and the Discriminator a 16x16 PatchGAN architecture [3].

Because a GAN needs a massive amount of training data to prevent overfitting satellite images from NASA [4], it is used and converted into the right format to act as training data. Each of the created height maps will get converted to a sketch map based on the height of the respective feature on the heightmap. These height and sketch maps are then saved as the training data for the GAN training. The whole process provides the GAN with a total of 5035 training examples.

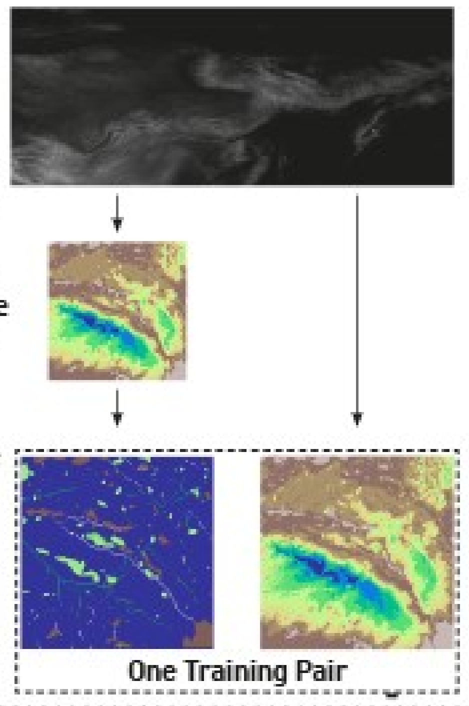


Once the model is trained, the user can simply create a drawing, and the generator will generate a heightmap from it. To enable the user to create such a drawing, a tool is created for the drawing process. The tool enables the user to draw the simple lines onto a small canvas and having the tool generate the heightmap in real-time, even during the drawing. This enables the user to see how the lines are affecting the terrain while drawing the input sketch. The heightmap is loaded into Blender, and a simple terrain shader is applied to the model.



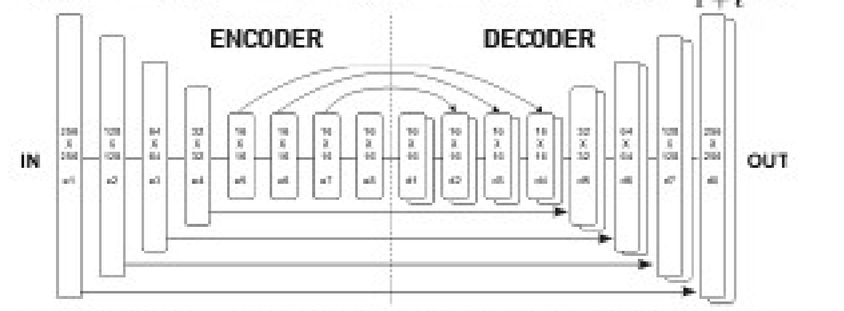
IMPLEMENTATION

The Preparation of the training data was implemented using the Python PySheds library and The Georaster Library. First, The large satellite height images from NASA were cut into patches of 256 x 256 pixels using the Georaster package. The height maps were then used as the ground truth images for the training data. The Sketches were extracted using a low detail variant of the high map patches. Through the PySheds package, rivers, basins, ridges, and peaks have been extracted and saved to a sketch map that was saved as the sketch map training data.

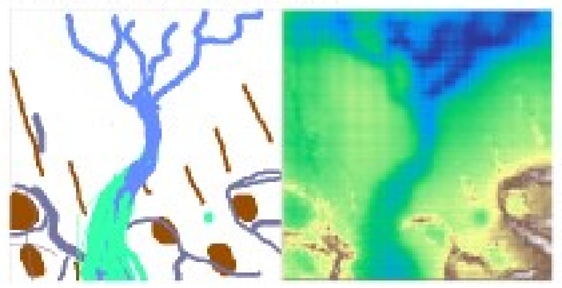


The GAN was implemented using the TensorFlow Keras library for python. The Generator is built with an UNet architecture consisting of an input layer, 8 encoder layers, 8 decoder layers, and one output layer. The encoder layers' activation function is Leaky ReLU, for the decoder layer ReLU and the output Than. The activation function for the discriminator layer is Leaky ReLU. The Discriminator is implemented with a 16x16 PatchGAN architecture comparable with the Generator's encoder part.

$$ReLU(x) = max(0, x) \quad LeakyReLU(x) = max(\alpha x, x) \quad tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$



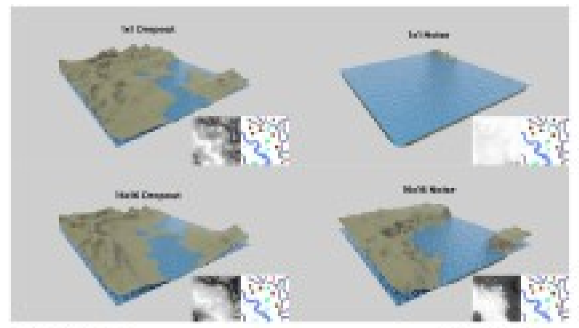
The Tool was implemented using the python TkInter GUI toolkit. The front end of the tool consists of 3 columns, one for settings, one for the drawing canvas, and one for a preview of the current heightmap. The settings allow for switching between the features to paint and the size of the brush. The features are buttons changing the brush's value and color to draw respective to the chosen filter, and the size is a slider changing the size value of the brush. Drawing is done by getting the x and y position of the mouse every iteration and drawing a line between the two positions. Additionally, every iteration, the sketch is sent to the generator, and the heightmap is generated, which is loaded into the third column.

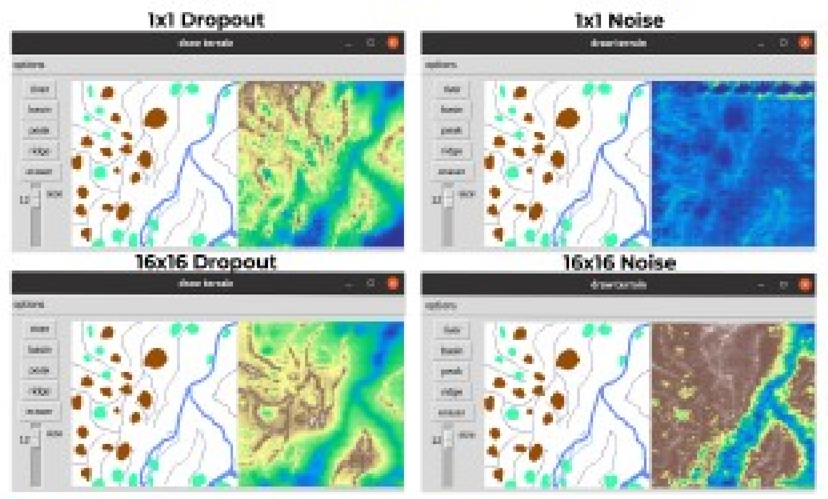


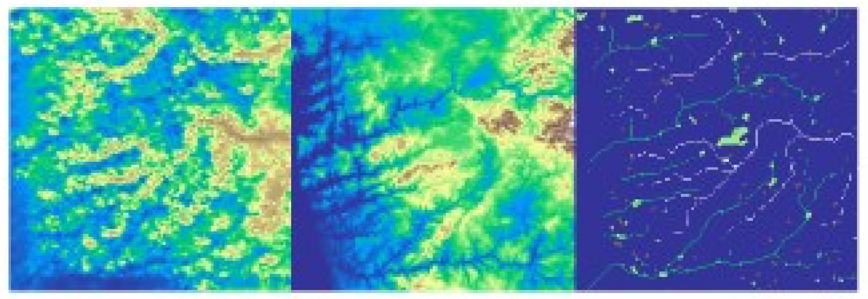
EXPERIMENTS

Four versions of the Generator network were tested to improve the quality of the generated heightmaps, two with random noise as input and two that introduced the randomness by a random dropout in the convolutional layers. In each of these cases, one network reduced the resolution to 16x16 in the encoder, and the other one reduced the resolution to 1x1 in the encoder.

RESULTS







CONCLUSION

The results show that realistic terrain can be generated in real-time for the generation from the sketch to the heightmap only takes 6 ms. Regarding the different versions that have been tested, the most realistic and smooth results in the renders came from the version that sampled the input down to 16x16 and used random dropouts instead of noise, so considering further research and tool development, this is the version which is to be used. The proposed approach allows the user to easily author terrain creation. Although the results are not perfect and do not create a production-ready terrain from just an input drawing, the approach still demonstrates that terrain defining features/characteristics can be created or edited by merely drawing or removing a single line or shape. However, for using this method in production, a combination with state of the art procedural generation techniques could improve the results' quality to a production-ready level.