WorldGAN: 3D Environment Generation Using Deep Adversarial Networks

Tim Whitaker
Colorado State University

December 6, 2019

1 Abstract

We introduce a method and application for generating 3D environments with generative adversarial networks. The outputs from our networks are able to be imported into popular three dimensional modeling software like Unreal, Unity and Blender. We've created a simple and easy to use web based interface to allow a user to control the output of the generated environment by painting with predefined brushes that represent the terrain varieties: mountains, hills, terraces, plains, and rivers.

2 Introduction

This project explores the usage of conditional generative adversarial networks to synthesize 3 dimensional environments.

The motivations for this project are to make it easier for game, world or environment designers to create realistic environments procedurally while giving the designer more control over the results.

We aim to create a simple and easy to use GUI interface for a user to paint a 2 dimensional semantically significant image map. We then pass the user created image map through a generative adversarial network trained on elevation data gathered from the Earth in order to generate a realistic terrain heightmap that approximately represents the user defined map. The resulting heightmap produced from the neural network can then be uploaded into any of the popular 3 dimensional rendering software including Unity, Unreal, Blender or World Machine.

3 Background

This project involves work touching on a number of fields of computer science. Combining machine learning, procedural generation, and three dimensional content is a relatively new area of research.

3.1 Procedural Content Generation

Procedural content generation is an algorithmic and stochastic process used in art, games, music and film. As a paradigm for content generation, it is used to increase replayability, variability, and flexibility. It can reduce production cost and effort and it can save on storage space. [1]

3.2 Generative Adversarial Networks

Neural networks have become a pervasive part of computer science. It allows for a powerful model of learning patterns from complex datasets. Generative adversarial networks work by synthesizing novel data according to patterns it learns in the training set. We evaluate two models of conditional generative adversarial networks to achieve image to image translation.

We are going to evaluate two implementations of a specialized form of generative adversarial networks designed to perform conditional image to image translation. The two models we are using are called Pix2Pix and CycleGAN.

3.2.1 Pix2Pix

3.2.2 CycleGAN

3.3 Related Work

Nvidia GauGAN.

4 Methodology

In this section we go over all of the pieces of work that went in to this project.

4.1 User Interface Application

The user interface is a web based javascript application. The application allows a user to paint a semantically significant image using predefined brushes.

The backend is powered by a Flask API.

4.2 The Dataset

Generative adversarial networks need a good amoount of training data to output good results and a key objective of this project is to generate realistic results. This leads to an interesting challenge of putting together a realistic yet large enough dataset. In order to achieve this, we use terrain data of the earth as opposed to fictional or manufactured terrain data. In the next two sections we go into the methodology behind building the dataset.

4.2.1 Training Data

I used an open source map tool https://github.com/tangrams/heightmapper. This tool is an interactive application elevation data gathered from Mapzen's global elevation service https://www.mapzen.com/blog/elevation/.

I then built a program that controls a headless browser using an open source library called puppeteer https://github.com/puppeteer/puppeteer. This script loads a random coordinate, checks if we are over land and saves an image of the elevation data at that coordinate. We resize the saved image to be 512 x 512 pixels.

I then started manually building the usermap training set using the user interface application I built. For each image in the heightmap training set, I would paint a usermap to represent that heightmap, using the semantically significant brushes that most closely matched the terrain (mountains, hills, plains, rivers, terrace).

After this process we are left with 258 paired usermaps and heightmaps. We group all of these together into batch one.

We then created a second batch by taking batch one and augmenting the images to increase the size of a training set by 5x. We performed 5 operations on each usermap and heightmap. We mirror horizontally, flip vertically, rotate 90 derees, rotate 180 degrees, and rotate 270 degrees.

Pix2pix requires the training data from both the heightmaps and usermaps need to be paired. We stack each pair of training images horizontally and save this new image into our training batch. The resulting images are 1024 x 512 pixels.

4.2.2 Test Data

The test data is a collection of 5 usermaps I drew that are designed to be representativ of a variety of environments a user might draw. These 5 environments include a mountainous region, a mountainous region fading into a hilly region, a hilly region fading into a flat region, a region with a main river and many sub rivers running through mountains, and a combined region containing mountains, hills, plains and a river.

4.3 Training The Models

We trained two models on this initial dataset, Pix2Pix and CycleGAN. GTX 1080ti GPU. i7 8700k. processor.

4.3.1 Pix2Pix

1 hour training time. 4 hours training time.

4.3.2 CycleGAN

4 hours training time. 24 hours training time.

4.4 Visualizations Using 3D Software

Visualizations of results with 3d software.

5 Results

5.1 Visualizations Using 3D Software

6 Conclusion

6.1 Future Work

References

[1] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgrd, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml), 2017.