WorldGAN: 3D Environment Generation Using Deep Adversarial Networks

Tim Whitaker

September 16, 2019

1 Introduction

Generative adversarial networks are a class of machine learning models that have become very popular since their inception in 2014. They've been the engine behind some of the most popular demos in the field, including Deep Fakes, This Person Does Not Exist, and Text-to-Image translation. These models are powerful in generating new and creative content, and I think they could be a good aid to humans in designing 3 dimensional environments for video games.

I am proposing a method of allowing a user to draw a top-down 2-dimensional segmentation map, using a GUI akin to MS Paint and using a generative adversarial network to translate that into a 3-dimensional environment in Unreal Engine.

2 Background

The motivation for this project is to help game designers build environments and worlds easier and more quickly. This method acts as a form of pseudo-procedural generation, and would allow for an endless amount of variation in content, without needing a user to manually define these variations.

Generative adversarial networks have only been applied to 3 dimensions very recently, and this work has only been done with modeling 3d objects like chairs and tables. The idea for this research is mostly inspired by the Nvidia GauGAN project http://nvidia-research-mingyuliu.com/gaugan/ and the relevant topics I'll be researching include procedural generation and conditional image synthesis.

2.1 Procedural Generation

Procedural generation has been used in the gaming industry for decades. It allows game designers to introduce variation to a game without needing to explicitly build each variant by hand. This can introduce millions of options on both the environment and the content, increasing the replayability value of a game and leading to a novel experience on each playthrough. This method is very popular in dungeon crawlers like nethack and rogue, and more recently it's been applied to massive 3-dimensional worlds like No Man's Sky and Minecraft.

2.2 Conditional Image Synthesis

Conditional image synthesis is the process of generating an image conditionally from a dataset based on a class label. The two big categories are text-to-image and image-to-image translation. The famous problems of style transfer, black and white photo colorization, and super resolution all fall into this category. GANs are a class of models that are often used for the task of conditional image synthesis.

2.3 Literature Review

The following is a collection of literature that are relevant to this project. Most of the content here deals with GANs and procedural generation.

- Physically Based Rendering: From Theory to Implementation
  https://www.pbrt.org/
- Semantic Image Synthesis with Spatially-Adaptive Normalization
  https://arxiv.org/abs/1903.07291
- Procedural Content Generation via Machine Learning (PCGML)
  https://arxiv.org/abs/1702.00539
- Generative Adversarial Networks
  https://arxiv.org/abs/1406.2661
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
  https://arxiv.org/abs/1511.06434
- Conditional Generative Adversarial Networks
  https://arxiv.org/abs/1411.1784
- Image-to-Image Translation with Conditional Adversarial Networks
  https://arxiv.org/abs/1611.07004

3 Methodology

I see this project broken up into three parts. First is the user interface. A simple GUI akin to MS Paint is sufficient and will allow a user to draw a 2d top-down semantic map. Second is creating and training the GAN to generate the terrain heightmap/mesh from the user input. Third is generating all of the style, textures, and content, like trees and rocks, that will be placed in/on the terrain. Due to the nature of Unreal Engine, I'm not sure if this step can be done with a machine learning model. I will need to experiment with Unreal Engine more to see if it's possible and if it's not, then more traditional forms of procedural generation may be used for this step.

I will need to build up a set of training data to bootstrap my GAN. I will do this manually by drawing both the input and output heightmaps using photoshop. It may be possible to augment this process by using volunteers and/or generating the inputs programmatically and creating the outputs manually.

3.1 Deliverables

- A simple user interface to allow a user to draw a 2d map using semantically significant brushes.
- A dataset of 2d images and heightmaps to train the GAN on.
- A GAN that will translate that 2d map to a 3d heightmap/mesh.
- A method for applying texture and objects to the 3d heightmap in Unreal Engine.