Proceedings of CONF-MLA 2024 Workshop: Securing the Future: Empowering Cyber Defense with Machine Learning and Deep Learning
DOI: 10.54254/2755-2721/110/2024MELB0114

Scene Modeling in Game Development Based on Generative Adversarial Networks

Haotian Liu

College of mathematics and informatics, South China Agricultural University, Guangzhou, China.

chengjuyi@ldy.edu.rs

Abstract. Due to booming advance on generative models, people have great interest on designing model structure to produce wonderful pictures, or even 3d shapes. The motivation of this work is that the 3d modeling manufacturing in game development is still challenging and time-consuming, and 3d shape produced from generative models may be powerful tools for the problem. The work tried to build game scenes, such as a city and a cave, the typical scene that requires many random but similar objects. This paper aims to explore a complete workflow for applying the GANs to the game development. The structure of the paper is introducing the background of the game development and the progress of the generative model, giving interpretation about the principle of Generative Adversarial Networks, and propose the process on how to utilize it to improve the productivity in developing game scene. Finally, this work found that it could considerably reduce the repetitive work on making massive and similar objects.

Keywords: Scene modeling, game development, generative adversarial networks.

1. Introduction

Game development industry improves due to the progress of generative model of artificial intelligence. It is noticeable that the game Scene Modelling, being considered as one of the most time-consuming and laborious, could be accomplished more efficiently with the help of the generative algorithm of computer. Unlike Character Modeling that requires high-quality and detailed 3d shape, it is acceptable for Scene Modelling to overlook some details, for example the location of objects in the scene is arbitrary, while things are well-organized in terms of characters.

There are two steps in Scene Modeling, constructing terrain and creating objects in the scene. Nowadays, thanks to Procedural Modeling technology, developer can generate slightly different terrains and objects. However, sometimes it is difficult for them to find out the generative rule for specific object in Procedural Modeling. In terms of deep learning, computer could learn the shape rules from 3d shapes by its own way, without the specific rules designed by developer.

As a result, recently, massive researchers propose various deep learning technology to implement the 3d shapes generation [1].

Research primarily focuses on improving the resolution and generation accuracy of models and rebuilding the scene for machine vision. So, this paper concentrates on generation accuracy, model lightweighting, and challenges in practical deployment within game environments.

DOI: 10.54254/2755-2721/110/2024MELB0114

This paper is served as a case to apply modern technology to real workplace. The goal is to create objects for large but structurally monotonous and irregular scenes (cities, forest asteroid belt or junkyard) by using GANs, meanwhile, build the whole scene by procedural modeling. The whole workflow is to input an initial 3d shape, then the algorithm will generate similar 3d model but with various features that can be put in a scene together, naturally. And the terrain is constructed by the Procedural Modeling. The experiment will give an instruction on how to improve our efficiency in building the game scene by leveraging on modern deep learning technology.

In the aspect of technology, the work is based on a modified GANs [2,3], a deep generative model that can learn from a 3d shape, then the model can generate the similar objects, without inputting massive data (for example many pictures in 3d reconstruction task), which relieve the burden on modeling a large number of same type of objects.

2. Related work

Scene Modeling, one of the most crucial and laborious steps in animation and game industry, has many different types of generative methods, consist of software (blender or Maya), Procedural Modeling and Artificial Intelligence algorithm generation. They have been widely used in the industry, while deep learning has not, the usage of Generative Adversarial Networks in game development is not that mature. Since, currently, the majority of studies on deep learning mainly focus on the novel work on generative models.

There are a large variety of novel generative models for modeling have been issued in the past decade. Variational Auto-Encoders (VAEs) [4], consist of an encoder and decoder, is initially fed many pictures, the encoder would gain the variation and expectation from the given data, fitting the approximately same variation and expectation by using Gaussian Distribution, but at present they are applied to generate images. Another prevalent technique is Neural Radiance Field, an effective approach to solve 3d reconstruction tasks, generating 3d shapes through learning different views of the models (several pictures). It relies on the data from Volume Rendering. And finally GANs is the technique that used in this paper.

Recent work on obtaining 3d shape from generative models has many novel ideas, including producing 2d images [5] and 3d shapes. People proposed Neural Radiance Fields (NeRF) to reconstruct the scene from the given 2d pictures, and recently, novel method, 3D Gaussian Splatting [6], has been proved that perform well in scene reconstruction. On the other hand, many technologies based on GANs [1], such as SinGAN [3] and StyleGAN, could produce good 3d shape.

Although these technologies are fascinated and novel, applying them to the industry is equally important [7,8]. There are some potential challenges in application. First of all, the compatibility between 3d shapes generator and game engine. According to the recent researches, the 3d shapes produced are point clouds, or voxels, while meshes tend to be acceptable in game engine. Furthermore, the workflow from training and inputting 3d shapes to gain different objects is not mature, which cannot virtually improve the efficiency in game development process.

Therefore, this paper will give a workflow to combine the GANs that produced 3d shapes with the game engine. We will show the process how to train and acquire many 3d shapes and construct a game scene based on GANs [2].

3. Method

The main workflow of this work includes obtaining the initial 3D shape, training a deep learning model using this shape as input, randomly generating new 3D shapes, selecting the shapes suitable for the scene, coloring the selected shapes, and ultimately constructing the complete scene.

3.1. Shape representation

- Encoder:

DOI: 10.54254/2755-2721/110/2024MELB0114

The encoder receives a three-dimensional Gaussian noise as input and transforms it into three planes, which store the key feature information of the shape. This process leverages the research on neural implicit functions, compressing the 3D shape information into 2D feature maps.

- Decoder:

Opposite to the encoder, the decoder is responsible for reconstructing the 3D shape from the three feature maps. This involves projecting each location point P (such as a voxel position that makes up the entire shape) onto the three directions to obtain its coordinates on each plane. Then, using bilinear interpolation, the corresponding position vectors are extracted from the feature planes based on these coordinates. Finally, the three position vectors are concatenated into a feature vector, which is then used to reconstruct the original 3D shape.

The representation of the 3d shape in Tri-plane feature maps was proposed by the recent research about neural implicit function, encoding from 3d shapes to 2d feature maps. After that, the position P (for example the location of certain voxel that constructed the whole shape) is projected onto three directions to acquire the 2d coordinates that have bilinear interpolation with position of planes to obtain position vectors. Concatenating these three vectors to get feature vector, the vector will be transformed into 3d shape.

Therefore, the generators virtually convolute the inputting 2d feature maps and noise to produce another random 2d feature planes, considered as a processors to realize our goal to acquire same type of 3d object but with various appearance.

3.2. GAN

GANs, mainly consist of Generators and Discriminators, learn the data by the competition between generators and discriminator. First of all, to generate random 3d models, the first Generators are fed with a 3d gaussian noise that projected along three axis onto three feature planes. Unlike voxels in 3d representation, the feature planes storage the 3d data on three 2d images. In terms of generating shapes in different scales, the generators are hierarchized, larger feature planes produced through computing 3d noise and upsampled planes. The enlarging data 2d maps in generators, virtually CNN (convolutional neural networks), pass through the whole hierarchy, finally to be certain feature planes with wanted size.

Furthermore, as for training the GANs, the goal of this process is to get samples from source shape, and learn the distribution of the wanted shape, making the generators produce the data in similar variance and expectancy....In terms of discriminators, WGAN-GP's training objective is adapted for discriminators, which has two terms, EM Distance (Wasserstein Distance), to gauge the distribution difference between the generation and the real shape, and Gradient Penalty to control the fitting degree. And it is obvious that the goal is to maximize the formula: D(parameter)= EM Distance - Gradient Penalty. Fit the model by using ADAM algorithm.

The Generators share the same structure, inputting i-level 3 feature planes with three 2s noise maps and outputting i+1 level planes, and the core is the convolutional networks that compute the source data. The task of generators is to receive three feature planes in i-level with 3d noise, and produce a higher level of mixed features maps that will be transformed into shape through decoder.

In order to verify the effectiveness of the application of deep learning networks in the practice, we give two typical scenes with massive similar objects or terrain in the game, city and cave.

- Generation city

According to the imported 3d triangular mesh model into Blender, the meshes are too rough, therefore the model surface is subdivided, also, unwrap the whole shape, allowing the texture to apply to the shape. In order to construct a complete city, several 3d models are prepared, namely building, tree and factory.

In terms of the terrain, we will use the procedural modeling to realize such as height map and texture image.

- Generation cave

DOI: 10.54254/2755-2721/110/2024MELB0114

The other example is cave. Unlike the city, the terrain will be constructed by GANs. At the beginning, we can gain a minimal terrain but has full information express the cave. Then we can get a customized scale of the scene, with various elements. Meanwhile, the objects in the scene come from the same approach. The challenge in this process is texturing the whole terrain.

After accomplish the static part of the scene (texturing map, normal map, and objects). we develop shaders that accommodate to the terrain and shapes to make the scene more vibrant.

4. Result and discussion

4.1. Experiment setup

The experiment was conducted on a computer equipped with an AMD Ryzen 7 4800H processor, an NVIDIA GeForce GTX 1660 Ti graphics card, and 16GB of RAM. The software system included the Windows 10 operating system, along with the necessary software: Blender for UV mapping and shading, and Unity 2022.3.20f1c1 for implementation. Additionally, the development environment for the GANs utilized Python 3.10, PyTorch 2.0.0 with CUDA cu118, and trimesh 4.4.7.

The GANs were trained using the Adam optimizer over the course of 2000 iterations, with a learning rate established at 1e-4. To achieve a diverse range of tree scales, the resizing parameters were configured such that the height (y-axis) varied between 1 and 1.5, while the dimensions along the other two axes remained constant. Furthermore, the model was configured to perform upsampling five times to enhance the resolution of the generated outputs.

4.2. Result and analysis

Once trained, the GANs can generate a large number of random 3d shapes with wanted. We acquire a variety of trees from GANs for city (Figure 1).

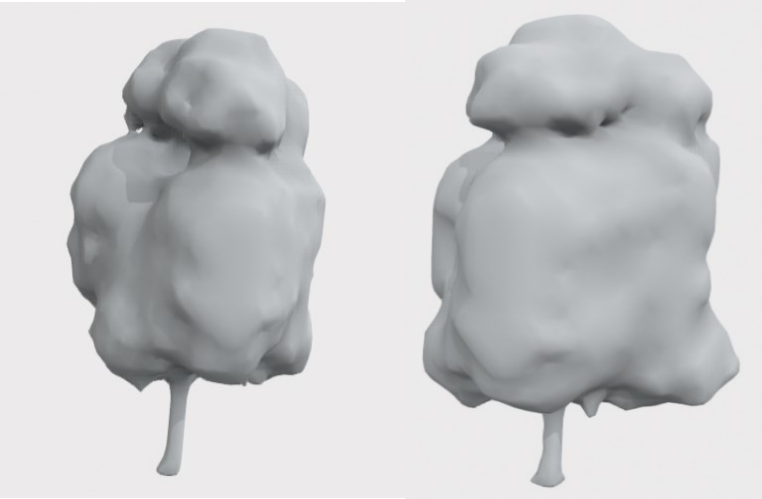

Figure 1. Generated trees (Photo/Picture credit : Original)

As figure 2, there are a range of stalagmites produced to construct the cave. The formation of stalagmite is a natural process with randomness, which is suitable for GANs to realize.

DOI: 10.54254/2755-2721/110/2024MELB0114

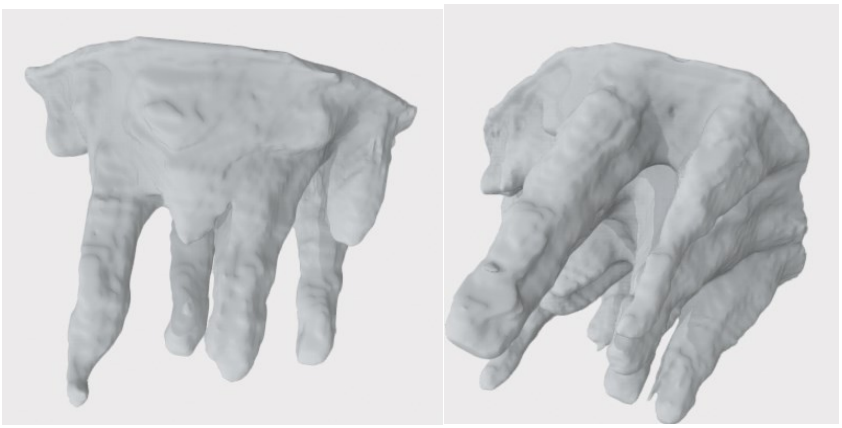

Figure 2. Generated stalagmites (Photo/Picture credit : Original)

The implementation of GANs in the modeling of trees demonstrates a notable advancement in the efficiency of 3D asset generation. As illustrated, the approach facilitates the easy acquisition of trees with varying heights, allowing users to manipulate size and orientation directly through the user interface and efficiently export the models to OBJ files. This capability significantly enhances the modeling workflow by streamlining the asset creation process. However, it is apparent that the GANs exhibit limitations in accurately replicating the intricate shapes of leaves, which restricts their applicability primarily to low-quality gaming environments or as distant background elements in scenes. This limitation underscores the necessity for ongoing improvements in the GAN architecture to enhance the fidelity of generated foliage.

In the Unity environment, a diverse assortment of randomly generated trees was imported into the terrain editor. The placement of these trees was accomplished with ease by utilizing the 'Paint Trees' tool, allowing for random distribution across the landscape through simple point-and-click interactions (Figure 3). For instance, in the construction of a cityscape utilizing objects generated by the GANs, it is acknowledged that while the detail in the trees may be lacking, the visual representation remains acceptable from a distance, thereby satisfying aesthetic requirements when finer details are not discernible to the viewer.

Figure 3. City (Photo/Picture credit : Original)

In comparison to traditional methodologies for creating large quantities of diverse objects, such as manual modeling or procedural modeling, the use of deep learning models like GANs presents significant advantages. The generation of 3D forms through interactive software can often be both costly and time-consuming. Although procedural modeling addresses these issues by employing specified generative rules, it inherently restricts variability and randomness, confining creations within predefined

DOI: 10.54254/2755-2721/110/2024MELB0114

parameters. By overcoming these limitations, GANs are capable of rapidly producing a wide variety of objects, yielding greater diversity and spontaneity in the asset generation process.

5. Conclusion

This paper proposes a scheme for applying cutting-edge technology to game scene creation, encompassing the design of original 3D shapes, the random duplication of shapes through GANs, the construction of scenes, and rendering, along with a simple explanation of the generative models. The work aims to address the challenges associated with transplanting 3D objects generated by models into 3D software and game engines. However, there are currently some unavoidable challenges, such as poor detail regeneration (e.g., tree leaves), the ability of deep learning generative models to produce only rough shapes, and the difficulty in rendering the meshes generated by these models (e.g., UV mapping). In conclusion, recent research on generative models for producing 3D shapes can significantly enhance the productivity of developing mini-games or games that do not require high resolution or high quality. In the future, there may be interest in designing generative models that can more accurately learn detailed features from objects, for example, through interactive structures where users can specify and provide detailed information to the models. Additionally, advanced geometric algorithms may be proposed to address the incompatibility between generative models and game engines.

References

[1] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *ArXiv. https://arxiv.org /abs/1406.2661*

[2] Wu, R., & Zheng, C. (2022). Learning to generate 3D shapes from a single example. *ACM Transactions on Graphics*, 41(6), Article 224, 19 pages.

[3] Shaham, T. R., Dekel, T., & Michaeli, T. (2019). SinGAN: Learning a generative model from a single natural image. *In 2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 4569-4579).

[4] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of Wasserstein GANs. *In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)* (pp. 5769–5779).

[5] Wang, T. -C., Liu, M. -Y., Zhu, J. -Y., Tao, A., Kautz, J., & Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional GANs. *In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 8798-8807).

[6] Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), Article 139.

[7] Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2019). A style-based generator architecture for generative adversarial networks. *In 2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 4401-4410).

[8] Jiang, Z., Yu, Y., & He, Z. (2020). Taming transformers for high-resolution image synthesis. *In Proceedings of the 37th International Conference on Machine Learning (ICML)* (Vol. 119, pp. 882-893).