Real-time View-Space Grass Generation for Video Games using GAN

Authors

Prashant Karn
Anju Chhetri
Tripti Sharma
Pramesh Shrestha

Abstract

Real-time visual element generation using GAN is a field gaining momentum in recent years. This project particularly explores the generation of grass in video games in the view-space, by using a Neural Rendering model. Since there is not much past work or data available related to this, the scope of this project includes specification of this model optimisation problem, creation of a mock game, highly efficient procedurally generated grass using traditional methods, a system to automatically generate datasets as required with a speed sufficient enough to supply data faster than can be used for training, a real-time testing environment using an integrated neural rendering engine within the render-pipeline, and research work conducting experiments using GAN. The target of this research work is to achieve subjectively good looking grass that can run at more than 10 fps. With the dataset generator and mock game working properly as verified by unit and relationship testing, the research work involved over a hundred experiments exploring five major neural network architectures seen in past work and created in accordance to insights gained from experiments in this project - UNet, Style Transfer, ResNet blocks, SPADE ResNet blocks, and Hybrid models. Many of these experiments produced models that successfully achieved the target of being real-time. Applications of this type of research include better rejection of unneeded information such as shape and identity of grass blades during their rendering, infinite LOD in procedural generation, and faster rendering of complex visual elements more resilient to factors such as number and density.

Introduction

Grass for a long time has been symbolic of a simple, ordinary aspect in reality that has been impossibly difficult to portray realistically in video games. The thousands to millions of grass blades stand out compared to other game elements. There is obviously redundant information being stored in valuable memory space with grass, which during game-play is easy enough to forget about, and the exact location of every single grass blade does not matter.

A compression approach therefore called for which selectively reduces storage of such irrelevant information, and is able to generate believable grass in the view without storing each grass element in world space. We explore the use of Generative Adversarial Networks (GAN) to achieve this.

Related work

Neural rendering [1] offers limiting grass-based information within the view-space, in one or multiple frame-buffers, which could be orders of magnitude cheaper than procedural grass. Generative Adversarial Networks (GANs) are a big part of this, being able to synthesise complex imagery. They have reached the stage of aiding procedural generation in 3D environments already [2]. Stable Diffusion [3] and similar image-to-image generation models can already take a simple, flat rendition of grass and add detail with depth to it. Wang et al. [4] present a way to replace all of the lighting and most of the rendering pipeline with GANs, to generate every visual element on the screen using Neural Rendering. Richter et al. [5] take a different approach, style-transferring realistic visuals on top of fully rendered video game scenes. Mittermueller et al. [6] provide some lighting information, and various maps, including untextured or lightly textured albedo, normal and depth map of a scene along with semantic labels on view-space segments to their CycleGAN based model to generate a realistic scene, combining the two approaches into a realistic, but crucially not yet real-time, application. The natural next step is real-time application of Neural Rendering.

Spatially Adaptive normalisation (SPADE) [7] proposes an alternative to the traditional U-Net architecture to generate images with only half as many layers while still retaining input information, which has been crucial for achieving real-time processing.

Approach

We chose a subset of Mittermueller et al. [6]'s list of types of inputs and generated a final dataset with about 15,000 entries through random movement in a mock video game from the point of view of a playable character, in a world full of procedurally generated

grass. Each dataset entry consists of three input and four output images, all of dimensions 800x480.

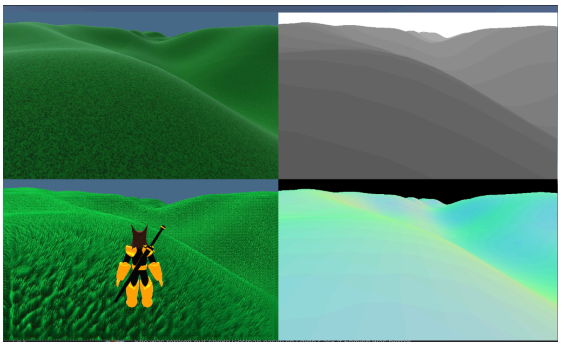

Fig: Constituents of the dataset:

- *View without grass, with world-space noise (top-left)*
- *Depth map of the view (top-right)*
- *Normals map of the land (bottom-right)*

Bottom-Left is a real-time output produced in our Unity environment through our Generator, with a character overlayed.

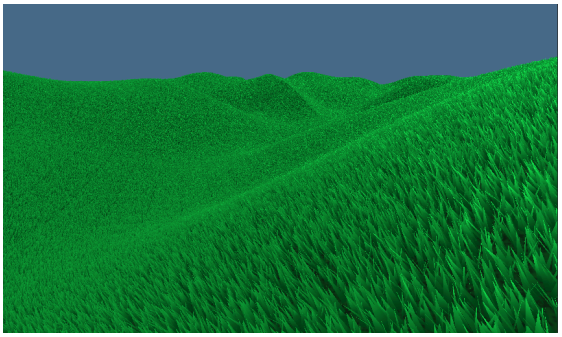

Fig: Procedurally generated grass used as example outputs in our dataset

We passed the three input images, which add up to an 800x480x7 tensor into a fine tuned Generator network consisting of a mixture of Half-SPADE Resblks and up-convolution layers. Two architectures produced results better than the rest, one suited for fast generation, and another for good quality.

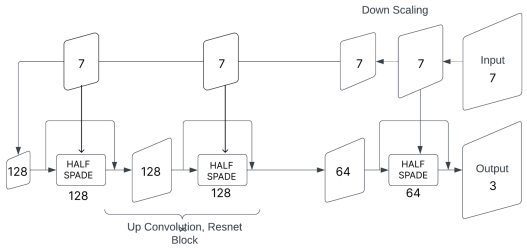

Fig: Fast Network Architecture

Half-SPADE Resblks are an altered version of SPADE Resblks [7] with the second SPADE layer omitted for faster processing, and considering the fact that image segmentation is trivial for our inputs, and primary concern is information retention. No noise was added at any point of the generation - instead, a fixed world-space noise was added to the input image.

Our high quality model consists of a single Half-SPADE Resblk with the others replaced with two generic Resnet Blocks each, as well as an extra Resnet Block near the end.

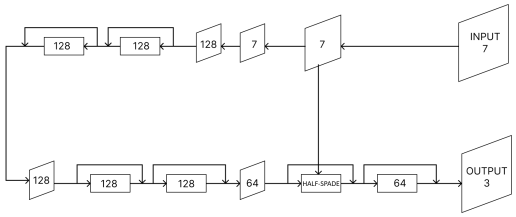

Fig.: High Quality Network Architecture

We used multiple PatchNets with spectral normalisation working at different input sizes as the discriminators, averaging their results while training our GAN. Training was done for 100 epochs each. Within the scope of this paper, we trained the best version of each network among different parameter counts based on different channel depths. The final model was run in Unity, using Barracuda, on an Nvidia GTX 1650 laptop GPU.

Results

The results produced by our best Fast Network, which consisted of 1.8 million parameters, yielded real-time speeds of 10 to 12 frames per second. There was no fast flicker, only gradual morphs in the grass shapes when moving around in the grassy landscape.

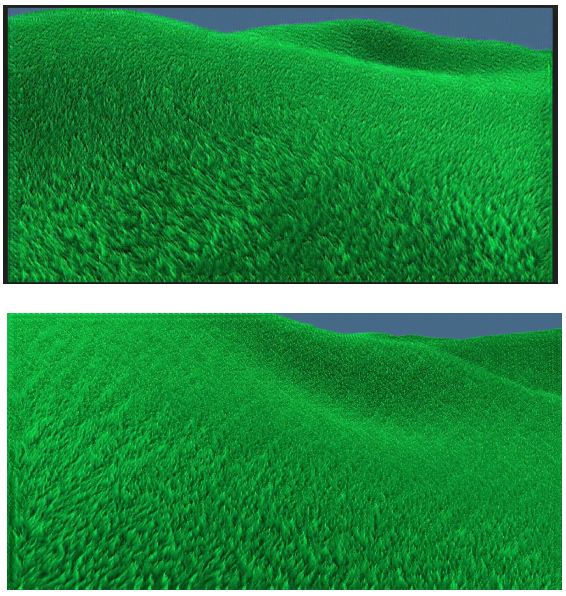

Fig: Images produced by our Fast Generator

Our fastest High Quality Network model produced better looking grass than the best Fast Network results in terms of distinction of grass blades. However, these blades appeared much more directionally biassed and repetitive. It ran at around 8 frames per second.

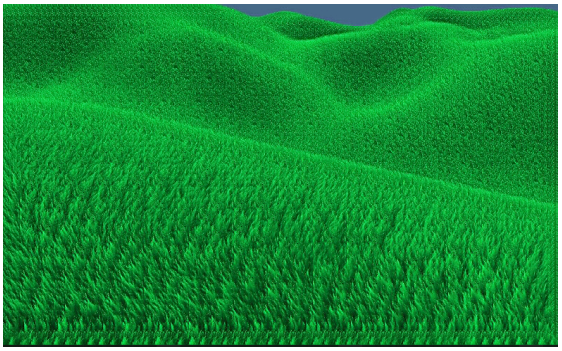

Fig: Result produced by our fastest acceptable High Quality Network

The results produced by our best High Quality Network, which consisted of over 10.5 million parameters, yielded real-time speeds of 3 to 5 frames per second. The grass produced had less morph and is far more visually similar to the example procedural grass, having much fewer artefacts and much more distinctly visible grass blades.

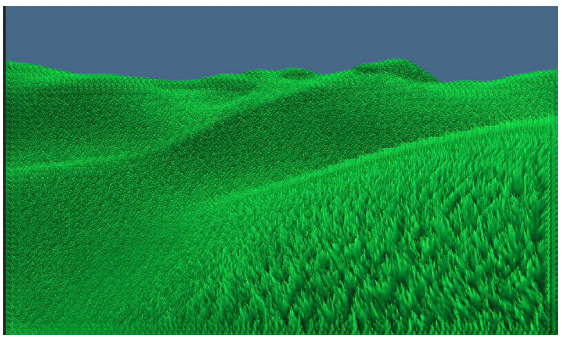

Fig: Result produced by our best High Quality Network

In all cases, Artefacts such as grid-patterns, border patterns and general repetitiveness was observed due to the usage of a smaller number of layers.

Conclusion

We successfully used Generative Adversarial Networks (GANs) to train a generator network to produce acceptable looking grass in the view-space of a video game in real-time at around 10 frames per second. This performance is not good enough to replace world-space procedural grass generation methods, however it successfully performs selective removal of the redundant information for such grass such as consistency of existence of individual grass blades. With further room for development, this could become a viable technology to perform such selective compression tasks for real-time generation of video game elements.

Future Work

There is potential for real time generation of grass using diffusion, using training techniques such as one presented in [Refer to Adversarial Diffusion Distillation — Stability AI]. We expect to reduce our repetitiveness issues and pattern artefacts using Diffusion models, and real-time generation using them seems to be on the horizon.

References

[1] Tiwari, A., & Fried, O. (n.d.). State of the art on Neural Rendering. Retrieved December 26, 2022, from https://onlinelibrary.wiley.com/doi/am-pdf/10.1111/cgf.14022

[2] Wulff-Jensen, A., Rant, N.N., Møller, T.N., Billeskov, J.A. (2018). Deep Convolutional Generative Adversarial Network for Procedural 3D Landscape Generation Based on DEM. In A. Brooks, E. Brooks, & N. Vidakis (Eds.), Interactivity, Game Creation, Design, Learning, and Innovation. ArtsIT DLI 2017 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 229 (pp. 95-105). Springer. https://doi.org/10.1007/978-3-319-76908-0_9

[3] Stable Diffusion 2-1 - a Hugging Face Space by stabilityai. (n.d.). Retrieved December 26, 2022, from https://huggingface.co/spaces/stabilityai/stable-diffusion

[4] 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 (pp. 5589-5597). IEEE. https://doi.org/10.1109/cvpr.2018.00917

[5] Cignoni, P., Scopigno, R., & Tarini, M. (2005). A simple normal enhancement technique for interactive non-photorealistic renderings. Computers & Graphics, 29(1), 125–133. https://doi.org/10.1016/j.cag.2004.11.012

[6] Mittermueller, M., Ye, Z., & Hlavacs, H. (2022). Est-Gan: Enhancing style transfer gans with intermediate game render passes. In 2022 IEEE Conference on Games (CoG) (pp. 1-8). IEEE. https://doi.org/10.1109/cog51982.2022.9893673

[7]Park, T., Liu, M. Y., Wang, T. C., & Zhu, J. Y. (2019). Semantic Image Synthesis with Spatially-Adaptive Normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2337-2346).