Journal of High School Science

A comparative analysis of generative models for terrain generation in open-world video games

Ishaan S

Submitted: October 16, 2023, Revised: version 1, November 30, 2023, version 2, January 21, 2024
Accepted: February 1, 2024

Abstract
Terrain generation in open-world video games plays a crucial role in creating immersive and visually appealing virtual environments. This paper presents a comparative analysis of generative models for terrain generation in open-world video games, encompassing Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Convolutional Neural Networks (CNNs), and Procedural Content Generation (PCG). The objective is to assess the effectiveness and suitability of these approaches in generating diverse and realistic terrains that enhance the user's experience. The study begins by explaining techniques of terrain generation that are commonly used in video games. It shows the disadvantages of these methods by explaining how these techniques of terrain generation in open-world video games can result in repetitive patterns, excessive time dedication, and dull landscapes. In contrast, generative models can create diverse and realistic terrain while consuming significantly fewer resources. To compare the different methods of terrain generation, the generated terrains are evaluated using quantitative and qualitative metrics through a meta-analysis of various research papers. The results of the evaluation reveal the strengths and limitations of each approach, providing valuable information for developers and researchers in selecting the most suitable method for various terrain generation scenarios in open-world video games.

Keywords
Terrain generation, Procedural generation, General adversarial networks, Convolutional neural networks, Variational autoencoders, Generative AI

Ishaan Srivastava, Woodinville High School, 23021 27th DR SE, Bothell, WA 98021, USA. ishaan0316@gmail.com

1. Introduction

The gaming community has experienced rapid growth, driven by advances in hardware and the increasing demand of players. In order to meet the demands of their target audience, developers now face the challenge of creating higher-quality games with greater complexity and realism. Unfortunately, despite recent advancements in gaming technology and hardware, the creation of terrain for open-world video games remains a manual process, consuming substantial time and money (1). This manual content creation process has led to financial constraints and risk avoidance, limiting creativity and resulting in emulation and stagnation within the industry (1).

Existing manual tools like Unity rely on the skills and experience of the designer, sometimes making it challenging to achieve the desired realism and immersion (2). Designing intricate and lifelike terrains requires significant expertise and time-consuming effort, often resulting in a bottleneck during the game development process. Moreover, the manual approach may lead to inconsistencies in the terrain design, as different designers might implement their creative ideas differently (3).

The limitations of manual content creation not only affect the overall quality of the game but also hinder the innovation and exploration of new and imaginative landscapes. Game developers are restricted by the time and resources required to create every detail of the terrain, leaving little room for experimentation and artistic expression. As a result, some developers may resort to reusing elements or adopting similar patterns across different areas of terrain, leading to repetitive and predictable environments that can reduce the players' sense of discovery.

Generative models offer a potential solution to this bottleneck. They have been used in many successful games such as Minecraft (4) and No Man's Sky (5), which allows these games to utilize vast worlds with an increased sense of replayability (6). Generative models enhance replayability by infusing each gaming session with unpredictability and uniqueness through the implementation of randomness and the creation of new terrain every time the game is played. Furthermore, generative models can save memory by only storing rendered regions on the RAM, helping mitigate the amount of hardware needed while still meeting user expectations for quality and diverse gaming experiences (7).

However, there are many limitations with current implementations of generative models, such as the massive amount of computer processing power required and the presence of repeating patterns, as shown in Figure 1. Therefore, there is a high demand for improved generative models to effectively surpass the limitations of traditional methods for developing terrain in open-word video games.

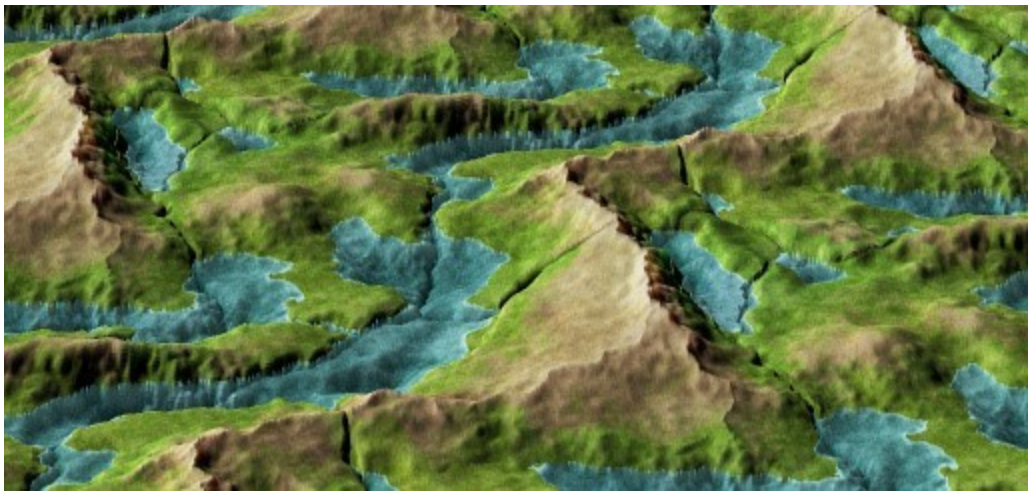

Figure 1. A Textured Region of Terrain. (27)

Given the rising demand for high-quality games, it is crucial to determine the most effective generative model for different terrain generation scenarios. This meta-analysis aims to identify the best generative model suited for various situations of terrain generation in open-world video games. By understanding and comparing the strengths and weaknesses of the different generative models, game developers can enhance creativity, reduce financial constraints, and deliver more immersive gaming experiences to their players.

2. Background

In this section, the essential context of each generative model is provided to compare the generative models. A deep understanding of each model is needed to compare the strengths and weaknesses of each model in order to evaluate the best one for terrain generation in open-world video games.

2.1 Procedural Generation

Procedural generation has emerged as the most popular and versatile generative model for terrain generation in video games (8). Its widespread adoption is evident in numerous successful titles, such as Minecraft and No Man's Sky, where Procedural Content Generation (PCG) has played a pivotal role in shaping immersive and expansive game worlds, as seen in Figure 2.

In the case of Minecraft, PCG is employed through a seed, which acts as a random number initializing the terrain generation process. This seed ensures that each player's world is unique and provides endless possibilities for exploration and creativity. Seeds can be chosen through fixed values, player input, time-based methods, or random generation to either provide consistent or unique gameplay experiences. To create diverse and natural-looking landscapes, Minecraft applies various noise algorithms, including perlin noise, simplex noise, and voronoi noise.

These algorithms generate visually appealing patterns, producing biomes and structures with remarkable realism.

Figure 2a. Terrain Generation in Minecraft using PCG (28).

Figure 2b. Terrain Generation in No Man's Sky using PCG (29).

Meanwhile, in No Man's Sky, the implementation of PCG takes the concept to an entirely new level. Here, PCG is used to generate diverse planetary systems and ecosystems. The sheer scale and variety of the universe in No Man's Sky demonstrate the vast potential of procedural generation in shaping gaming experiences.

One of the key advantages of Procedural generation over other generative models is that it does not require training like traditional artificial intelligence (AI) algorithms. AI-based generative models often rely on extensive training using large datasets to learn patterns and generate content accordingly. This training process

can be time-consuming and resource-demanding.

In contrast, PCG operates using algorithms and mathematical rules, which means it does not rely on learning from data. The terrain generation process in PCG is based on predefined rules and noise functions. As a result, PCG can produce diverse and visually appealing terrains in real time without the need for extensive pre-processing or data training.

The absence of training in PCG provides game developers with greater flexibility and efficiency in the game development process. They can immediately generate terrain and content as needed without having to wait for AI models to be trained. This allows for rapid testing, thus allowing developers to experiment with their game designs more effectively. Additionally, PCG enables games to be more resource-efficient since it doesn't require large trained models or massive datasets. This aspect is particularly beneficial for games on platforms with limited resources, such as mobile devices or older hardware.

Another compelling advantage of PCG is its ability to produce diverse terrain variations with each game session or when rendering new regions. Procedural generation achieves this by using a multitude of noise algorithms which are shown in Figure 3. These noise algorithms introduce random and natural-looking patterns into the terrain generation process, ensuring that players encounter fresh landscapes and unexpected obstacles. The algorithms' clever use of linear and bilinear interpolation plays a pivotal role in calculating the distances between data points, enhancing the seamless integration of diverse terrain features.

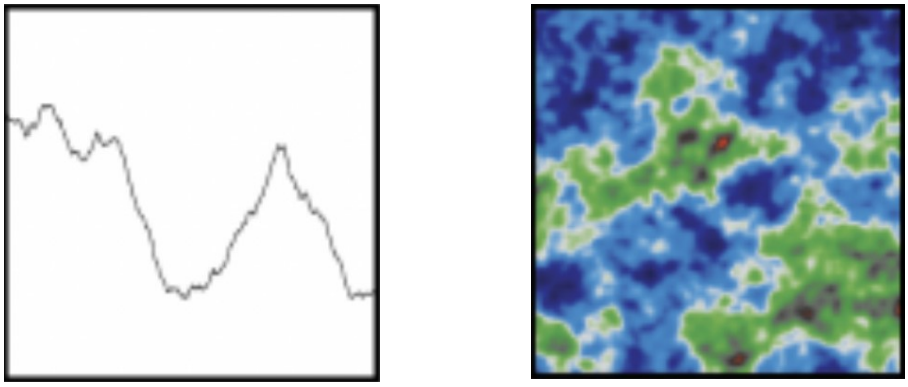

Figure 3. Types of Noise Algorithms. Left: Midpoint Displacement, Right: Diamond Square(6)

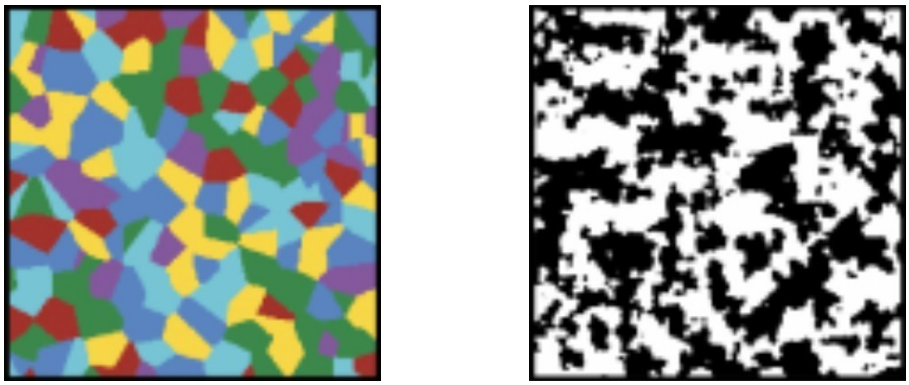

Figure 3. Types of Noise Algorithms. Left:Voronoi/Worley Noise, Right: Perlin Noise(6)

2.1.1 *Midpoint Displacement and Diamond Square Algorithms*

Midpoint displacement is a fractal algorithm that initiates with random values at two endpoints, computes their average at the midpoint, introduces random noise, and iteratively repeats this process for the midpoints between the endpoints and their midpoint (9). Building upon this concept, the diamond-square algorithm refines the technique by employing squares instead of line segments, resulting in terrain with smoother and less jagged features (1). Moreover, the diamond-square algorithm adopts bilinear interpolation as opposed to random displacements, effectively smoothing out the terrain height values and producing visually appealing landscapes (2).

2.1.2 *Voronoi/Worley Noise*

Voronoi/Worley noise is an algorithm that operates by distributing random seeds across a grid (9). Every point in the space belongs to the cell corresponding to the nearest seed which creates an intricate pattern of cells, each representing the influence of its closest seed (9). The value or color associated with each cell can be determined by various rules or functions, leading to diverse and visually compelling results.

2.1.3 *Perlin and Simplex Noise*

Although perlin noise is complicated and hard to implement, it is a widely used algorithm in procedural generation due to its ability to produce natural-looking patterns (9). This makes it the most popular choice for generating various elements in procedural content creation. One of its significant advantages is its capacity to be layered multiple times, allowing for the creation of complex and detailed structures (1). Moreover, Perlin noise is well-suited for creating fractal Brownian Motion, adding depth and complexity to the generated content (10). The algorithm's effectiveness lies in its utilization of a gradient vector with multiple dimensions, which allows for the generation of randomness while maintaining a smooth and coherent overall output (9). This not only results in visually pleasing patterns but also ensures fast computation and optimized memory usage (11). Expanding

on this idea, simplex noise uses simplexes, which are like triangles, instead of a square grid for the gradient vector, making it extremely difficult to implement (9). The use of simplexes in Simplex noise contributes to its speed and efficiency, particularly in higher dimensions. By operating on these simplexes, the algorithm reduces the number of necessary calculations and linear interpolations, resulting in faster computation times compared to Perlin noise (11). Moreover, Simplex noise's efficiency does not come at the expense of image quality. In fact, it is renowned for producing high-quality images and smooth, natural-looking patterns.

2.2 General Adversarial Networks

Generative Adversarial Networks (GANs) have revolutionized various fields, including image synthesis and data augmentation. Their widespread adoption is evident in numerous applications such as image generation. One of the most remarkable uses of GANs has been in video games, where they are used to enhance visual realism and expand gameplay possibilities.

In the gaming industry, GANs have been harnessed for texture synthesis, enabling the creation of high-quality, detailed textures for game environments. By training the generator to produce textures similar to those found in real-world images, GANs can generate diverse and visually captivating game assets, significantly reducing the need for manual texture creation.

Additionally, GANs have enabled the creation of virtual worlds by training them on real-world environmental data, such as satellite imagery and elevation maps, to generate landscapes with astounding accuracy (6). This not only saves development time but also offers players endless exploration opportunities in vast open-world terrain.

GANs excel in generating highly realistic and diverse data, making them particularly effective in capturing intricate patterns and maintaining the precision of the original data distribution. Unlike traditional generative models that rely on fixed functions, GANs learn the underlying data distribution through a dynamic adversarial process between the generator and discriminator, as shown in Figure 4 (2).

The generator plays a crucial role in producing data that closely resembles real data from a given dataset. Its objective is to learn the probability distribution of the training data and generate new samples that are virtually indistinguishable from real data when presented to the discriminator (2). Essentially, the generator learns to capture the most common features and patterns present in the dataset, enabling it to create realistic and diverse samples (12). On the other hand, the discriminator's primary task is to classify images as either real (belonging to the dataset) or fake (generated by the generator) (12).

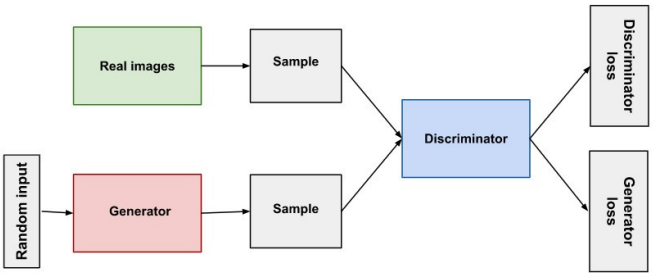

Figure 4. An Overview of the Structure of GANs. (11)

Figure 5 is an example of a function that is commonly referred to as the BCE cost function or the Gradient Descent equation in the context of GANs. It is an optimization algorithm used to minimize the loss function during the training process of GANs. The goal is to find a set of model parameters that minimize the loss function and utilize the generator and discriminator to their fullest efficiency.

$$J(\theta) = \frac{-1}{m}\sum_{i=1}^{m}\left[y^{(i)}log(h)(x^{(i)},\theta) + (1-y^{(i)})log(1-h(x^{(i)},\theta))\right]$$

Average loss of the whole batch; Prediction; Parameter; Label; Features

Figure 5. BCE Cost Function / Gradient Descent Equation.

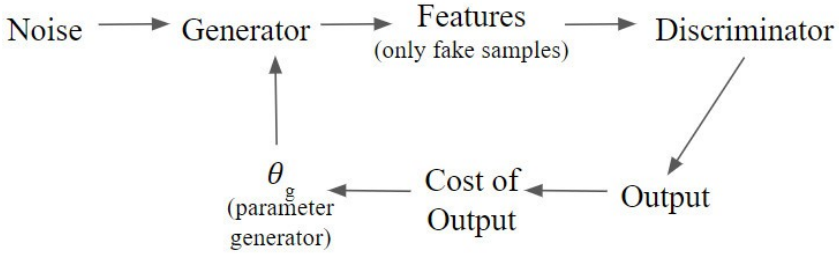

Figure 6. The Equation for Training the Generator.

When training the generator, it receives feedback based on the discriminator's ability to distinguish between the dataset and the generated data, as shown in Figure 6 (12). The generator aims to produce samples that can convince the discriminator

to classify them as real data. As the generator improves, it becomes more skilled at generating data that closely resembles the real dataset, making it increasingly challenging for the discriminator to differentiate between the two.

As shown in Figure 7, when training the discriminator, it receives feedback based on the accuracy of its predictions regarding whether an image was generated by the generator or if it belongs to the original dataset (12). The discriminator improves by correctly classifying the source of the input data. As training progresses, it becomes more skilled at distinguishing real from generated data, pushing the generator to generate even more realistic samples.

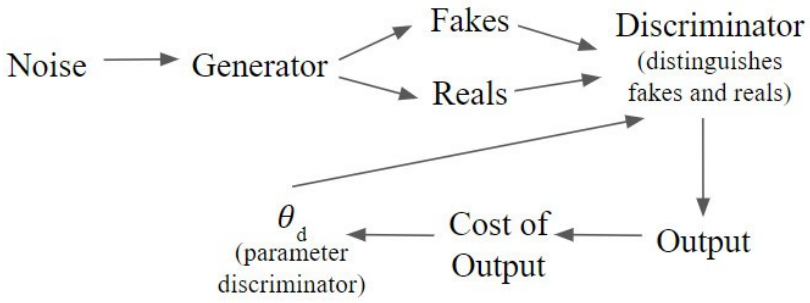

Figure 7. The Equation for Training the Discriminator.

This process drives both the generator and discriminator to improve their skills, ultimately leading to the generation of high-quality data that closely matches the real data. To maintain a balance between the generator and discriminator, it is essential to prevent either from becoming significantly stronger than the other (12). This ensures that the GAN achieves the desired results, where the generator produces high-quality and diverse data, and the discriminator accurately differentiates between real and generated samples.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a powerful deep learning algorithm for image generation. One of the key strengths of CNNs lies in their ability to learn spatial features from input data. In the context of terrain generation, this means that CNNs can automatically discover patterns, textures, and other relevant features present in real-world terrains. This feature learning capability enables them to generate landscapes that closely resemble natural environments, adding realism and authenticity to virtual worlds.

Another advantage of CNNs is their data efficiency. Unlike other generative models, CNNs can be trained on relatively small datasets, which is valuable in terrain generation where acquiring vast amounts of data may be challenging or impractical (13). This ease of preprocessing allows game developers to create captivating terrains even with limited training samples.

Furthermore, CNNs demonstrate excellent generalization capabilities. Once trained, they can effectively generate new terrain samples based on an initial seed or random noise input. The generated terrains share similarities with the training data, but they also possess unique characteristics such as feature hierarchies, filters, and pooling layers, resulting in diverse and realistic landscapes for different game worlds.

Figure 8 shows that CNNs are composed of three main layers: the convolutional layer, the pooling layer, and the fully connected layer. This plays a crucial role in image processing by reducing images into a format that is easier to analyze without losing essential features (1).

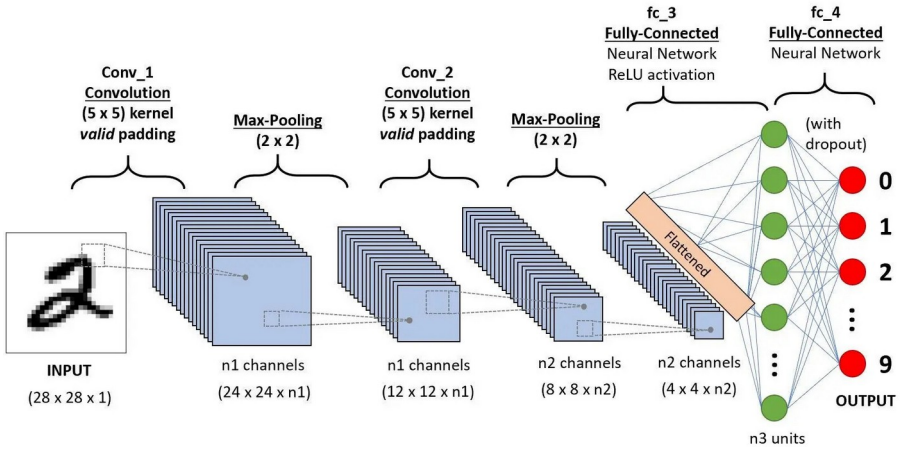

Figure 8. An Overview of the Structure of CNNs. (12)

In the convolutional layer, which is shown in Figure 9a, kernels are utilized to extract high-level features while capturing low-level features (13). These kernels are small filters that slide over the input image, detecting different patterns and textures. By learning from the data, CNNs can automatically identify meaningful visual features, enhancing their ability to recognize objects, edges, and textures within the image.

Padding is another important aspect of the convolutional layer. Valid padding decreases the dimensions of the feature maps, while same padding maintains or increases the dimensionality (13). Padding is used to control the spatial size of the feature maps and can affect the amount of information available to other layers in the network.

Next, the pooling layer, which is shown in Figure 9b, is encountered. The pooling layer reduces the spatial size of the feature maps obtained from the convolutional layer (6). This serves two main purposes: first, it decreases the computational power required

for further processing by reducing the dimensionality of the data (13). Second, pooling extracts the most important features, focusing on the most prominent information while discarding less relevant details.

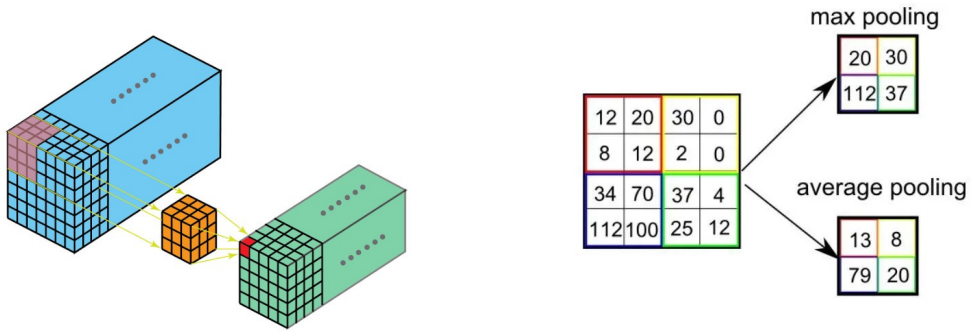

Figure 9. The Convolutional Layer and the Pooling Layer. Left: Kernels Used in the Convolutional Layer, Right: Max Pooling and Average Pooling. (12)

The two types of pooling are max pooling and average pooling. Max pooling takes the maximum value within each pooling window, while average pooling calculates the average value (13). Average pooling is commonly used when you want to preserve information from all regions of an image, but max pooling is often preferred because it acts as a noise suppressant, capturing the most significant information and disregarding less relevant elements (13).

Finally, after all the layers of the CNN process the image, the model gains a comprehensive understanding of the features present. The fully connected layer combines all the high-level features learned throughout the network, preparing the model for making predictions or classifications based on the extracted information (13).

2.4 Variational Autoencoders

Variational Autoencoders (VAEs) have found intriguing applications beyond the traditional domains of image and data generation. When applied to terrain generation, VAEs offer a powerful approach to creating realistic and diverse landscapes for video games, simulations, and other interactive experiences.

In the context of terrain generation, VAEs can capture the underlying patterns and characteristics of real-world terrains. The process typically involves training a VAE on a dataset of existing terrains, such as elevation maps or topographical data. By learning from these samples, the VAE can generate new terrains that exhibit similar features, textures, and structures.

The key advantage of using VAEs for terrain generation lies in their ability to create continuous and meaningful latent representations of terrains. The latent space in a VAE captures essential features of input terrains like elevation data, allowing for smooth interpolation between different terrains in the latent space. This means that

by navigating the latent space, it can generate terrains that smoothly transition from one type to another, creating a variety of landscapes.

Figure 10 shows that VAEs consist of two neural networks: the encoder and the decoder. These components work in conjunction to achieve dimensionality reduction, a crucial process that decreases the complexity of data descriptions. However, this compression can sometimes lead to loss of information during the encoding phase, creating a challenge in the decoding process (14). The primary objective of dimensionality reduction is to identify the optimal encoder and decoder pair within a given set of possibilities. Essentially, it seeks the pair that retains the maximum information during encoding, resulting in minimal reconstruction errors during decoding (14).

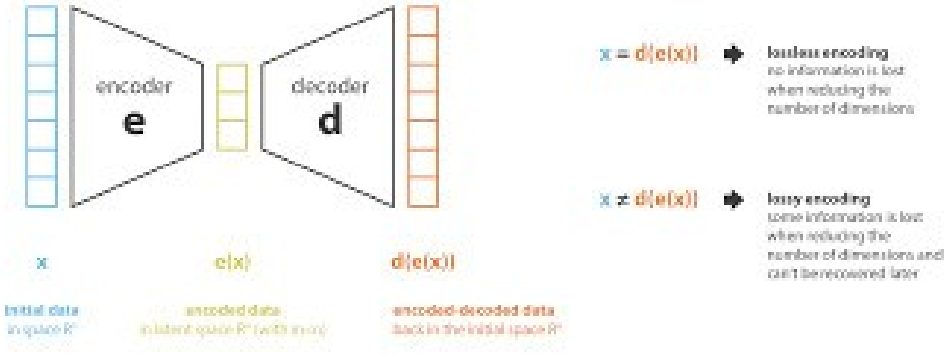

Figure 10. An Overview of the Structure of VAEs. (14)

VAEs progressively refine the optimal pair of the decoder and encoder through an iterative optimization process called gradient descent, an effective algorithm for finding the local minimum in differential functions in order to optimize the model and avoid overfitting (14). By creating a bottleneck for data, VAEs ensure that only the essential structural information passes through for reconstruction.

In the VAE's loss function, a lower value indicates a superior ability to model the dataset. This is similar to Principal Components Analysis (PCA), where the goal is to identify the optimal linear subspace within the initial feature space, minimizing the errors when approximating data (14).

As shown in Figure 11, the latent space, an integral part of VAEs, represents compressed data, aligning with the best linear subspace for data projection, thereby reducing dimensions (14). VAEs feature regularization during training to prevent overfitting, which can lead to the presence of meaningless points in the latent space (14). This regularization ensures that the latent space possesses favorable properties that support effective generative processes.

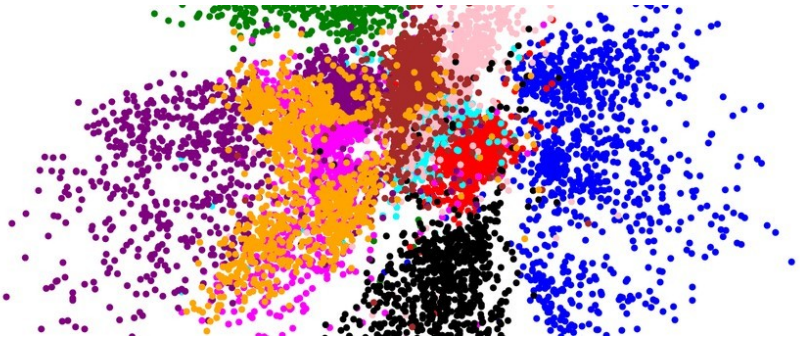

Figure 11. A Visualization of Latent Space. (22)

A critical regularization term in VAEs is the Kullback-Leibler (KL) divergence, which aids in regulating the structure of the latent space. This regularization promotes continuity (close points in the latent space should yield similar outputs when decoded) and completeness (points in the latent space should yield meaningful content upon decoding) (14). In contrast, the reconstruction loss serves as a metric that evaluates the proximity of the decoder's output to the initial input data. Together, these aspects enhance the latent space's capacity for meaningful representation and generative capabilities.

3 Meta-Analysis to Compare the Generative Models

This section will discuss the comprehensive comparison of the generative models, clarify the rationale behind this comparative meta-analysis, detail the data collection methodology, as well as explain the quantitative metrics employed in the evaluation. Furthermore, the outcomes resulting from this comparison will be thoroughly presented and analyzed.

3.1 Methodology and Data Collection

This section presents the methodology employed for comparing generative models for terrain generation in open-world video games and the data collection process supporting the findings. The objective is a comprehensive evaluation of the various generative models in application to terrain generation, considering their distinct strengths and weaknesses while including a broad range of research.

The examination of the generative models focuses on a detailed assessment of their unique attributes. This involves a comprehensive review of each model's architecture, key features, and capabilities. The core principles that enable these models to generate data are analyzed, with a focus on their effectiveness in capturing essential terrain patterns and producing realistic and diverse terrain.

Additionally, a rigorous meta-analysis is conducted, drawing information from a diverse collection of research papers that have explored generative models for terrain generation. By identifying trends and utilizing tables from various research papers, a broader perspective on the

generative models' capabilities in terrain generation is gained, which allows for a reputable ranking of the generative models for terrain generation.

3.2 Qualitative Analysis

3.2.1 *Strengths of each generative model*

PCG: PCG stands out by eliminating the need for training, relying instead on mathematical functions. This accelerates terrain generation significantly, providing a cost-effective alternative compared to other generative AI models. Additionally, PCG's ability to create new terrain in real-time as regions are rendered contributes to memory and resource savings.

GAN: GANs excel in producing realistic images, surpassing other generative models, thanks to their unique adversarial process. This process plays a crucial role in creating lifelike terrain, enhancing overall realism and immersion, as seen in Figure 12a.

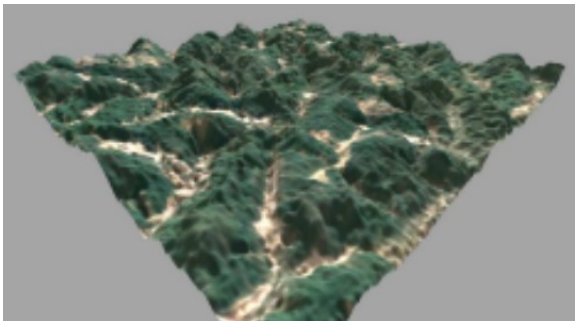

Figure 12a. Terrain Generated with SGAN

CNN: CNNs exhibit the strength of effective training on small datasets while demonstrating great generalization capabilities. Their capacity to learn patterns from limited data sets contributes to their versatility in various terrain generation scenarios.

VAE: VAEs are particularly adept at detecting and replicating repetitive structures, making them valuable for generating terrain that contains patterns and nuances. They accomplish this by capturing image data by utilizing their latent space.

3.2.2 *Weaknesses of each generative model*

PCG: A disadvantage of PCG is that their reliance on mathematical functions and algorithms might occasionally lead to terrain that appears more structured and less natural compared to the results obtained from AI models that can learn from extensive datasets. This could potentially limit the level of realism and intricacy that PCG-generated terrains can achieve in certain cases.

GAN: GANs are prone to a phenomenon known as "mode collapse," where the discriminator gains dominance over the generator (6). This occurs when the generator network primarily produces a limited set of similar samples, leading to a lack of diversity in the generated data. In the context of terrain generation, this can lead to unfavorable outcomes, undermining

the quality and diversity of the generated landscapes (15). Additionally, GANs can be challenging to train and require careful parameter tuning to achieve the optimal balance between the generator and discriminator, which can be time-consuming and resource-intensive.

CNN: One disadvantage of CNNs is their high susceptibility to overfitting, where the model learns to perform exceptionally well on the training data but struggles to generalize to new, unseen data, resulting in low-quality terrain generation. Furthermore, CNNs are primarily designed for image-related tasks, and while they can capture 3D features and patterns well, they might not inherently understand the geological or topographical characteristics that influence terrain composition, as seen in Figure 12b.

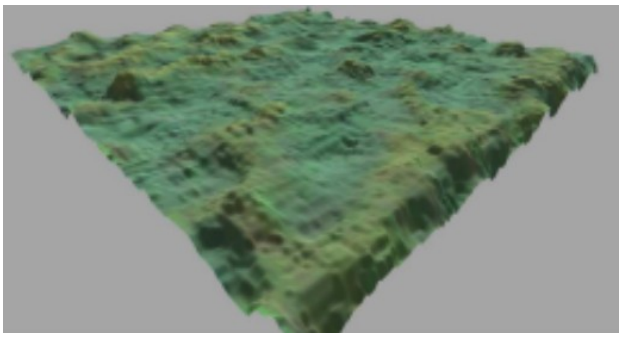

Figure 12b. Terrain Generated with CNN

VAE: One significant limitation of VAEs is that they can sometimes produce blurry or low-quality outputs (16). This is due to the trade-off between ensuring data precision and generating new samples from the learned latent space. VAEs aim to approximate the true data distribution, but this can result in blurriness as the generated samples try to cover a wider range of possibilities, leading to a loss of sharpness and detail in the generated terrains as seen in Figure 12c.

Figure 12c. Terrain Generated with PCG in a modded version of minecraft (30).

3.3 Quantitative Analysis

This section presents an in-depth exploration of the quantitative metrics that form the comparison among the trainable generative models: GANs, CNNs, and VAEs. These metrics serve as valuable tools for assessing the performance and effectiveness of these models in the context of terrain generation. Each metric shows a distinct aspect of the generative models, collectively providing a comprehensive view of how well these models capture and replicate real-world landscapes. While these metrics are individually powerful, their true strength emerges when they are considered as a whole. The comparative analysis of SSIM, MSE, L-MSE, R-MSE, M-MSE, Entropy, FID, and NLL creates a multidimensional evaluation that assesses the models' capabilities from various angles. This allows for the identification of trends and nuances that may have otherwise remained hidden.

Table 1: Comparison of GAN, CNN, and VAE

	SSIM (2)	MSE (2)	L-MSE (17)	R-MSE (17)	M-MSE (17)	Entropy (18)	FID (19)	Accuracy (20)
GAN	0.5870	5876	0.4300	0.02840	0.03350	0.000	31.40	71.87%
CNN	0.3100	11950						64.02%
VAE			0.03000	0.007300	0.01030	2.266	45.84	

SSIM (Structural Similarity): used to evaluate the amount of similarity in structure between images (2). Higher values are indicative of better similarity.

MSE (Mean Squared Error): quantifies pixel-level errors in capturing image details (2). Lower MSE values indicate more accurate predictions.

L-MSE (Latent Mean Squared Error): computes the mean squared error between target and model-computed vectors, serving as the model's loss (17). Lower values reflect better model performance.

R-MSE (Reconstruction Error): measures the mean squared error between an image from the dataset and its reconstruction via the generative model (17). Lower values signify improved image reconstruction.

M-MSE (Mapped Error): represents the mean squared error between images from the dataset and images reconstructed by the generative model following linear mapping (17). Lower values represent a more accurate reconstruction.

Entropy: evaluates image diversity and fidelity, offering insights into the balance between variety and image quality (18). Its interpretation varies based on the level of consistency desired in generated terrain, but lower entropy values are usually preferred.

FID (Frechet Inception Distance): a widely used metric for assessing generative model performance. By quantifying the vector

distance between real and generated data, it assesses image quality (19). Lower FID scores correlate with higher-quality generated images.

NLL (Negative-Log Likelihood): serves as a common loss function in generative models (21). Reduced NLL values indicate that the generated data closely matches the real data, demonstrating better model performance.

Table I shows distinct patterns: GANs showcase heightened consistency and precision in predictions, evident in their elevated SSIM value and minimized MSE value. Moreover, GANs prove superior in dataset replication compared to CNNs, supported by their higher accuracy percentage. Meanwhile, VAEs stand out for their reduced errors, as shown by their lower L-MSE, R-MSE, and M-MSE values in comparison to GANs.

3.4 Ranking

This section ranks various aspects of the generative models using their respective strengths and weaknesses and a comparison of their quantitative metrics.

Table 2: Ranking each Generative Model in Different Subcategories

	PCG	GAN	CNN	VAE
Quality	4	1	3	2
Training Difficulty and Resource Consumption	1	4	2	3
Errors	1	4	3	2
Diversity/Patterns	4	1	2	2

Quality: 1. Best model at capturing complex terrain patterns and details, 2. May struggle with highly detailed terrain, 3. Does not capture the same level of realism as GANs and VAEs, 4. Does not match the quality of the learning models.

Training difficulty and resource consumption: 1. Due to the mathematical rules, it does not require training, generates terrain quickly and requires few resources, 2. Less complex than VAEs and GANs but complex cCNNs may require high resource demands, 3. Need to balance reconstruction and latent space, requires significant processing power and memory, 4. Due to the adversarial process, training is complex and has high resource demands.

Errors: 1. Uses mathematical rules that mitigate errors that are common in other models, 2. Errors occur if the latent space is not well structured, but it is not very prone to overfitting and mode collapse, 3. Overfitting is more common in CNNs than in other models which can cause errors when generating complex terrain, 4. Mode collapse is frequent and can cause errors in terms of terrain diversity.

Diversity/Patterns: 1. The adversarial training process favors intricate and varied terrain patterns, 2. May struggle to generate complex terrain patterns compared to GANs but better than CNNs at capturing nuanced features, 3. Struggles to generate complex patterns at a large scale, 4. Limited by mathematical rules, causing it to struggle to capture varies/complex terrain.

3.4.1 *Analysis of the Rankings*

PCG: PCG is often an efficient choice for terrain generation due to its low resource consumption and infrequent occurrence of errors, as it relies on mathematical functions and rules rather than training on

data. Furthermore, it offers the advantage of immediate terrain creation without resource-intensive training processes. While it might lack the intricate detail and realism that deep learning models can achieve, its speed, low resource consumption, and ability to generate terrain with hardly any errors make it a top choice for large-scale open-world environments.

GAN: GANs excel in capturing complex patterns and generating realistic data. In terrain generation, GANs have the potential to create visually stunning and diverse landscapes by learning the data distribution from existing terrain samples. However, they require careful tuning to avoid common errors like mode collapse and might demand more computational resources for training. In addition, the adversarial training process is very complex and time-consuming because it consists of two neural networks competing against each other. The challenges posed by these drawbacks of GANs render them the most complex to effectively deploy for terrain generation. Despite these challenges, GANs exhibit the greatest potential in terrain generation due to their pronounced advantages in quality and diversity of generated terrain, despite their challenging implementation.

CNN: While CNNs are primarily used for image classification tasks, they can be adapted for terrain generation. However, they might not be as suitable as the other models for this specific task because they can't learn complex features very well. CNNs can potentially generate terrains, but their strength lies more in recognizing patterns in images rather than creating landscapes. Furthermore, CNNs might require substantial computational resources for training if a large dataset is used. Additionally, CNNs face challenges associated with overfitting more often than the other models, potentially leading to inaccuracies in generating intricate terrain features.

VAE: VAEs offer a balance between GANs and PCG. They can capture latent representations of terrain data and generate new terrains, combining realism with creative exploration. VAEs often provide smoother and more continuous latent spaces, allowing for controlled generation of terrains with specific attributes. Moreover, they exhibit fewer errors compared to other generative models, attributed to their reduced susceptibility to overfitting and mode collapse. However, VAEs might suffer from blurriness in generated images and could require careful hyperparameter parameter tuning to achieve desired results. They also demand substantial resource consumption. VAEs stand as the most balanced model, securing second place in three categories and third in one category.

3.5 Optimal Utilization of Each Generative Model

Each generative model has its strengths and weaknesses, making it suitable for different terrain-generating scenarios. This assessment of their optimal utilization depends upon their rankings across various categories and a comprehensive consideration of their individual attributes.

PCG: Leveraging its exceptional speed, minimal resource demands, and absence of training requirements, PCG stands out as an ideal choice for generating terrain in real-time as regions unfold. While the terrains created through PCG might exhibit a seemingly reduced image quality when compared to certain alternative models, the game Minecraft sidesteps the demand for high-resolution visuals. By embracing a distinctive visual style that employs block-based graphics, Minecraft harnesses PCG's swift terrain generation capabilities without the need for intricate or highly detailed images. In this context, the game prioritizes the creation of varied landscapes, encompassing diverse biomes and structures, which aligns with the strengths of PCG. This serves as a clear illustration of how PCG, despite yielding visuals of potentially lower quality, can be optimally employed in scenarios where a game's visual appeal depends on its gameplay mechanics and artistic approach rather than on executing realistic graphics.

GAN: Effectively harnessing the strengths of GANs in terrain generation involves a strategic approach that capitalizes on their exceptional quality and diversity capabilities. GANs are best suited for scenarios where quality and realism are of high importance. In applications such as geographical modeling, where authenticity is crucial, GANs can be employed to generate landscapes that mirror real-world details. To optimize their usage, developers can utilize a diverse training dataset that encompasses a wide array of terrain features, enabling GANs to capture the intricate patterns present in the natural environment. Furthermore, to mitigate challenges such as mode collapse and resource-intensive training, developers can explore techniques such as progressive growth of GANs (ProGANs) or other specialized variants that address these limitations. By embracing these strategies, game developers can leverage GANs' unique ability to produce visually compelling and authentic terrains, ultimately enhancing the immersive experience for users navigating virtual environments.

CNN: CNNs excel in capturing intricate patterns and are particularly well-suited for scenarios where detailed patterns play a crucial role. Leveraging their ability to learn hierarchical features from data, CNNs can effectively model and generate terrain with complex patterns. Their strength in generalizing from relatively small datasets makes them valuable in situations where training data may be limited. However, due to the substantial computational resources demanded by CNNs, especially when dealing with larger datasets, and their high susceptibility to overfitting, utilizing CNNs effectively involves carefully managing their training process, employing techniques to mitigate overfitting, and capitalizing on their pattern capture capabilities to create visually engaging landscapes.

VAE: Effectively utilizing VAEs requires recognizing their adaptability and employing them in scenarios where a well-rounded performance is preferred over extreme specialization. Their balanced nature, with acceptable performance across

various aspects of terrain generation, makes them a reliable choice for generating terrain. Their capability to capture latent features and maintain a respectable level of performance in different metrics enables them to address a wide range of terrain generation needs. By strategically utilizing VAEs' balanced attributes, developers can efficiently generate terrains that contain stability between different aspects of terrain generation, ultimately enhancing the overall gaming experience.

Leveraging the strengths and weaknesses of each generative model could allow developers to strategically employ them in areas where they excel, optimizing the terrain generation process. For example, PCG could be used to generate uncomplicated terrain sections due to its ease of implementation, even though it may yield lower-quality output. In contrast, GANs could be utilized for the most intricate sections of terrain, because they can produce high-quality and realistic terrain. Meanwhile, VAEs and CNNs should be employed where a balance between quality and implementation ease is desired.

However, this hybrid approach also comes with its own set of advantages and disadvantages. On the positive side, it allows developers to harness the unique strengths of each model, optimizing the overall efficiency and quality of the generated terrain. Conversely, managing a multitude of complex AI models introduces challenges in terms of integration and complexity. Moreover, the need to train and maintain multiple models increases the computational resources required and the time investment.

4 Conclusion

This paper presented a comprehensive ranking of generative models achieved through qualitative and quantitative analysis. The intention behind these results is to offer valuable guidance to future developers when selecting the most suitable model for generating terrain in open-world video games. Moreover, the insights from this research can extend beyond terrain generation, benefiting broader applications in general image generation. This approach seeks to optimize model utilization, streamline development processes, and elevate the quality of video games. Ultimately, this study has the potential to revolutionize the way designers conceptualize and craft virtual environments. While generative models may not fully replace human designers, they offer an automated means to create a vast array of diverse environments efficiently.

To ensure the accuracy and impartiality of the comparison, this paper draws upon the foundation of various research papers. It leverages this knowledge to create a comparison framework that eliminates bias. However, the paper does have limitations. Notably, there are some missing data points, including incomplete entries in Table 1. Moreover, the quantitative aspect of the comparative analysis is missing the context of PCG against other generative models. More comprehensive datasets comparing these models in terrain generation could enhance the paper's insights. Additionally, the paper has not

explored the practical application of generated terrain images in video games, particularly in their conversion into 3D environments.

Future research could expand the scope by incorporating Recurrent Neural Networks (RNNs), which specialize in handling sequential or time-series data. Embracing transformer-based, discriminative, and predictive AI models could provide deeper insights into their applications and effectiveness. Moreover, a detailed exploration of various GAN types like ProGANs, StyleGANs, and DCGANs could offer further insights. Research could also delve into additional facets of terrain generation, such as building generation or other landscape elements (22). Beyond terrain, exploring text generation and other aspects of generative AI could enrich the understanding of generative models' versatility and potential applications. By considering these potential avenues for future research, this paper contributes to the ongoing exploration of generative models' capabilities, enhancing our understanding and opening the door to improved future utilization of generative models for terrain generation in open-world video games.

References

1. Matos, R. (2020, October). 3D Terrain Generation using Neural Networks. https://repositorio.iscte-iul.pt/bitstream/10071/22222/1/master_rodrigo_tavares_almeida.pdf

2. Spick, R. J.,Walker, J. A. (2019, October). Realistic and Textured Terrain Generation using GANs. https://eprints.whiterose.ac.uk/153088/

3. Tschang, F. T. (2005). Videogames as Interactive Experiential Products and their Manner of Development. International Journal of Innovation Management, 9(1), 103–131. https://doi.org/10.1142/s1363919605001198

4. Minecraft. (2009). [Video game]. Xbox 360.

5. No Man's Sky. (2016). [Video game]. PS4.

6. Panagiotou, E., Charou, E. (2020, October). Procedural 3D Terrain Generation using Generative Adversarial Networks. https://arxiv.org/abs/2010.06411

7. Raffe, W. L., Zambetta, F., Li, X. (2011). Evolving patch-based terrains for use in video games. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (pp. 363-370). http://dx.doi.org/10.1145/2001576.2001627

8. Liu, J., Snodgrass, S., Khalifa, A., Risi, S., Yannakakis, G. N., Togelius, J. (2020). Deep Learning for Procedural Content Generation. https://par.nsf.gov/servlets/purl/10231830

9. Anon. Fundamentals of Terrain Generation (Carnegie Mellon University). https://www.cs.cmu.edu/~112/notes/student-tp-guides/Terrain.pdf

10. Wang, J. L. (2016). Simulation of Fractional Brownian Motion with Conditionalized Random Midpoint Displacement. https://project.inria.fr/fraclab/files/2016/01/Simulation-of-Fractional-Brownian-Motion-with-Conditionalized-Random-Midpoint-Displacement.pdf

11. Archer, T. Procedurally Generating Terrain (Morningside College). https://micsymposium.org/mics_2011_proceedings/mics2011_submission_30.pdf

12. Zhou, S., Zhou, E., Zelikman, E. (2020). Build Basic Generative Adversarial Networks (GANs) [Online course]. Coursera. https://www.coursera.org/learn/build-basic-generative-adversarial-networks-gans?specialization=generative-adversarial-networks-gans

13. Saha, S. (2023, June). A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way [Blog post]. https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

14. Rocca, J. (2021, March). Understanding Variational Autoencoders (VAEs) [Blog post]. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

15. Sajjadi, M. S. M., Bachem, O., Lucic, M., Bousquet, O., Gelly, S. (2018). Assessing Generative Models via Precision and Recall. In *Proceedings of the Neural Information Processing Systems Conference* (NeurIPS). https://proceedings.neurips.cc/paper/2018/file/f7696a9b362ac5a51c3dc8f098b73923-Paper.pdf

16. El-Kaddoury, M., Mahmoudi, A., Himmi, M. M. (1970). Deep Generative Models for Image Generation: A Practical Comparison Between Variational Autoencoders and Generative Adversarial Networks. In *Mobile, Secure, and Programmable Networking* (pp. 1-8). Springer. https://link.springer.com/chapter/10.1007/978-3-030-22885-9_1

17. Asperti, A., Tonelli, V. (2022). Comparing the Latent Space of Generative Models. In *Neural Computing and Applications*, 35(30), 3155-3172. https://link.springer.com/article/10.1007/s00521-022-07890-2

18. Mi, L., Shen, M., Zhang, J. (2018). A Probe Towards Understanding GAN and VAE Models. https://arxiv.org/pdf/1812.05676

19. Peters, H., Celis, N. (2022, June). An empirical comparison of generative capabilities of GAN vs VAE (Master's thesis, KTH). https://www.diva-portal.org/smash/get/diva2:1703317/FULLTEXT01.pdf

20. Ravuri, S., Vinyals, O. (2019). Classification Accuracy Score for Conditional Generative Models. In *Proceedings of the Neural Information Processing Systems Conference* (NeurIPS). https://proceedings.neurips.cc/paper/2019/file/fcf55a303b71b84d326fb1d06e332a26-Paper.pdf

21. Bond-Taylor, S., Leach, A., Long, Y., Willcocks, C. G. (2022). Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. arXiv, 2103.04922. https://arxiv.org/abs/2103.04922

22. Dahrén, M. (2021). The Usage of PCG Techniques Within Different Game Genres (Master's thesis, Malmö University). https://www.diva-portal.org/smash/get/diva2:1604550/FULLTEXT02.pdf

23. Davidson, T., Falorsi, L., De Cao, N., Kipf, T., Tomczak, J. (2018). Hyperspherical Variational Auto-Encoders (University of Amsterdam). https://www.researchgate.net/publication/324182043_Hyperspherical_Variational_Auto-Encoders

24. Gribbon, K., Jonathon, C., Bailey, D. (2003, January). A Real-time FPGA Implementation of a Barrel Distortion Correction Algorithm with Bilinear Interpolation. https://www.researchgate.net/publication/228546868_A_real-time_FPGA_implementation_of_a_barrel_distortion_correction_algorithm_with_bilinear interpolation

25. Google Developers. Overview of GAN Structure. https://developers.google.com/machine-learning/gan/gan_structure

26. Howard, D., Kannemeyer, J., Dolcetti, D., Munn, H., Robinson, N. (2022, March). Assessing Evolutionary Terrain Generation Methods for Curriculum Reinforcement Learning. https://arxiv.org/abs/2203.15172

27. Filter Forge. (2014). Terrain Generator. Terrain Generator by Filter Forge is licensed under CC by 2.0. https://www.flickr.com/photos/93421824@N06/13690180714

28. Anon. (2022). Peaks. IGN. https://www.ign.com/wikis/minecraft/Peaks

29. Anon. (2016). A Tour of 5 New Planets. IGN Southeast Asia. https://sea.ign.com/no-mans-sky/103909/video/no-mans-sky-a-tour-of-5-beautiful-planets

30. Wright, J. (2022). Minecraft 1.19 Seeds for Beautiful Landscapes. Sportskeeda. https://www.sportskeeda.com/minecraft/top-5-minecraft-1-19-seeds-beautiful-landscapes