3D Character Generation using PCGML

Balika J. Chelliah, Vijay Krishna Vallabhaneni, Saikiran Reddy Lenkala, Mithran J, M Kesava Krishna Reddy

Abstract: In video games, both the characters and the world play an important role in creating a sense of immersion to the player. Although each character can be modelled by hand to make them feel more life-like, the process becomes inherently complex when populating a game map with more than 100000+ characters. In cases like this, developers must look towards incorporating new tools to automate and accelerate their creation pipeline. We present a new method to procedurally generate the Non- Playable Characters (NPCs) in video games using a modified Style-based generative adversarial network (StyleGAN) which is a type of neural network. PCGML acronym for Procedural Content Generation using Machine learning is the most cost- effective method for game content generation, it is employed to reduce production effort and to save storage space. Our approach adapts the use of PCGML with styleGAN to generate NPCs that are unique in both appearance and behaviour. The properties or traits influence the generation of characters making the game environment diverse and interesting for the players.

Index Terms: Procedural Content Generation, Machine Learning, StyleGAN, Video Games, Characters, 3D

I. INTRODUCTION

The level of detail and complexity of 3D assets in video games has increased tremendously over the years. As a result, developers tend to depend highly on automation or procedural tools to create huge-scale virtual environments. If the developers were to create the assets manually it would take a lot of time and effort which would drive the production costs [1]. Procedural content generation (PCG) is a technique where the game content can be generated algorithmically, when applied in game development it helps in creating game worlds that are essentially infinite and detailed. With PCG, developers can now take a hands-off approach to the development process so that they can focus on the core game and functionalities. PCG via Machine Learning (PCGML) is an advanced approach to creating game content, it relies heavily on machine learning models to generate game content like levels, maps procedurally [2].

Revised Manuscript Received on April 14, 2019.

Balika J. Chelliah, Associate Professor, Undergraduate Student Department of Computer Science & Engineering, SRM Institute of Science and Technology, Chennai, India

Vijay Krishna Vallabhaneni, Undergraduate Student Department of Computer Science & Engineering, SRM Institute of Science and Technology, Chennai, India

Saikiran Reddy Lenkala, Undergraduate Student Department of Computer Science & Engineering, SRM Institute of Science and Technology, Chennai, India

Mithran J, Undergraduate Student Department of Computer Science & Engineering, SRM Institute of Science and Technology, Chennai, India

M Kesava Krishna Reddy, Undergraduate Student Department of Computer Science & Engineering, SRM Institute of Science and Technology, Chennai, India

By adapting PCGML techniques to a 3D environment, it opens up a lot of new and exciting opportunities, especially in character creation.

This paper's main focus is on the population of the game world with procedurally generated non-playable characters (NPC) that are unique and immersive. Most of the games today reuse the same assets when rendering the in-game environment with NPCs. As a result, most of the characters tend to have identical facial features and behave the same. This makes the player lose interest in the game as the game world feels static and limited. With machine learning the 3D models can be generated using a class of deep learning algorithms like Style-based Generative adversarial networks (StyleGAN).

With Traditional GANs we have no control over the generated output [8]. NVIDIA proposed an alternative GAN called StyleGAN, which can fabricate hyper-realistic human faces with granular control over synthesis [6]. By using StyleGAN as the generation algorithm, The generated output is highly detailed with little to no imperfections, provided that the sample size is large.

Before the character can be rendered in a game engine, it has to be validated by the machine learning model. The model improves continuously by receiving feedback from the output mesh after every iteration. Normally creating a humanoid mesh requires a lot of vertices representing various facial landmarks, because of this the game engine can experience a lag when rendering more than 10 models at once. This can be optimized by making the machine learning model generate a UV map, a normal map for the NPC to apply texture on top of a base mesh. The base mesh is a simple 3D model resembling that of a low poly humanoid, By applying a high-quality texture to the normal map facial landmarks can be reconstructed without sacrificing performance. This process is called baking and is used widely in the industry to reduce render times.

Games that rely heavily on open-world game mechanics can make use of this approach to populate the world with NPCs that convey a sense of variety to the player. This technique can also be used in film making to create crowds fairly quickly. Machine Learning methods often take time to produce the expected output, Real-time generation of NPCs can be achieved by using a series of optimizations which help in cutting the time for an NPC mesh to render.

By procedurally generating NPCs, the game population reflects that of the real world where each person live their own life unconnected from the main character. Developers can easily populate their maps with thousands of NPCs with ease instead of the traditional model. This approach can save time in a developer's character creation pipeline.

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

Developers can also use the procedurally generated character as a base model and work on top of the model to make any refinements.

II. BACKGROUND AND RELATED WORK

A. Procedural Content Generation via Machine Learning (PCGML)

The idea behind PCGML is to use machine learning models to generate game content. The PCGML technique can be classified based on multiple data representation and training methods. In the paper, the author's focus mainly on procedural generation of levels, maps in 2D games [1]. This approach does not cover the procedural generation of content in a 3D environment. The authors explore some of the open problems in PCGML such as lack of training data, small datasets. It is important to mention that this approach allows for repair, critique and content analysis which is lacking in traditional PCG.

B. Photorealistic Facial Texture Inference Using Deep Neural Networks

The authors made use of a neural network to generate a photorealistic 3D model from a single image [7]. The approach is focused mainly on facial reconstruction, and can only work when there is an input image. By combining this approach with a procedural based machine learning model, the whole of the NPC mesh can be generated and not just the face.

C. Controlling NPC Behavior Using Constraint-Based Story Generation System

The authors introduced the concept of 'properties' that affect NPCs behaviour. This keeps the game world interesting as each NPC behaves differently making the interaction dynamic [4] [10]. By using 'properties' or 'attribute', NPC generation can be done procedurally with the help of a machine learning model with features like age, nationality, build, ethnicity, etc.

D. Adaptive Content Generation

Adaptive Content Generation introduces a method that takes into consideration the player, context and the game to gener- ate meaningful procedural content [5]. By incorporating this method into the NPC generation, we can adapt the content generation in real time that peaks the curiosity of the player. For example, when the player interacts with an NPC, the probability of that NPC to reappear in future instances can be increased. This helps the player develop a sense of familiarity within a video game.

E. Learning-Based Procedural Content Generation

The authors present a PCG framework that uses data obtained during game development and player testing to create content procedurally. This helps in generating content that matches the developer's intended style of asset creation [3]. Thereby making the generated content indistinguishable from manually created content

F. GAN

The prospect of creating a falsified image with the modern age technology stack has become a clear issue for the Cyber security. The deep-fake neural networks are able to generate realistic videos of a US president with ease, this raises doubts in many about the authenticity of digital content. But put into right use these technology stacks can be of great use to media and game development.

GAN(Generative Adversarial Networks) can be of excellent use in creating the NPCs (Non Playable Characters) of a game. GAN is made up of two neural networks trained and played at each other. The two neural networks are discriminator and a generator, the discriminator is useful in determining the obtained image is real or fake and generator works on building new images based on the data discriminator thinks as real [8]. In the simplified GAN the generator works with randomized data and learns to convert to distribution of real time data.

The generator never works with real time data, it is made to generate real time images based on the criticism it gains from the discriminator. But one of the major disadvantages of this two based neural network system is that if one of the neural networks gets dominant and generates a 100 percentage positive data on every outcome then the learning process of the other neural network is damaged.

G. WGAN (Wasserstein GAN)

WGAN is introduced by which both the neural networks can be at a constant process of learning and developing [11], the development of the neural networks are graphed based on sigmoid function so if a discriminator is always able to anticipate the fake data then it returns a value 0 to the generator using sigmoid graph in such case the use of generator neural network diminishes ,similarly if the generator is able to fool the discriminator 100% time then the sigmoid function generates 1 as a value stopping the learning process of the discriminator this can be prevented using WGAN which prevents from a 100% generation of any neural network thus maintaining a balanced system.

III. DESIGN AND IMPLEMENTATIONS

A. Problem Statement

The existing workflow of game development is tedious and requires game designers to create each asset individually. This waste of development resource can be decreased by using procedurally generated content to accelerate the development workflow. The characters generated by search based PCGs often look bleak and lifeless because there are a lot of constraints that the game developer has to consider to make the procedural algorithm works as expected. By using a machine learning based approach to PCG, the generated content will automatically infer constraints and evaluate the output with limited human intervention.

The paper proposes a method which generates diversified non-playable characters by combining the use of both procedural and machine learning based strategies. The primary motivation behind the idea is to make the game world feel more realistic by making the player feel more immersed in the game than when compared to the traditional NPCs in the existing games.

The characters that are not vital to the game's storyline are generally bland, predictable and often have similar facial features, apparel. Procedural generation of NPCs can be achieved by using deep learning to train the model to understand the correlation between the input attributes and the facial landmark information. This assists the machine learning model in creating a UV map where the normal map, textures can be applied to the base mesh to produce a character mesh. Gaming Industry stands to benefit considerably from the adoption of this new approach in populating the game world with characters that have distinct features.

B. Module Description

The central focus of this paper is to help reduce the work-load of character creation in the game development pipeline. This requires coordination between several modules to produce high-quality results that are expected by the development team. Continuous improvement of the learning algorithm is key in detecting and repairing the generated mesh.

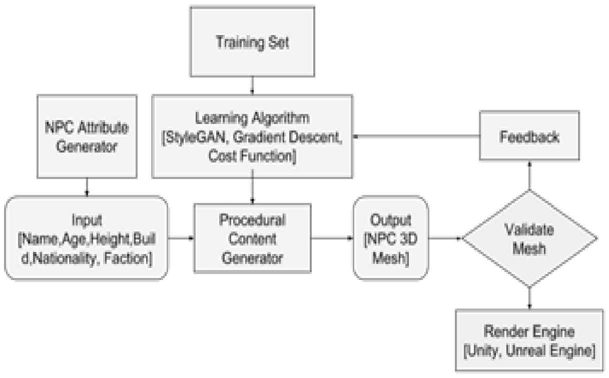
Training Set
NPC Attribute Generator
Learning Algorithm [StyleGAN, Gradient Descent, Cost Function]
Feedback
Input [Name,Age,Height,Build,Nationality, Faction]
Procedural Content Generator
Output [NPC 3D Mesh]
Validate Mesh
Render Engine [Unity, Unreal Engine]

Fig. 1 System Architecture

1) Training Set: The dataset must contain NPC features or attributes like age, sex, race, build, height, skills etc that define the character of an NPC. The training set must also have a target image so that the learning algorithm can determine the similarities between the facial features for a certain sample of attributes.

2) Prime number based StyleGAN: In traditional GAN algorithm there is the usage of sigmoid functions where the functionality ranges from -1 to +1 in a best case scenario as we have seen that the discriminator neural network achieves +1 for all the scenarios there by fooling the generator neural network which has been overcome in the WGAN algorithm by providing the generator neural network with x values of the sigmoid function instead of the f (x) values, but still the algorithm is limited to a range of 2 which can be overdrawn by the proposed algorithm. The algorithm is based on the prime numbers where as the discriminator neural network generates a noise value which is fed into the function as a x value and the generator neural network obtains the function fed f (x) values using the following method, Lets say the generated noise value is 47 the function sums the two numbers 4 + 7 thereby generating a sum of 11, The function again sums the two numbers 1 + 1 generating a 2 which is a prime number so the functionality sums the number generated by the discriminator neural network noise value until it is reduced to a single digit number

$97 \Longrightarrow 9 + 7 = 16$

$16 \Longrightarrow 1 + 6 = 7$

The discriminator noise values are fed to the x values of the function and the prime algorithm generated values that range between 1 and 9 on the f (x) side of the standard graph are fed into the generator neural network always when the graph obtains one of the numbers of the series (1, 2, 4, 5, 7, 8) generator distinguishes as real data and goes ahead for the series of (3, 6, 9) it assumes as fake data ,the (1, 2, 4, 5, 7, 8) series is prime number series and the (3, 6, 9) series is not.

3) Facial Landmark Analysis: Facial Landmark analysis helps in identifying the key facial features that are common to every character. So the locations of universal features like nose, ears, eyes, eyebrows, jaw can be used to predict the shape of a face. By understanding how these features map to the target image, the algorithm can learn to generate faces that are similar to the real world. This can help in avoiding the generation of abnormal or alien-like characters that have irregularities in their generated structure. By using a cost function the learning algorithm can fine tune its generation parameters and eventually reach a local maxima.

4) Attribute Generator: To generate a character procedurally, first, the character's attributes have to be generated ran- domly and then passed on as input to the learning algorithm. The learning algorithm then uses its experience gained from the training set to procedurally generate a character with the given sample of input.

5) Mesh Validator: A validator that checks if the created mesh is free from artifacts. Also to clean up the mesh object by removing double vertices, Back-face Culling. If the mesh is invalid it sends feedback to the learning model to improve the hypothesis. If the model is valid it is then sent to the rendering engine to display it in the 3D scene.

6) Mesh Analysis: Perform Mesh analysis to use quads instead of triangles wherever possible. Mesh analysis is useful in essentially detecting and employing strategies to repair any defects in the generated model.

7) UV mapping: The process of applying 2D image tex-tures to a 3D model is called UV mapping. The generated model is unwrapped to a 2D image, where the texture can be applied to the 3D model without any seams. The machine learning model gets progressively better at unwrapping the model and applying base textures to the UV map.

8) Bump Map generator: Creating realistic models often requires a high polygon model, this can impact rendering performance. One way to fix this issue without sacrificing the details on the models is by using a normal or bump map.

This helps in enhancing the details of a low poly model. The generation of normal maps is achieved by using the procedurally generated image to estimate the height difference. All faces have similar normal maps because of the frequently occurring pattern of facial landmarks, but the parameters of the base normal map can be slightly varied to produce different results.

9) Weight and Texture Painting: To make the generated faces look more natural, weight and texture painting can be done to the mesh to add facial hair, scars, makeup. Weight painting essentially decides what region of the mesh a particular texture is to be applied. Eyebrows, Beards, Eyelashes can be added to the face by increasing specific regions with hair density.
10) Apparel Generator: Using a layered approach, apparel textures can be placed on top of each other to make the character seem fully clothed. The generation of the types of apparel depends on the character's attributes like wealth, build, job/faction
11) Character Rigging: To make the character move, the character has to be rigged with a series of interconnected bones which forms a skeletal structure inside the polygon mesh. The movement is enacted by using Inverse Kinematics (IK). IK also prevents clipping by setting constraints that limit the bone movement either by rotation or location. Animations can be achieved by using reinforcement learning coupled with Inverse Kinematics to perform a particular task like walking, running etc
12) Behaviour: A Character's behaviour is mainly influenced by their attributes. There are many behaviours including, but not limited to posture, frequent spots, daily routines, job, relation with the characters nearby. By sticking to these simple rules, complex behaviour can be composed.

IV. RESULTS AND DISCUSSIONS

A. Results

Our initial goal of developing the prototype was to focus on the generation of 3D face using StyleGAN. The python libraries Tensorflow and Scikit Learn have been used to imple- ment the modified styleGAN algorithm that can help generate 3D assets. We have used the Chicago Face Database (CFD) to train the neural network, as it is a standardized face corpus with norming data that fits the project's use case. Each photograph from CFD has a resolution of 2444 x 1718 px, as a result the images had to be downscaled to 50% of the original resolution so that the prototype can be optimized to run on mid-range GPUs. Each pixel of an image is passed on as input to the neural network. The training sample image is split into multiple layers at each hidden layer of the network, allowing more granular control in the generation process. We have found that StyleGAN produces better results than compared to other deep learning techniques. Although StyleGAN takes longer and better hardware to train the quality of the generated artifacts is unmatched.

The sample set has been divided into 2 sets so that the training set is isolated from the test set. Having a test set allows learning algorithms to measure the performance with new or unseen sample instances.

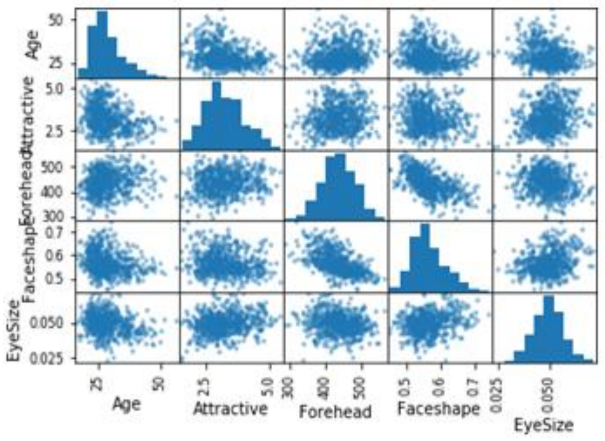

Fig. 2 Scatterplot of features from CFD norming data

Performing feature correlation test in the Chicago face database helped immensely in feature reduction to keep the model simple and also to find hidden relations between the attributes. Overfitting and Under fitting is one of the major problems that exists in any machine learning project, by using a stratified mini batch sampling approach the learning algorithm began producing legible results even when processing new instances. The discriminator uses a similarity check to validate the uniqueness of the generated character with that of the previously generated output.

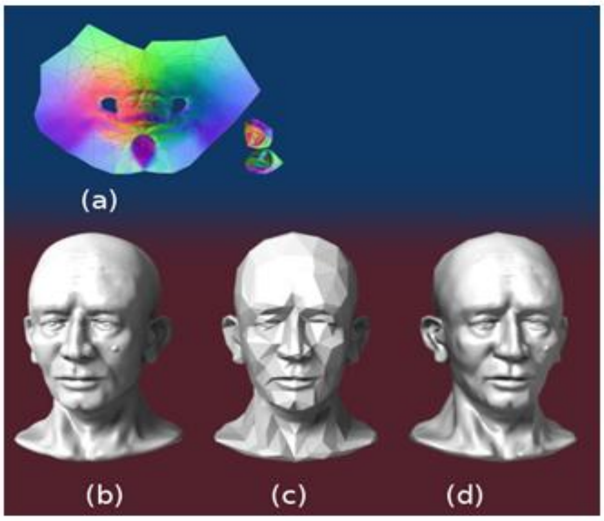

Fig. 3 Example Normal Map (a) on top of a Base Model (c)

The quality of the textures, normal map generated by the modified styleGAN made the base mesh appear realistic. As the sample set we chose did not contain the picture of the whole body, The generation algorithm needed to be fine tuned to use base meshes as a starting point to sculpt the desired features. This allowed features like build, height, age to influence the shape of the final character body by using style transfer techniques.

Generating apparel can also be achieved using StyleGAN, but we decided not to go ahead with that because it requires extensive datasets which we are lacking.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

As most humanoid characters have similar skeletal structure, rigging the model became fairly easy. With the help of inverse kinematics, the character's animation feels more fluid and expressive.

B. Render Engine

After the character has been generated, it is forwarded to the pipeline of the render engine where it is rendered in the scene with shadows, textures, reflections. Game engines like unity, unreal engine have specific requirements before the 3D model can be rendered, such as polygon limit, maximum VRAM usage. Generating characters in real-time requires GPUs that have tensor cores built-in to the architecture.

C. Face Database

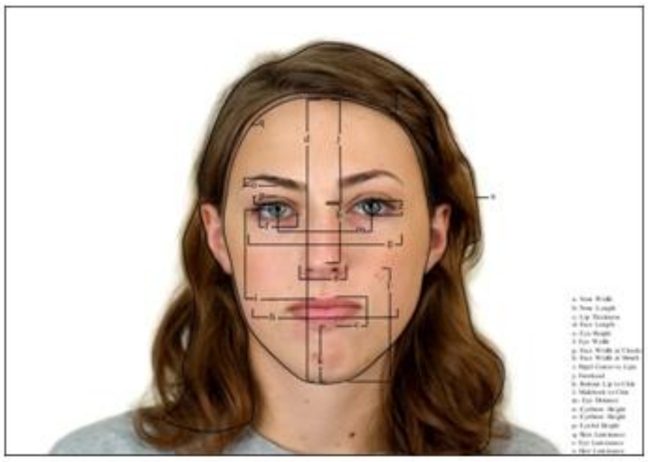

Fig. 4 Measurement Guide of CFD

For the approach presented in the paper, We've decided to use Chicago Face Database(CFD) [9], because the dataset contains high-quality target images with physical attributes like age, race, gender and facial landmarks which forms a basis for the learning algorithms

D. Optimizations

To optimize the rendering times of the generated NPCs, Level of Detail can be implemented into the render pipeline. This helps in reducing the number of details on a model when they are far away and increase fidelity when near. LOD can be achieved by generating textures with varying levels of quality.

E. Limitations

Dynamic animation of the rigged character model needs extensive datasets to train the model. But the datasets available for 3D assets is extremely small.

F. Future Work

To make the NPC interaction more realistic, we wish to develop ways to generate animations based on the context or environment that they are in. Next, we would like to explore means to make the game microcosmic, where each NPC has a role to play in the world. For example, if the player destroys a building, NPCs who are construction workers start working on making renovations to the old one. Also, NPCs don't grow old in games, we would like to look into ways to age NPCs, Player, Game World by using procedural modification techniques.

V. CONCLUSION

We have proposed a method to create NPCs that are inherently complex by adding depth to their character and appearance. Using Machine Learning to generate procedural content takes the menial task of designing out of the hands of the game developer. This automation of design tasks can give game developers a competitive advantage in the market, as they can now create complex environments fairly quickly. With decreased development costs, the game can now undergo many iterations to create a product that is polished and refined.

REFERENCES

1. N. Shaker, J. Togelius, and M. J. Nelson, Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer, 2016
2. A. Summerville et al., "Procedural Content Generation via Machine Learning (PCGML)," in IEEE Transactions on Games, vol. 10, no. 3, pp. 257-270, Sept. 2018
3. J. Roberts and K. Chen, "Learning-Based Procedural Content Gener- ation," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 7, no. 1, pp. 88-101, March 2015.
4. S. H. Moon, Minhyung Lee, S. Kim and S. Han, "Controlling NPC behaviour using constraint-based story generation system," 4th Interna- tional Conference on New Trends in Information Science and Service Science, Gyeongju, 2010, pp. 104-1
5. S. Oliveira and L. Magalhes, "Adaptive content generation for games," 2017 24 Encontro Portugułs de Computao Grfica e Interao (EPCGI), Guimaraes, 2017, pp. 1-8
6. Tero Karras, Samuli Laine and Timo Aila A Style-Based Generator Architecture for Generative Adversarial Networks in arXiv-eprints, Dec 2018
7. Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano and Hao Li Photorealistic Facial Texture Inference Using Deep Neural Networks in CoRR 2016
8. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in Advances in Neural Information Processing Systems, 2014, pp. 26722680.
9. D. S. Ma, J. Correll, and B. Wittenbrink. The Chicago face database: A free stimulus set of faces and norming data. Behaviour Research Methods, 47(4):11221135, 2015.
10. G. N. Yannakakis and J. Togelius, Experience-driven procedural content generation, IEEE Transactions on Affective Computing, vol. 99, pp. 147161, 2011
11. Arjovsky, Martin and Chintala, Soumith and Bottou, Le´on "Wasserstein GAN" in arXiv-eprints, Jan 2017

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*