SELF-EVOLVING PLATFORMER

Shreyash Srivastava[1], Sarthak Sharma[1], Rahul Tripathi[1], Samanvitha S[1] and K Deepa Shree[2]

[1] *Student, Department of CSE, Dayananda Sagar Academy of Technology and Management, Bangalore, India*

[2] *Assistant Professor, Department of CSE, Dayananda Sagar Academy of Technology and Management, Bangalore, India*

ABSTRACT

Digital games have been one of the most popular genres of software ever since the boom of computers. The current market of video games holds a volume of $250Bn and is growing yearly. Hence video games hold the potential to have the largest share of global audience. Video games evolve every year and machine learning and artificial intelligence have been a recent trend. The Self-evolving Platformer software development kit aims to be the layer that bridges the gap between classic video games and the artificial intelligence layer. SEP-SDK leverages the features of machine learning tools to create a generative and interactive environment for the player giving a personalized and immersive environment.

Keyword: - *Game, Platformer, SDK, Artificial Intelligence, GAN*

1. INTRODUCTION

The self-evolving platformer SDK serves as a compelling proof of concept, showcasing the utilization of artificial intelligence in crafting a dynamic environment within an existing game. This innovative SDK has been seamlessly integrated into Super Mario, one of the industry's longest-standing games, enabling the generation of levels based on prompts provided by users. Within this paper, we delve into the integration of cutting-edge stable diffusion technology, employed to produce in-game assets in real-time. The introduction of artificial intelligence has rendered traditional GAN models, which developers have relied upon for years, obsolete. By leveraging stable diffusion and generative AI, we are empowered to process information swiftly and generate game assets on-the-fly, thereby significantly reducing the overall size of the game build. Consequently, players are afforded a more immersive and unparalleled gaming experience that is truly unique.

2. PROBLEM STATEMENT

The existing approaches and methodologies for game development heavily rely on traditional architectural design, which unfortunately falls short in seamlessly integrating artificial intelligence (AI) into games. This limitation arises due to the lack of provisions within the current tools and frameworks, impeding the seamless incorporation of AI capabilities. Moreover, these tools suffer from a lack of programmability, constraining developers from customizing and extending their functionalities to meet specific requirements. Additionally, the absence of robust support for external plugins restricts the integration of third-party AI libraries and services, further impeding the utilization of AI in game development. Consequently, there is a pressing need for novel methods and tools that address these shortcomings, empowering game developers to effortlessly leverage AI and explore its vast potential for enhancing gameplay, creating dynamic environments, and delivering immersive player experiences.

3. RELATED WORKS

These publications have served as valuable sources of inspiration for our research, highlighting the immense potential of adversarial networks in reaching a wider audience within popular games [1]. The study on game assessment using GANs elucidates the application of generated adversarial networks in dynamically generating levels within games, leveraging an analysis of prevalent patterns [2]. Furthermore, delving deeper into the exploration of game levels and dynamics, we encountered a publication that elucidates the workings of procedural level generation and its applicability in games like Super Mario [3]. Additionally, we came across alternative approaches such as N-gram models, which utilize existing datasets to generate novel levels, offering another perspective on level generation techniques. These insights from the aforementioned papers have significantly influenced our research direction, shaping our understanding of the possibilities and challenges associated with leveraging AI techniques for game content generation.

4. METHODS AND MODULES

The SEP-SDK's functionality can be broadly divided into seven abstract modules which are:

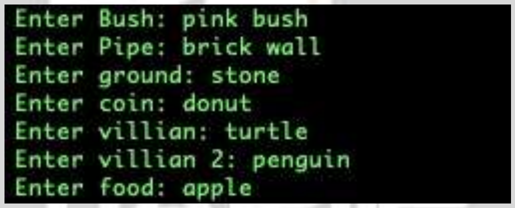
4.1 Game-Engine

The current implementation uses the pygame package as the game engine. Pygame offers a number of utilities for drawing game frames and synchronizing the rendering of those frames. It also provides utilities for implementing 2D physics in the game. The SEPSDK uses the game engine as an interface, and as such, it can be swapped with any other game engine. The module also imports the other sprites and assets needed in the game. It then initializes the SDK with a skeleton setup, which the other modules use.

The game state is updated by the game logic. The game logic is responsible for determining the behavior of the sprites in the game. The game frames are rendered by the renderer. The renderer is responsible for drawing the sprites to the screen. The next frame is waited for by the clock object. The game loop continues until the user quits the game. When the user quits the game, the module cleans up the resources that it used.

4.2 Prompt input interface

After receiving the user's prompt, it is crucial to clean and prepare it for further processing. This involves utilizing a prompt tokenizer, which takes the user's input and extracts significant keywords and objects. The tokenizer then generates a bag of words, comprising the essential elements from the prompt. This bag of words is then seamlessly transferred to the subsequent step of the pipeline, where it undergoes further processing and analysis to produce the desired outcome. By effectively cleaning the prompt and extracting its key components, the pipeline can effectively understand and address the user's needs or inquiries, leading to more accurate and relevant responses.



```
Enter Bush: pink bush
Enter Pipe: brick wall
Enter ground: stone
Enter coin: donut
Enter villian: turtle
Enter villian 2: penguin
Enter food: apple
```

Fig -1: Taking user prompt

4.3 Object sentiment classifier

Once the bag of words is obtained, it serves as a crucial input for determining the sentiment associated with each word or object. This sentiment analysis helps categorize the words into different in-game characters for the current game, such as power-ups, walls, coins, in-game currency, non-playing characters, and enemies like Goomba in the case of Super Mario. Additionally, the sentiment analysis assigns a weight to each object, indicating the percentage of certain behaviors and in-game interactions that the object would exhibit.

Subsequently, these categorized objects and their corresponding properties are utilized to generate images through the utilization of object names and properties as input for stable diffusion. The images generated from stable diffusion undergo further refinement to eliminate any background noise using a Python package called "bg-remover." This cleaning process ensures that the resulting images are clear and free from any distracting or unwanted elements, providing a visually pleasing and immersive gaming experience.

4.4 Real-world Object to in-game Mapper

After undergoing the necessary cleaning procedures, the images from the previous step undergo a crucial transformation into the spruce format. This format conversion serves as a pivotal preparatory stage before the images are seamlessly integrated into the game engine. The game engine plays a pivotal role in the overall gaming experience by serving as the core component responsible for rendering and executing the game's logic. Within the engine, the converted images are meticulously mapped onto in-game objects, enabling a seamless fusion of the visual and interactive elements. This mapping process establishes a vital connection between the appearance and behavior of the in-game objects.

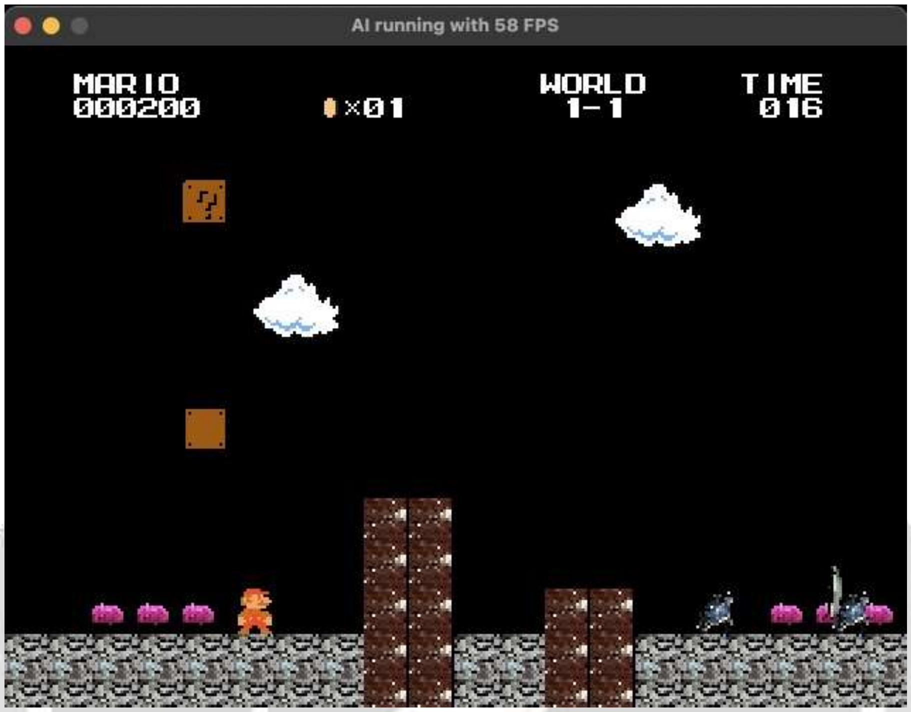


Fig -2: Generated Level Start

Once successfully imported into the game, these transformed objects become interactive entities that players can engage with, imbuing the game with a heightened level of realism and immersion, by integrating the cleaned images into the game engine and mapping them onto in-game objects, players are presented with a visually appealing and dynamic virtual world. The inclusion of interactive objects possessing unique characteristics not only introduces a range of options and variations to the gameplay but also cultivates a profound sense of involvement and active participation.

This elevated degree of interactivity guarantees that players establish a deeper bond with the virtual world, culminating in a more gratifying and absorbing gaming venture. By incorporating interactive objects with distinct properties, players are granted agency and the ability to shape their experiences within the game.

The described process is of significant importance in game development, as it plays a crucial role in enhancing immersion and interactivity. By transforming cleaned images into interactive objects with distinct properties, players are able to fully immerse themselves in the virtual environment. This heightened level of engagement empowers players to have a sense of control and influence over their gaming experiences. Moreover, the inclusion of diverse objects adds a much-needed variety, ensuring that the gameplay remains fresh and captivating. Additionally, integrating visuals into the storytelling aspect of the game enhances the overall narrative experience, making it more compelling and memorable for players. Ultimately, the aim of this process is to maximize player satisfaction and retention by creating an engaging and immersive gameplay journey filled with interactive elements.

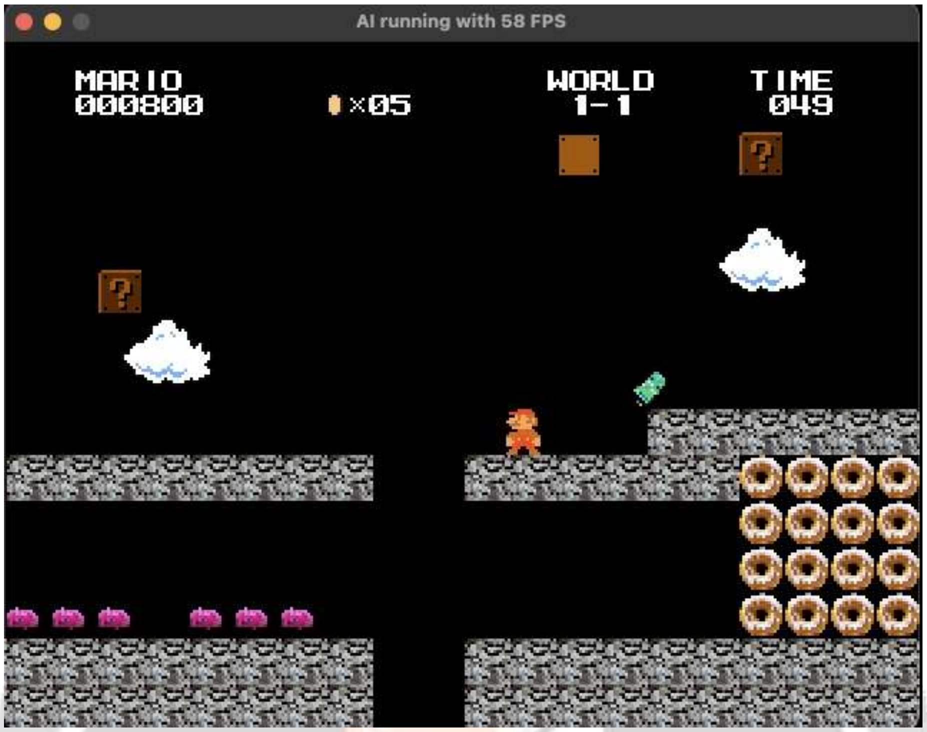


Fig -3: Generated Level End

4.5 Level generator

In the final step, the process begins with the utilization of a statistical language model known as an n-gram model. This model plays a vital role in generating new dynamic levels. It examines the textual content of existing levels and identifies patterns and correlations in the word order. By understanding these patterns, the n-gram model becomes capable of generating fresh levels that bear resemblance to the existing ones while incorporating some degree of variation.

To further enhance the generated levels, newly created sprite sheets are employed. These sprite sheets consist of images that represent various objects within the game. The generated levels are then adorned with these sprite sheets, effectively assigning visual representations to the corresponding in-game objects. Each level object is associated with a weight, which determines its likelihood of being selected. This weighting mechanism helps to introduce diversity and prevent the gameplay from becoming monotonous or predictable.

In summary, this process harnesses the power of an n-gram model to analyze and understand existing level patterns, enabling the generation of new dynamic levels. The incorporation of sprite sheets and their weighted attachment to level objects adds visual appeal and injects variation into the gameplay, contributing to a more engaging and enjoyable gaming experience.

5. CONCLUSIONS

Stable diffusion is a type of generative model that can be used to generate images from noise. It works by gradually adding noise to an image until it resembles the desired image. This process is called diffusion, and it is what gives stable diffusion its name. Our method uses stable diffusion to generate high-quality images of game assets. We start with a random noise image and then gradually add noise to it until it resembles the desired game asset. We can control the amount of noise that we add to the image, which allows us to generate a variety of different game assets. Our method can be used to generate game assets for a variety of different genres. For example, we can use our method to generate characters for first-person shooters, objects for role-playing games, and environments for real-time strategy games. We can also use our method to generate game assets that are tailored to the specific needs of the player. For example, we can generate game assets that are more challenging or more enjoyable to play. Our method is still under development, but it has the potential to be a valuable tool for game developers who want to create high-quality game assets quickly and easily. Our method is able to generate game assets in real-time, which means that game developers can create new levels or characters without having to wait for the assets to be generated. This can save game developers a significant amount of time and effort. Overall, we believe that our method is a promising new tool for generating dynamic game assets in real-time. We believe that our method has the potential to revolutionize the way that game assets are created.

6. REFERENCES

[1]. Matthew C. Fontaine1 et al - Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network. 21 Jun 2021.

[2]. Karp, Rafał & Swiderska, Zaneta. (2021). Automatic generation of graphical game assets using GAN. 7-12. 10.1145/3477911.3477913.

[3]. Kerssemakers, Manuel & Tuxen, Jeppe & Togelius, Julian & Yannakakis, Georgios. (2012). A procedural procedural level generator generator. 2012 IEEE Conference on Computational Intelligence and Games, CIG 2012. 335-341. 10.1109/CIG.2012.6374174.

[4]. Dahlskog, Steve & Togelius, Julian. (2014). A Multi-level Level Generator. 10.1109/CIG.2014.6932909.

[5]. Awiszus, M.; Schubert, F.; and Rosenhahn, B. 2020. TOADGAN: coherent style level generation from a single example. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 16, 10–16.

[6]. J. Togelius, N. Shaker, and M. J. Nelson, "Introduction," in Procedural Content Generation in Games: A Textbook and an Overview of Current Research, N. Shaker, J. Togelius, and M. J. Nelson Eds. Springer, 2014.

[7]. L. Johnson, G. N. Yannakakis, and J. Togelius, "Cellular Automata for Real-time Generation of Infinite Cave Levels," in Proceedings of the ACM Foundations of Digital Games. ACM Press, June 2010.