Stable Diffusion X Dreamhold

Lukas Schmalzer, Stefan Schwaiger

January 31, 2023

Abstract

This project aims to use the text-to-image model *Stable Diffusion* for visualizing the interactive fiction game *The Dreamhold* during live play.

As soon as the player enters a new location, the respective description is taken from the game and used as part of a prompt to create an image and visualize the current game environment for the player.

In addition to this implementation, we tried to modify the prompts in order to optimize the resulting images and find the best generation parameters for this specific use case.

Contents

Chapter 1

Introduction

Stable Diffusion X Dreamhold is a semester project of the course Media-Technology and -Design at the campus Hagenberg of the University of Applied Sciences Upper Austria. The goal of the project is to combine state-of-the-art AI image generation technology with classic interactive fiction games in order to expand the gameplay of a text-based adventure by adding a visual representation of the in-game environments.

While the goal of the project was specified as mentioned above, we were given full freedom concerning the choice of interactive fiction game and image generation model as well as how the interaction between the two should be implemented.

Chapter 2

Background and Related Work

2.1 The Dreamhold

The interactive fiction game *The Dreamhold*, created by Andrew Plotkin a.k.a. Zarf in 2004 [3], offers detailed descriptions of its fantastical environments, mixing nature, technology as well supernatural elements, making it a great candidate for AI-based visualization.

When it came to choosing a text-based game as a basis for image generation, however, not only personal preference was a factor, but also how we could access the text describing the game areas. This turned out to be quite the challenge, as we could not access the game's output directly.

One of the approaches we implemented used optical character recognition to interpret the text from a specified section of the screen. While this worked surprisingly well, we then ran into problems when it came to distinguishing between non-relevant game text and text actually describing environments. Therefore, this method proved to create more problems for us than it would solve and we ultimately decided against it.

We needed to find a way to only get the text relevant for visualizing the environments while ignoring the rest, preferably without having to analyze the output with resource-intensive natural language processing methods. The solution for this turned out to be an online interpreter for interactive fiction games called Parchment found on iplayif.com [1] [5]. It allows users to upload and run z-code files (a format for running text adventure games) directly in their browser. This method of running the game turned out to be perfect for our purposes as it displays the name of each individual area that the player enters within a dedicated subheader class in its HTML code. Dreamhold on the other hand tends to always give a detailed description of a new area directly beneath the area's name. This allowed us to simply access this relevant part of the HTML using the beautiful soup package [4] for python and return the contents of the first element under each area-subheader as a string.

[1] Parchment can be accessed online through: https://iplayif.com/

2.2 Stable Diffusion

We decided to use Stable Diffusion 2.1 [2] for generating the environments as the code (including model weights) is publicly available. Additionally, Stable diffusion can be run locally, even on moderate GPUs, and doesn't require cloud services to be used. Specifically, we used the Stable Diffusion web UI [1], as it allows for relatively seamless access to the image generation process through an API running in Python. After the web UI of Stable Diffusion has been started, the prompt as well as all parameters relevant to creating the image can be sent to the model through the local IP. The created image is then returned to the python application where it can be processed further or displayed. Our prototype opens up a browser window (Chrome specifically) running Dreamhold while a separate GUI window is shown next to it in order to display the generated environments.

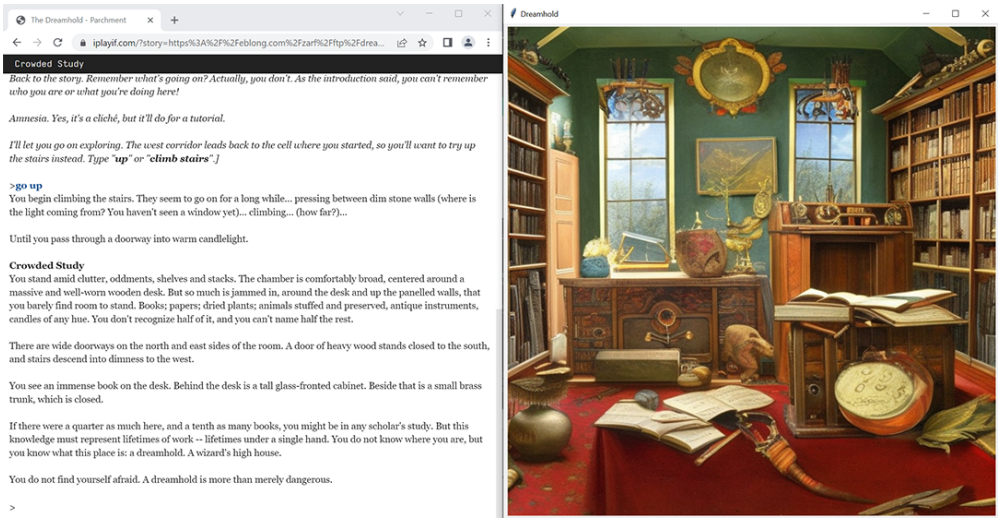

Figure 2.1: Screenshot of the project UI

Chapter 3

Testing and Results

3.1 What We Tested

The following section features examinations we made in order to find optimal settings for our use case. Specifically, we investigated the impact of various parameters of Stable Diffusion on image generation. These adjustments include the modification of the input prompt, the number of sampling steps, the resolution, and the CFG scale. It is shown that these parameters can have a significant effect on the generated image, even when the noise pattern, determined by the seed, remains unchanged. By comparing test results, parameter values were determined which provide the best results in terms of image quality and computation time. Since the images are generated in a live gaming scenario, it is crucial that the computation time does not exceed an acceptable delay for the player. Additionally, the project should be able to run without requiring a high-end GPU. Finally, future possibilities are presented that aim to further improve the generation of images in the context of live image generation in games.

3.2 Prompt Optimization

In the course of the project, it quickly became apparent that stable diffusion performs better with default settings when it receives more information and the prompt is structured differently. Since the text used for prompts is directly taken from the games' environmental descriptions, the extracted strings can become rather long. In order to still provide succinct instructions for Stable Diffusion, we thought it might help to shorten the descriptions by automatically summarizing them. Our first attempt at this was by using the Python package Natural Language Toolkit. While this method could summarize the descriptions quite inexpensively in terms of computational resources, the resulting prompts often lacked specific details important for correctly displaying the area. Because of this, we decided to try out a more resource-intensive approach by summarizing the prompts using the GPT-3 NLP model and comparing it to a prompt directly taken from the game. To test for the resulting behavior and see the differences, we used the prompt in three different variants: One without modifications, one summarized with GPT-3, and a self-modified version. In the modified version, a pre-prompt was inserted describing that an image of an environment should be created. At the end

of the description, adjectives, and descriptions were added to improve the structure, layout, and lighting situation. Then, a specific style of an artist or era is chosen to give the image a unique charm and characteristics. Finally, a negative prompt is defined to suppress features and aspects during the generation process. To see the best results, the seed value has always been kept the same, so the same random values are generally used.

The test images now show the different behavior through the changed descriptions.

No changes:

Figure 3.1: The unmodified prompt does a good job of representing the environment, but there is a lack of clearly defined structures and interesting objects in the image.

Summarized:

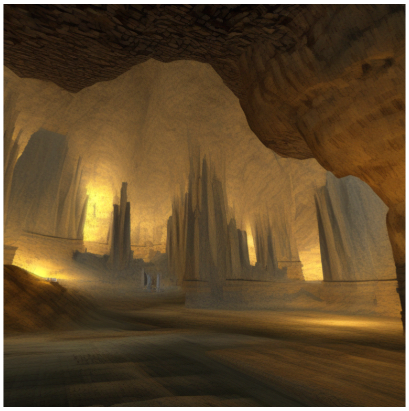

Figure 3.2: The summarized image, in contrast, shows a better structure and a real room. However, by shortening and summarizing, some described details are missing, and thus the image does not look as interesting to the viewer's eye.

Modified:

Figure 3.3: The modified image offers a dramatic difference at first glance. The image consists of many more textures and geometry that can be distinguished from each other. More light sources provide a very contrasty and dark image, with the colors in particular being much more pronounced and impressive. However, some described features are missing or at least leave room for interpretation as to whether and how they were incorporated into the image. Additionally, a complete castle was inserted, which does not occur in the description.

Prompt Optimization - Summary:

The unmodified description provides a good basis in principle, as many objects and features from the text are transferred to the finished image. However, it lacks interesting color, geometric, and structural aspects. The same can be said about the summary, as too little information is transmitted here. Even if some aspects are composed, the modified prompt is convincing due to its colors, perspective, and general features in the finished image. To further refine the result, the negative prompt would have to be expanded and the given circumstances further adapted.

3.3 Sampling Methods

Stable Diffusion is a Latent Diffusion Model, which roughly means that the input text is analyzed and transformed into a sort of multiple layers. These layers then dictate how the final image should look. Using a randomly generated noise pattern and differential equations, Stable Diffusion's different algorithms create an image. The more samples or iterations that are run, the more detailed the final image becomes. The current version of Stable Diffusion's web UI comes with a total of 19 different sampling methods. The difference between these methods relates to the calculations used to implement the noise pattern. Many of the methods produce similar-looking images. However, it's interesting to note that some methods can generate the exact same image, but with variations in sharpness and art style.

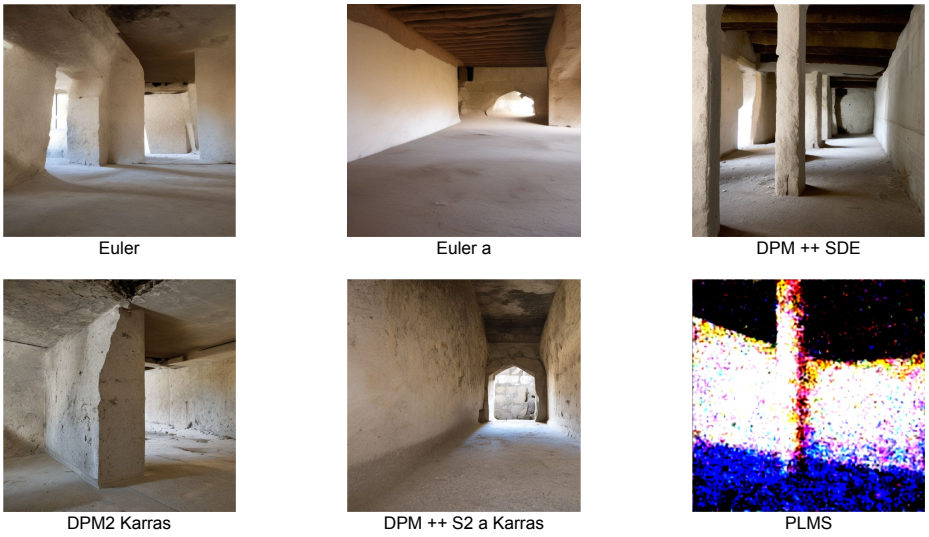

Figure 3.4: Different sampling methods used

Sampling Methods - Summary:

In the test, the 19 different methods generated 5 distinct images that varied greatly in structure, light, and sharpness. Some of the samplers tend towards a more photorealistic style, while others tend towards a more artistic style. Four methods, in particular, stand out: Euler, DDIM, DPM++ SDE and DPM++ 2S a Karras all produced very detailed and not overly sharp or too soft results compared to the other methods. DPM fast and the PLMS methods, however, produced results that were not usable.

3.4 Sampling Steps

Different sampling rates produce noticeably different results. While the general structure of the image remains unchanged, textures and geometries become sharper and the image becomes more detailed overall. In this section, we test using two different scenes on how many samples are needed to generate a good-looking image. The sample count is contrasted with the computation time to find the best mean values.

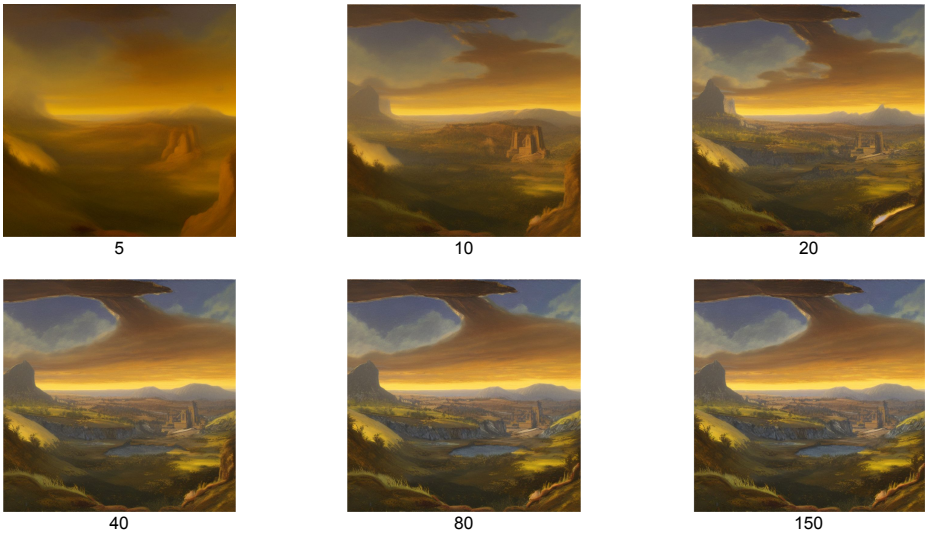

Figure 3.5: Different number of sampling steps used

Sampling Steps - Summary:

The results now confirm the original claims by showing that the images become more detailed and pronounced with increasing sampling steps. At 20 steps, all basic shapes and colors are present in the image and are only refined from this point on, which in turn takes a lot of extra time. In general, it can be stated that the 20 steps in our experiment strike a good balance between generating speed and detail richness. However, an interesting aspect also arises during the testing that was not previously considered. While the image of the "Narrow Hallway" with 5 sampling steps is not usable due to many artifacts and unclear lines, the "Ledge Night" image shows a very abstract but visually appealing interpretation of the finished image. The image is reminiscent of a watercolor painting, which only hints at the most important features in the corresponding regions but is still recognizable. Of course, objects and details are not included that are specifically described in the text, but this would also be a great approach for a new game mode if the prompt is short and leaves a lot of room for interpretation.

3.5 Sampling Steps vs Sampling Methods

In the previous test, we observed that a rough value of 20 sampling steps provides a sweet spot for the details and computation time. Therefore, we next investigate which sampling method yields the best results in this range. To do so, we test the four best methods (Euler, DDIM, DPM++ SDE, and DPM++ 2s a Karras) from the sampling method test with various sampling steps ranging from 1 to 25.

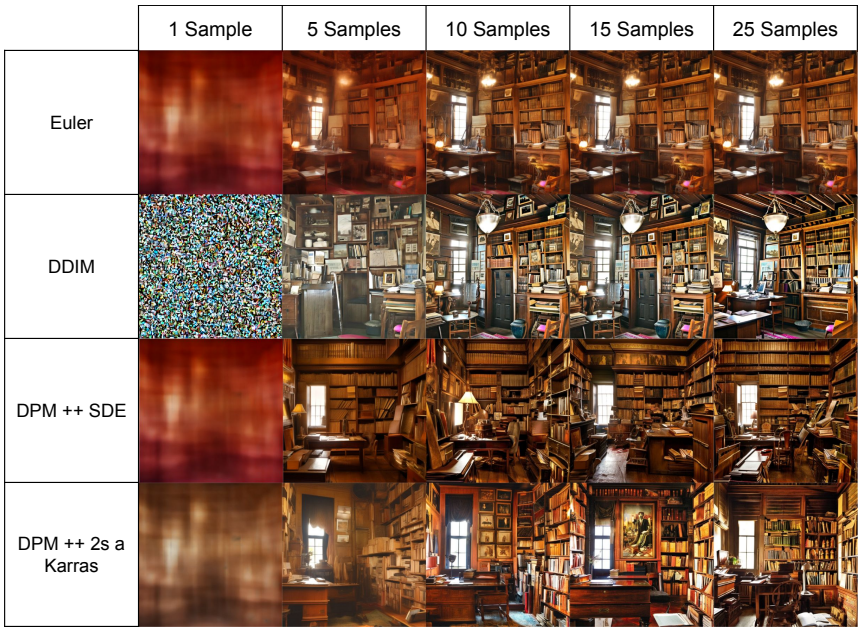

Figure 3.6: Showcase of differences in sampling methods and steps

Sampling Steps vs Sampling Methods - Summary:

It is very interesting to see how the various methods calculate images from the initial noise. With only one sample, different colors and structures can be observed, from which the geometry is formed. It is also fascinating to see how the DDIM algorithm greatly differs from the others and that the Euler and DPM++ SDE methods generate exactly the same image with only one sample. All four methods deliver visually appealing and detailed images from 10 to 15 steps onward. However, all methods, except for Euler, change the objects with each iteration and thus deliver a completely different image every time. Only the Euler algorithm remains relatively constant during the creation process and becomes increasingly detailed with each iteration.

3.6 CFG Scale

The Classifier-free-guidance scale, or CFG scale, is used to determine the level of alignment between the generated image and the input prompt or image during the image generation. The default value for the CFG scale in stable diffusion is seven, as it represents a good balance. In the next section, we will test different values of the CFG scale between one and 30 and attempt to determine which range is most suitable for image generation.

Figure 3.7: Showcase of different CFG scales used

CFG Scale Summary:

As can be seen in the images, the results are visually striking at a low value. But as the value increases, the image becomes increasingly more accurate in its representation of the input prompt but also introduces more noise and artifacts. At a high value, the algorithm struggles to fit the described elements into the image. At a very low value, the final image resembles a work of art, and all the objects in the image are harmoniously coordinated, but the image is not accurate. Some information is missing and some elements are interpreted differently because they are either not described or cannot be generated. As we expect visually interesting results for the game, we find a value of four to be very appealing. It strikes the best balance between details and visual appeal.

3.7 Input Complexity vs CFG Scale

In the previous test, it was noted that the algorithm seems to struggle with the interpretation of non-defined inputs when the CFG value exceeds a certain threshold. Therefore, in this experiment, we aim to investigate how the generated images change when presented with a long and detailed prompt versus a short and open-to-interpretation text. In this test, CFG values between one and ten will be tested.

Figure 3.8: Varying degrees of prompt complexity with adjusted CFG scale (1, 5, 10): low complexity on the left to high complexity on the right

Input Complexity vs CFG Scale - Summary:

The resulting images show mixed results. On one hand, the "Crowded Study" prompt demonstrates a clear improvement in image quality with higher CFG values, producing more detailed representations of the prompt. On the other hand, the short "Narrow Hallways" prompt did not produce a convincing image with a low CFG value, as originally expected. Given that no modified prompts were used as inputs, it can be argued that these simply provide insufficient context for scene creation and lack a default artist or style to rely on.

3.8 Summary of the Tests and Final Implementation

Finally, several observations can be made. The most important parameter by far is the prompt itself. A good input consistently gives the best results in terms of richness in detail and visually appealing images. It is equally interesting to see how different sampling methods alter the image and how some of them tend towards a more photorealistic representation, while others produce a more artistic image. The sampling rate plays a secondary role; it is important not to fall below a certain threshold of 15-20 sampling steps, as results will then suffer from heavy artifacts. However, all methods produce a very detailed image above this threshold, even though some sampling methods continually reshape the image. The CFG Scale is also an important parameter that can help to a great extent when the quality of the input prompt is poor. However, it really only works wonders when the prompt itself is very descriptive about the player's surroundings. All of these parameters together can greatly improve the image during creation. If you know how, photorealistic or artistic masterpieces can be created in a

playful manner, which can greatly enhance the gaming experience.

Which Parameters did we use:

In order to achieve a seamless experience with minimal wait time for the player, we opted for a standard parameter of 20 sampling steps per generated image. This value strikes a nice balance between performance and visual detail. Additionally, the standard value of 7 for the CFG Scale is used, as it also strikes a nice balance in most tested cases. Given the varying levels of detail in the prompts, a compromise must be made. Ultimately, the image is most strongly influenced by the input prompt, which is where our primary focus lies. Our pre-prompt and additional details result in visually appealing images. We have also added the option to use different artists or artistic styles from different eras, providing a visually unique result with each new playthrough.

Chapter 4

Outlook

The results we have obtained thus far demonstrate a great and exciting possibility of supporting purely text-based games with visual content. While we are mostly satisfied with the parameters we have chosen, they are ultimately middle-ground solutions in some areas due to the diverse nature of the prompts. To address this issue, two possible approaches have come to mind that can be tested in the future: 1. Training a custom machine learning model that converts the input prompts into prompts that work best for Stable Diffusion. The program would likely achieve the best results, but a lot of data would be required for the training and validation of such an approach. 2. Another possibility would be to use an existing NLP model and test the prompt for its complexity. If the complexity and description of the environment are minimal, we could counteract with our own presets and adjust the CFG Scale accordingly. On the other hand, if a good description of the environment is available, additional pre-defined prompts could be added and the CFG Scale increased. This way, a good balance for creating detail-rich images would be achieved even with different prompts and complexities.

References

[1] AUTOMATIC1111. 2022. URL: https://github.com/AUTOMATIC1111/stable-diffusion-webui (cit. on p. 5).

[2] CompVis group LMU Munich; Runway; Stability AI. 2022. URL: https://github.com/CompVis/stable-diffusion (cit. on p. 5).

[5] *Parchment*. 2020. URL: https://github.com/curiousdannii/parchment (visited on 01/31/2023) (cit. on p. 4).

[3] Andrew Plotkin. *The Dreamhold*. 2004. URL: https://zarfhome.com/dreamhold/ (cit. on p. 4).

[4] Leonard Richardson. 2004. URL: https://www.crummy.com/software/BeautifulSoup/ (cit. on p. 4).